Simultaneous Optimistic Optimization on the Noiseless BBOB Testbed

Bilel Derbel*, Philippe Preux†
*Université Lille 1, †Université Lille 3, CRIStAL CNRS UMR 9189
*Dolphin, †Sequel — Inria Lille Nord Europe
*bilel.derbel@univ-lille1.fr, †philippe.preux@univ-lille3.fr

***Abstract*—We experiment the SOO (Simultaneous Optimistic Optimization) global optimizer on the BBOB testbed. We report results for both the unconstrained-budget setting and the expensive setting, as well as a comparison with the DiRect algorithm to which SOO is mostly related. Overall, SOO is shown to perform rather poorly in the highest dimensions while agreeably exhibiting interesting performance for the most difficult functions, which is to be attributed to its global nature and to the fact that its design was guided by the goal of obtaining theoretically provable performance. The greedy exploration-exploitation sampling strategy underlying SOO design is also shown to be a viable alternative for the expensive setting which gives rooms for further improvements in this direction.**

I. Introduction

Blackbox optimization refers to a specific setting where the function to be optimized is viewed as a blackbox that, provided a point in a given domain, returns the value of the function at that point; and nothing else but the so-returned value can be used in the optimization process. Additionally, only a limited budget (e.g., in terms of computing time) is typically available when attempting to optimize the given function which makes the optimization process even more challenging. Under such assumptions, one seeks a general purpose optimizer which can exhibit provable performance for a large panel of functions with variable and unknown properties. In this respect, the performance of a blackbox optimizer can be assessed using a purely theoretical approach where analytical bounds on approximation quality or running time are derived, or in contrast by adopting an experimental methodology for which careful considerations are to be taken. Of course the two approaches are complementary since they seemingly have the same goal: provide indisputable evidence about the relative performance of a given optimizer while informing about its behavior. In this context, we are interested in benchmarking a novel algorithm called "Simultaneous Optimistic Optimization" (SOO) originally introduced in [1] and falling in the family of global optimizers [2].

Generally speaking, the SOO algorithm finds its foundations in the machine learning field. It is a deterministic global optimizer which has theoretically provable performance for functions being locally 'smooth' near their global optima but where the actual smoothness is not known. In fact, only the performance of the algorithm is expressed as a function of smoothness (in a specific sense), but the algorithm design does not rely on this knowledge in any way. Interestingly, SOO is the only existing global optimizer for which the discrepancy between the best found point after a given number of function evaluations and the optimum can be bounded analytically under such a very weak assumption. Such a result being derived from a purely theoretical perspective, we aim in this paper at conducting an experimental analysis of SOO in order to complement the so-obtained theoretical results and to inform about the performance and the behavior of SOO using the BBOB testbed.

Algorithm 1: High level pseudocode of SOO.

Parameters: K, and $h_{\max}$
$\mathcal{T} \leftarrow$ initial tree with one node representing the whole domain ;
while maximum number of evaluations not exhausted **do**
  $\mathcal{C} \leftarrow \varnothing$; $v_{\min} \leftarrow +\infty$;
  for $h \in \{0, \dots, \min(\mathit{depth}(\mathcal{T}), h_{\max})\}$ **do**
    Among all leaves of $\mathcal{T}$ at depth h, select the one, denoted $C_h^\star$, having the best fitness value at its center, denoted $x_h^\star$;
    if $f(x_h^\star) \le v_{\min}$ **then** $\mathcal{C} \leftarrow \mathcal{C} \cup C_h^\star$; $v_{\min} \leftarrow f(x_h^\star)$;
  for $C \in \mathcal{C}$ **do**
    split C in K subcells and add them accordingly in $\mathcal{T}$;

II. Algorithm Presentation

SOO is a very simple algorithm as can be seen in the pseudocode of Algorithm 1. It can be viewed as a divide-and-conquer tree search algorithm. It proceeds iteratively by dynamically expanding a tree whose nodes are mapped to cells, also called hyper-rectangles, representing decreasing subdomains; with the initial root node representing the entire domain of the objective function. Having this in mind, SOO considers each depth in the tree and for each depth, selects the cell having the lowest (in the case of minimization, largest in the case of maximization) value among the leaves at this depth (breaking ties arbitrary) and splits this cell only if its value is lower than all previously selected cells of lower depths — the value of a cell being defined as the value of the objective function f at the center point x of the subdomain represented by that cell. Notice that selected cells are split and added in the tree only after all the depths of the current tree were processed. The iterations go on until the budget of function evaluations is exhausted and the point with the best fitness is returned.

SOO is very closely related to the so-called DiRect (Dividing Rectangle) algorithm [3]. DiRect also samples the search space by choosing at each iteration the next subset of hyperrectangles to divide both based on the fitness of center points (the lower the better) and the size of hyperrectangles (the larger the better). Although, SOO and DiRect share similar design components, DiRect assumes the objective function to be globally Lipschitz (with unknown Lipschitz constant) whereas SOO needs only a much weaker assumption to work properly which is: local smoothness around the global optimum. In addition, finite-time performance guarantees can be obtained for SOO whereas DiRect only enjoys an asymptotic convergence result. More specifically, after N function evaluations, the difference $f(x^*) - f^*$ between the value of the global optimum of the objective function $f^* = \min_{x \in X} f(x)$ and the value of the best point x^* returned by SOO is decreasing in $O(N^{-1/d})$ where d is the near-optimality dimension of f [1]. Note that exponential rate $e^{-O(N)}$ can also be achieved in the non-trivial case $d = 0$. That said from the theoretical side, we can find a first step towards the experimental evaluation of SOO in [4]. In the remainder, we extend on such an experimental evaluation using the BBOB testbed.

III. Experimental Procedure

We consider studying the behavior of SOO in the BBOB default scenario where the number of function evaluations is not constrained; and also, when considering an expensive setting where only a restricted number of function evaluations is available. Besides, we consider the study of the performance of SOO when compared to the DiRect algorithm for which experimental data is already available for the BBOB testbed [5].

We use the C implementation of SOO available for free download at [6] to which small modifications were made in order to adapt the definition of objective functions to fit the BBOB testbed. SOO has only three simple parameters: (i) the number of cells K resulting from a split, (ii) the split policy, and (iii) the maximum height $h_{\max}$ of the tree. Motivated by previous setups [4], [7], these parameters are set as following: (i) $K = 3$, (ii) the dimensions being numbered, the direction of a split is merely the next dimension, and (iii) $h_{\max} = 10 \cdot \sqrt{(\log N)^3}$ where N is the maximum number of function evaluations which is set to $D \cdot 10^5$ with D being the problem dimension. We also notice that there is no restart mechanism in SOO since it is deterministic and provides exactly the same result once the function and the domain are fixed; that is, all the 15 instances of BBOB2013 defined within $[-5, 5]^D$ for $D \in \{2, 3, 5, 10, 20\}$ in our setting.

IV. CPU Timing

In order to evaluate the CPU timing of the algorithm, we have run the requested BBOB timing experiments on the function f_8 until a maximum budget equal to $D \cdot 10^5$ is reached using a MacBook Pro 2.6 Ghz Intel Core i7 (8Go). The time per function evaluation for dimensions 2, 3, 5, 10, 20 equals 2.1, 2.3, 2.6, 3.1, 3.4 times 10^{-5} seconds respectively.

V. Results

Experimental results according to [8] on the benchmark functions given in [9], [10] are presented in Figures 1, 2 and in Table I for the standard setting; and in Figures 3, 4 and in Table II for the expensive setting. Results showing the relative performance of SOO when compared with DiRect are only shown in the expensive setting. Our main observations are summarized in the following.

