Benchmarking of the Indicator-based evolution algorithm using the Bi-Objective BBOB Test Suite

Final report *

Daro Heng Karim Kouki Ahmed Mazari
Mihaela Sorostinean Aris Tritas

ABSTRACT

The objective of this project was to study, implement and benchmark the Indicator Based Evolutionary Algorithm (IBEA) using the Comparing Continuous Optimizer (COCO) platform. Firstly we provide a brief overview of the algorithm. Secondly, we describe our implementation and experimental setup. Finally, we discuss the obtained results and compare them to both baseline approaches and related work.

1. SETTING

In the context of a multi-objective optimization, the main goal is to find a good approximation of the set of Pareto-optimal solutions. An evolutionary algorithm is an exploration strategy of the domain space $\mathbb{R}^n$ which seeks optimal solution vectors defined in the objective space $\mathbb{R}^k$ (here $k = 2$).

The performance measure used by IBEA for determining the relative quality of two solution sets in the objective space is a binary indicator function $I : \Omega \times \Omega \to \mathbb{R}$. In terms of two decision vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, the *domination* relation is defined as : $\boldsymbol{x}^1 \succ \boldsymbol{x}^2 \leftrightarrow (f_i(\boldsymbol{x}^1) \leq f_i(\boldsymbol{x}^2)\ \forall i \in \{1, ..., n\}$ and $f_j(\boldsymbol{x}^1) < f_j(\boldsymbol{x}^2)$ for at least one objective). The epsilon indicator is compliant with this dominance relationship and defined for pairwise comparisons as:

$$I_{\epsilon^+}(\{\boldsymbol{x}^1\}, \{\boldsymbol{x}^2\}) = \min_{\epsilon} f_i(\boldsymbol{x}^1) - \epsilon \leq f_i(\boldsymbol{x}^2)\ \forall i \in \{1, ..., n\}$$

. It reduces the minimum distance to improve on the Pareto set approximation to a single scalar. Intuitively, this may incur a loss of information with respect to each dimension of the multi-objective optimization. Indeed, we are required to choose the minimum distance of improvement across all objectives. However, the fact that we choose to *conservatively* improve on the fitness estimate means that we may miss on potential improvement on one or more objectives.

*Submission deadline: October 21st.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The fitness value is defined in the sequel as a measure of the usefulness of each individual with regards to the optimization goal. As such, the algorithm tries to maximize it.

$$F(\boldsymbol{x}^1) = \sum_{\boldsymbol{x}^2 \in P \setminus \{\boldsymbol{x}^1\}} -\exp \frac{I(\{\boldsymbol{x}^1\}, \{\boldsymbol{x}^2\})}{\kappa}$$

where P is the population set, and $\boldsymbol{x} \in \mathbb{R}^n$. The fitness function of an approximation set is defined as a *dominance preserving relation.*

The domain space typically has axes with different scales. Therefore, Adaptive-IBEA scales the objective function values as well as the range of values taken by the indicator function. This scheme decreases the need parameter tuning in face of problem and indicator function diversity.

2. IMPLEMENTATION

The input of the algorithm is the size of the population (α), a maximum number of generations (corresponding to the budget) set as termination criterion and a fitness scaling factor (κ). The output of the algorithm is an approximation of the Pareto-set.

1. **Initialization:** generate an initial population P of given size uniformly between the lower and upper bound of the domain space.
2. **Fitness assignment:** compute and assign a fitness value to each individual in P.
3. **Environmental selection:** detect and remove individuals which have the smallest fitness values from the population until the current size of the population P does not exceed α. Update the fitness values for all remaining individuals.
4. **Termination:** after the environmental selection is performed, the termination criterion of the algorithm is checked; if the maximum number of generations is reached or another termination criterion is met, the algorithm returns the set of decision vectors A.
5. **Mating Selection:** consists in creating as temporary mating pool P' which is filled with individuals from P by performing binary tournament selection with replacement on P.
6. **Variation:** finally the variation step consists applying recombination and mutation operators to the previously created mating pool. The offspring resulting

from variation is added to P and the generation counter is incremented. The algorithm is then performed again from step 2, until a termination criterion is met.

