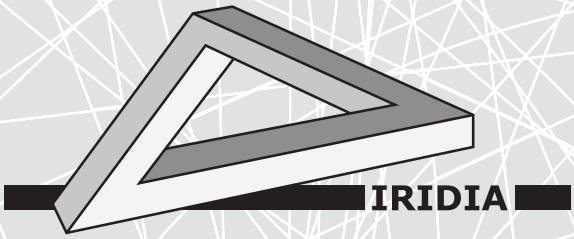


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Improving Performance via Population Growth and Local Search: The Case of the Artificial Bee Colony Algorithm

Doğan Aydin, Tianjun Liao, Marco A. Montes de Oca, Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2011-015

August 2011

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/2011-015

Improving Performance via Population Growth and Local Search: The Case of the Artificial Bee Colony Algorithm

Doğan Aydın[1], Tianjun Liao[2], Marco A. Montes de Oca[2], and Thomas Stützle[2]

[1] Dept. of Computer Engineering, Dumlupınar University, Kütahya, Turkey
dogan.aydin@dpu.edu.tr,
[2] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{tliao,mmontes,stuetzle}@ulb.ac.be

Abstract. We modify an artificial bee colony algorithm to (i) make the population size grow over time, and to (ii) apply local search on strategically selected solutions. The modified algorithm obtains very good results on a set of large-scale continuous optimization problems. This is not the first time we see that these algorithmic components make an initially non-competitive algorithm produce state-of-the-art results. In previous work, we have shown that the same modifications substantially improve the performance of particle swarm optimization (PSO) and ant colony optimization (ACO) algorithms. Altogether, these results suggest that population growth and local search are key components for obtaining high-quality results.

1 Introduction

Thinking of optimization algorithms made of components has changed the way optimization algorithms are designed. Research based on components and not on specific implementations of algorithms promises to lead to breakthroughs. In previous work, we have taken this perspective and we have obtained interesting results for IPSOLS [1, 2] and IACO$_{\mathbb{R}}$-LS [3], which instantiate the hybridization of the incremental social learning framework [1] with a local search procedure. Incremental social learning improves the scalability of a particle swarm or an ant colony composed of multiple learning individuals while a local search enables an individual a fast exploitation to reach good solutions. The resulting IPSOLS and IACO$_{\mathbb{R}}$-LS algorithms significantly improved PSO and ACO$_{\mathbb{R}}$, respectively. Their performance was competitive with state-of-the-art algorithms in the special issue of the Soft Computing journal on large-scale continuous optimization [4]. (Throughout the rest of the paper, we will refer to this special issue as SOCO.) We illustrate that improving particle swarm optimization (PSO) and ant colony optimization (ACO) algorithms via population growth and local search is not just a lucky case, by using a third case study.

In this paper, we propose an incremental Artificial Bee Colony algorithm with local search (IABC-LS), which applies the incremental social learning framework and a local search procedure to an Artificial Bee Colony (ABC) algorithm [5,6]. Powell's conjugate directions set [7] and Lin-Yu Tseng's Mtsls1 [8] methods are considered as local search procedures. By further benchmarking the algorithms

on SOCO's large scale function suite, we analyze the scalability of ABC-based algorithms. The experiments indicate that IABC-LS significantly improves ABC and that it is competitive with other state-of-the-art algorithms. Meanwhile, the improved performance of IABC-LS suggests once more that population growth and a local search are effective to improve a population-based optimization algorithm.

2 Related Work

The incremental social learning framework [1] facilitates the scalability of systems composed of multiple learning agents by adding to the population one agent at a time according to a schedule. The population is initially composed of a small number of agents to allow a faster learning. Then, agents are incrementally added to the population to learn socially from the interaction with those agents that have been in the population. The incremental social learning framework with a local search procedure was first applied to improve the performance of PSO algorithms [1]. The local search procedure employed in that paper was Powell's direction set method [7]. The tuned IPSO+Powell algorithm [2] was successfully benchmarked on SOCO for large scale optimization problems and compared favorably to other state-of-the-art algorithms that include differential evolution algorithms, memetic algorithms, particle swarm optimization algorithms and other types of optimization algorithms [4]. In SOCO, a differential evolution algorithm (DE) [9], the covariance matrix adaptation evolution strategy with increasing population size (G-CMA-ES) [10], and the real coded CHC algorithm (CHC) [11] were used as reference algorithms.