Default unconstrained budget setting. In this scenario, SOO performs rather poorly in comparison to the artificial best algorithm from BBOB'2009, and it scales at least in a quadratic manner as a function of problem dimension. For low dimensions, the difference between SOO and the best artificial algorithm is surprisingly *more* pronounced for the group of functions f-1 to f-14 compared to the group of functions f-15 to f-24 — which are the most difficult ones. We attribute this to the global nature of SOO which keeps splitting subdomain-cells although it might exist other cells deeper in the tree with better fitness values at their centers. This is actually an exploration component in the design of SOO which ensures search consistency; but which makes SOO less efficient for the a priori easiest functions like the sphere (f-1) and the linear slope (f-5). In contrast, for the more difficult, multimodal and less structured functions, the sampling induced by the SOO tree search strategy is performing relatively much better, but it still has some issues when scaling dimensions.

Expensive setting. Similarly, there is a clear difference when examining the performance of SOO over the two groups of functions (f-1 to f-14) and (f-15 to f-24). However, SOO is now relatively competitive when compared to the best artificial algorithm for the multi-modal and weak-structure functions; with even slightly improved ERTs for a couple of functions in this group and for the lowest dimensions. Notice that the SOO domain sampling can simply be viewed as a greedy heuristic implying a very basic exploration-exploitation search trade-off. Nonetheless, and with regards to the observed performance, this greediness appears to be seemingly a viable strategy in order to sample relatively good points using a small budget for several low dimensional 'difficult' functions.

Comparaison with DiRect. Recall that we show comparison results only in the expensive setting. This is not only because of lack of space; but also because we did not observe much differences between SOO and DiRect using higher budget. In the expensive setting, we can yet see some small differences between the two algorithms with SOO's ERTs being overall slightly better especially in the highest dimensions (few exceptions can be found when examining Table II and most often differences are not statistically significant). Notice that the DiRect algorithm was not previously benchmarked in the expensive setting. Thus, our comparative results also indicate that this kind of global algorithms, that are based on heuristically dividing and systematically sampling the function domain, can be effective.

VI. Conclusion

We conclude this paper by remarking that SOO is originally coming from the machine learning community and its unique characteristic is to offer the first theoretically provable performance under very weak assumptions. It is our opinion

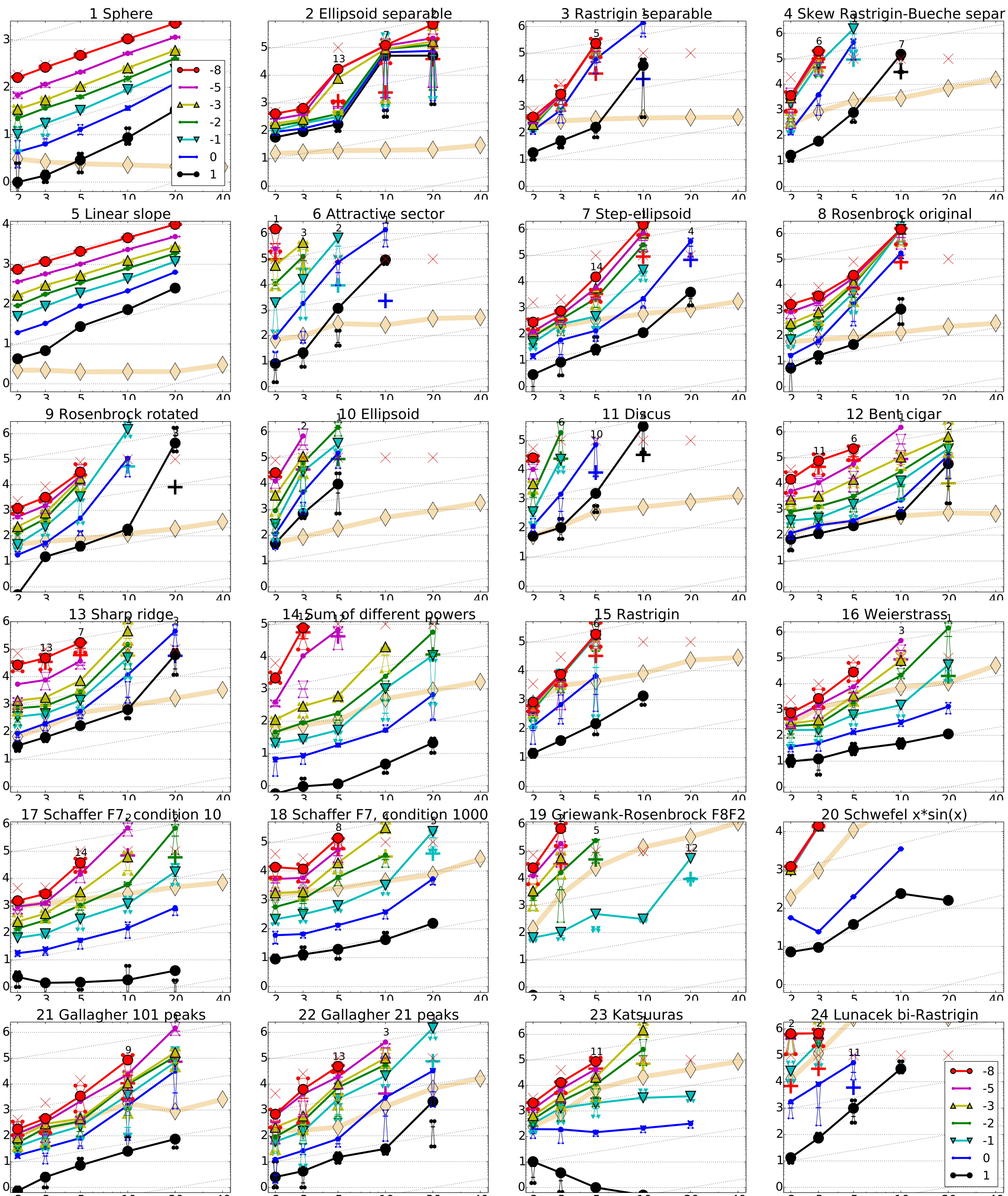

Fig. 1. Expected number of f-evaluations (ERT, lines) to reach $f_{\mathrm{opt}} + \Delta f$; median number of f-evaluations (+) to reach the most difficult target that was reached not always but at least once; maximum number of f-evaluations in any trial (×); interquartile range with median (notched boxes) of simulated runlengths to reach $f_{\mathrm{opt}} + \Delta f$; all values are divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown are $\Delta f = 10^{\{1,0,-1,-2,-3,-5,-8\}}$. Numbers above ERT-symbols (if appearing) indicate the number of trials reaching the respective target. The light thick line with diamonds indicates the respective best result from BBOB-2009 for $\Delta f = 10^{-8}$. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling.