The Python implementation keeps all population data (vector, objective and fitness value) in a dictionary for optimal access time. All numerical computation is done with NumPy. The main algorithm algorithms were inlined in a single optimization function, except for the recombination and mutation operators which were implemented in separate functions to achieve modularity during testing. The optimization function does not use multiple cores. However, run-level parallelism is achieved

Budget pitfall

Using too small a budget was not caught before the intermediate report. As such, benchmarking was substantially delayed.

3. EXPERIMENTAL SETUP

3.1 La galere

For the first part of the project our milestone was to have a working version of the algorithm, and potentially reasonable results on at least some groups of functions.

Without further changes to the original strategy, our algorithm does not seem to reach the optimal Pareto set of solution vectors. We tried applying recombination and mutation with high and low probabilities respectively, following the literature. This did not yield improved results, which leads us to think that either our implementation lacks robustness, or has a subtle bug.

We intended to:

1. Experiment with self-adaptive strategies for the mutation step, and
2. Implement Self-Adaptive SBX which seeks an optimal index for the approximation distribution used

3.2 Setup

The proposed implementation of the IBEA algorithm was tested with the collection of benchmark problems comprised in the COCO platform. We performed several experiments with various combinations of parameters in order to evaluate the influence of specific parameters on the results for different categories of benchmark problems. Therefore we varied the following parameters:

- The budget - various values between 300, 500, 750, 850, 1000
- Population size - varied between 50, 60, 80, 100, 200
- Number of offspring - 20, 30 or 40
- Mutation probability low (0.0, 0.05, 0.1) and high (0.7, 0.8, 0.9, 1.0)
- Crossover probability - 0.4, 0.6, 0.7, 0.8, 0.9, 1.0
- Variance - 1.3 , 2.5, 5, 10, 15
- Offspring : - 20, 30, 35, 40
- Dimension : 2, 5, 10, 20, 40
- Max iterations: 10^6, 10^9
- Distribution index for the SBX operator: 2, 5 or 20 (the value suggested by the authors)
- Operators : Isotropic, Derandomized

Since the number of variable parameters was considerable, carrying experiments with all combinations between them was impossible, so we selected a limited number of combinations which we considered might give significant results. The sets of parameters that we used in our experiments are illustrated in table 1. Running a bi-objective algorithm in a laptop is not an easy task. It's both time and RAM consuming. The execution takes from 5 to 23 hours to finish with a budget of 1000. The timing experiment is intrinsically associated with the size of the population and the dimension. For that reason, we tried only one configuration for 40 dimensions that took almost 48h with a moderate budget of 300. In order to try several parameterizations of the algorithm we were obliged to make non stopping running of the algorithm on different machine with different parameter in order to evaluate how the algorithm performs under different parameterizations. We conclude from this empirical study that a high variance and low variance impacts negatively the performance of the algorithm. The trick is to find a trade-off between the size of population, variance and the recombination values.

3.3 Recombination

The operator we used were intermediate weighting, discrete recombination and finally, the Simulated Binary Crossover [2]. However, results using these operators were not encouraging. The reason for that may be a rather naive exploration of the search space.

In this case, the Simulated Binary Operator was chosen for the recombination part.

3.4 Variation

For the mutation step, we simply added isotropic Gaussian noise with fixed variance to the produced offspring. However, it is well known that fixing variance does not speed up search optimally. a polynomial distribution was used for the mutation part.

4. CPU TIMING

In order to evaluate the CPU timing of the algorithm, we have run the IBEAwith restarts on the entire bbob-biobj test suite [5] for $2D$ function evaluations. The Python code was run on a Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz with 4 processors and 8 cores. The time per function evaluation for dimensions 2, 3, 5, 10, 20 equals $1.7e{-}3$, $1.9e{-}3$, $2.0e{-}3$, $2.2e{-}3$, $2.4e{-}3$ seconds respectively.

5. RESULTS

Results of IBEA from experiments according to [4] and [1] on the benchmark functions given in [5] are presented in Figures 1, 2, **??**, and **??**, and in Table **??**. The experiments were performed with COCO [3], version 1.0.1, the plots were produced with version 1.0.4.

