

Empirical assessment of the effects of update synchronization in Particle Swarm Optimization

Luca Musci¹, Stefano Cagnoni¹, and Fabio Daolio²

¹ University of Parma, Department of Information Engineering
musci, cagnoni@ce.unipr.it

² University of Lausanne, Information Systems Institute **
fabio.daolio@unil.ch

Abstract. Despite considerable popularity, the mechanisms that govern the behavior of Particle Swarm Optimization (PSO) are still a subject of research. Regarding communication between particles, for example, many authors have discussed the effects of swarm topology, but few have studied the dynamics of the information exchange among particles. In this paper we show that a synchronous update of the social attractors, which is necessary when parallel versions of PSO are implemented, may influence the effectiveness of the algorithm. To do so we compare the synchronous and asynchronous variants of PSO on a standard benchmark. The results show that the ‘global best’ topology is sensitive to the policy update, especially in the presence of high-dimensional search spaces. In contrast, sparsely-connected topologies seem to be much less sensitive to synchronization.

1 Introduction

Particle Swarm Optimization (PSO) was first introduced by Kennedy and Eberhart in 1995 [1]. PSO searches the optima of a function (generally termed fitness for analogy with evolutionary algorithms) by mimicking the movements of flocks of birds in search of food. The particles ‘fly’ over the domain of the fitness function of which, in each step, the values associated with new positions of each particle are computed. In the most basic version of PSO, each time its state is updated, the particle keeps part of its speed (inertia), being also subject to two forces of attraction: cognitive attraction draws it towards the best position it has visited so far, while social attraction draws it towards the best position found so far by the whole swarm. A particle’s motion is therefore represented by the following equations:

$$\begin{aligned} V_i(t) &= wV_i(t-1) \\ &\quad + C_1R_1[X_i^{pb}(t-1) - X_i(t-1)] \\ &\quad + C_2R_2[X_i^{gb}(t-1) - X_i(t-1)] \end{aligned} \tag{1}$$

$$X_i(t) = wX_i(t-1) + V_i(t) \tag{2}$$

** formerly with University of Parma

where i refers to the i -th dimension of the search space, V is the particle's velocity, C_1 and C_2 are positive constants, w is the coefficient of inertia, $X(t)$ the position, X^{pb} the position with better fitness visited so far by the particle, X^{gb} ('global best') the best-fitness point found by the entire swarm so far, R_1 and R_2 are random numbers drawn from a uniform distribution in $[0, 1]$. Most of the many possible variants [2] are based on a modified topology of the swarm [3, 4]. The most typical one replaces X^{gb} with X^{lb} ('local best'), which is defined as the best location found so far within a predefined neighborhood of the particle. Further variants are based on different topologies of such a neighborhood. They have been compared in [4], where random topologies and Von Neumann neighborhoods are reported to yield good performances; however, in the same paper, authors also show that the most efficient type of neighborhood is, in general, dependent on the problem. The two algorithms termed 'Standard PSO' are detailed in [5, 6]: they use a fixed topology and a stochastic star ring topology, respectively. In the latter, each particle 'informs' itself and $K - 1$ neighbors chosen at random, while the neighborhoods of each particle are initialized randomly and are updated after each iteration in which the global best does not improve. Since the suggested value for K is 3, the behavior and performance of this topology are very similar to those of a fixed ring topology with two neighbors per particle. The reference version we used for our tests is the one described in [6].

Another issue which affects performance is the strategy by which X^{gb}/X^{lb} are updated. The 'synchronous' version of PSO updates the positions of all particles in turn in each iteration, usually called, inappropriately, a 'generation' for analogy with evolutionary computation. The fitness values associated with the new positions are evaluated as well. X^{gb}/X^{lb} is updated only after the end of the current generation. Listing 1a details this strategy by means of pseudo-code. The 'asynchronous' version, instead, updates X^{gb}/X^{lb} immediately after assessing the fitness of each particle, such that the swarm is more readily attracted by the new optimum (see Listing 1b). If the asyn-