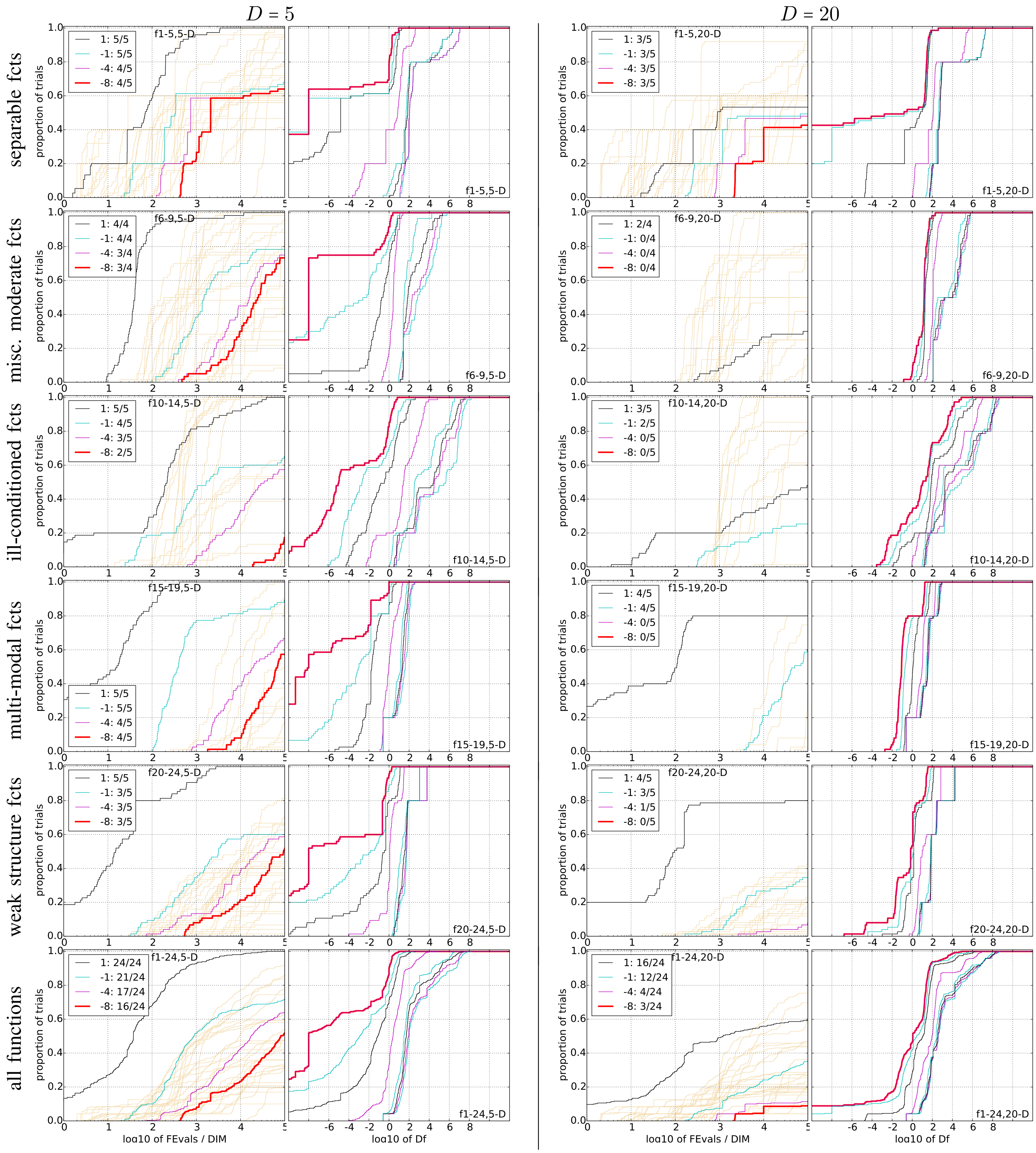

Fig. 2. Empirical cumulative distribution functions (ECDF), plotting the fraction of trials with an outcome not larger than the respective value on the x-axis. Left subplots: ECDF of the number of function evaluations (FEvals) divided by search space dimension D, to fall below $f_{\rm opt} + \Delta f$ with $\Delta f = 10^k$, where k is the first value in the legend. The thick red line represents the most difficult target value $f_{\rm opt} + 10^{-8}$. Legends indicate for each target the number of functions that were solved in at least one trial within the displayed budget.Right subplots: ECDF of the best achieved Δf for running times of $0.5D, 1.2D, 3D, 10D, 100D, 1000D, \ldots$ function evaluations (from right to left cycling cyan-magenta-black…) and final Δf-value (red), where Δf and Df denote the difference to the optimal function value. Light brown lines in the background show ECDFs for the most difficult target of all algorithms benchmarked during BBOB-2009.

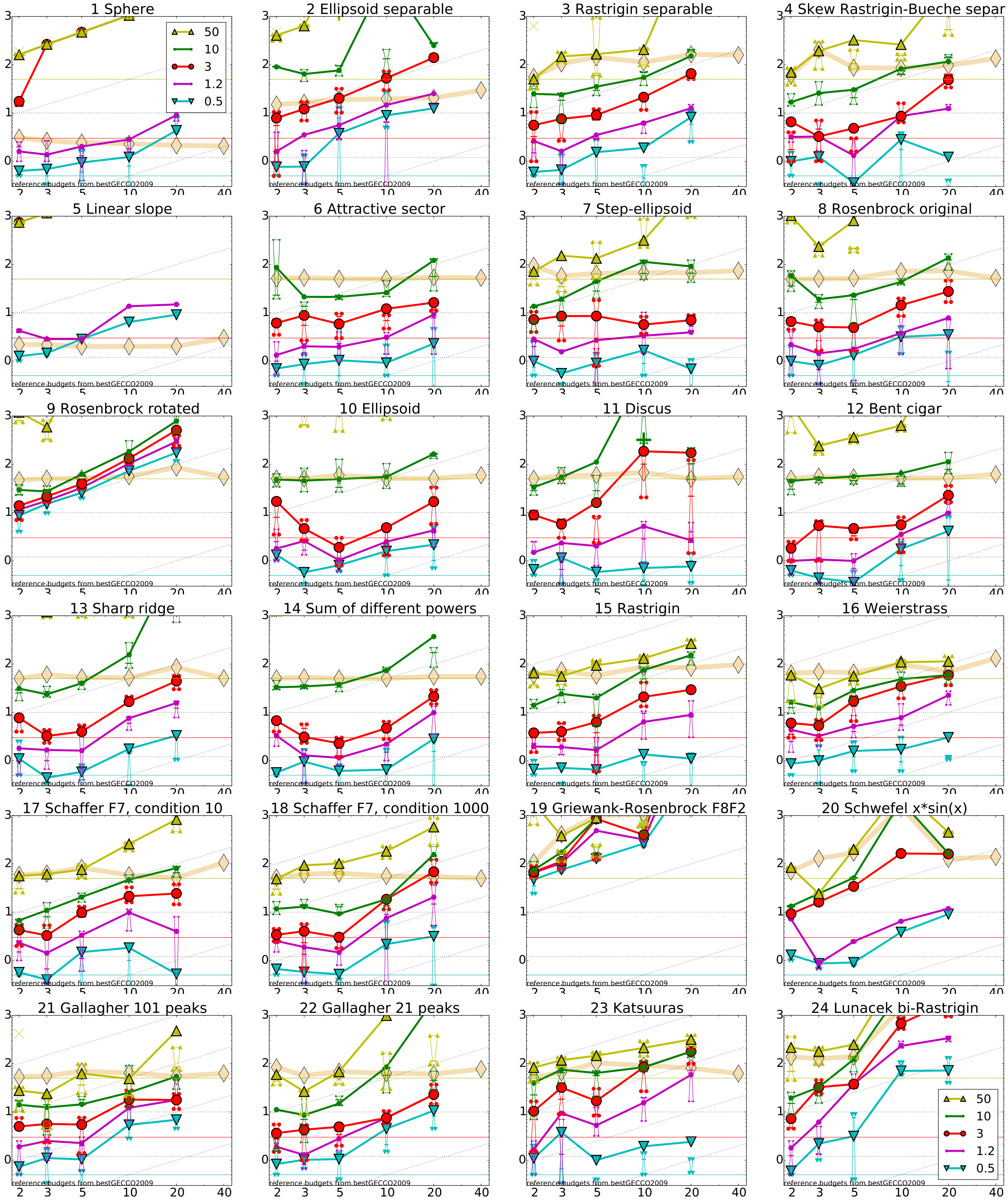

Fig. 3. Expected number of f-evaluations (ERT, lines) to reach $f_{\mathrm{opt}} + \Delta f$; median number of f-evaluations (+) to reach the most difficult target that was reached not always but at least once; maximum number of f-evaluations in any trial (×); interquartile range with median (notched boxes) of simulated runlengths to reach $f_{\mathrm{opt}} + \Delta f$; all values are divided by dimension and plotted as $\log_{10}$ values versus dimension. Shown is the ERT for targets just not reached by the artificial GECCO-BBOB-2009 best algorithm within the given budget $k \times \mathrm{DIM}$, where k is shown in the legend. Numbers above ERT-symbols indicate the number of trials reaching the respective target. The light thick line with diamonds indicates the respective best result from BBOB-2009 for the most difficult target. Slanted grid lines indicate a scaling with $\mathcal{O}(\mathrm{DIM})$ compared to $\mathcal{O}(1)$ when using the respective 2009 best algorithm.