5.1 Comparison of isotropic and derandomized mutation operators

Derandomized mutation performs better on the following functions: (initial variance is 5, mutation prob is 0.8, crossover is SBX-5 and Pr(0.8))

- Rosenbrock function 28.
- Function 23
- Much better on functions 26 & 34
- Function 2

By using α = 80 and offsprings 30, better results were obtained for the following functions:

- function 13 (as well as function 12)
- function 16
- function 19
- 2-moderate 4-multi-modal: no good results in function group overall except for a slight improvement with this configuration on function 25

Function 15: Isotropic : Variance of 5 better than variance of 10, Derandomized: Smaller population as good as isotropic with small variance

1-separable 2-moderate (3d)

: low mutation (0.1) is better for functions 12 and 13 and/or high recombination probability (>=0.8) is bad (why?)
smaller initial variance (3) is better for f13

1-separable 4-multi-modal

: high variance (10) is bad

1-separable 5-weakly-structured

function 18: High recombination probability (0.7-0.9) High mutation probability (around 0.7-0.8) (along with isotropic pop 100) High variance is bad

2-moderate 2-moderate

Variance is a critical factor. Mutation with probability 1 without adaptation of the variance gives worse results. Function 20 gives good results with mut0.3 in 5d - best mut0.3 recomb0.7 28-2d as good for pop100 offs20 mut0.1 recomb0.7 var5.0 as for pop100 offs20 mut0.8 recomb0.6

2-moderate 5-weakly-structured

Better results with low mutation probability (0.1-0.3) Function 33: good results with relatively higher variance. Improved for a smaller population and moderate variance 26, 33: best with mut0.8 recomb0.6

3-ill-conditioned 3-ill-conditioned

variance is not the determinating factor. Isotropic a little worse.
Function 42: Derandomized and/or Higher population are better than pop=60

3-ill-conditioned 5-weakly-structured

High variance doesn't hurt w.r.t a smaller one. - Can't say much though because the problems are unstructured and hard) f44-5d: mut0.8 recomb0.7 var5.0derandomized sbx2

4-multi-modal 4-multi-modal

Derandomized improves a tiny bit on isotropic.

4-multi-modal 5-weakly-structured

Function 48: high variance does not perform worse, on other functions of derandomized is better
Functions 54, 55: Can't say much. Derandomized/Smaller variance better.
Cases in which low mutation probability works: 2-moderate 5-weakly-structured: best is 0.3 3-ill-conditioned 5-weakly-structured: function 44 has good results
For 5-d space: Function 53 performs very well with pop80, mut0.8, recombination1.0, derandomized On the other hand it performs well with low mutation probability (0.1-0.3) on 3d Find solutions really quick with var 3.

5.2 aRT

We observe with no surprise that our algorithm has a runtime of $1e{-}3$ iff the variance is adapted. gets to 10-3 on f12
We also observe that using a better indicator tends to find targets much faster.

6. RESULTS

Results from experiments according to [4], [**?**] and [**?**] on the benchmark functions given in [5] are presented in Figures 1, 2, 3 and **??** and in Tables 1 and **??**. The experiments were performed with COCO [3], version 1.0.1, the plots were produced with version 1.1.1.

The **average runtime (aRT)**, used in the tables, depends on a given quality indicator value, $I_{\text{target}} = I^{\text{ref}} + \Delta I$, and is computed over all relevant trials as the number of function evaluations executed during each trial while the best indicator value did not reach I_{target}, summed over all trials and divided by the number of trials that actually reached I_{target} [4, **?**]. **Statistical significance** is tested with the rank-sum test for a given target I_{target} using, for each trial, either the number of needed function evaluations to reach I_{target} (inverted and multiplied by -1), or, if the target was not reached, the best ΔI-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

7. RELATED WORK

Comparison with results stemming from the same approach was performed, and to an extent is it quite revealing. For lack of time, no comparison was done with the state-of-the-art. However, using the tools provided by COCO as well as the datasets shared by other groups, and after comparison with the implementation of IBEA-ϵ in the C language as well as results of groups studing IBEA-HV. We can safely say that the ϵ indicator may be overly simplistic for the wide range of optimization functions benchmarked.

