2012 Third International Conference on Networking and Computing

Asynchronous Strategy of Parallel Hybrid Approach of GA and EDA for Function Optimization

Said Mohamed Said
Information Engineering Department
University of the Ryukyus
1 Senbaru, Nishihara, Okinawa 903-0213 JAPAN
saidy87@hotmail.com

Morikazu Nakamura
Information Engineering Department
University of the Ryukyus
1 Senbaru, Nishihara, Okinawa 903-0213 JAPAN
morikazu@ie.u-ryukyu.ac.jp

***Abstract*—This paper adapts parallel master-slave estimation of distribution and genetic algorithms (GAs and EDAs) hybridization. The master selects portions of the search space, and slaves perform, in parallel and independently, a GA that solves the problem on the assigned portion of the search space. The master's work is to progressively narrow the areas explored by the slave's GAs, using parallel dynamic K-means clustering to determine the basins of attraction of the search space. Coordination of activities between master and slaves is done in an asynchronous way (i.e. no waiting is entertained among the processes). The proposed asynchronous model has managed to reduce computation time while maintaining the quality of solutions.**

***Keywords*-Hybrid, Estimation of Distribution Algorithm, Genetic Algorithms, Synchronous, Asynchronous, Parallel processing, Master-Slave, K-means clustering;**

I. Introduction

Researches in [1][3][4] and many others hint high capability of Evolutionary Computation (EC) to solve various problems in computer science domain. However, with rapid growth of real world problem size and complexity, higher computational cost is needed to solve these problems. While hybridization boosts the performance of EC [10], parallel processing helps to speed up searching process as shown in [11] and [12]. This paper is a continuation of proposed algorithms in [6] and [18] to achieve both higher quality solutions and good computational speed based on GAs and EDAs hybrid approach. The latter outperformed the former by improving solution quality but in a cost of additional computation time. The increased complexity was due to additional load in dynamic K means clustering algorithm even when parallelly executed. In [18] strategic synchronous master-slave formulation of EDA and GA was used similar to that in [6] except that master part was emphasized by parallel dynamic K means clustering for more reasonable estimation.

Both GAs and EDAs have shown promising achievements and have been used in variety of problem domains [5],[14],[13],[2]. The aim of this research is to reduce computation time by introducing asynchronous strategy to the present master-slave hybrid scheme. The proposed change has been well analyzed and proved to be fruitful in [9] and [20]. Both of them assure significant reduction of computation time in an asynchronous mode. Our approach suits well in an asynchronous mode due to the fact that it uses shared memory multiprocessor unit with several parallel threads working over one common population. From a parallel processing point of view, reducing unnecessary communication among processors is essential to avoid performance degradation [20]. In asynchronous master-slave scheme, slaves perform independent evolutionary computation using GA with un-identical number of generations(i.e slave terminates searching when predefined target fitness value has been reached) and master controls the searching using EDA, whenever pre-defined number of solutions in the Database(DB) returned by slaves is reached. Furthermore the master EDA has to follow four phase strategy adopted in [6], with every phase initiated by parallel dynamic K means clustering except in the first and last phases. The phase defines the manner in which EDA probabilistic estimation vectors are obtained by the master. The experiment was done using Real-Parameter Black Box Optimization Benchmarking system on noiseless testbed and compared with performance of [18] and GA. The results suggest maintained solution quality with notable reduction in computation time.

II. Hybrid Asynchronous Master-Slave Scheme

Among the merits of master-slave formulation are; i) it is a simple transposition of the single processor evolutionary algorithm onto multiple processor architectures that allows reproducibility of results, ii) there is no permanent loss of information when a slave fails or is unreachable by the master, iii) it is appropriate for networks of computers where availability is sometimes limited (e.g. available only during night time or when screen saver is on) as nodes can be added or removed dynamically with no loss of information, and iv) it is made of a centralized repository of the population which simplifies data collection and analysis as elaborated in [17]. In changing our approach to be asynchronous, we maintained its basic master-slave architecture to retain the above mentioned benefits. The whole algorithm runs in a fixed number of repetitive iterations, with all slaves executing GA and re-initialization taking place at the beginning of every iteration. The manner in which population members are initialized in each iteration is controlled by master using probabilistic estimations of EDA with the help of K means clustering algorithm. Using EDA on local optimal solutions returned by slaves, master can guess the areas

978-0-7695-4893-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICNC.2012.80

within search space with high potentials. The order of probabilistic search space exploration(Strategy) is of vigor importance, as experimentally proved in [6], hence master performs this task in four sequentially arranged phases as follows;

Wide Range Search (WRS): At the beginning of searching in early iterations, initialization has to be random to explore much wider area of the search space (No idea where the global optimum lies).

Outside Clusters Search (OCS): When there is enough number of solutions returned as a result of slaves GA, parallel dynamic K means clustering is performed. The results of clusters are used to form vectors $p(x)$ as per EDA probabilistic model, in our case simplest Univariate Marginal Distribution Algorithm (UMDA). The vectors represent specific areas within search space. In this phase Master instructs slaves to perform initialization outside clusters to avoid immature convergence (i.e. use complements of determined vectors).