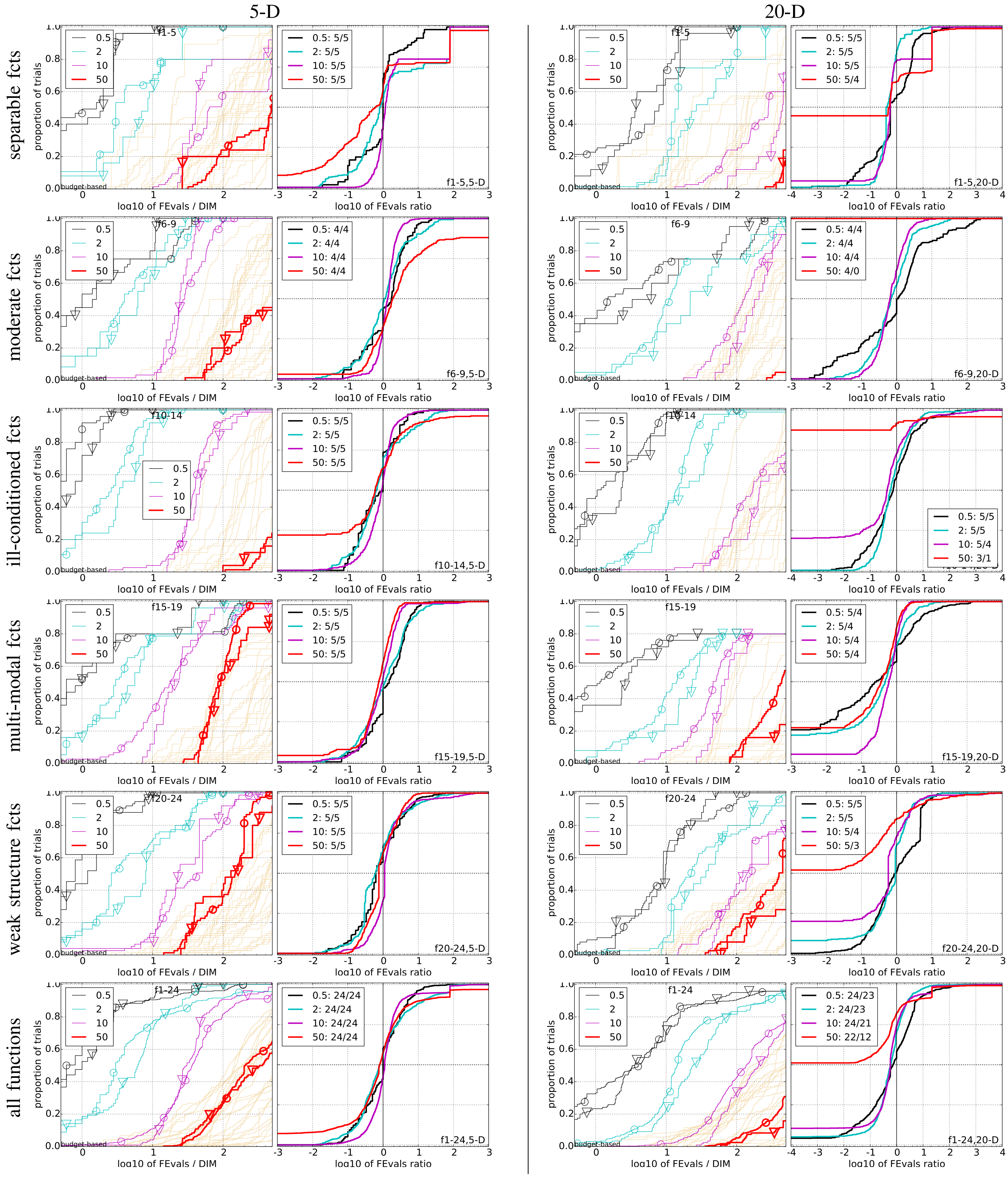

Fig. 4. Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to fall below $f_{\mathrm{opt}} + \Delta f$ for SOO ($\circ$) and DIRECT ($\triangledown$) where Δf is the target just not reached by the GECCO-BBOB-2009 best algorithm within a budget of $k \times \mathrm{DIM}$ evaluations, with k being the value in the legend. Right sub-columns: ECDF of FEval ratios of SOO divided by DIRECT for run-length-based targets; all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the target budget of $k \times \mathrm{DIM}$ evaluations and, after the colon, the number of functions that were solved in at least one trial (SOO first).

5-D

Δf	1e+1	1e+0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	12	12	15/15
	1.3(0.5)	5.3(2)	13(2)	26(3)	43(2)	86(7)	156(7)	15/15
f_2	83	87	88	89	90	92	94	15/15
	10(3)	14(3)	18(3)	23(6)	425(3)	874(1361)	867(1329)	13/15
f_3	716	1622	1637	1642	1646	1650	1654	15/15
	1.2(2)	185(229)	698(610)	696(1239)	695(440)	694(814)	693(1186)	5/15
f_4	809	1633	1688	1758	1817	1886	1903	15/15
	4.9(2)	1358(2039)	4424(6296)	∞	∞	∞	∞*5.0e5*	0/15
f_5	10	10	10	10	10	10	10	15/15
	14(0.0)	45(0.0)	95(0.1)	172(0.1)	263(0.1)	505(0.1)	843(0.1)	15/15
f_6	114	214	281	404	580	1038	1332	15/15
	52(93)	1740(1456)	11726(21787)	∞	∞	∞	∞*5.0e5*	0/15
f_7	24	324	1171	1451	1572	1572	1597	15/15
	5.7(8)	2.1(1)	2.0(3)	7.8(3)	18(27)	18(37)	27(26)	14/15
f_8	73	273	336	372	391	410	422	15/15
	3.1(0.7)	32(20)	71(70)	133(87)	150(93)	220(310)	255(432)	15/15
f_9	35	127	214	263	300	335	369	15/15
	5.7(3)	19(5)	81(164)	242(108)	297(279)	326(280)	413(199)	15/15
f_{10}	349	500	574	607	626	829	880	15/15
	136(140)	1507(1974)	3125(1275)	12256(3090)	∞	∞	∞*5.0e5*	0/15
f_{11}	143	202	763	977	1177	1467	1673	15/15
	54(155)	1762(5042)	∞	∞	∞	∞	∞*5.0e5*	0/15
f_{12}	108	268	371	413	461	1303	1494	15/15
	11(2)	6.8(1)	21(43)	39(47)	153(174)	220(219)	587(271)	6/15
f_{13}	132	195	250	319	1310	1752	2255	15/15
	6.3(2)	14(13)	28(28)	43(41)	27(34)	105(102)	298(391)	7/15
f_{14}	10	41	58	90	139	251	476	15/15
	0.59(0.3)	2.2(0.6)	4.5(5)	10(10)	22(25)	1342(1490)	7484(8237)	0/15
f_{15}	511	9310	19369	19743	20073	20769	21359	14/15
	1.4(1)	3.5(6)	40(14)	46(38)	46(63)	44(76)	43(37)	6/15
f_{16}	120	612	2662	10163	10449	11644	12095	15/15
	1.2(0.7)	1.1(0.6)	1.2(0.4)	0.96(0.4)	1.6(2)	3.2(4)	7.7(8)	15/15
f_{17}	5.2	215	899	2861	3669	6351	7934	15/15
	1.4(3)	1.2(0.5)	1.7(0.7)	1.8(0.3)	4.4(12)	11(12)	19(16)	14/15
f_{18}	103	378	3968	8451	9280	10905	12469	15/15
	0.95(0.5)	1.8(0.5)	0.80(0.9)	3.6(6)	10(4)	25(22)	50(25)	8/15
f_{19}	1	1	242	1.0e5	1.2e5	1.2e5	1.2e5	15/15
	1	1	10(0.9)	12(13)	∞	∞	∞*5.0e5*	0/15
f_{20}	16	851	38111	51362	54470	54861	55313	14/15
	12(0.0)	1.2(0.0)	∞	∞	∞	∞	∞*5.0e5*	0/15
f_{21}	41	1157	1674	1692	1705	1729	1757	14/15
	0.88(0.5)	0.35(0.2)	0.70(0.9)	1.1(0.3)	1.4(0.3)	6.2(14)	8.5(14)	15/15
f_{22}	71	386	938	980	1008	1040	1068	14/15
	1.0(0.7)	0.98(0.3)	14(19)	37(77)	50(49)	92(78)	205(263)	13/15
f_{23}	3.0	518	14249	27890	31654	33030	34256	15/15
	1.7(3)	1.4(0.8)	0.71(0.8)	1.9(4)	3.3(8)	6.5(11)	10(10)	11/15
f_{24}	1622	2.2e5	6.4e6	9.6e6	9.6e6	1.3e7	1.3e7	3/15
	3.0(2)	1.2(1)	∞	∞	∞	∞	∞*5.0e5*	0/15