<pre> <initialize particles' positions/velocities> <initial fitness evaluation> <initialize personal/global bests> for(g=0; g < genNumber; g++){ for(p=0; p < partNumber; p++){ <update the position of particle p> <re-evaluate the fitness of particle p> <update the personal best of particle p> <update the global best> } } <output the global best position> </pre> <p style="text-align: center;">(a) Asynchronous PSO</p>	<pre> <initialize particles' positions/velocities> <initial fitnesses evaluation> <initialize personal/global bests> for(g=0; g < genNumber; g++){ <update the position of all particles> <re-evaluate all fitness values> <update all personal/global bests> } <output the global best position> </pre> <p style="text-align: center;">(b) Synchronous PSO</p>
--	---

Fig. 1: Pseudo-code listings for the (a) Asynchronous and (b) Synchronous PSO.

chronous version is implemented on a distributed architecture, this sequential temporal structure is lost and the update equations can be applied to any particle at any time, without following any specific order. Regarding the effects that such a change might have, results reported in [7] are particularly interesting: in such a paper, a genetic algorithm evolves order and update rate of the particles within a swarm, often achieving better performance than standard PSO. However, the authors also notice that there are many features and algorithm parameters, such as particle quality, update rate, swarm size, etc., that affect performance. Focusing on synchronization and on convergence, we measured the frequency with which absolute optima could be found on benchmarks as, for example, the one described in [8]. Indeed, [3] shows that, in general, the synchronous version requires a greater number of fitness evaluations to reach convergence than the asynchronous version. However, the effect of synchronization on the ability to converge to a good solution is not discussed in that paper.

We have empirically evaluated the performance of the two policies, taking also into account the particles' topology, by applying them to two different sequential implementations of the PSO which are based on the 'global best' and 'local best' topologies.

2 Results

The tests were performed using the Standard PSO 2006 (SPSO 2006) [6] to optimize some of the functions included in the 2009 Black Box Optimization Benchmark [8], for which the functions are described in [8, 9] and the evaluation procedure in [10]. Since we are only interested in how the performance of the algorithm is affected by the update strategy, we limited our tests to separable functions, on which the PSO has generally the best performance [11], to make results more sensitive to performance changes and as much independent of problem hardness as possible. We evaluated both the 'local best' and, modifying the original code, the 'global best' topology setting the algorithm parameters to the values suggested in [6].

The results (see Figure 2 and Figure 3) confirm the hypotheses about the dependence of convergence speed on the update strategy. The graphs show the Expected Running Time (ERT) of the four variants of PSO we took into consideration on the functions Sphere and Ellipsoid (unimodal) and Rastrigin (multimodal) versus problem size. When the size of the search space is small, there is only a slight difference in the number of evaluations after which the optimum found by PSO falls below the required tolerance. With 'global best' topology, however, as problem size increases, the synchronous strategy becomes not only less efficient but also less effective: fewer executions (numbers above the symbol ●) are successful and more trials (×) never reach the goal; performance is similar for successful trials (symbols +). The point at which the ERT departs from a linear trend is clearly visible.

This is most evident in the plots of the distribution of running times (see Figure 4 and Figure 5; for more details on the graphs and the evaluation procedure from which they are generated see [10]) that summarize the performance of the four algorithms on all functions: on the left, with problem size equal to 5, the distributions are equal, although the asynchronous version still solves one more function with the stricter tolerance (see legends). With problem size equal to 50, however, the plots on the right

Table 1: Global success rate on the separable functions described in [9] vs. number of particles; for each problem/size pair 25 independent trials were performed, with values of problem size DIM in (2, 3, 5, 10, 20, 30, 40, 50). A trial was considered successful if the distance between the optimum found by PSO and the actual optimum of the function was lower than 10^{-8} , otherwise it was stopped without success after $2 \cdot 10^4$ DIM fitness evaluations.

Number of Particles	SPSO2006	Sync-SPSO2006	Async-gBest	Sync-gBest
$10 + 2\sqrt{\text{DIM}}$	0.647	0.648	0.485	0.373
32	0.704	0.713	0.587	0.447
64	0.727	0.729	0.673	0.520
100	0.729	0.731	0.686	0.547
128	0.730	0.732	0.691	0.568

show that the synchronous PSO with fully connected topology is much less effective. The tests were performed with relatively small swarms: standard PSO [6] suggests that a number of particles equal to $n_P = 10 + 2\sqrt{D}$ is used for problems in D dimensions.