Cumulative Clustering (CC): In this phase same process as in OCS is used but with the increased number of solutions to be clustered. Now the initialization take place using vectors (i.e. within clusters).

Best Cluster Focusing (BCF): Using the results of CC, master can have very good guess where the best solution lies. Hence final searching concentration is done only in promising area.

In initializing new population using vectors (sampling), slaves assume gaussian distribution with mean and standard deviation calculated using cluster members.

A. Synchronous vs Asynchronous Model

In synchronized version of our approach all slaves performed GA searching in exactly same number of generations at each iteration. When one slave finished its execution, it needed to wait for other slaves to finish so that they can start next iteration at the same time again. At the master side, it also needed to wait for the slaves to reach specified number of iterations to start the estimation process. The purpose of this synchronization was to provide smooth timing and activity scheduling between master and slaves. For example, master can know after how many iterations it will start clustering and phase change was after fixed iterations. As for the salves it is easy to know the type of initialization(phase) they are going to use at a given iteration.

Figure 1 is an activity-time diagram, that shows sequence of events from the start to the end of algorithm for synchronous model. In the figure M and S represent master and slave processes respectively. Sync means the starting point of synchronization (i.e. the time when the quickest slave has finished its searching) and Es means probabilistic estimation of search space. The phase change occurs at dark dashed horizontal lines. Within CC phase, Sync and Es occur at each iteration. The vertical dark lines on M and S1 indicate time intervals when master and slaves become active respectively. Thin parts of M and S1 are when they are idle. Master is inactive when it is waiting for slaves to complete EC and return best solutions, while slaves are idle when waiting for master to complete estimation.

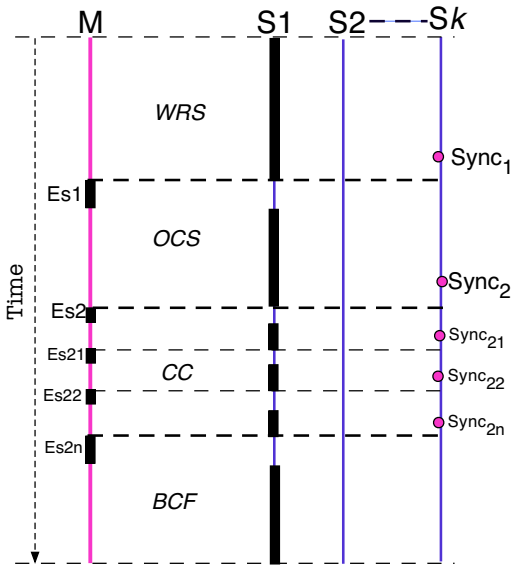

Figure 1. Synchronous model activity-time sketch showing searching, estimation and synchronization intervals.

In contrast to synchronous, asynchronous version accepts differences in computational speeds between slaves. If the stopping criteria is met by any of the slaves after any number of generations, it can terminate and start another iteration initialization without waiting for others to finish. It also sets the number of stored solutions to reach a specific value for master's estimation to begin. Hence the algorithm obeys the following conditions to ensure correctness and effectiveness of estimation, initialization and idle time reduction;

- If the slave finishes the phase before master has finished the estimation needed for next phase, it has to reinitialize according to the phase it has just finished using best local optimal solution values.
- To reduce master's idle times, we must set the number of solutions to be reached before start of estimation to be smaller than total possible number of solutions returned during one complete phase of synchronous model.
- Slave's number of iterations in phases are not necessarily the same. For example, a slave may have more iterations in OCS than in WRS.

Figures 2 and 3 show pseudo codes that summarize the slaves and master processes respectively in an asynchronous model. In figures T means strategy and x denotes a real gene of chromosome in a population. In figure 3, step 10 indicates the start of EDA by using solutions stored in database DB. In step 13 the result of clustering algorithm stored in database DB are used to generate probabilistic vectors ($p_i(x)$) governed by strategy T.

Figure 4 shows diagrammatically the result of changes

```
1: for i = 0... MAXITERATIONS do
2:    Recieve(p(x)) from Master;
3:    P ← GenerateInitialPopulation(p(x));
4:    Evaluate(P);
5:    while target fitness value is not reached do
6:       P' ← Recombine(P);
7:       P'' ← Mutate(P');
8:       Evaluate(P'');
9:       P ← Select(P, P'');
10:   end while
11:   Send the best solutions to Master;
12: end for
```

Figure 2. Pseudo code for Slaves

```
1: Initialize(DB);
2: for i = 0...k − 1 do
3:    T ← WRS;
4:    p_i(x) ← GenerateProbabilityVector(T, DB);
5:    Send p_i(x) to Slave i;
6: end for
7: while termination condition is not met do
8:    ReceiveSolutionFromSlaves(DB, i);
9:    if EnoughSolutions then
10:      EstimationOfDistribution(DB);
11:      T ← Strategy();
12:      Perform parallel K means clustering(DB);
13:      p_i(x) ← GenerateProbabilityVector(T, DB);
14:      Send p_i(x) to Slave i;
15:   end if
16: end while
```

Figure 3. Pseudo code for Master

made in terms of idle and active intervals of master and slaves. Continuous vertical thick dark line on S shows complete removal of idle times in all slaves. On M line we can notice that estimation starts prior to the finish of phases (or iterations in CC). Early estimation ensures early completion of next phase and decreases master's idle time.

