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[3], and Thomas Stützle[2]

[1] Dept. of Computer Engineering, Dumlupınar University, 43030 Kütahya, Turkey
dogan.aydin@dpu.edu.tr
[2] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{tliao,stuetzle}@ulb.ac.be
[3] Dept. of Mathematical Sciences, University of Delaware, Newark, DE, USA
mmontes@math.udel.edu

Abstract. We modify an artificial bee colony algorithm as follows: we make the population size grow over time and apply local search on strategically selected solutions. The modified algorithm obtains very good results on a set of large-scale continuous optimization benchmark problems. This is not the first time we see that the two aforementioned modifications make an initially non-competitive algorithm obtain state-of-the-art results. In previous work, we have shown that the same modifications substantially improve the performance of particle swarm optimization and ant colony optimization algorithms. Altogether, these results suggest that population growth coupled with local search help obtain high-quality results.

1 Introduction

Thinking of optimization algorithms as being composed of components has changed the way high-performance optimization algorithms are designed. Research based on algorithm components and not on specific implementations of optimization algorithms promises to lead to breakthroughs because a more systematic approach can be taken in order to explore the space of algorithm designs. In previous work, we have taken a component-based approach to the design of optimization algorithms for continuous optimization and have obtained promising results. In particular, we integrated a growing population size and a local search components into a particle swarm optimization (PSO) algorithm and an ant colony optimization algorithm for continuous domains. The resulting algorithms, called IPSOLS [3,4] in the case of PSO, and IACO$_{\mathbb{R}}$-LS [5] in the case of ACO$_{\mathbb{R}}$, exhibited a significantly better performance than the original algorithms. In fact, their performance was competitive with state-of-the-art algorithms in the special issue of the Soft Computing journal on large-scale continuous optimization [6]. (Throughout the rest of the paper, we will refer to this special issue as SOCO.) In this paper, we present a third case study based on the artificial

J.-K. Hao et al. (Eds.): EA 2011, LNCS 7401, pp. 85–96, 2012.

bee colony (ABC) algorithm [7, 8]. The goal of this case study is to test the "pure chance" hypothesis, which explains the results obtained with PSO and $ACO_{\mathbb{R}}$ as consequence of mere good luck. The results obtained in this third case study, shown in Section 3.4, suggest that the pure-chance hypothesis is false. We find that population growth together with a local search procedure are algorithmic components that make swarm intelligence algorithms for continuous optimization obtain high-quality results.

2 Related Work

The idea of increasing the size of the population in swarm intelligence algorithms for continuous optimization derives from the incremental social learning (ISL) framework [3]. The ISL framework's aim is to reduce the time needed by a swarm intelligence system to reach a desired state. In the case of swarm intelligence systems for optimization, the desired state may be associated with a solution better than or equal to a desired quality. The ISL framework achieves this goal by starting the system with a small population and increasing its size according to some addition criterion. By starting a swarm-based optimization algorithm with a smaller population than usual, the framework encourages a rapid convergence toward promising regions of the search space. Adding new solutions increases the diversity of the swarm. The new solutions are generated using information from the "experienced" swarm; thus, new solutions are not completely random.

Modifying the size of the population while an algorithm is operating is not a new idea. For example, a number of researchers in the field of evolutionary computation (EC) have proposed schemes that increase or reduce the size of the population in order to adjust the diversification-intensification properties of the underlying optimization algorithms (see e.g. [9–11]). The ISL framework is different from most previous dynamic population sizing approaches. In most of these cases, diversification is favored during the first phases of the optimization process through a large population, and intensification toward the end by reducing its size. In contrast, the ISL framework modifies the population size in one direction only: increasing its size. A notable exception of the strategy used in most EC algorithms is the one used in the G-CMA-ES [12] algorithm. In this algorithm, the population size also increases over time. It is important to note that G-CMA-ES is considered to be a state-of-the-art algorithm for continuous optimization.

The utilization of an auxiliary local search method is a common approach to enhance the intensification properties of EC algorithms. In ISL, local search can be integrated as a means to simulate individual learning, that is, learning without any social influence.