Conclusion

Covariance matrix adaptation, different indicator function.

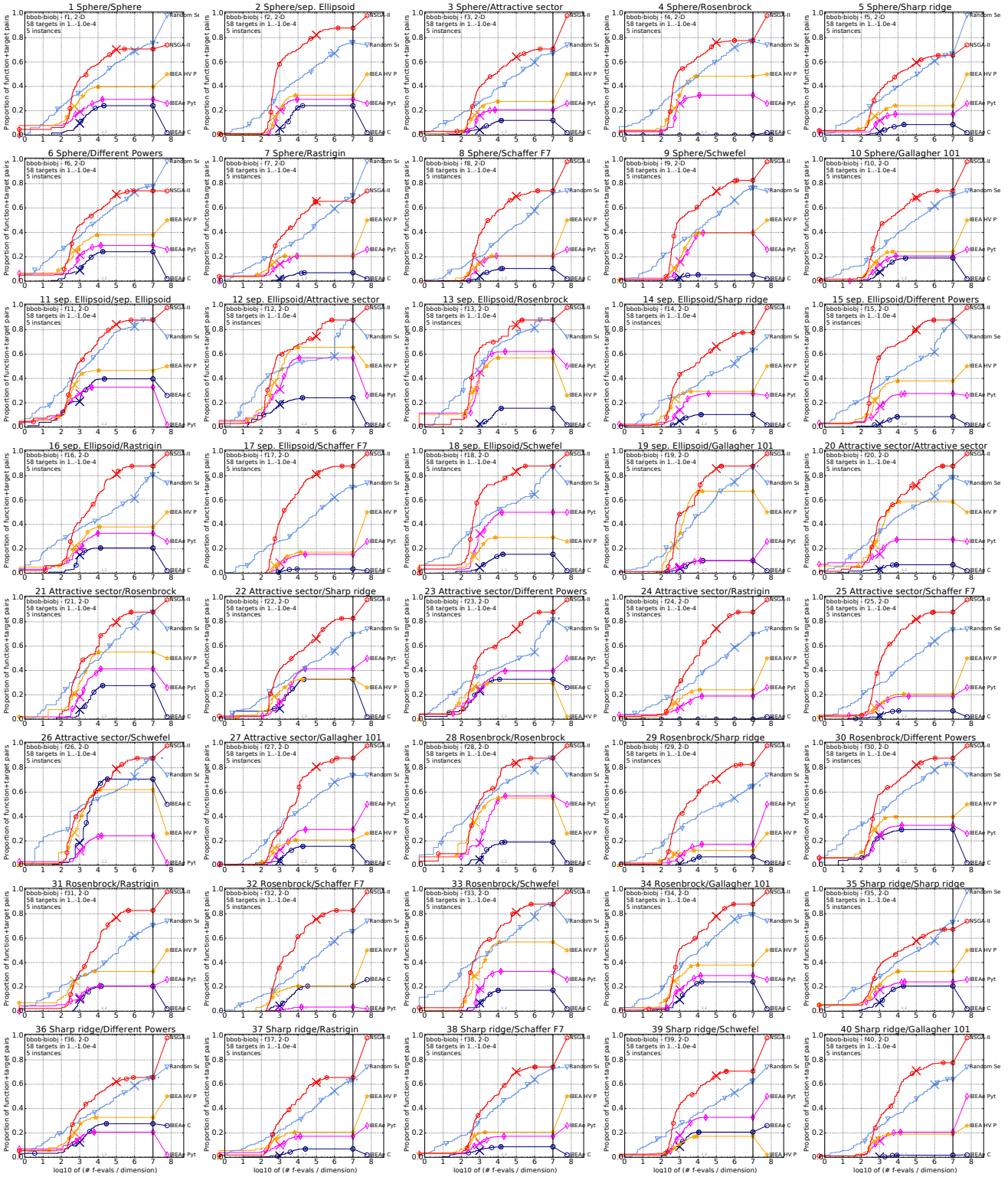

Figure 1: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for each single function f_1 to f_{40} in 10-D.