Removal of slaves waiting time and reduction in master's idle time make the overall algorithm to complete execution earlier than the previous synchronous model.

III. Parallel Processing

The used hybrid approach includes parallelism in both master and slaves.

A. Platform

Parallel processing is very important tool for achieving better computational speeds in complex algorithms. Its implementation can either be in shared memory, or distributed memory architecture. The former uses multiple processor cores in a single processing unit and is characterized by the ability for all processors to access all memory as single addressing space. The latter uses processors distributed in multiple processing units, and requires a communication network to connect inter-processor memory. We in this research have adopted shared memory

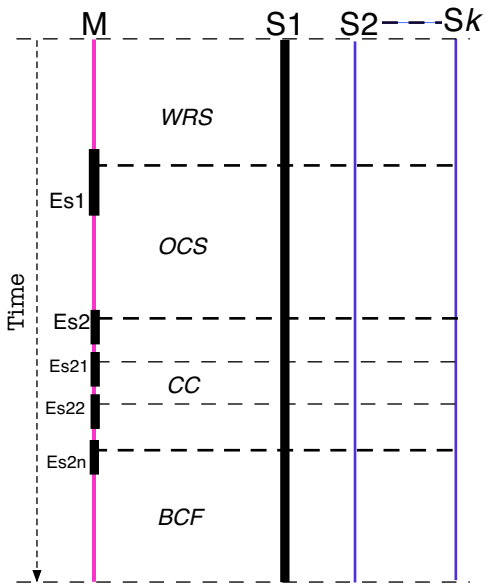

Figure 4. Asynchronous model activity-time sketch showing searching and estimation intervals

parallel processing technique using multi-core processing unit.

B. Master's Parallel Control

Master's search space control is initialized by dynamic K means clustering adopted in [16] to optimize the results of clustering algorithm by increasing inter-cluster distance and decreasing intra-cluster distance. Other approaches for optimized clustering results are well explained in [7][8][15]. The clustering is performed with different values of K, each running in an independent subthread initiated by Master process. Subthreads execute their work in parallel and their results are compared using validity measure. Master then uses the clusters of optimal K to form the probabilistic vectors of EDA and standard deviation within clusters with the guide of Strategy explained above.

C. Slaves Parallel Searching

All slaves need to initialize their population for GAs based on probabilistic vectors and standard deviations returned by master process to sample new solution strings at the beginning of every iteration. Under the guidance of master process, the searching is performed independently in parallel by all slaves. Slaves parallel GA is by itself asynchronous in a sense that a slave's GA can converge at any time if the target value is reached.

IV. Experimental Evaluation

To evaluate the performance of our algorithm, we implemented our method using the C programming language with the POSIX Threads (Pthreads) library on a Mac Pro (Intel Dual Quad Core, i.e., 8 cores in total; 3.0 GHz; 12 GB RAM). The algorithm can also be implemented using OpenMP as a shared memory parallelism. However in

distributed memory environment of multicomputer system, message passing library (MPI) might as well be used for parallelization. All the libraries are available in C/C++ and Fortran programming languages. The experiment has been performed to compare the performance of asynchronous hybrid (ASYNCH) approach with synchronous hybrid approach (SYNCH) and Genetic Algorithm GA. GA was also executed using parallel Pthreads to match the conditions with other two algorithms. Furthermore the results are compared with best results of BBOB 2009. For every algorithm, each dimension was executed in 15 instances to try to reach target value f_{t}.

A. Test Functions

We used 24 benchmark functions whose parameter settings are well explained in [21], [23]. For every function the execution was done in dimensions 2, 3, 5, 10, 20 and 40 in [-5 5] search space. In 24 functions we have 5 subgroups categorized as separable functions, moderate functions, ill-conditioned functions, multi-modal functions and weakly structured multimodal functions.

B. Algorithms Parameter Settings

Crossover and mutation probabilities used in GAs are 1 and 0.08 respectively with population size of 50. GA uses Stochastic Remainder method for selection. 3 Pthreads were used to represent slaves and 1 Pthread to represent master process. We used three iterations in each tactic (phase), hence making total of twelve sequentially executed iterations in every instance. The phase change was set to occurs after every three iterations. The clustering algorithm used values of K from 2 to 6 executed concurrently by 5 Pthreads created by Master. Number of solutions to be reached before start of clustering and EDA estimation in an asynchronous model were 400, 800, 1000, 1200, and 1400.