Later, we proposed IACO$_{\mathbb{R}}$-LS [3], which is a variant of ACO$_{\mathbb{R}}$ that uses a local search and that features a growing solution archive. We experiment with Powell's conjugate directions set [7], Powell's BOBYQA [12], and Lin-Yu Tseng's Mtsls1 [8] as local search procedures. Automatic parameter tuning results show that IACO$_{\mathbb{R}}$-LS with Mtsls1 (IACO$_{\mathbb{R}}$+Mtsls1) is not only a significant improvement over ACO$_{\mathbb{R}}$, but that it is also competitive with the state-of-the-art algorithms described in SOCO. Further experimentation with IACO$_{\mathbb{R}}$+Mtsls1 on an extended benchmark functions suite, which includes functions from SOCO and CEC 2005 [13], showed that it obtained better performance than IPSO+Powell and IPSO+Mtsls1.

Dynamic population sizing has also been studied within the field of evolutionary computation for many years [14–17]. Most strategies found in the literature consider the possibility of reducing the size of the population during an algorithm's run. However, there are algorithms in which the population size is not decreased. In G-CMA-ES [10], the population size of a CMA-ES algorithm is doubled each time it is restarted. The goal of combining local search techniques with population-based algorithms is to accelerate agents to search good locations. This research issue has been addressed by the PSO community [18–22]. Differently, we combined an increasing population with local search procedures in IPSOLS and IACO$_{\mathbb{R}}$-LS, in which an adaptive initial position of added individuals and an adaptive step size for the local search are used.

Algorithm 1 The Artificial Bee Colony Algorithm

initialization
while termination condition is not met **do**
  Employed Bees Stage
  Onlookers Stage
  Scout Bees Stage
end while

3 An Artificial Bee Colony Algorithm with Population Growth and Local Search

3.1 The Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm [5,6], which simulates the intelligent foraging behavior of a honeybee swarm, is one of the most recently introduced swarm-based optimization techniques [23]. In a real bee swarm, the employed bees, the onlookers, and the scout bees (or unemployed bees) search for food. The employed bees search the food around the food source, and then they pass their knowledge to the onlookers. The onlookers assess the food sources found by the employed bees to select and search for a new food source in the vicinity of the selected food source. Some of the employed bees, whose food sources are exhausted, are transformed to scouts. These scouts look around for new food sources.

In the original ABC algorithm, the position of a food source represents a solution and its nectar amount corresponds to it fitness value. The employed bees and onlooker bees search around the food sources to find better food locations, and the scouts are assigned to find new food sources if few food sources reach their limits similar as real bee swarms do. The general steps of the algorithm are shown in Algorithm 1.

At the initialization step of the algorithm, ABC generates a number of randomly located food sources, and it creates a number of employed and onlooker bees. The number of food sources, which is denoted by SN, is equal to the number of employed and onlooker bees . Each cycle of the algorithm consists of three successive steps. At the first step, the employed bees search for new, better food sources by modifying the locations of existing food sources according to

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}), i \neq k \tag{1}$$

where $k \in \{1, 2, \ldots, SN\}$, $j \in \{1, 2, \ldots, D\}$ (D is the problem dimension), $\phi_{i,j}$ is a uniform random number between $[-1, 1]$, $x_{i,j}$ and $x_{k,j}$ is the position of the reference food source and the randomly selected food source in dimension j, respectively. The candidate food source created by an employed bee can be better than the reference food source. In this case, the reference food source is replaced with the candidate food source. At the second step, the onlooker try to find new food sources near existing food sources as onlooker bees do. The difference is that each of the onlooker bees randomly selects a food source i with a food source selection probability, p_i, which is determined as:

$$p_i = \frac{\mathit{fitness}_i}{\sum_{n=1}^{SN} \mathit{fitness}_n} \tag{2}$$

Algorithm 2 The Incremental ABC Algorithm with Local Search

```
initialization {Initialize SN number of food sources}
while termination condition is not met do
  if FailedAttempts = Failures_Max then
    Invoke local search on randomly selected food source
  else
    Invoke local search on the best food source
  end if
  Employed Bees Stage {Use x_gbest,j as the reference food source}
  Onlookers Stage
  Scout Bees Stage {Use eq. 5}
  if Food source addition criterion is met then
    Add a new food source to the environment {Use eq. 4}
    SN ← SN + 1
  end if
end while
```

where $fitness_i$ is the fitness value of the food source i, which is inversely proportional to the objective value of the food source i for function minimization.