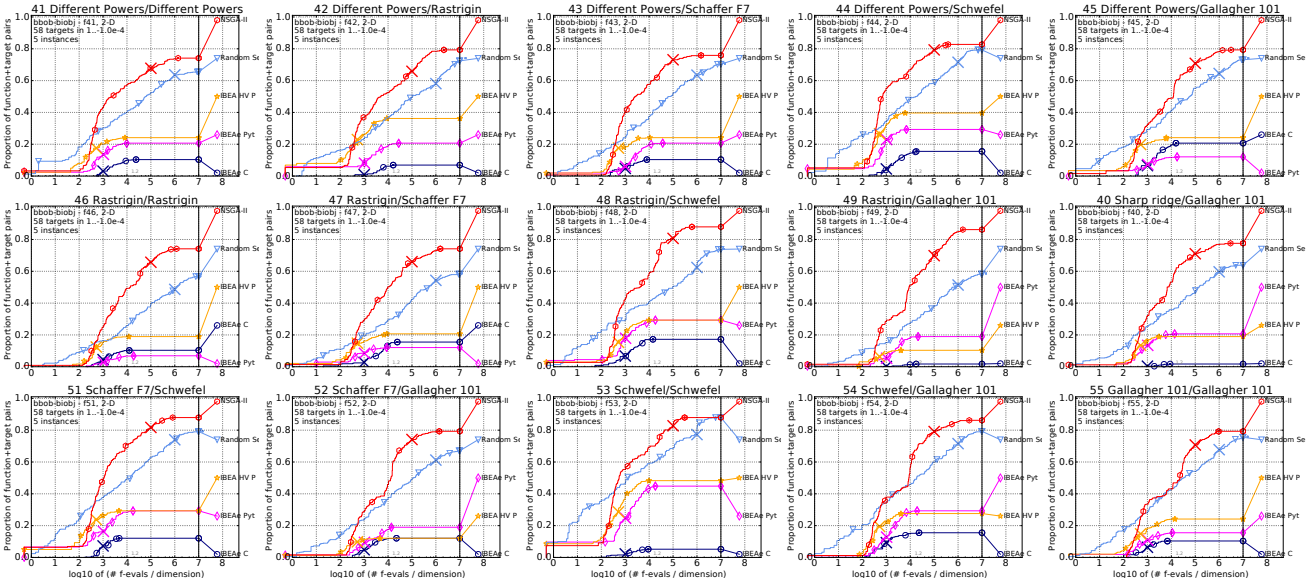

Figure 2: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) as in Fig. 1 but for functions f_{41} to f_{55} in 10-D.