C. Results

Results from experiments according to [22] on the benchmark functions given in [21], [23] are presented in Figures 5, 6 and 7 and in Tables I to VIII. The **expected running time (ERT)**, used in the figures and tables, depends on a given target function value, $f_{\mathrm{t}} = f_{\mathrm{opt}} + \Delta f$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach f_{t}, summed over all trials and divided by the number of trials that actually reached f_{t} [22], [19]. **Statistical significance** is tested with the rank-sum test for a given target Δf_{t} (10^{-8}) using, for each trial, either the number of needed function evaluations to reach Δf_{t} (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration. However the best algorithm in BBOB-2009 is far better than all of the three compared algorithms. As always expected, extending the model to be asynchronous has some challenges. Particularly for this algorithm initialization conflicts sometimes occurred due to some slaves starting their next iteration before the intended masters estimation has taken place. However we solved this problem by telling slaves to initialize using previous iterations local minima as probabilistic vectors. This sometimes caused performance degradation in both solution quality and computation time.

Tables I to VIII give the Expected Running Time (ERT) for targets $10^{1,\,-1,\,-3,\,-5,\,-7}$ divided by the best ERT obtained during BBOB-2009 (given in the $\mathrm{ERT}_{\text{best}}$ row), respectively in 5-D and 20-D. Bold entries correspond to the best values. The median number of conducted function evaluations is additionally given in *italics*, if $\mathrm{ERT}(10^{-7}) = \infty$. #succ is the number of trials that reached the final target $f_{\mathrm{opt}} + 10^{-8}$. Entries with the $\downarrow$ symbol are statistically significantly better (according to the rank-sum test) compared to the best algorithm in BBOB-2009, with $p = 0.05$ or $p = 10^{-k}$ where $k > 1$ is the number following the $\downarrow$ symbol, with Bonferroni correction of 24. In the tables we can observe that with different functions, number of successful trials vary for each algorithm. For example in table V all algorithms could reach Δf_{t} (10^{-8}) in all 15 trials, while in table VIII only best algorithm in BBOB 2009 could reach Δf_{t} (10^{-8}). This variation suggest that solution quality depends highly on target functions. Since number of evaluations required reflects the computation time, it is normal to expect that comparing speeds of algorithms can be decided by how many function evaluations it has gone to reach certain target value. This situation is true when comparing GA with other two (ASYNCH and SYNCH). Between ASYNCH and SYNCH time difference was because of wait overheads due to synchronization. Therefore similar number of evaluations for less computation time with similar quality of solutions.

Figures 6 and 7 and Tables I to VIII justify the dominance of ASYNCH and SYNCH in both quality of solutions and algorithm computational speed over GA. Figure 5 suggests that ASYNCH outperforms SYNCH.

D. CPU Timing Experiments

For the timing experiments, all three algorithms were run on f_8 and restarted until at least 30 seconds (according to [23]). All experiments have been conducted on Mac Pro (Intel Xeon Dual Quad Core, i.e., 8 cores in total; 3.0 GHz; 12 GB RAM) with OS X 10.6.8. For GA the results were 2.0; 2.0; 2.0; 2.0; 2.1 and 2.2 x10^{-5} seconds per function evaluations in dimensions 2; 3; 5; 10; 20 and 40 respectively. For SYNCH the recorded times in seconds are exactly similar to those in GA in all dimensions. The results for ASYNCH were 1.8; 1.9; 2.0; 2.0; 2.0 and 2.0 x10^{-5} seconds per function evaluations in dimensions 2; 3; 5; 10; 20 and 40 respectively. Furthermore overall execution of 24 functions, in all dimensions and all 15 instances finds ASYNCH algorithm finishing 44.4 minutes before GA and 20.4 minutes before SYNCH. These results show that if asynchronous model is used, we can reduce overall algorithm processing time while maintaining solutions quality.

V. Conclusion and Future work

In this paper, we proposed asynchronous parallel hybrid method of EDAs and GAs using master-slave cooperation. Master process estimates the probability distribution of the search space on the basis of the non-dependency model. The slaves generate new initial population based on estimation. Upon convergence the slaves proceed with another search and do not wait for other slaves to converge or master to finish estimation. Both master and slaves processes have been parallelized for speedup. Furthermore in theory as the number of processor cores increases, computation time is expected to be reduced in some manner. The experiment has been done using only fixed number of processor cores, hence in the future we will study how does our approach behave under variable number of processor cores. The experiments conducted verify the increased speed of asynchronous model over synchronous under fixed number of processor cores and parallel processes. However, with comparison to BBOB-2009 best algorithm, our method does not reach good results in higher dimensions, hence the introduction of other significant EDAs model is suggested for further improvement of our algorithm.

References

[1] L. Changhe, and Y. Shengxiang, *An Island Based Hybrid Evolutionary Algorithm for Optimization*, Simulated Evolution and Learning, LNCS, vol. 5361, pp. 180–189, 2008.

[2] H. Mühlenbein, and G. Paa *From recombination in genes to the estimation of dis- tribution 1, binary parameters*, Lecture Notes in Computer Science 1411, Parallel Problem Solving from Nature, 1996.

[3] T. Niknam, B. Amiri, J. Olamaei, and A. Arefi, *An Efficient Hybrid Evolutionary Optimization Algorithm based on PSO and SA for clustering*, JZU SC. A , vol 10, No 4, 512–519, 2008.

[4] T. Tometzki, and S. Engell, *Hybrid evolutionary optimization of two-stage stochastic integer programming problems*, An empirical investigation, Evol Comput, Winter 17(4) 511–526, 2009.