In [3], we introduced IPSOLS, a PSO-local search hybrid algorithm with increasing population size as an instantiation of the ISL framework. In subsequent work [4], we used iterated F-Race [14], an automatic algorithm configuration software, throughout the redesign process of IPSOLS. The redesigned IPSOLS algorithm [4] was benchmarked on SOCO's large-scale benchmark 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 [6]. In SOCO, a differential evolution algorithm (DE) [15], G-CMA-ES, and the real coded CHC algorithm (CHC) [16] were used as reference algorithms. A second instantiation of ISL was presented in [5] in the context of $ACO_{\mathbb{R}}$. The introduced algorithm, called $IACO_{\mathbb{R}}$-LS, features the same two components: a growing solution population size and a local search procedure. $IACO_{\mathbb{R}}$-LS was also benchmarked on SOCO's benchmark problems and on the IEEE CEC 2005 functions [17]. We can say that $IACO_{\mathbb{R}}$-LS is a state-of-the-art algorithm because it obtained better results than IPSOLS and it is competitive with G-CMA-ES.

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

In this section, we present our third case study of the utilization of a growing population size and a local search procedure in a swarm intelligence algorithm for continuous optimization.

3.1 The Artificial Bee Colony Algorithm

The ABC algorithm [7, 8] is inspired by the foraging behavior of a honeybee swarm. 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. In the first step, each employed bee selects successively a food source i and then produces a candidate food, v_i, by mutating the location of the selected food source as reference 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's dimension), $\phi_{i,j}$ is a uniform random number in $[-1, 1]$, $x_{i,j}$ and $x_{k,j}$ is the position of the reference food source i and a randomly selected food source k in dimension j. The candidate food source created by an employed bee can be better than the reference food source. In this case, the reference food source, x_i, is replaced with the candidate food source, v_i. In the second step, onlooker bees try to find new food sources near existing food sources as the employed bees do. Different from employed bees, onlooker bees randomly select a food source i with a food source selection probability p_i, which is determined as

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

where $\mathit{fitness}_i$ 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.

Algorithm 1. The Incremental ABC Algorithm with Local Search

```
initialization {Initialize SN 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
```

Onlooker bees explore in the vicinity of good food sources. This behavior is responsible for the intensification behavior of the algorithm since information about good solutions is exploited. In the last step, a few food sources, which have not been improved during a predetermined number of iterations (controlled by a parameter *limit*), are detected and abandoned. Then, scout bees search randomly for a new food source, x_i, and replaces with it the location of the abandoned food source, x_i^{old}. The new food source is found according to

$$x_{i,j} = x_j^{min} + \varphi_{i,j}(x_j^{max} - x_j^{min}) \tag{3}$$

where $\varphi_{i,j}$ is a uniform random number in $[0, 1]$ for dimension j and food source i, and x_j^{min} and x_j^{max} are the minimum and maximum limits of the search range on dimension j, respectively. Clearly, in the ABC algorithm employed and onlooker bees intensify the algorithm's search and the scout bees diversify it [18].

3.2 Integrating Population Growth and Local Search

In this section, we describe the integration of a growing population size and a local search procedure into the ABC algorithm. The outline of the proposed algorithm, called IABC-LS, is shown in Algorithm 1. In IABC-LS, the number of food sources and, indirectly, the population size of the bee colony (that is, the number of onlooker and employed bees), is increased according to 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-LS begins with few food sources. New food sources are placed biasing their location toward the location of the best-so-far solution. This is implemented as

$$x'_{new,j} = x_{new,j} + \varphi_{new,j}(x_{gbest,j} - x_{new,j}), \tag{4}$$

where $x_{new,j}$ is the randomly generated new food source location in dimension j, $x'_{new,j}$ is the updated location of the new food source, $x_{gbest,j}$ refers to best-so-far food source location, and $\varphi_{new,j}$ is a number chosen uniformly at random in $[0, 1]$. A similar replacement mechanism is applied by the scout bees step in IABC-LS. The difference is a replacement factor, 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:

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

Another difference between the original ABC algorithm and IABC-LS is that employed bees search in the vicinity of $x_{gbest,j}$ instead of a randomly selected food source. This modification enhances the intensification behavior of the algorithm and helps it to converge quickly toward good solutions. Although intensification in the vicinity of the best-so-far solution may lead to premature convergence, the algorithm detects stagnation by using the *limit* parameter, and the scouts can discover another random food source to avoid search stagnation.

IABC-LS is a hybrid algorithm that calls a local search procedure at each 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 paper, the IABC-LS algorithm is implemented with Powell's conjugate direction set [19] (IABC-Powell) and Lin-Yu Tseng's Mtsls1 [20] (IABC-Mtsls1) as local search procedures. Both local search procedures are terminated after a maximum number of iterations, itr_{max}, or when the tolerance $FTol$ is reached. $FTol$ is the threshold value of the relative difference between two successive iterations' solutions. 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 enforce bound constraints. To enforce bound constraints, the following penalty function is used in both local search procedures [4, 5]:

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

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.