This behavior of the onlookers leads to explore good food sources in the vicinity of qualified food sources thus increasing the exploitation of the best food sources. At the last step, few food sources, which are not improved during a predetermined number of iterations (controlled by a parameter *limit*), are detected and abandoned. Then, the scout bees search for a new food source randomly according to

$$x_{i,j} = x_j^{min} + rand\,[0,1]\,(x_j^{max} - x_j^{min}) \tag{3}$$

where x_j^{min} and x_j^{max} are the minimum and maximum limits of the search range on dimension j, respectively. Eventually, in the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process [24].

3.2 Integrating Population Growth and Local Search

The outline of the incremental ABC algorithm (IABC) with local search is illustrated in Algorithm 2. In IABC, the number of food sources and indirectly the population size of the bee colony (that is, the number of onlookers and employed bees) is increased with a control parameter g. Every g iterations, a new food source is added to the environment until a maximum number of food sources is reached.

IABC begins with few food sources. New food sources are placed biasing their location towards the location of the best-so-far solution. This is implemented as

$$x'_{new,j} = x_{new,j} + rand\,[0,1]\,(x_{gbest,j} - x_{new,j}), \tag{4}$$

where $x_{new,j}$ is the randomly generated new food source location, $x'_{new,j}$ is the updated location of the new food source, $x_{gbest,j}$ refers to best-so-far food source

location. A similar replacement mechanism is applied by the scout bees step in IABC. The difference is a replacement factor parameter, R_{factor}, that controls how much the new food source locations will be closer to the best-so-far food source. This modified rule is:

$$x'_{new,j} = x_{gbest,j} + R_{factor}(x_{gbest,j} - x_{new,j}) \tag{5}$$

The other difference between the original ABC algorithm and IABC is that employed bees search in the vicinity of $x_{gbest,j}$ instead of a randomly selected food source. This modification boosts the exploitation behavior of the algorithm and helps to converge quickly towards good solutions. Although intensification in the vicinity of the best-so-far solution may lead to early stopping at local optima, the algorithm detects stagnation by using the *limit* parameter, and the scouts can discover another random food source to escape a local optimum.

The IABC algorithm is hybridized with local search by calling a local search procedure in every algorithm iteration. The best-so-far food source location is usually used as the initial solution from which the local search is called. The result of the local search replaces the best-so-far solution if there is an improvement on the initial solution. In this study, the IABC algorithm is hybridized with Powell's conjugate direction set [7] (IABC+Powell) and Lin-Yu Tseng's Mtsls1 [8] (IABC+Mtsls1). Both local search procedures are terminated after a maximum number of iterations, itr_{max}, or when the tolerance $FTol$, which refers to the threshold value of the relative difference between two successive iterations' solution, is reached. An adaptive step size for each local search procedure is used. The step size is set to the maximum norm ($\|\cdot\|_\infty$) of the vector that separates a randomly selected food source from the best-so-far food source.

For fighting stagnation, the local search procedures are applied to a randomly selected location if local search calls cannot improve the results after a maximum number of repeated calls, which is controlled by a parameter $Failures_{max}$. The original versions of the local search algorithms do not check the bound constraints. To enforce boundary constraints, the following penalty function is used in both local search procedures [2, 3]:

$$P(x) = fes \times \sum_{j=1}^{D} Bound(x_j), \tag{6}$$

where $Bound(x_j)$ is defined as

$$Bound(x_j) = \begin{cases} 0, & if\ x_j^{max} > x_j > x_j^{min} \\ (x_j^{min} - x_j)^2, & if\ x_j < x_j^{min} \\ (x_j^{max} - x_j)^2, & if\ x_j > x_j^{max} \end{cases} \tag{7}$$

where fes is the number of function evaluations that have been used so far.