20-D

Δf	1e+1	1e+0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	43	43	15/15
	15(6)	56(7)	111(16)	189(16)	279(15)	533(25)	847(21)	15/15
f_2	385	386	387	388	390	391	393	15/15
	2648(5200)	3897(2602)	6485(10351)	6482(10316)	8388(11951)	11073(12772)	34956(1e5)	2/15
f_3	5066	7626	7635	7637	7643	7646	7651	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_4	4722	7628	7666	7686	7700	7758	1.4e5	9/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_5	41	41	41	41	41	41	41	15/15
	124(0.0)	312(0.0)	579(0.0)	928(0.0)	1349(0.0)	2439(0.0)	4028(0.0)	15/15
f_6	1296	2343	3413	4255	5220	6728	8409	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_7	1351	4274	9503	16523	16524	16524	16969	15/15
	59(79)	1603(2174)	∞	∞	∞	∞	∞*2.0e6*	0/15
f_8	2039	3871	4040	4148	4219	4371	4484	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_9	1716	3102	3277	3379	3455	3594	3727	15/15
	5000(4147)	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{10}	7413	8661	10735	13641	14920	17073	17476	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{11}	1002	2228	6278	8586	9762	12285	14831	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{12}	1042	1938	2740	3156	4140	12407	13827	15/15
	1110(1610)	1284(2100)	1597(3747)	2610(2853)	3190(4606)	∞	∞*2.0e6*	0/15
f_{13}	652	2021	2751	3507	18749	24455	30201	15/15
	1927(2272)	4490(3011)	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{14}	75	239	304	451	932	1648	15661	15/15
	5.7(3)	55(88)	712(427)	2527(1182)	∞	∞	∞*2.0e6*	0/15
f_{15}	30378	1.5e5	3.1e5	3.2e5	3.2e5	4.5e5	4.6e5	15/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{16}	1384	27265	77015	1.4e5	1.9e5	2.0e5	2.2e5	15/15
	1.6(0.5)	0.96(0.4)	14(28)	204(197)	∞	∞	∞*2.0e6*	0/15
f_{17}	63	1030	4005	12242	30677	56288	80472	15/15
	1.3(2)	16(6)	87(34)	1159(1185)	∞	∞	∞*2.0e6*	0/15
f_{18}	621	3972	19561	28555	67569	1.3e5	1.5e5	15/15
	5.0(3)	27(36)	241(252)	∞	∞	∞	∞*2.0e6*	0/15
f_{19}	1	1	3.4e5	4.7e6	6.2e6	6.7e6	6.7e6	15/15
	1	1	3.2(6)	∞	∞	∞	∞*2.0e6*	0/15
f_{20}	82	46150	3.1e6	5.5e6	5.5e6	5.6e6	5.6e6	14/15
	39(0.0)	∞	∞	∞	∞	∞	∞*2.0e6*	0/15
f_{21}	561	6541	14103	14318	14643	15567	17589	15/15
	2.7(1)	98(233)	100(224)	162(179)	232(308)	1893(1831)	∞*2.0e6*	0/15
f_{22}	467	5580	23491	24163	24948	26847	1.3e5	12/15
	90(325)	117(180)	1257(1490)	∞	∞	∞	∞*2.0e6*	0/15
f_{23}	3.2	1614	67457	3.7e5	4.9e5	8.1e5	8.4e5	15/15
	1.6(2)	3.9(3)	1.1(1)	∞	∞	∞	∞*2.0e6*	0/15
f_{24}	1.3e6	7.5e6	5.2e7	5.2e7	5.2e7	5.2e7	5.2e7	3/15
	∞	∞	∞	∞	∞	∞	∞*2.0e6*	0/15

TABLE I

EXPECTED RUNNING TIME (ERT IN NUMBER OF FUNCTION EVALUATIONS) DIVIDED BY THE BEST ERT MEASURED DURING BBOB-2009. THE ERT AND IN BRACES, AS DISPERSION MEASURE, THE HALF DIFFERENCE BETWEEN 90 AND 10%-TILE OF BOOTSTRAPPED RUN LENGTHS APPEAR IN THE SECOND ROW OF EACH CELL, THE BEST ERT IN THE FIRST. THE DIFFERENT TARGET Δf-VALUES ARE SHOWN IN THE TOP ROW. #SUCC IS THE NUMBER OF TRIALS THAT REACHED THE (FINAL) TARGET $f_{\rm opt} + 10^{-8}$. THE MEDIAN NUMBER OF CONDUCTED FUNCTION EVALUATIONS IS ADDITIONALLY GIVEN IN *italics*, IF THE TARGET IN THE LAST COLUMN WAS NEVER REACHED. **BOLD** ENTRIES ARE STATISTICALLY SIGNIFICANTLY BETTER (ACCORDING TO THE RANK-SUM TEST) COMPARED TO THE BEST ALGORITHM IN BBOB-2009, WITH $p = 0.05$ OR $p = 10^{-k}$ WHEN THE NUMBER $k > 1$ IS FOLLOWING THE ↓ SYMBOL, WITH BONFERRONI CORRECTION BY THE NUMBER OF FUNCTIONS.

that SOO can be improved in order to perform much more efficiently in practice while still providing strong performance guarantees. For example, revisiting the SOO tree exploration heuristic by introducing more intensification when choosing the next cells to split could result in improved ERTs in the case of structured functions or functions with moderate multi-modality. Besides, it is worth noting that hybridizing SOO with local and stochastic search algorithms could be an interesting option; especially after observing that SOO is able to deterministically provide relatively good points using a small budget — such points could typically serve for initialization or restart purposes.

Notice also that in the case of noisy functions, a variant of SOO based on multi-armed bandits, and called StoSOO, is available with theoretically provable performance [11]. It would be particularly interesting to benchmark StoSOO using the noisy functions form the BBOB testbed and to analyze its behavior experimentally. This is on-going work.

REFERENCES

[1] R. Munos, "Optimistic optimization of a deterministic function without the knowledge of its smoothness," in *Advances in Neural Information Processing Systems*, 2011, pp. 783–791.