Table 1. The best parameter configurations obtained through iterated F-race for ABC algorithms in 10-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	$10^{-5.6}$

3.3 Experimental Setup

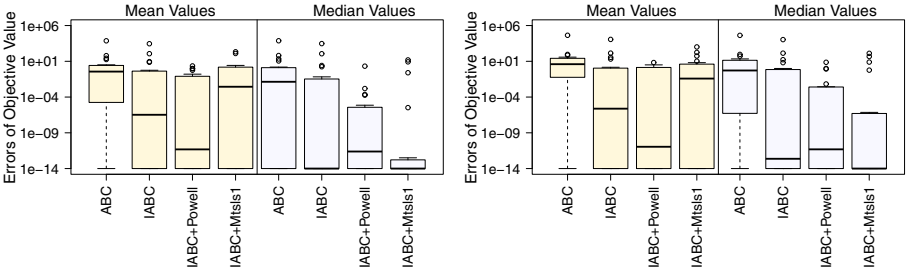
In our study, we performed three sets of experiments. First, we ran the original ABC algorithm, IABC (IABC-LS without local search) and the two instantiations of IABC-LS on the 19 SOCO benchmark functions proposed by Herrera et al. [21] for SOCO. A detailed description of the benchmark function set can be found in [21]. All experiments were conducted under the same conditions, and grouped by the dimensionality of the optimization problems ($D \in \{50, 100, 200, 500, 1000\}$). All investigated algorithms were run 25 times for each SOCO function. Each run stops when the maximum number of evaluations is achieved or the solution value is lower than 10^{-14}, which is approximated to 0. The maximum number of evaluations is $5000 \times D$.

The parameters of all ABC algorithms used in this paper were tuned using iterated F-race [14]. The best parameters found 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 tuning budget was set to 50,000 algorithm runs. The number of function evaluations used in each run is 50,000.

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 the SOCO special issue.

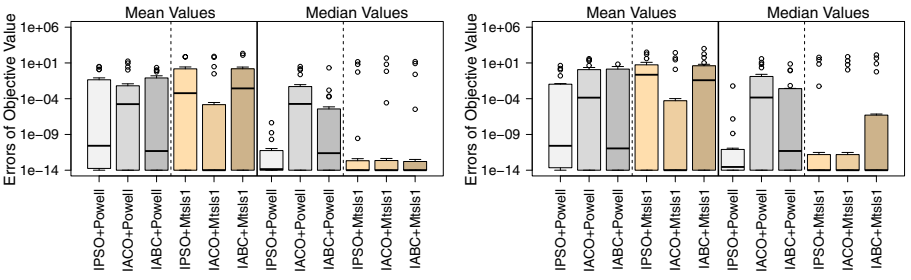
3.4 Results

The results of the first set of experiments, where we compare the ABC algorithm variants, are shown in Figure 1. 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 box-plot measures the mean and median performance for 25 independent trials on each function.) In all cases, except in 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. Statistical significance was determined with Wilcoxon's test at $\alpha = 0.05$ using Holm's method for multiple test corrections. There are interesting differences in performance depending on whether we base our conclusions