3.3 Experimental Setup

In the experiments, we used a set of 19 scalable functions proposed by Herrera et al. [25] for SOCO. The first 6 functions were originally proposed for the special

Table 1: The best parameter configurations obtained through iterative F-race for ABC algorithms in ten-dimensional instances

Algorithm	SN	$limit$	SN_{max}	$growth$	R_{factor}	itr_{max}	$Failures_{max}$	$FTol$
ABC	5	98	—	—	—	—	—	—
IABC	8	96	13	8	10^{-6}	—	—	—
IABC+Mtsls1	4	13	80	6	$10^{-2.94}$	199	17	—
IABC+Powell	7	99	44	9	$10^{-0.94}$	82	9	$10^{-5.6}$

session on large scale global optimization organized for the CEC 2008 conference, and the latter 5 functions were proposed at the ISDA 2009 conference. The 8 remaining functions were created by combining two functions belonging to the first 11 functions. A detailed description can be found in [25].

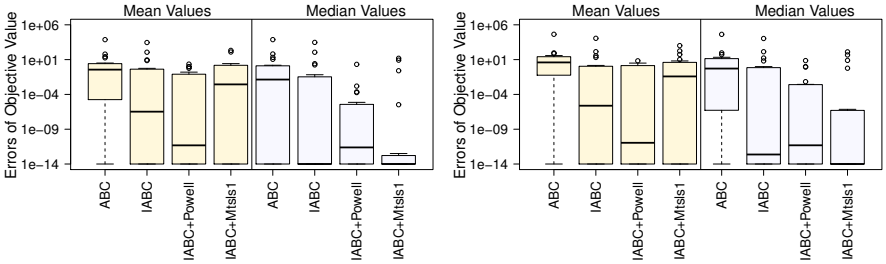
All experiments were conducted under the same conditions, and grouped by the dimensionality of the optimization problems. These problem dimensions are 50, 100, 200, 500, and 1000, respectively. All investigated algorithms were run 25 times for each SOCO function. Each run stops when the maximum number of evaluations is achieved or solution value is lower than a threshold, 10^{-14}, which is approximated to 0. The maximum number of fitness evaluations is $5000 \times D$, where D is the problem dimension.

The parameters of all ABC algorithms used in this paper were tuned automatically by Iterated F-race [26,27]. The best parameters for each algorithm are given in Table 1. Iterated F-race was run using the 10-dimensional versions of the 19 benchmark functions as tuning instances and the maximum budget of Iterated F-race was set to 50,000 algorithm runs each of 50,000 evaluations.

3.4 Results

In our study, we performed three sets of experiments. First, we ran the original ABC algorithm, IABC and the hybridized IABC variants on the 19 SOCO benchmark functions. Then, we compared the computational results of IABC+Powell and IABC+Mtsls1 with $\text{IACO}_{\mathbb{R}}$-LS and IPSOLS. Finally, all three IABC variants were compared with the 16 algorithms featured in SOCO special issue.

The results of the first set of experiments, where we compare the ABC algorithm variants, are indicated in Figure 1 by using box-plots. The box-plots indicate the distribution of average results (left side) and the median results (right side) obtained by each algorithm on the 19 SOCO benchmark functions. (Each point for the boxplot measures the mean and median performance for 25 independent trials on each function.) In all cases except of the comparison between IABC+Mtsls1 and ABC on the 50 dimensional functions, the mean and median results obtained by the proposed algorithms are statistically significantly better than those of the original ABC algorithm as verified by a Wilcoxon test at $\alpha = 0.05$ and using Holm's method for multiple test corrections. There are interesting differences in performance depending on whether we base conclusions on median or mean performance. According to mean errors, IABC+Powell performs better than IABC and IABC+Mtsls1. However, if we consider the median errors (that is, for each function, we measure the median result across the 25