In view of designing a parallel implementation of PSO, where the number of particles has little influence on running time, we also evaluated how the increase in this parameter affects PSO effectiveness, i.e. its capacity to converge to global optima. Table 1 provides an overall estimate of the probability of success [12], which can be used to quantify the influence of swarm size on optimization performance. PSO confirms to be less sensitive to population size [3] than other metaheuristics, but very sensitive to the type of inter-particles communication [13].

Figure 6 finally summarizes our observations and shows that, for the two topologies, the likelihood of success changes differently with problem size for each number of particles and update policy combination.

3 Comments

Space available for this paper does not permit to discuss observations in depth. However, if equation (1) is viewed as a recombination operator [13], in the synchronous global best version all solutions in the same iteration are generated from the same parent (the global best), that changes only after all particles have moved and corresponding fitness has been evaluated. In the asynchronous version, however, in the same iteration particles are influenced by different social attractors. This may justify what was observed, since the loss of diversity reduces the exploration ability of population-based metaheuristics [14]. In fact, since PSO is very sensitive to topology, we can infer that diversity is better preserved if the particles interact within small neighborhoods, regardless of the policy update. The SPSO2006, used in the tests as local-best PSO variant, uses a random topology [15] where each particle derives its local attractor from a constant number (default 3) of informants which include the particle itself: this actually creates a sparsely-connected network that offers good performance (better than the global-best variant) even with a synchronous policy.

4 Conclusions

The results suggest that postponing the update of the attractors at the end of each generation does not undermine the effectiveness of PSO if (and, apparently, only if) a sparsely-connected topology is adopted. In particular, when the updates of attractors are synchronized in the presence of a fully-connected topology, performance degrades as problem size increases, apparently because this reduces population diversity and the search capacity of the algorithm.

As a future development of this study, we are undertaking experiments to collect data and statistics about swarm diversity to try and demonstrate this assumption. In doing so we are going to consider also non-separable and, in general, harder-to-optimize functions, to assess possible dependence of performances also on problem hardness.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proc. IEEE Int. Conf. on Neural Networks. Volume IV., Washington, USA, IEEE Comp. Soc. (1995) 1942–1948
2. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization: an overview. *Swarm Intelligence* **1**(1) (June 2007) 33–57
3. Carlisle, A., Dozier, G.: An off-the-shelf PSO. In: Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indianapolis, IN (2001) 1–6
4. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proc. of the Congress on Evolutionary Computation - CEC, Washington, USA, IEEE Comp. Soc. (2002) 1671–1676
5. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: IEEE Swarm Intelligence Symposium. (2007) 120–127
6. Kennedy, J., Clerc, M.: Standard PSO 2006 (2006)
[online] available at: http://www.particleswarm.info/Standard_PSO_2006.c.
7. Dioşan, L., Oltean, M.: What else is evolution of PSO telling us? *Journal of Artificial Evolution and Applications* **2008**(1) (2008) 1–12
8. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions. Technical Report 2009/20, Research Center PPE (2009)
9. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical Report RR-6829, INRIA (2009)
10. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. Technical Report RR-6828, INRIA (2009)
11. Hansen, N., Ros, R., Mauny, N., Schoenauer, M., Auger, A.: PSO Facing Non-Separable and Ill-Conditioned Problems. Research Report RR-6447, INRIA (2008)
12. Auger, A., Hansen, N.: Performance evaluation of an advanced local search evolutionary algorithm. In: The 2005 IEEE Congress on Evolutionary Computation, 2005. Volume 2., Washington, USA, IEEE Comp. Soc. (Sept. 2005) 1777–1784
13. Miranda, V.: Evolutionary Algorithms with Particle Swarm Movements. In: Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems. (Nov. 2005) 6–21
14. Wilke, D.N., Kok, S., Groenwold, A.A.: Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. *International Journal for Numerical Methods in Engineering* **70**(8) (2007) 962–984
15. Clerc, M.: Back to random topology. Technical report (2007)