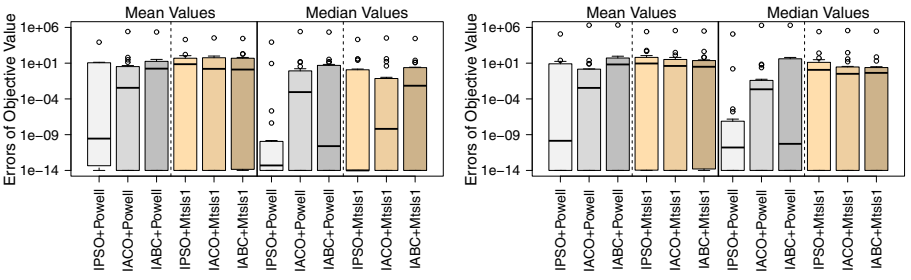


(a) 50 dimensions (b) 100 dimensions

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



(a) 50 dimensions (b) 100 dimensions



(c) 500 dimensions (d) 1000 dimensions

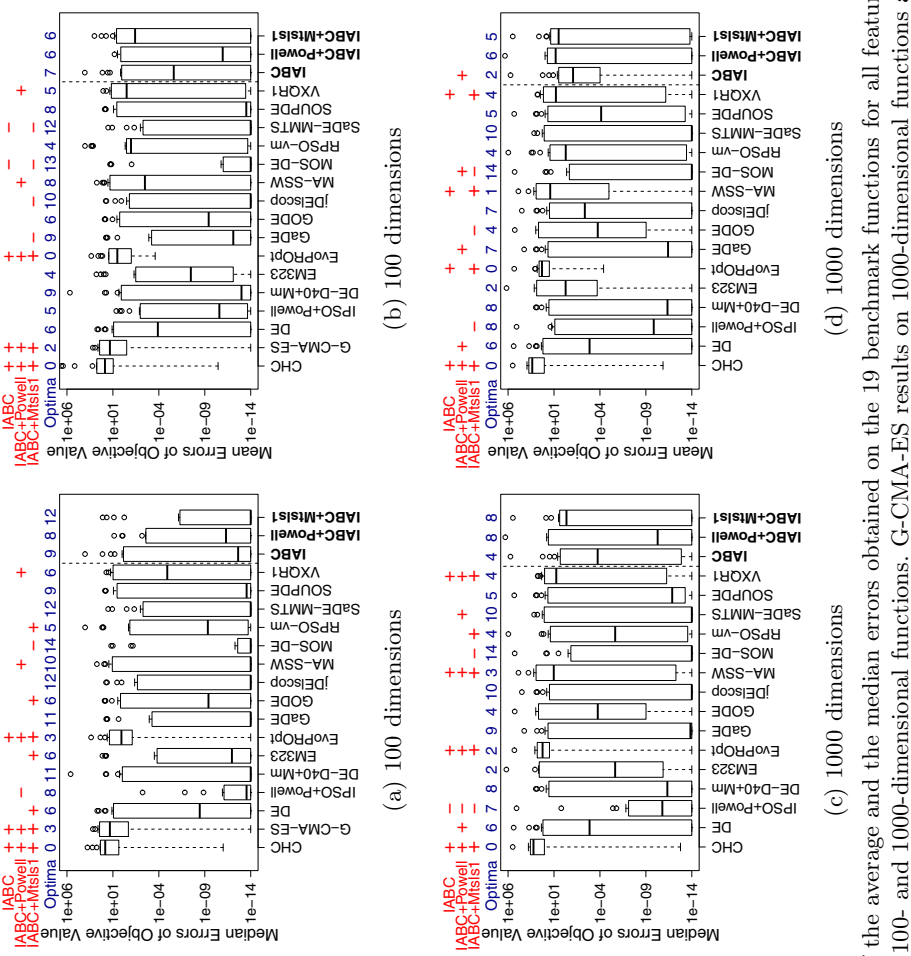
Fig. 2. Distribution of the mean and the median errors obtained on the 19 benchmark functions

on the median or the mean performance. Based on mean errors, IABC-Powell performs better than IABC and IABC-Mtsls1. However, based on median errors (that is, for each function, we measure the median result across the 25 independent trials), IABC-Mtsls1 outperforms all others. This difference is the result of the fact that IABC-Mtsls1 shows early stagnation effects in few trials in which

it obtains rather poor results. This difference also indicates that a further refinement of IABC-Mtsls1 could be interesting for future work. Notice that the results obtained with IABC 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 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 with that of other algorithms that have been obtained by adopting the ISL 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 [4, 5]. The distributions of the mean and median errors obtained on the 19 SOCO test functions for various dimensions are shown in Figure 2. Based on the mean errors, IABC-Powell appears to be competitive with IPSOLS and $\text{IACO}_{\mathbb{R}}$-LS variants, especially when $D = 50$ and $D = 100$. It is also apparent that the performance of IABC-Powell degrades for higher dimensions. Based on median performance, IABC-Powell seems to be slightly better than $\text{IACO}_{\mathbb{R}}$-Powell. When using Mtsls1 as local search procedure, the results are different. In this case, Mtsls1 seems to work better with $\text{IACO}_{\mathbb{R}}$. Another interesting observation is that the hybrids with Mtsls1 appear to degrade in performance when compared to the hybrids with Powell's direction set method, indicating a better scaling behavior for the latter.

Finally, we compare all IABC variants with the 16 algorithms featured in SOCO. To test the significance of the observed differences, we again conducted pairwise Wilcoxon tests with Holm's corrections for multiple comparisons, at the 0.05 significance level. Figure 3 shows these results on the 100- and 1000-dimensional functions. A "+" symbol on top of a box-plot denotes a significant difference at the 0.05 level between the results obtained with the indicated algorithm and those 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 denote 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 than 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 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. The IABC variants exhibit a performance similar to rest of the state-of-the-art algorithms with the exception of IPSO-Powell and MOS-DE.