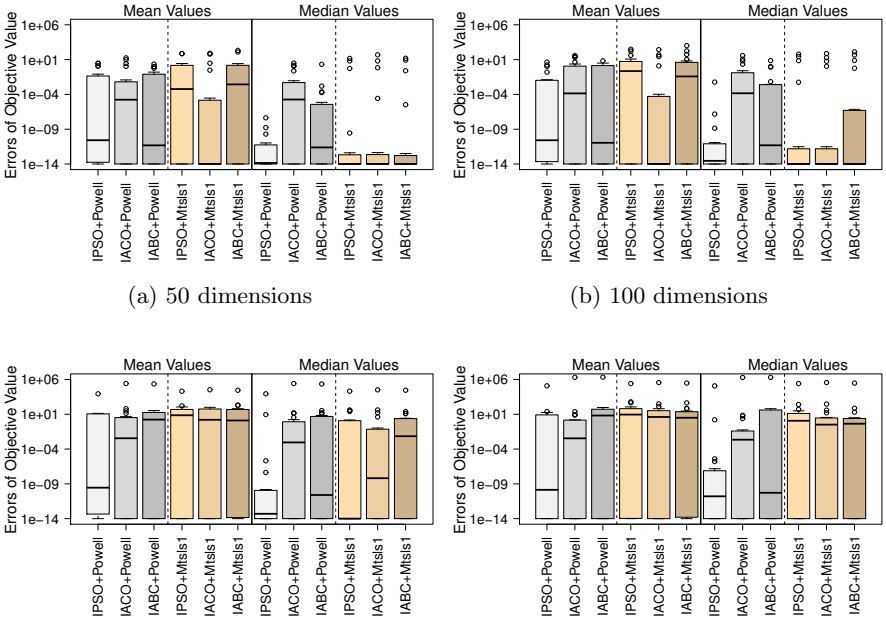
(a) 50 dimensions (b) 100 dimensions

Fig. 1: Distribution of mean and median errors of ABC, IABC, IABC+Powell and IABC+Mtsls1 on the 19 SOCO benchmark functions.

independent trials), IABC+Mtsls1 outperforms all others. This difference is because the IABC+Mtsls1 shows early stagnation effects in few trials in which it obtains rather poor results. This also indicates that a further refinement of the IABC+Mtsls1 algorithm could be interesting for future work. It is also important to note that the IABC (without local search) results are very promising and competitive with IABC+Mtsls1 and IABC+Powell for several functions. This indicates that the improved performance over ABC is not only due to the usage of a local search procedure, but that the incremental social learning mechanism and the, in general, stronger exploitation of the best found solutions are decisive to improve over ABC.

It is interesting to compare the performance of IABC and its variants to that of other algorithms that have been obtained by adopting the incremental social learning framework to improve them. Therefore, we compared IABC-LS with IPSOLS (IPSO+Powell and IPSO+Mtsls1) and $\text{IACO}_{\mathbb{R}}$-LS ($\text{IACO}_{\mathbb{R}}$+Powell and $\text{IACO}_{\mathbb{R}}$+Mtsls1). All experiments were conducted using tuned parameter configurations based on [2,3]. The distribution of the mean and median errors obtained on the 19 SOCO test functions for various dimensions are shown in Figure 2. According to the mean errors, IABC with Powell direction set appears to be competitive with IPSOLS and $\text{IACO}_{\mathbb{R}}$-LS variants especially for the problem dimensions 50 and 100, while it degrades a bit for higher dimensions. Concerning median performance it seems to be slightly preferable over $\text{IACO}_{\mathbb{R}}$ with Powell. This is different when considering Mtsls1 local search, which seems to be more profitably combined with $\text{IACO}_{\mathbb{R}}$. Another interesting observation is that the hybrids with Mtsls1 local search appear to degrade in performance when compared to the hybrids with Powell's direction set local search, indicating a better scaling behavior for the latter.

Finally, we compare all IABC variants with the 16 state-of-the-art algorithms featured in SOCO. To test the significance of the observed differences, we again conducted pairwise Wilcoxon tests with Holm's corrections of multiple comparisons, at the 0.05 α-level. Figure 3 shows these results on the 100 and 1000 dimensional functions. A + symbol on top of a box-plot denotes a signicant