[5] R. Salomon, *Re-evaluating genetic algorithm performance under coordinate rotation of benchmark func- tions. a survey of some theoretical and practical aspects of genetic algorithms*, ELSEVIER, Biosystems 39, 263278, 1996.

[6] S. M. Said, and M. Nakamura, *A Hierarchical Hybrid Evolutionary Computation for Continuous Function Optimization*, IJNGC, Vol. 3, No 1, 2012.

[7] J.C. Bezdek, and N.R. Pal, *Some new indexes of cluster validity*, IEEE Trans. Syst. Man. Cybern., vol. 28, pp. 301–315,1998.

[8] G. W. Milligan, and M.C. Cooper, *An examination of procedures for determining the number of clusters in a data set*, Psychometrika, vol. 50, pp. 159–179,1985.

[9] S. S. Abdelwahed, M. F. Hassan, M. A. Sultan, *Parallel asynchronous algorithms for optimal control of large scale dynamic systems*, J. Optimal Control applications and methods. 18, 1997.

[10] C. Grosan, and A. Abraham *Hybrid evolutionary algorithms: methodologies, architectures, and reviews* In: Grosan C, Abraham A, Ishibuchi H, editors. Hybrid evolutionary algorithms. Berlin, Heidelberg: Springer; p. 117, 2007.

[11] J. Ocenasek, *Parallel estimation of distribution algorithms*, Ph.D. thesis, Brno University of Technology, 2002.

[12] A. Mendiburu-Alberro, *Parallel implementation of estimation of distribution algorithms based on proba- bilistic graphical models. application to chemical calibration models*, Ph.D. thesis, The University of the Basque Country, 2006.

[13] P. Larranaga, and J. A. Lozano, *Estimation of Distribution Algorithms*, Kluwer Academic Publishers, 2002.

[14] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[15] G. W. Milligan, *Clustering validation: Results and implications for applied analyses*, In P. Arabie, L.J. Hubert and G. De Soete (Eds.), Clustering and Classification, Singapore: World Scientific, pp 341–375, 1996.

[16] Ray and R. H. Turi, *Determination of number of clusters in K-means clustering and application in colour image segmentation*, (invited paper) in N R Pal, A K De and J Das (eds), Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99), Calcutta, India, Narosa Publishing House, New Delhi, India, ISBN: 81-7319-347-9, pp 137-143, 27-29 December, 1999.

[17] C. Gagne, and M. Parizeau, *A Robust Master-Slave Distribution Architecture for Evolutionary Computations*, Proc of Genetic and Evolutionary Computation Conference, Late Breaking, pp. 80–87, 2003.

[18] S. M. Said, and M. Nakamura, *Parallel Enhanced Hybrid Evolutionary Algorithm for Continuous Function Optimization*, Proc of 7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Victoria, Canada, (Accepted), 2012.

[19] K. Price, *Differential evolution vs. the functions of the second ICEO*, In Proceedings of the IEEE International Congress on Evolutionary Computation, pages 153157, 1997.

[20] M. Golub and L. Budin *An Asynchronous Model of Global Parallel Genetic Algorithms*, 2nd ICSC Symposium on Engineering of Intelligent Systems EIS2000, pp 353–359, 2000.

[21] S. Finck, N. Hansen, R. Ros, and A. Auger, *Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions*, Technical Report 2009/20, Research Center PPE, 2009, Updated February 2010.

[22] N. Hansen, A. Auger, S. Finck, and R. Ros *Real-parameter black-box optimization benchmarking 2010: Experimental setup*, Technical Report RR-7215, INRIA, 2010.

[23] N. Hansen, S. Finck, R. Ros, and A. Auger, *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*, Technical Report RR-6829, INRIA, 2009, Updated February 2010.

Table I
ERT ON f_1 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f1	11	12	12	12	12	12	15/15
ASYNCH	36(99)	691(379)	1210(349)	2150(361)	4088(537)	**1.3e4**(8371)	15/15
SYNCH	**4.5**(4)	**531**(367)	1001(383)	1889(513)	3919(535)	1.9e4(2e4)	15/15
GA	9.3(6)	575(381)	**984**(349)	**1864**(454)	**3829**(777)	2.4e4(2e4)	6/15

Table II
ERT ON f_2 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f2	83	87	88	90	92	94	15/15
ASYNCH	**353**(154)	523(231)	882(566)	**2925**(832)	**4908**(1957)$^{\star 2}$	**9594**(7580)*	10/15
SYNCH	394(127)	515(146)	833(466)	4479(2170)	8046(1403)	2.4e4(2e4)	4/15
GA	353(130)	**507**(127)	**658**(148)	6183(7653)	∞	∞ *1e6*	0/15

Table III
ERT ON f_3 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f3	716	1622	1637	1646	1650	1654	15/15
ASYNCH	**17**(8)	**39**(34)	94(64)	**119**(61)	**254**(125)*	**439**(29)$^{\star 2}$	14/15
SYNCH	22(11)	49(33)	**93**(71)	150(133)	404(63)	528(61)	7/15
GA	18(14)	73(87)	113(133)	220(220)	4028(4420)	∞ *1e6*	0/15