[2] P. Pošík, W. Huyer, and L. Pál, "A comparison of global search algorithms for continuous black box optimization," *Evol. Comput.*, vol. 20, no. 4, pp. 509–541, 2012.

[3] D. Jones, C. Perttunen, and B. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[4] P. Preux, R. Munos, and M. Valko, "Bandits attack function optimization," in *CEC*, 2014, pp. 2245–2252.

[5] P. Pošík, "Bbob-benchmarking the direct global optimization algorithm," in *GECCO Companion*, 2009, pp. 2315–2320.

[6] P. Preux, "yaStoSOO (yet another StoSOO) library (v0.3)," http://www.grappa.univ-lille3.fr/~ppreux/software/StoSOO/, 2014.

[7] M. Valko, A. Carpentier, and R. Munos, "Stochastic simultaneous optimistic optimization," in *the* 30^{th} *International Conference on Machine Learning (ICML)*, vol. 28, no. 2, 2013, pp. 19–27.

[8] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2012: Experimental setup," INRIA, Tech. Rep., 2012. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads

[9] S. Finck, N. Hansen, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions," Research Center PPE, Tech. Rep. 2009/20, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2010-downloads

[10] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Tech. Rep. RR-6829, 2009, updated February 2010. [Online]. Available: http://coco.gforge.inria.fr/bbob2012-downloads

[11] R. Munos, "From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning," *Foundations and Trends in Machine Learning*, vol. 7(1), pp. 1–130, 2014.

5-D

#FEs/D	0.5	1.2	3	10	50	#succ
$\mathbf{f_{1}}$	*2.5e+1*:4.8	*1.6e+1*:7.6	*1.0e-8*:12	*1.0e-8*:12	*1.0e-8*:12	15/15
1: SOO	0.99(0.8)	1.3(1)	194(16)	194(10)	194(16)	15/15
2: DIRECT	1(1)	2.2(2)	190(28)	190(27)	190(27)	5/5
$\mathbf{f_{2}}$	*1.6e+6*:2.9	*4.0e+5*:11	*4.0e+4*:15	*6.3e+2*:58	*1.0e-8*:95	15/15
1: SOO	6.6(2)	2.5(3)	6.7(5)	6.6(2)	872(1322)	13/15
2: DIRECT	5.2(9)	2.3(3)	5.6(4)	4.2(0.9)	456(347)	4/5
$\mathbf{f_{3}}$	*1.6e+2*:4.1	*1.0e+2*:15	*6.3e+1*:23	*2.5e+1*:73	*1.0e+1*:716	15/15
1: SOO	1.9(2)	1.2(0.5)	1.9(2)	2.4(2)	1.2(2)	15/15
2: DIRECT	2.4(1)	2.6(1)	3.1(1)	3.0(0.7)	45(106)	4/5
$\mathbf{f_{4}}$	*2.5e+2*:2.6	*1.6e+2*:10	*1.0e+2*:19	*4.0e+1*:65	*1.6e+1*:434	15/15
1: SOO	0.69(1)	0.68(0.7)	1.3(2)	2.3(1)	3.7(0.8)	15/15
2: DIRECT	1(2)	2.9(2)	2.5(0.7)	2.5(0.7)	52(89)	5/5
$\mathbf{f_{5}}$	*6.3e+1*:4.0	*4.0e+1*:10	*1.0e-8*:10	*1.0e-8*:10	*1.0e-8*:10	15/15
1: SOO	3.6	1.4	1054(0.1)	1054(0.1)	1054(0.1)	15/15
2: DIRECT	3.6(0.1)	3.1(0.1)	13(0.1)	13(0.1)	13(0.1)	5/5
$\mathbf{f_{6}}$	*1.0e+5*:3.0	*2.5e+4*:8.4	*1.0e+2*:16	*2.5e+1*:54	*2.5e-1*:254	15/15
1: SOO	1.7(2)	1.2(0.6)	1.8(2)	2.0(0.8)	6105(5271)	4/15
2: DIRECT	2.1(1)	1.7(1)	2.2(1)	1.5(0.4)	361(536)	3/5
$\mathbf{f_{7}}$	*1.6e+2*:4.2	*1.0e+2*:6.2	*2.5e+1*:20	*4.0e+0*:54	*1.0e+0*:324	15/15
1: SOO	1.1(1.0)	2.2(4)	2.1(2)	4.1(3)	2.1(2)	15/15
2: DIRECT	1.1(1)	1(1.0)	1.8(1)	2.0(0.6)	1.7(0.8)	5/5
$\mathbf{f_{8}}$	*1.0e+4*:4.6	*6.3e+3*:6.8	*1.0e+3*:18	*6.3e+1*:54	*1.6e+0*:258	15/15
1: SOO	1.4(2)	1.3(2)	1.4(1.0)	2.1(0.6)	16(37)	15/15
2: DIRECT	1(0.8)	1.1(0.5)	2.5(3)	2.5(1)	5.4(5)	5/5