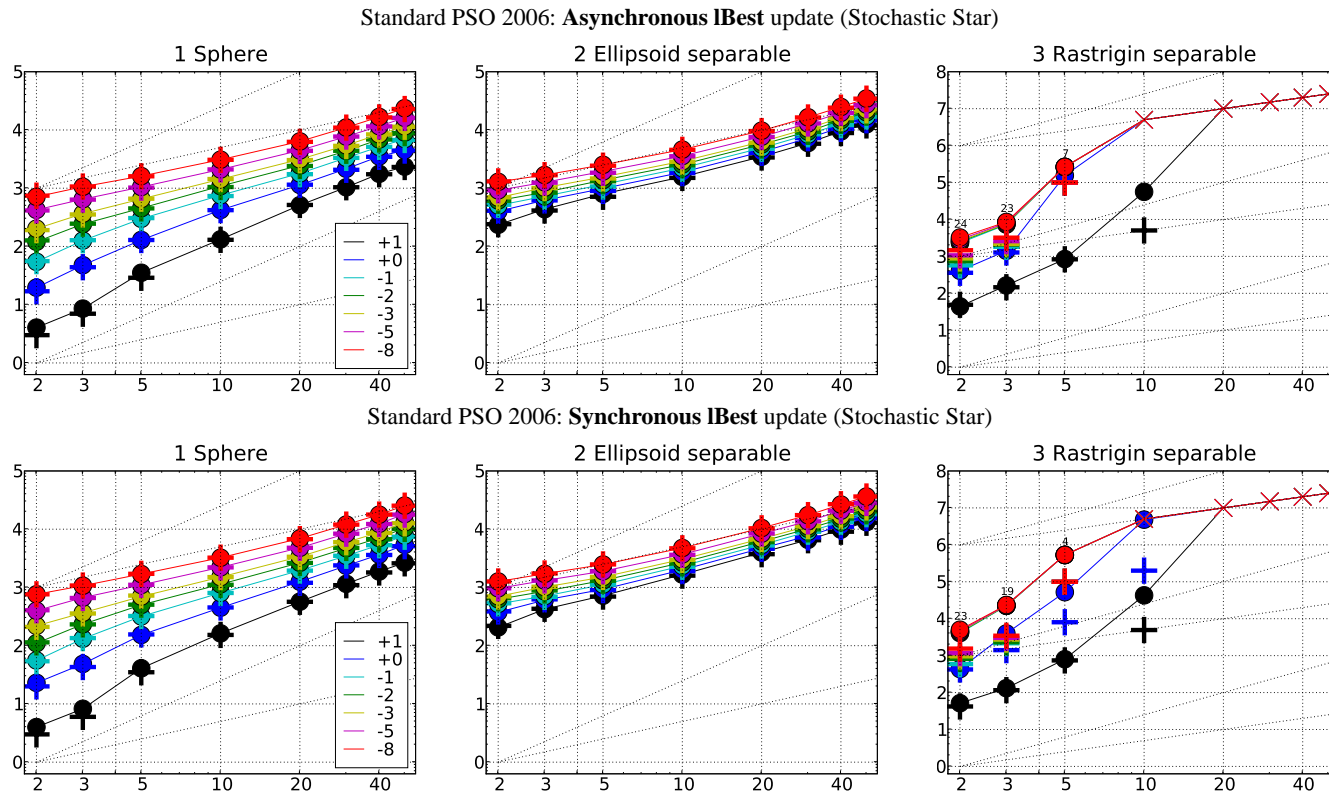


Fig. 2: Expected Running Time (ERT, ●) for the PSO with ‘local best’ topology to reach the target value $f_{opt} + \Delta f$, $\Delta f = 10^k$ (k as in legends), and mean (\log_{10}) fitness evaluations in successful runs (+) versus problem size. $ERT(\Delta f)$ is equal to #FEs divided by the number of successful runs. #FEs is the total number of fitness evaluations to reach $f_{opt} + \Delta f$ in all (successful and unsuccessful) runs. The crosses (×) denote the test which have failed in all runs. The numbers above the symbols ● denote the number of successful runs.