Table IV
ERT ON f_4 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f4	809	1633	1688	1817	1886	1903	15/15
ASYNCH	31(18)	80(72)	**100**(77)	**158**(103)	**326**(104)$^{\star 2}$	**397**(46)	11/15
SYNCH	**29**(26)	80(55)	183(138)	231(154)	390(40)	755(517)	5/15
GA	34(26)	**71**(54)	181(244)	511(530)	∞	∞ *1e6*	0/15

Table V
ERT ON f_5 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f5	10	10	10	10	10	10	15/15
ASYNCH	**1028**(851)	**2675**(1311)	**3061**(1309)	**3177**(1230)	**3254**(1281)	**3313**(1244)	15/15
SYNCH	1070(751)	2709(787)	3063(839)	3218(785)	3293(752)	3368(733)	15/15
GA	1044(1000)	2971(1274)	3375(1330)	3507(1248)	3598(1248)	3654(1170)	15/15

Table VI
ERT ON f_6 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f6	114	214	281	580	1038	1332	15/15
ASYNCH	130(90)	166(100)	428(788)	**1562**(1254)	**6743**(6956)	∞ *1e6*	0/15
SYNCH	**94**(83)	**147**(116)	**234**(127)	1818(1097)	∞	∞ *1e6*	0/15
GA	124(102)	170(104)	377(350)	2.4e4(3e4)	∞	∞ *1e6*	0/15

Table VII
ERT ON f_7 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f7	24	324	1171	1572	1572	1597	15/15
ASYNCH	**210**(336)	410(301)	**513**(542)	**1400**(1234)	**1400**(1228)	**1393**(1219)	6/15
SYNCH	227(185)	**409**(570)	670(562)	2099(2142)	2099(2184)	2073(2150)	4/15
GA	335(280)	484(530)	783(822)	∞	∞	∞ *1e6*	0/15

Table VIII
ERT ON f_8 IN 5-D OVER ERT_{BEST} OBTAINED IN BBOB 2009

Δf_{opt}	1e1	1e0	1e-1	1e-3	1e-5	1e-7	#succ
f8	73	273	336	391	410	422	15/15
ASYNCH	213(77)	740(650)	2640(3253)	1.8e4(2e4)	∞	∞ *1e6*	0/15
SYNCH	202(46)	542(619)	2157(2362)	1.2e4(1e4)	∞	∞ *1e6*	0/15
GA	**180**(77)	**385**(437)	**898**(970)	**8410**(9093)	∞	∞ *1e6*	0/15