$\mathbf{f_{9}}$	*2.5e+1*:20	*1.6e+1*:26	*1.0e+1*:35	*4.0e+0*:62	*1.6e-2*:256	15/15
1: SOO	6.5(3)	6.3(1.0)	5.7(1)	5.1(2)	202(65)	15/15
2: DIRECT	2.7(0.3)	3.4(2)	3.2(2)	3.0(0.5)	54(42)	5/5
$\mathbf{f_{10}}$	*2.5e+6*:2.9	*6.3e+5*:7.0	*2.5e+5*:17	*6.3e+3*:54	*2.5e+1*:297	15/15
1: SOO	1.4(1)	0.74(1)	0.56(0.6)	4.6(8)	74(51)	15/15
2: DIRECT	1.4(0.9)	1.3(1)	1.1(2)	4.6(3)	34(36)	5/5
$\mathbf{f_{11}}$	*1.0e+6*:3.0	*6.3e+4*:6.2	*6.3e+2*:16	*6.3e+1*:74	*6.3e-1*:298	15/15
1: SOO	0.98(2)	1.6(1)	5.0(9)	7.5(1)	2683(5084)	6/15
2: DIRECT	1.4(1)	1.4(0.9)	5.0(3)	7.4(6)	1510(2351)	1/5
$\mathbf{f_{12}}$	*4.0e+7*:3.6	*1.6e+7*:7.6	*4.0e+6*:19	*1.6e+4*:52	*1.0e+0*:268	15/15
1: SOO	0.50(0.6)	0.66(1)	1.2(0.8)	5.4(2)	6.8(1)	15/15
2: DIRECT	1.2(1)	1.1(0.8)	1.8(0.9)	4.2(0.4)	8.7(5)	5/5
$\mathbf{f_{13}}$	*1.0e+3*:2.8	*6.3e+2*:8.4	*4.0e+2*:17	*6.3e+1*:52	*6.3e-2*:264	15/15
1: SOO	1.0(2)	0.96(1)	1.2(0.7)	3.9(0.6)	29(18)	15/15
2: DIRECT	1.6(2)	2.1(2)	2.8(1)	5.3(2)	48(45)	5/5
$\mathbf{f_{14}}$	*1.6e+1*:3.0	*1.0e+1*:10	*6.3e+0*:15	*2.5e-1*:53	*1.0e-5*:251	15/15
1: SOO	1.0(1)	0.59(0.2)	0.74(0.8)	3.6(3)	1342(760)	12/15
2: DIRECT	1.9(2)	1(1.0)	1.0(1)	4.3(2)	1892(2587)	1/5
$\mathbf{f_{15}}$	*1.6e+2*:3.0	*1.0e+2*:13	*6.3e+1*:24	*4.0e+1*:55	*1.6e+1*:289	5/5
1: SOO	1.1(0.8)	0.64(0.4)	1.3(0.8)	1.8(1)	1.6(0.7)	15/15
2: DIRECT	1.3(1)	1.2(1)	1.7(1)	1.8(0.5)	1(0.1)	5/5
$\mathbf{f_{16}}$	*4.0e+1*:4.8	*2.5e+1*:16	*1.6e+1*:46	*1.0e+1*:120	*4.0e+0*:334	15/15
1: SOO	1.7(0.9)	1.6(1)	1.9(2)	1.2(0.6)	0.85(0.4)	15/15
2: DIRECT	1.2(1)	1.1(0.6)	1.2(1)	1.2(0.7)	2.2(0.9)	5/5
$\mathbf{f_{17}}$	*1.0e+1*:5.2	*6.3e+0*:26	*4.0e+0*:57	*2.5e+0*:110	*6.3e-1*:412	15/15
1: SOO	1.4(3)	0.64(0.6)	0.86(0.6)	0.93(0.3)	0.92(0.5)	15/15
2: DIRECT	1(1)	1.1(0.8)	1.2(0.4)	1.1(0.5)	1.0(0.7)	5/5
$\mathbf{f_{18}}$	*6.3e+1*:3.4	*4.0e+1*:7.2	*2.5e+1*:20	*1.6e+1*:58	*1.6e+0*:318	15/15
1: SOO	0.76(0.5)	1.0(0.8)	0.76(0.5)	0.79(1)	1.6(0.6)	15/15
2: DIRECT	1.1(0.3)	1(1)	1.0(1)	1.3(1)	2.1(0.7)	5/5
$\mathbf{f_{19}}$	*1.6e-1*:172	*1.0e-1*:242	*6.3e-2*:675	*4.0e-2*:3078	*2.5e-2*:4946	15/15
1: SOO	3.7(0.7)	10(0.9)	6.3(20)	1.4(0.1)	0.90(0.1)	15/15
2: DIRECT	1.0(0.2)	1.1(0.4)	1(0.1)	11(32)	6.7(1)	4/5
$\mathbf{f_{20}}$	*6.3e+3*:5.1	*4.0e+3*:8.4	*4.0e+1*:15	*2.5e+0*:69	*1.0e+0*:851	15/15
1: SOO	0.88(0.1)	1.5(0.1)	11(0.0)	3.7(0.0)	1.2(0.0)	15/15
2: DIRECT	1.6(0.1)	1(0.1)	4.0(0.0)	3.4(0.0)	1.5(0.0)	5/5
$\mathbf{f_{21}}$	*4.0e+1*:3.9	*2.5e+1*:11	*1.6e+1*:31	*6.3e+0*:73	*1.6e+0*:347	5/5
1: SOO	1.3(0.8)	1.1(0.6)	0.88(0.6)	0.99(0.9)	0.90(0.8)	15/15
2: DIRECT	2.6(5)	1.6(2)	1.0(0.8)	1(0.4)	1(0.7)	5/5
$\mathbf{f_{22}}$	*6.3e+1*:3.6	*4.0e+1*:15	*2.5e+1*:32	*1.0e+1*:71	*1.6e+0*:341	5/5
1: SOO	1.5(2)	0.94(0.6)	0.77(0.6)	1.0(0.6)	1.00(0.3)	15/15
2: DIRECT	1(1.0)	1.4(1)	1.1(1)	1(0.6)	1(0.6)	5/5
$\mathbf{f_{23}}$	*1.0e+1*:3.0	*6.3e+0*:9.0	*4.0e+0*:33	*2.5e+0*:84	*1.0e+0*:518	15/15
1: SOO	1.7(3)	2.9(3)	2.5(4)	3.8(3)	1.4(0.5)	15/15
2: DIRECT	1.5(0.8)	1.7(2)	2.0(3)	1.5(1)	3.5(3)	5/5
$\mathbf{f_{24}}$	*6.3e+1*:15	*4.0e+1*:37	*4.0e+1*:37	*2.5e+1*:118	*1.6e+1*:692	15/15
1: SOO	1.1(0.2)	5.1(2)	5.1(3)	5.2(3)	1.8(0.9)	15/15
2: DIRECT	1.3(1)	5.4(2)	5.4(2)	5.6(6)	2.0(2)	5/5