(a) 50 dimensions

(b) 100 dimensions

(c) 500 dimensions

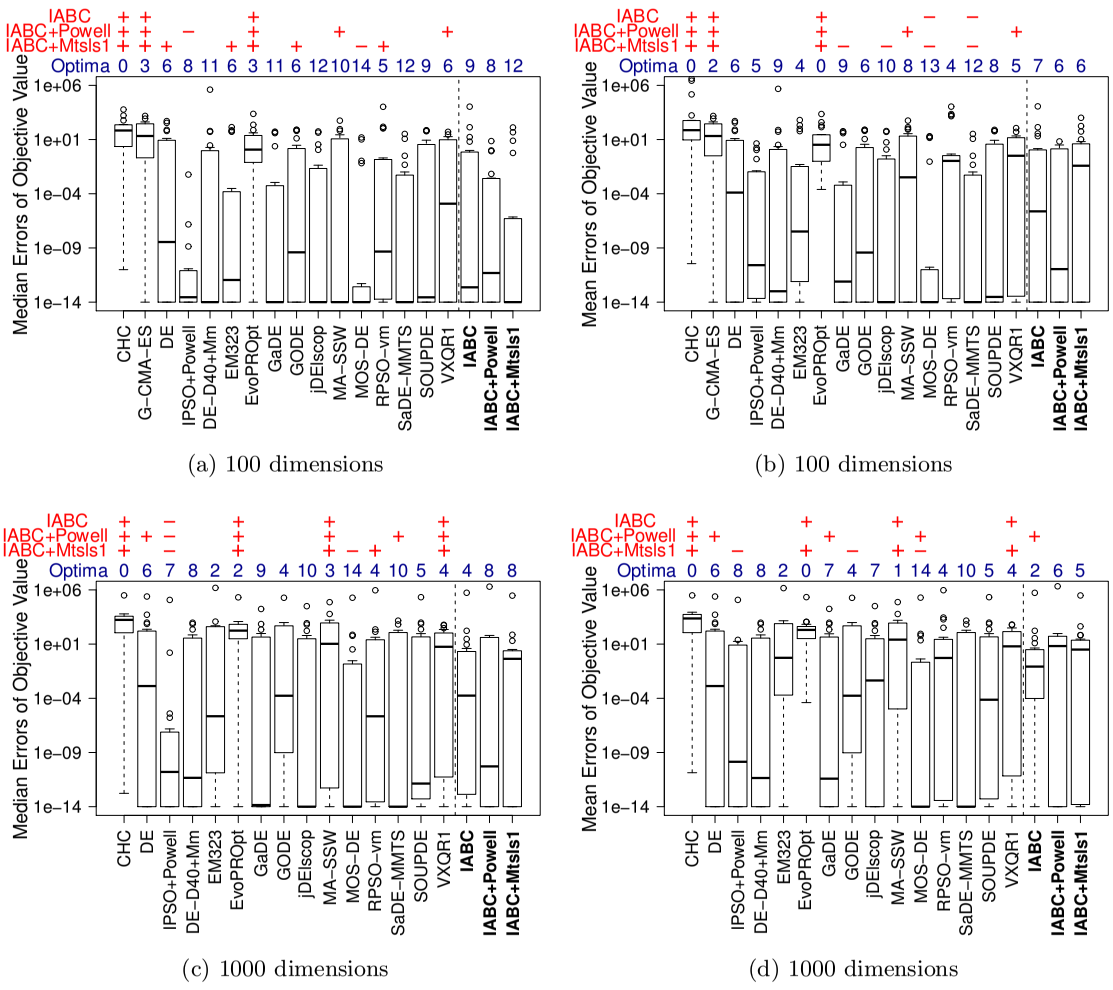
(d) 1000 dimensions

Fig. 2: Distribution of the mean and the median errors obtained on the 19 benchmark functions

difference at the 5% level between the results obtained with the indicated algorithm and those obtained with the indicated IABC variant. If an algorithm is significantly better than an IABC variant, a $-$ symbol is put on top of a box-plot for indicating this difference. The numbers on top of a box-plot denotes the number of optima found by the corresponding algorithm (in other words, the number of functions on which the statistic, either mean or median, is smaller then the zero threshold 10^{-14}. Figure 3 indicates that IABC, IABC+Powell and IABC+Mtsls1 significantly outperform CHC and G-CMA-ES in each case. This is noteworthy since, in particular, G-CMA-ES is an acknowledged state-of-the-art algorithm for continuous optimization. When taking into account the median errors of the algorithms, IABC variants outperform CHC, G-CMA-ES, EvoPROpt, MA-SSW, RPSO-vm, and VXQR1 in almost all dimensions. Except IPSO+Powell and MOS-DE, the algorithms exhibit performance similar to rest of the state-of-the-art algorithms.