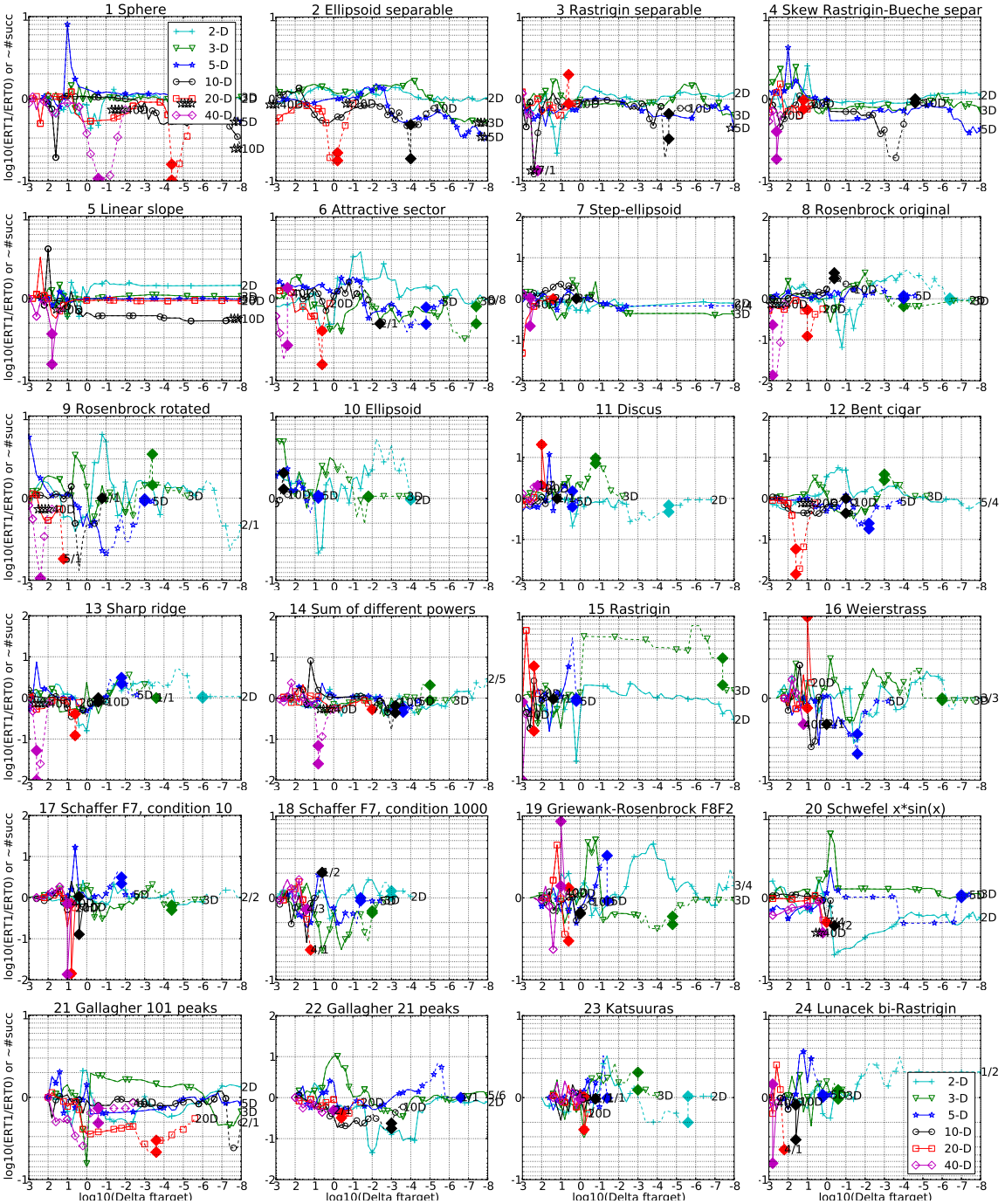

Figure 5. Ratio of ERT for ASYNCH over ERT for SYNCH versus $\log_{10}(\Delta f)$ in 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40-D:♢. Ratios $< 10^0$ indicate an advantage of ASYNCH, smaller values are always better. The line becomes dashed when for any algorithm the ERT exceeds thrice the median of the trial-wise overall number of f-evaluations for the same algorithm on this function. Filled symbols indicate the best achieved Δf-value of one algorithm (ERT is undefined to the right). The dashed line continues as the fraction of successful trials of the other algorithm, where 0 means 0% and the y-axis limits mean 100%, values below zero for ASYNCH. The line ends when no algorithm reaches Δf anymore. The number of successful trials is given, only if it was in $\{1 \ldots 9\}$ for ASYNCH (1st number) and non-zero for SYNCH (2nd number). Results are significant with $p = 0.05$ for one star and $p = 10^{-\#\star}$ otherwise, with Bonferroni correction within each figure.