(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.

4 Discussion

In this paper, we have shown that the ISL framework with local search can enhance the performance of ABC. In earlier work we have shown the same result for two other swarm intelligence algorithms: PSO and $ACO_{\mathbb{R}}$. In swarm intelligence algorithms for optimization, individual agents can learn from each other. However, the specific knowledge-sharing mechanism used in any given algorithm can be slow. In the ISL framework, new agents can obtain knowledge directly from experienced ones. Thus, ISL 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, the 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 another agent's position.

Concerning the impact of the ISL 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 $ACO_{\mathbb{R}}$ 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) the bee colony metaphor has not been as explored as the metaphors behind PSO or $ACO_{\mathbb{R}}$ have; thus, improvements are easier to obtain; and (ii) ABC algorithms have a better local search type behavior for the refinement of solutions than PSO and $ACO_{\mathbb{R}}$ algorithms. A more in-depth analysis of the ABC algorithm could be an interesting direction for future research. In addition to the usage of ISL, we note that another possible reason for the observed performance improvements is the stronger focus around the best-so-far solutions, which is a feature also present in other ABC variants [22].

The performance of the hybrid method, that is, the combination of IABC with local search (but also that of $IACO_{\mathbb{R}}$ 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 $ACO_{\mathbb{R}}$, 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, an adaptive choice of the local search algorithm to be used would be desirable. As example, we have tried 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 best-so-far solution have contributed to make the proposed IABC algorithm much better performing than the original ABC algorithm. For the hybridization with local search two different local search procedures were used, Powell's conjugate direction set and Mtsls1. The parameters of IABC and the hybrid IABC with local search were tuned using iterated F-Race, an automatic algorithm configuration tool. For the benchmark functions of 50 and 100 dimensions we found that the hybrid algorithm typically improves over IABC. When compared with other algorithms for large scale continuous optimization from the SOCO special issue, the hybrid IABC algorithm was found to reach high-quality solutions; it was outperformed, according to the median results, only by an incremental PSO algorithm and the overall best performing algorithm from the SOCO benchmark competition.

It is maybe more noteworthy that also in the ABC case, extending this 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 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 to obtain high performing swarm intelligence algorithms for continuous optimization.

Acknowledgments. The research leading to the results presented in this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n°246939, and by the Meta-X project, funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a Research Associate.

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE International Conference of Neural Networks, vol. 4, pp. 1942–1948 (1995)
2. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185, 1155–1173 (2008)
3. Montes de Oca, M.A., 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), 368–384 (2011)
4. 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 15(11), 2233–2255 (2011)
5. 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, pp. 125–132. ACM Press, New York (2011)

6. Lozano, M., Molina, D., Herrera, F.: Editorial: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Computing 15(11), 2085–2087 (2011)
7. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes Universitesi, Computer Engineering Department (2005)
8. 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)
9. Eiben, A.E., Marchiori, E., Valkó, V.A.: Evolutionary Algorithms with On-the-Fly Population Size Adjustment. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN VIII. LNCS, vol. 3242, pp. 41–50. Springer, Heidelberg (2004)
10. Chen, D., Zhao, C.: Particle swarm optimization with adaptive population size and its application. Applied Soft Computing 9(1), 39–48 (2009)
11. 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), 444–456 (2009)
12. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: CEC 2005, pp. 1769–1776. IEEE Press (2005)
13. Balaprakash, P., Birattari, M., Stützle, T.: Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HM 2007. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
14. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. In: Experimental Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer, Heidelberg (2010)
15. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
16. Eshelman, L., Schaffer, J.: Real-coded genetic algorithms and interval-schemata. Foundations of Genetic Algorithms 2, 187–202 (1993)
17. 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)
18. Karaboga, D., Akay, B.: A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation 214(1), 108–132 (2009)
19. Powell, M.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal 7(2), 155–162 (1964)
20. Tseng, L., Chen, C.: Multiple trajectory search for large scale global optimization. In: CEC 2008, pp. 3052–3059. IEEE Press, Piscataway (2008)
21. Herrara, F., Lozano, M., 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`
22. Diwold, K., Aderhold, A., Scheidler, A., Middendorf, M.: Performance evaluation of artificial bee colony optimization and new selection schemes. Memetic Computing 3(3), 149–162 (2011)