4 Discussion

In this paper, we have shown that the incremental social learning framework with local search can enhance the performance ABC; in earlier work we have



(a) 100 dimensions

(b) 100 dimensions

(c) 1000 dimensions

(d) 1000 dimensions

Fig. 3: Distribution of the average and the median errors obtained on the 19 benchmark functions for all featured algorithms in SOCO and IABC variants on 100- and 1000-dimensional functions.G-CMA-ES results on 1000-dimensional functions are unknown.

shown the same result for two other population-based algorithms, a PSO algorithm and an ACO algorithm. In population-based optimization algorithms, individual agents can learn from each other. However, this sharing knowledge mechanism can be slow as in the ABC algorithm. In incremental social learning, new agents can obtain knowledge directly from experienced ones. Thus, incremental social learning allows the new agent to more directly focus the search in the vicinity of the best results. On the other hand, although local search procedures cannot always find good-enough solutions alone, local search procedures play an important role in population-based algorithms. At the same time, behavior of a population-based algorithm affects directly the behavior of the local search procedure. For example, information from the population may be used to select restart positions and adaptively determine the step size of the local search procedure. In our case, the step size is determined by the position of the best-so-far agent and two agent positions, respectively.

Concerning the impact of the incremental social learning framework on algorithm performance, we observed that IABC improved strongly upon the original ABC algorithm, more than what we observed when moving to the incremental ACO and PSO algorithms. In fact, the IABC algorithm alone was sufficient to find optimal solutions for a number of benchmark functions. Possible reasons for this strong improvements may be that (i) on one side the bee colony metaphor is a less explored than other metaphors and that further improvements are therefore easier; and (ii) on the other side that ABC algorithms have a better local search type behavior for the refinement of solutions than PSO and ACO algorithms. A more in depth analysis would be an interesting direction for future research. Besides the incremental social learning framework, we note that another possible reason for the observed performance improvements is the stronger focus around the best-so-far solutions, in which our IACO algorithm is similar to earlier proposals [23].

The performance of the hybrid method, that is, the combination of IABC with local search (but also that of IACO and IPSO with local search) depends significantly on the local search algorithm chosen. In fact, we observed that for "low" dimensions of 50 and 100, Mtsls1 seems to work well with IABC and IACO, while Powell's direction set method seems to be less affected by increasing dimensionality. Since the best local search may further depend on the particular benchmark function, we think that an adaptive choice of the local search algorithm to be used would be desirable. As example, we have tried to a simple strategy in which the algorithm selects the better local search procedure after the first iteration of the algorithm, resulting in further improvements of the IABC results.

5 Conclusions

In this paper, we have introduced an ABC algorithm with a growing population size and we hybridized it with a local search procedure for tackling large-scale benchmark functions. The increasing population size and a stronger focus on the search space regions around the best-so-far solution have contributed to make the proposed IABC algorithm much more performing than the original ABC algorithm. For the further hybridization with local search two different local

search procedures were used, using either Powell direction set or Mtsls1. The parameters of IACO and the hybrid IACO with local search were further tuned using the automatic parameter tuning tool Iterated F-race. For the benchmark functions of 50 and 100 dimensions we found that the hybrid algorithm typically still further improves over IACO either with respect to the mean results (when hybridized with Powell direction set) or the median results (when hybridized with Mtsls1). When compared to other algorithms for large scale continuous optimization from the SOCO special issue, the hybrid IABC algorithm was found to reach high performance; it was outperformed, according to the median results, only by an incremental PSO algorithm and the overall best performing from the SOCO benchmark competition.

It is maybe more noteworthy that also in the ABC case, extending this type of algorithm with the incremental social learning framework led to significant performance improvements. Certainly, further analysis of the hybrid IABC algorithm is necessary to determine the contribution of the specific algorithm components. Nevertheless, the that fact that apart from ABC we also could improve PSO and continuous ACO algorithms by embedding them into the incremental social learning framework, gives strong evidence that an increasing population size and the hybridization with local search algorithms are important components to obtain high performing algorithms.