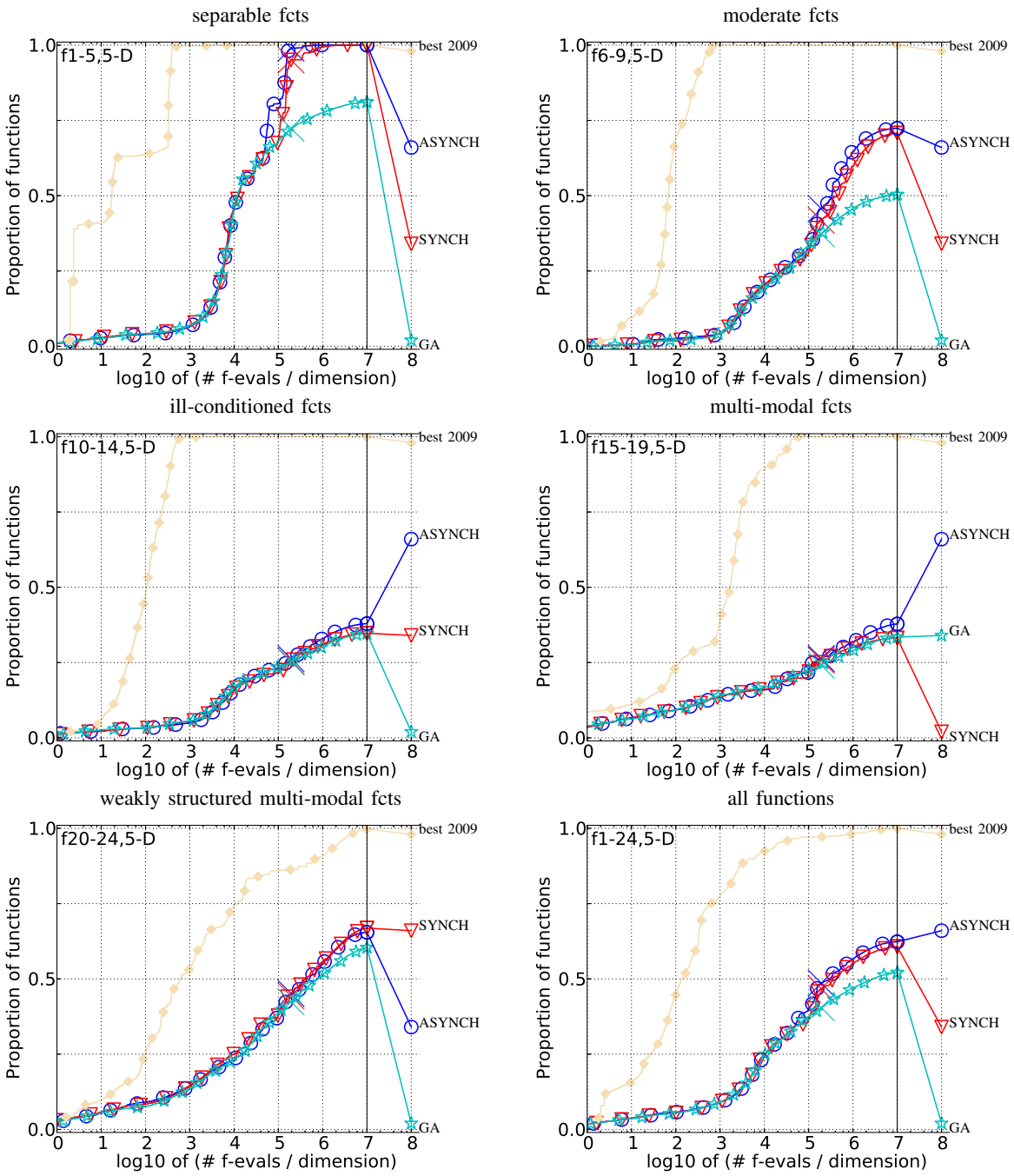

Figure 6. Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 5-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

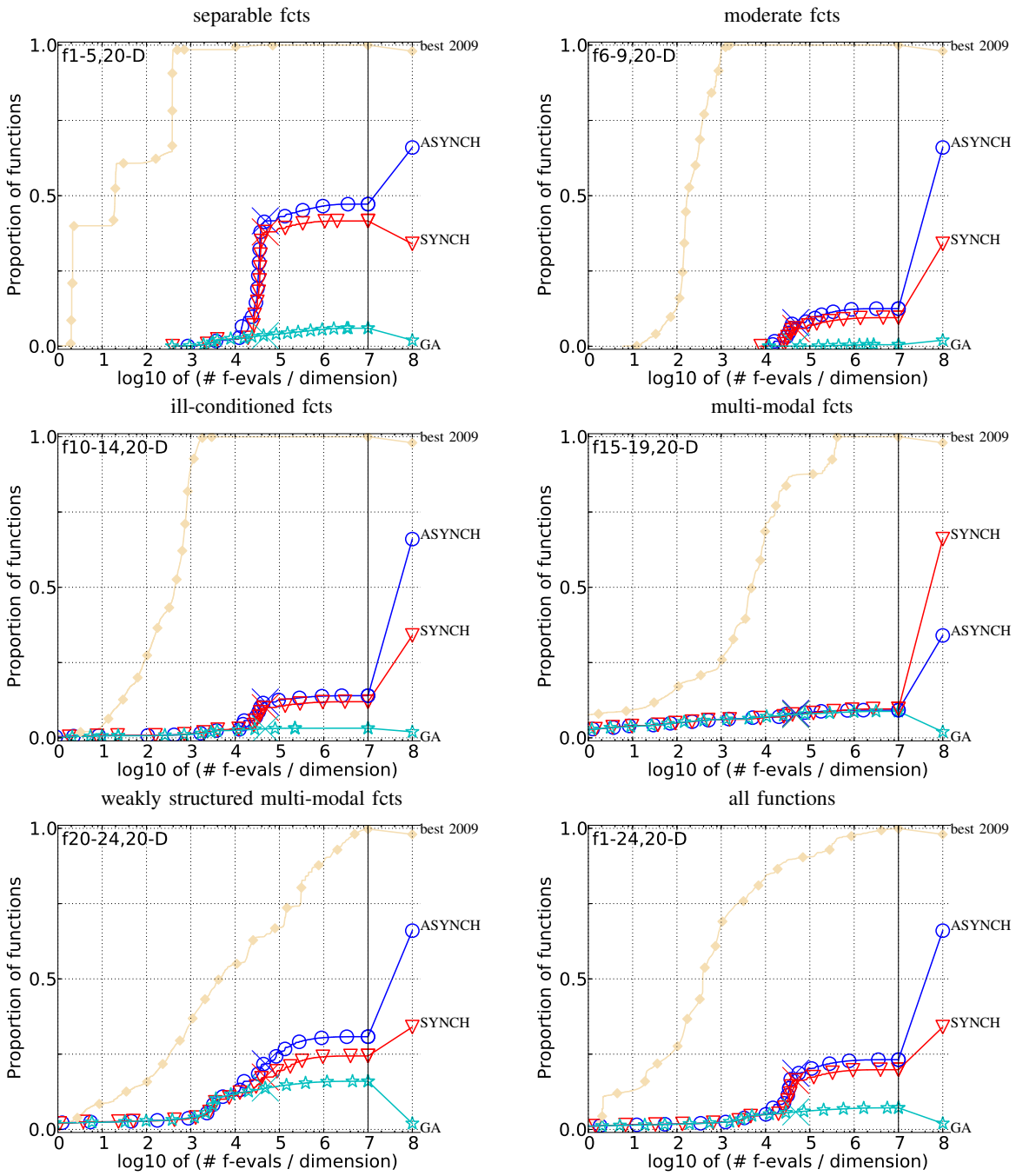

Figure 7. Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.