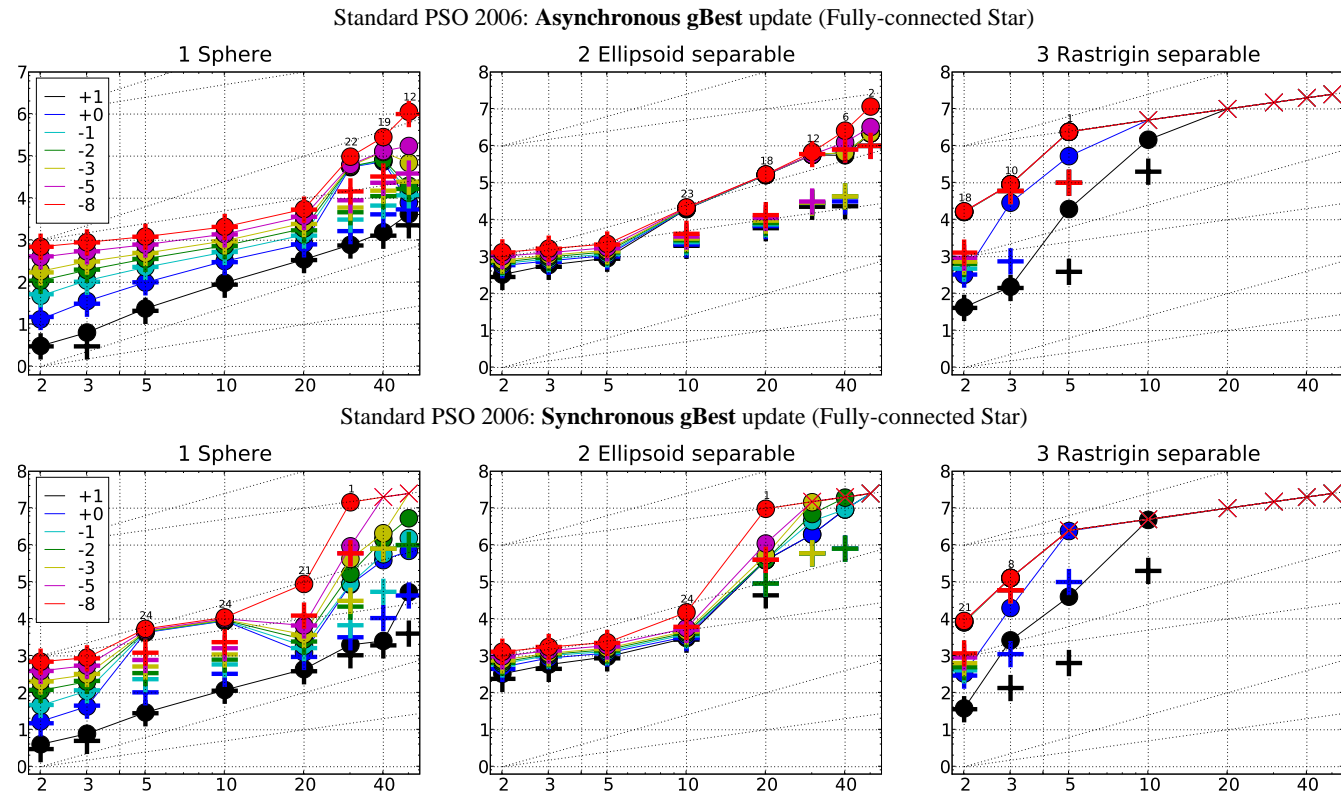


Fig. 3: Expected Running Time (ERT, ●) for the PSO with ‘global best’ topology to reach the target value $f_{\text{opt}} + \Delta f$, $\Delta f = 10^k$ (k as in legends), and mean (\log_{10}) fitness evaluations in successful runs (+) versus problem size. ERT(Δf) is equal to #FEs divided by the number of successful runs. #FEs is the total number of fitness evaluations to reach $f_{\text{opt}} + \Delta f$ in all (successful and unsuccessful) runs. The crosses (×) denote the test which have failed in all runs. The numbers above the symbols ● denote the number of successful runs.