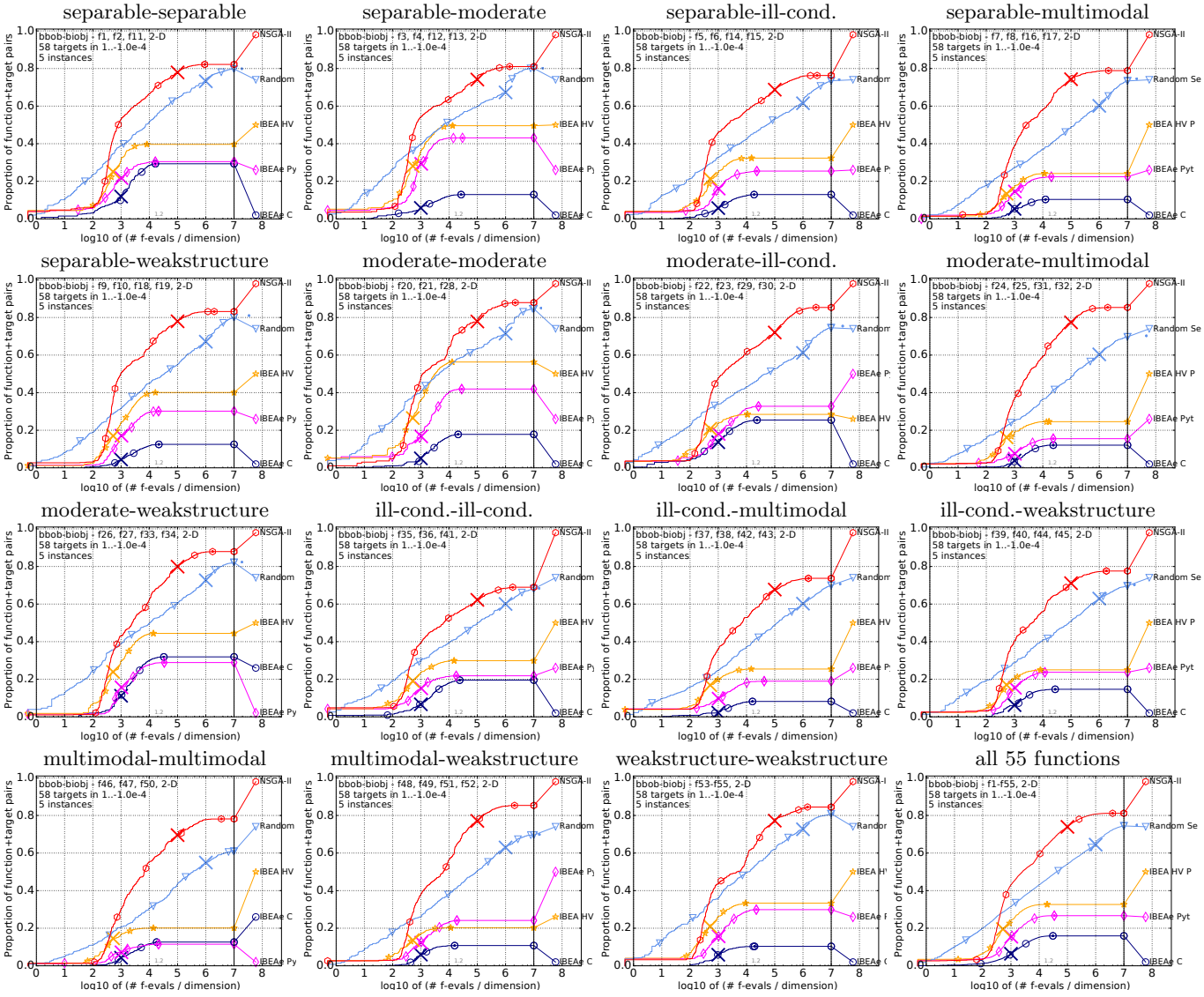

Figure 3: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for all functions and subgroups in 4-D.

8. REFERENCES

[1] D. Brockhoff, T. Tušar, D. Tušar, T. Wagner, N. Hansen, and A. Auger. Biobjective performance assessment with the coco platform. *arXiv preprint arXiv:1605.01746*, 2016.

[2] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(3):1–15, 1994.

[3] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *ArXiv e-prints*, arXiv:1603.08785, 2016.

[4] N. Hansen, T. Tusar, O. Mersmann, A. Auger, and D. Brockhoff. Coco: The experimental procedure. *arXiv preprint arXiv:1603.08776*, 2016.

[5] T. Tusar, D. Brockhoff, N. Hansen, and A. Auger. Coco: The bi-objective black box optimization benchmarking (bbob-biobj) test suite. *arXiv preprint arXiv:1604.00359*, 2016.