20-D

#FEs/D	0.5	1.2	3	10	50	#succ
$\mathbf{f_{1}}$	*6.3e+1*:24	*4.0e+1*:42	*1.0e-8*:43	*1.0e-8*:43	*1.0e-8*:43	15/15
1: SOO	3.6(3)	4.3(2)	1042(21)	1042(24)	1042(25)	15/15
2: DIRECT	7.9(7)	10(5)	2437(305)	2437(2061)	2437(1986)	4/5
$\mathbf{f_{2}}$	*4.0e+6*:29	*2.5e+6*:42	*1.0e+5*:65	*1.0e+4*:207	*1.0e-8*:412	15/15
1: SOO	8.8(5)	12(7)	43(10)	24(6)	33424(44934)	2/15
2: DIRECT	14(28)	21(15)	75(19)	43(7)	∞*1.0e5*	0/5
$\mathbf{f_{3}}$	*6.3e+2*:33	*4.0e+2*:44	*1.6e+2*:109	*1.0e+2*:255	*2.5e+1*:3277	15/15
1: SOO	5.0(3)	5.8(2)	12(5)	12(5)	234(655)	12/15
2: DIRECT	1.7(0.6)	8.0(2)	43(9)	28(4)	127(184)	1/5
$\mathbf{f_{4}}$	*6.3e+2*:22	*4.0e+2*:91	*2.5e+2*:250	*1.6e+2*:332	*6.3e+1*:1927	15/15
1: SOO	1.1(1)	2.7(1)	3.9(3)	7.0(3)	47(2)	15/15
2: DIRECT	1.1(0.4)	9.2(7)	10(5)	15(4)	212(260)	1/5
$\mathbf{f_{5}}$	*2.5e+2*:19	*1.6e+2*:34	*1.0e-8*:41	*1.0e-8*:41	*1.0e-8*:41	15/15
1: SOO	10(0.0)	8.7(0.0)	4928(0.0)	4928(0.0)	4928(0.0)	15/15
2: DIRECT	17(0.0)	27(0.0)	226(0.0)	226	226(0.0)	5/5
$\mathbf{f_{6}}$	*2.5e+5*:16	*6.3e+4*:43	*1.6e+4*:62	*1.6e+2*:353	*1.6e+1*:1078	15/15
1: SOO	2.9(5)	4.2(4)	5.3(2)	6.9(22)	12599(12308)	2/15
2: DIRECT	8.1(9)	10(9)	12(10)	9.1(5)	∞*1.0e5*	0/5
$\mathbf{f_{7}}$	*1.0e+3*:11	*4.0e+2*:39	*2.5e+2*:74	*6.3e+1*:319	*1.0e+1*:1351	15/15
1: SOO	1.3(3)	2.0(2)	1.9(1)	5.7(4)	59(86)	15/15
2: DIRECT	3.5(6)	3.1(3)	3.3(2)	5.6(2)	∞*1.2e5*	0/5
$\mathbf{f_{8}}$	*4.0e+4*:19	*2.5e+4*:35	*4.0e+3*:67	*2.5e+2*:231	*1.6e+1*:1470	15/15
1: SOO	3.8(5)	4.5(4)	8.2(7)	12(3)	2107(1702)	6/15
2: DIRECT	4.3(6)	5.4(4)	24(12)	22(6)	∞*1.0e5*	0/5
$\mathbf{f_{9}}$	*1.0e+2*:357	*6.3e+1*:560	*4.0e+1*:684	*2.5e+1*:756	*1.0e+1*:1716	15/15
1: SOO	10(3)	11(4)	15(6)	21(27)	5000(4372)	3/15
2: DIRECT	4.7(0.6)	8.8(6)	30(49)	79(55)	∞*1.0e5*	0/5
$\mathbf{f_{10}}$	*1.6e+6*:15	*1.0e+6*:27	*4.0e+5*:70	*6.3e+4*:231	*4.0e+3*:1015	15/15
1: SOO	2.9(4)	3.1(3)	4.8(2)	14(7)	1743(1477)	10/15
2: DIRECT	5.2(6)	8.1(11)	8.4(10)	21(24)	∞*1.0e5*	0/5
$\mathbf{f_{11}}$	*4.0e+4*:11	*2.5e+3*:27	*1.6e+2*:313	*1.0e+2*:481	*1.0e+1*:1002	15/15
1: SOO	1.4(2)	2.0(3)	11(2)	343(277)	∞*2.0e6*	0/15
2: DIRECT	2.4(2)	1.5(0.9)	1(0.6)	23(44)	∞*1.0e5*	0/5
$\mathbf{f_{12}}$	*1.0e+8*:23	*6.3e+7*:39	*2.5e+7*:76	*4.0e+6*:209	*1.0e+1*:1042	15/15
1: SOO	3.6(5)	5.0(5)	6.0(3)	11(9)	1110(1775)	10/15
2: DIRECT	5.0(6)	6.6(5)	13(7)	17(3)	421(457)	1/5
$\mathbf{f_{13}}$	*1.6e+3*:28	*1.0e+3*:64	*6.3e+2*:79	*4.0e+1*:211	*2.5e+0*:1724	15/15
1: SOO	2.4(2)	4.9(1)	11(5)	1339(2762)	5103(5321)	3/15
2: DIRECT	3.6(3)	15(5)	31(7)	∞	∞*1.0e5*	0/5
$\mathbf{f_{14}}$	*2.5e+1*:15	*1.6e+1*:42	*1.0e+1*:75	*1.6e+0*:219	*6.3e-4*:1106	15/15
1: SOO	3.9(5)	4.7(4)	5.7(4)	34(10)	∞*2.0e6*	0/15
2: DIRECT	2.1(1)	3.8(2)	8.4(4)	46(66)	∞*1.0e5*	0/5
$\mathbf{f_{15}}$	*6.3e+2*:15	*4.0e+2*:67	*2.5e+2*:292	*1.6e+2*:846	*1.0e+2*:1671	15/15
1: SOO	1.4(0.6)	2.6(2)	2.0(0.8)	3.6(1)	3.2(0.9)	15/15
2: DIRECT	1.9(2)	3.9(4)	4.4(2)	4.1(0.6)	109(134)	2/5
$\mathbf{f_{16}}$	*4.0e+1*:26	*2.5e+1*:127	*1.6e+1*:540	*1.6e+1*:540	*1.0e+1*:1384	15/15
1: SOO	2.3(2)	3.6(2)	2.2(0.9)	2.2(1)	1.6(0.3)	15/15
2: DIRECT	2.1(2)	4.1(0.4)	1.9(0.4)	1.9(0.3)	8.3(9)	5/5
$\mathbf{f_{17}}$	*1.6e+1*:11	*1.0e+1*:63	*6.3e+0*:305	*4.0e+0*:468	*1.0e+0*:1030	15/15
1: SOO	0.99(0.7)	1.3(3)	1.6(2)	3.5(2)	16(6)	15/15
2: DIRECT	3.7(4)	1.8(2)	3.1(1)	5.8(1)	55(64)	4/5
$\mathbf{f_{18}}$	*4.0e+1*:116	*2.5e+1*:252	*1.6e+1*:430	*1.0e+1*:621	*4.0e+0*:1090	15/15
1: SOO	0.54(0.4)	1.6(1)	3.2(3)	5.0(1)	11(8)	15/15
2: DIRECT	1.5(0.4)	3.1(2)	5.3(0.8)	9.1(2)	29(14)	5/5
$\mathbf{f_{19}}$	*1.6e-1*:2.5e5	*1.0e-1*:3.4e5	*6.3e-2*:3.4e5	*4.0e-2*:3.4e5	*2.5e-2*:3.4e5	3/15
1: SOO	0.55(0.1)	3.2(2)	26(24)	83(51)	∞*2.0e6*	0/15
2: DIRECT	∞	∞	∞	∞	∞*1.0e5*	0/5
$\mathbf{f_{20}}$	*1.6e+4*:38	*1.0e+4*:42	*2.5e+2*:62	*2.5e+0*:250	*1.6e+0*:2536	15/15
1: SOO	4.8(0.0)	5.6(0.0)	51(0.0)	13(0.0)	3.6(0.0)	15/15
2: DIRECT	1(0.0)	6.1(0.0)	37(0.0)	23(0.0)	∞*1.0e5*	0/5
$\mathbf{f_{21}}$	*6.3e+1*:36	*4.0e+1*:77	*4.0e+1*:77	*1.6e+1*:456	*4.0e+0*:1094	15/15
1: SOO	3.8(3)	4.6(3)	4.6(3)	2.4(1)	8.7(24)	15/15
2: DIRECT	6.4(3)	8.4(6)	8.4(6)	3.1(2)	34(92)	4/5
$\mathbf{f_{22}}$	*6.3e+1*:45	*4.0e+1*:68	*4.0e+1*:68	*1.6e+1*:231	*6.3e+0*:1219	15/15
1: SOO	4.7(4)	6.8(3)	6.8(4)	180(10)	69(133)	15/15
2: DIRECT	12(6)	14(8)	14(8)	14(18)	8.4(11)	5/5
$\mathbf{f_{23}}$	*6.3e+0*:29	*4.0e+0*:118	*2.5e+0*:306	*2.5e+0*:306	*1.0e+0*:1614	15/15
1: SOO	1.7(3)	10(11)	12(9)	12(9)	3.9(2)	15/15
2: DIRECT	4.1(3)	5.4(4)	5.3(4)	5.3(4)	52(65)	3/5
$\mathbf{f_{24}}$	*2.5e+2*:208	*1.6e+2*:918	*1.0e+2*:6628	*6.3e+1*:9885	*4.0e+1*:31629	15/15
1: SOO	6.9(4)	7.3(3)	5.6(6)	13(25)	20(23)	15/15
2: DIRECT	2.0(2)	76(164)	∞	∞	∞*1.0e5*	0/5

TABLE II

EXPECTED RUNNING TIME (ERT IN NUMBER OF FUNCTION EVALUATIONS) DIVIDED BY THE RESPECTIVE BEST ERT MEASURED DURING BBOB-2009 IN DIMENSIONS 5 (LEFT) AND 20 (RIGHT). THE ERT AND IN BRACES, AS DISPERSION MEASURE, THE HALF DIFFERENCE BETWEEN 90 AND 10%-TILE OF BOOTSTRAPPED RUN LENGTHS APPEAR FOR EACH ALGORITHM AND RUN-LENGTH BASED TARGET, THE CORRESPONDING BEST ERT (PRECEDED BY THE TARGET Δf-VALUE IN *italics*) IN THE FIRST ROW. #SUCC IS THE NUMBER OF TRIALS THAT REACHED THE TARGET VALUE OF THE LAST COLUMN. THE MEDIAN NUMBER OF CONDUCTED FUNCTION EVALUATIONS IS ADDITIONALLY GIVEN IN *italics*, IF THE TARGET IN THE LAST COLUMN WAS NEVER REACHED. BOLD ENTRIES ARE STATISTICALLY SIGNIFICANTLY BETTER COMPARED TO THE OTHER ALGORITHM, WITH $p = 0.05$ OR $p = 10^{-k}$ WHERE $k \in \{2, 3, 4, \dots\}$ IS THE NUMBER FOLLOWING THE $\star$ SYMBOL, WITH BONFERRONI CORRECTION OF 48. A $\downarrow$ INDICATES THE SAME TESTED AGAINST THE BEST ALGORITHM OF BBOB-2009.