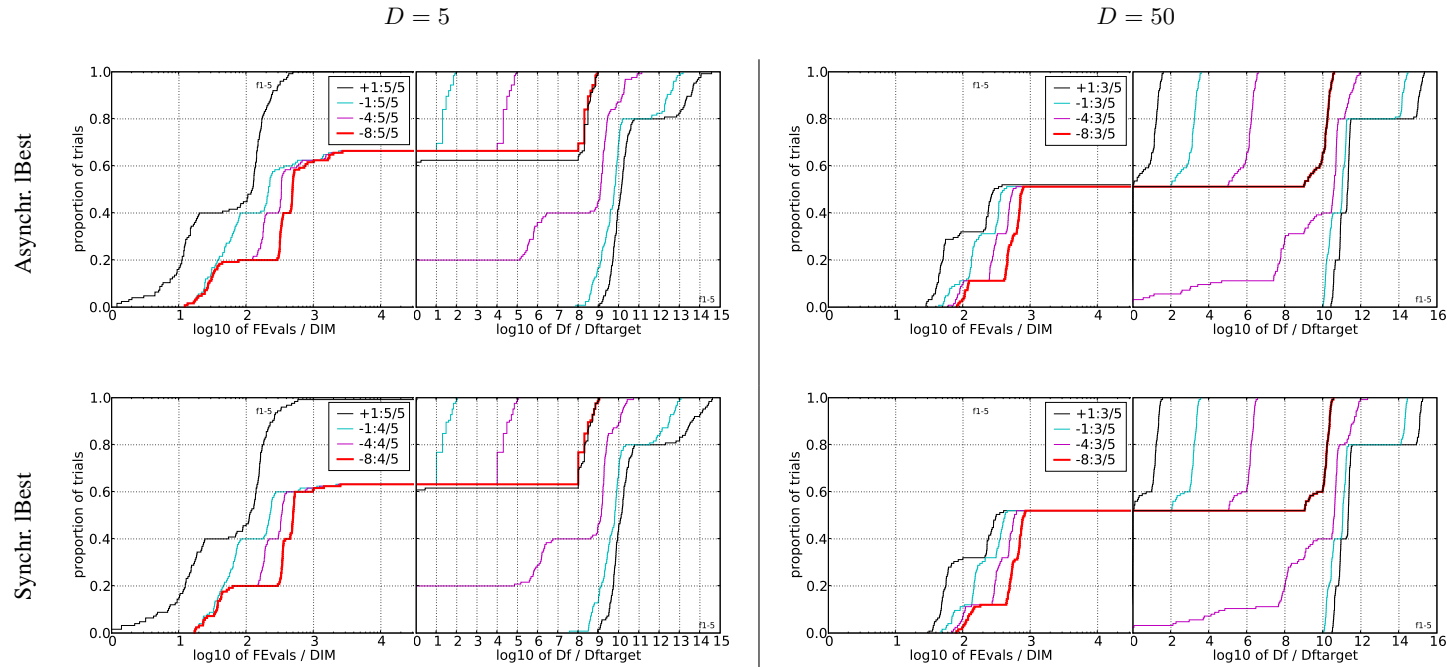


Fig. 4: PSO with 'local best' topology. Empirical cumulative distribution (ECDFs) of success rate vs. running time (subplots on the left) or vs. distance from the global optimum (Δf) (subplots on the right). The thickest lines denote the best results. Subplots on the left: ECDF vs. number of fitness evaluations to achieve $f_{\text{opt}} + \Delta f$ (with $\Delta f = 10^k$, k being the first value in the legend), divided by the dimension D of the search space. Subplots on the right: ECDF of the best Δf divided by 10^k (top left in the continuation of the subplots on the left), and best Δf divided by 10^{-8} for D , $10D$, $100D$ fitness evaluations (from right to left, black, cyan, and magenta plots). Each pair of plots (line-wise) represent a different combination of topology (local/global best) and communication strategy (synchronous/asynchronous update). The second value in the legends is the number of functions for which at least one execution has been successful. $FEvals$ is the number of evaluations, D or DIM the size of the search space and Δf or Df the distance from the global optimum.