References

1. Montes de Oca, M., Stützle, T., Van den Enden, K., Dorigo, M.: Incremental social learning in particle swarms. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **41**(2) (2011) 368–384
2. Montes de Oca, M.A., Aydın, D., Stützle, T.: An incremental particle swarm for large-scale optimization problems: An example of tuning-in-the-loop (re)design of optimization algorithms. Soft Computing (2011) In press.
3. Liao, T., Montes de Oca., M.A., Aydın, D., Stützle, T., Dorigo, M.: An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization. In: GECCO 2011, New York, ACM Press (2011) accepted.
4. Lozano, M., Molina, D., Herrera, F.: Editorial: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Computing (2011) In Press.
5. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes Universitesi, Engineering Faculty, Computer Engineering Department (2005)
6. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39(3), 459–471 (2007)
7. Powell, M.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal **7**(2) (1964) 155
8. Tseng, L., Chen, C.: Multiple trajectory search for large scale global optimization. In: CEC 2008, Piscataway, NJ, IEEE Press (2008) 3052–3059
9. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4) (1997) 341–359
10. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of IEEE Congress on Evolutionary Computation, CEC 2005, Piscataway, NJ, IEEE Press (2005) 1769–1776

11. Eshelman, L., Schaffer, J.: Real-coded genetic algorithms and interval-schemata. Foundations of Genetic Algorithms **2**(1993) (1993) 187–202
12. Powell, M.: The BOBYQA algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06, University of Cambridge, UK (2009)
13. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University (2005)
14. Eiben, A., Marchiori, E., Valk, V.: Evolutionary algorithms with on-the-fly population size adjustment. In Yao, X. et al., eds.: PPSN VIII. Vol. 3242 of LNCS. Springer, Heidelberg (2004) 41–50
15. Fernandes, C., Rosa, A.: Self-regulated population size in evolutionary algorithms. In Runarsson, T. et al., eds.: PPSN IX. Vol. 4193 of LNCS. Springer, Heidelberg (2006) 920–929
16. Chen, D., Zhao, C.: Particle swarm optimization with adaptive population size and its application. Applied Soft Computing **9**(1) (2009) 39–48
17. Hsieh, S., Sun, T., Liu, C., Tsai, S.: Efficient population utilization strategy for particle swarm optimizer. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **39**(2) (2009) 444–456
18. Chen, J., Qin, Z., Liu, Y., Lu, J.: Particle swarm optimization with local search. In: ICNN&B'05. Volume 1., IEEE Press, Piscataway, NJ (2005) 481–484
19. Liang, J., Suganthan, P.: Dynamic multi-swarm particle swarm optimizer with local search. In: CEC 2005. Volume 1., IEEE Press, Piscataway, NJ (2005) 522–528
20. Gimmler, J., Sttzle, T., Exner, T.: Hybrid particle swarm optimization: An examination of the influence of iterative improvement algorithms on performance. In Dorigo, M. et al., eds.: ANTS 2006. Vol. 4150 of LNCS. Springer Heidelberg (2006) 436–443
21. Das, S., Koduru, P., Gui, M., Cochran, M., Wareing, A., Welch, S., Babin, B.: Adding local search to particle swarm optimization. In: CEC 2006., IEEE Press, Piscataway, NJ (2006) 428–433
22. Petalas, Y., Parsopoulos, K., Vrahatis, M.: Memetic particle swarm optimization. Annals of Operations Research **156** 99–127
23. Zhu, G., Kwong, S.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation 217(7), 3166 – 3173 (2010)
24. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation 214(1), 108 – 132 (2009)
25. F. Herrara, M.L., Molina, D.: Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems (2010), http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf
26. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In: HM 2007. pp. 108–122. Springer, Heidelberg (2007)
27. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. Experimental Methods for the Analysis of Optimization Algorithms pp. 311–336 (2010)
28. Liao, T., Montes de Oca., M.A., Stützle: Tuning Parameters across Mixed Dimensional Instances: A Performance Scalability Study of Sep-G-CMA-ES. In: Proceedings of the Workshop on Scaling Behaviours of Landscapes, Parameters and Algorithms of GECCO 2011, New York, ACM Press (2011) accepted.