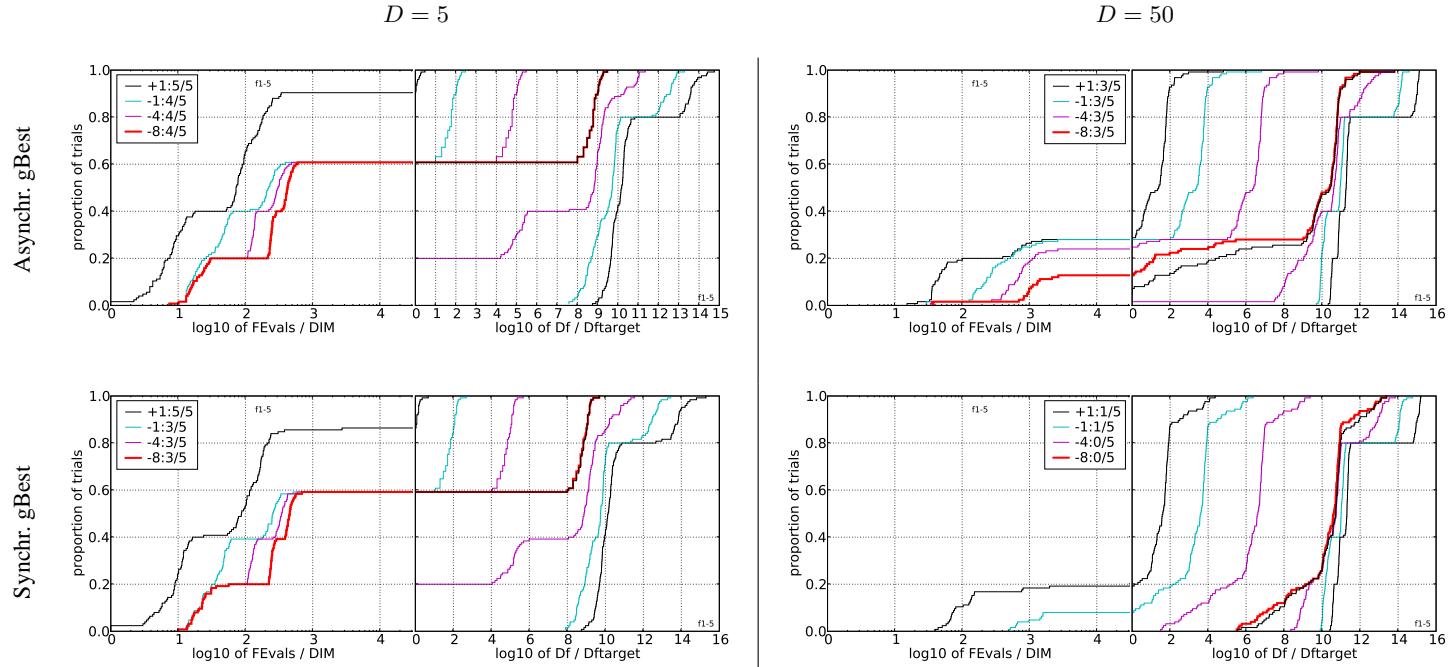


Fig. 5: PSO with ‘global best’ topology. Empirical cumulative distribution (ECDFs) of success rate vs. running time (subplots on the left) or vs. distance from the global optimum (Δf) (subplots on the right). The thickest lines denote the best results. Subplots on the left: ECDF vs. number of fitness evaluations to achieve $f_{\text{opt}} + \Delta f$ (with $\Delta f = 10^k$, k being the first value in the legend), divided by the dimension D of the search space. Subplots on the right: ECDF of the best Δf divided by 10^k (top left in the continuation of the subplots on the left), and best Δf divided by 10^{-8} for D , $10D$, $100D$ fitness evaluations (from right to left, black, cyan, and magenta plots). Each pair of plots (line-wise) represent a different combination of topology (local/global best) and communication strategy (synchronous/asynchronous update). The second value in the legends is the number of functions for which at least one execution has been successful. $FEvals$ is the number of evaluations, D or DIM the size of the search space and Δf or Df the distance from the global optimum.

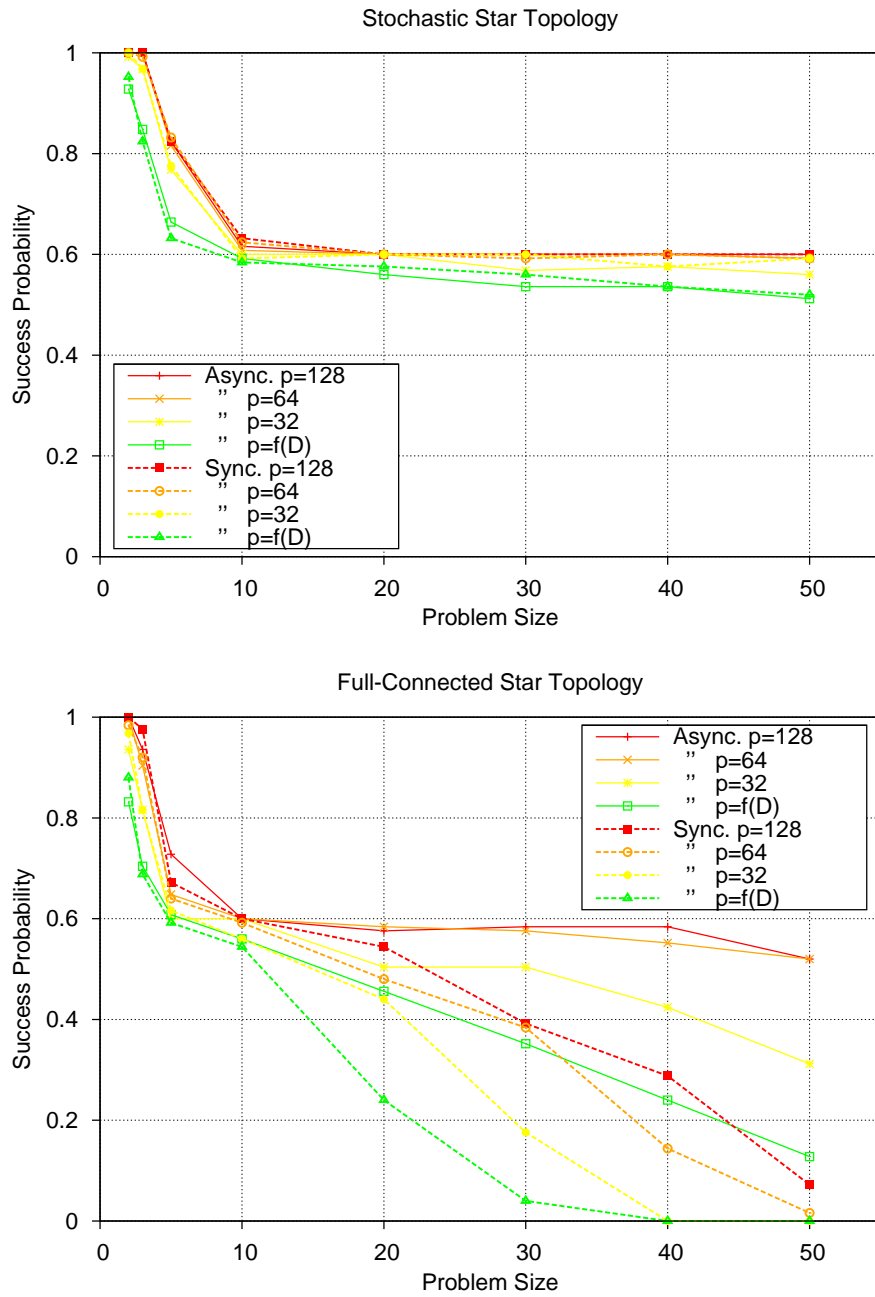


Fig. 6: Success Probability versus number of dimensions. The results are split into two graphs according to the swarm topology (see titles). Success rates are evaluated with the same criteria adopted in Table 1. Line style (dotted or solid) relates to the attractors' update policy (synchronous or asynchronous); line color is related to swarm size (see legends).