Δf	1e0	1e-1	1e-2	1e-3	#succ
f1					
IBEA	1226(457)	8440(5000)	∞	∞ *2000*	0/5
IBEA	**1**(0)	1650(554)	∞	∞ *2063*	0/5
IBEA	**1**(0)	1097(1126)	4827(2602)	∞ *1041*	0/5
Rand	4.4(6)	**76**(44)	2202(1709)	6.9e4(5e4)	0/5
NSGA	90(28)	547(58)	**847**(156)	**5343**(363)	0/5
f2					
IBEA	4518(2000)	9879(6000)	∞	∞ *2000*	0/5
IBEA	653(172)	2509(4674)	∞	∞ *2063*	0/5
IBEA	781(642)	1475(1301)	∞	∞ *1041*	0/5
Rand	**18**(28)	**409**(572)	5282(5659)	1.6e5(3e5)	0/5
NSGA	191(180)	596(56)	**882**(155)	**1526**(616)	2/5
f3					
IBEA	3128(4042)	∞	∞	∞ *2000*	0/5
IBEA	620(1755)	2880(2668)	∞	∞ *2063*	0/5
IBEA	314(312)	1311(2358)	∞	∞ *1041*	0/5
Rand	663(826)	1.1e4(3e4)	3.8e5(1683)	6.8e5(2e5)	0/5
NSGA	**191**(214)	**704**(202)	**1365**(345)	**2.2e4**(7873)	0/5
f4					
IBEA	∞	∞	∞	∞ *2000*	0/5
IBEA	367(411)	1326(295)	∞	∞ *2063*	0/5
IBEA	312(520)	1160(1404)	2119(1590)	∞ *1041*	0/5
Rand	**31**(0)	**194**(420)	1490(1372)	5.2e4(7e4)	0/5
NSGA	152(189)	562(132)	**808**(113)	**1795**(1003)	0/5
f5					
IBEA	2483(3144)	∞	∞	∞ *2000*	0/5
IBEA	378(628)	∞	∞	∞ *2063*	0/5
IBEA	36(44)	4774(4945)	∞	∞ *1041*	0/5
Rand	**5.6**(4)	**213**(182)	1.1e4(2714)	8.6e5(2e5)	0/5
NSGA	148(216)	595(80)	**1832**(1406)	**5.3e4**(3e4)	0/5
f6					
IBEA	1135(1646)	8695(7500)	∞	∞ *2000*	0/5
IBEA	**1**(0)	2510(2162)	∞	∞ *2063*	0/5
IBEA	**1**(0)	853(449)	5086(7547)	∞ *1041*	0/5
Rand	2.8(4)	**62**(52)	1440(1233)	4.4e4(6e4)	0/5
NSGA	71(133)	493(180)	**868**(308)	**7131**(3863)	0/5
f7					
IBEA	3763(5362)	∞	∞	∞ *2000*	0/5
IBEA	178(442)	4024(2012)	∞	∞ *2063*	0/5
IBEA	261(520)	1422(1390)	∞	∞ *1041*	0/5
Rand	**5.2**(5)	990(159)	4.5e4(4e4)	9.5e5(2e5)	0/5
NSGA	170(274)	**726**(317)	**3942**(5600)	**2.5e4**(1e4)	0/5
f8					
IBEA	3988(3500)	∞	∞	∞ *2000*	0/5
IBEA	583(572)	4764(2020)	∞	∞ *2063*	0/5
IBEA	512(580)	2225(781)	∞	∞ *1041*	0/5
Rand	**50**(96)	4980(8919)	9.4e4(1e5)	1.7e6(1e6)	0/5
NSGA	382(228)	**825**(173)	**1648**(183)	**9484**(3741)	0/5
f9					
IBEA	3940(2788)	∞	∞	∞ *2000*	0/5
IBEA	837(768)	4448(2855)	9453(1e4)	∞ *2063*	0/5
IBEA	605(687)	4561(8328)	5049(2863)	∞ *1041*	0/5
Rand	**21**(24)	**344**(296)	8842(1e4)	3.5e5(3e5)	0/5
NSGA	311(354)	678(192)	**923**(126)	**3555**(1952)	0/5
f10					
IBEA	880(2118)	9578(6500)	∞	∞ *2000*	0/5
IBEA	347(359)	4266(5331)	∞	∞ *2063*	0/5
IBEA	375(110)	1539(521)	∞	∞ *1041*	0/5
Rand	**7.2**(10)	**491**(452)	1.3e4(9389)	4.0e5(4e5)	0/5
NSGA	190(260)	606(166)	**1376**(455)	**9039**(6929)	0/5
f11					
IBEA	583(562)	1581(2000)	8628(3000)	∞ *2000*	0/5
IBEA	162(381)	726(651)	∞	∞ *2063*	0/5
IBEA	278(33)	533(1064)	900(804)	∞ *1041*	0/5
Rand	**17**(0.5)	**289**(697)	8054(9886)	1.8e5(5e5)	4/5
NSGA	107(264)	382(300)	**596**(317)	**1812**(703)	3/5
f12					
IBEA	322(360)	3550(7500)	∞	∞ *2000*	0/5
IBEA	302(752)	1646(2508)	3552(4747)	**4105**(5673)	0/5
IBEA	72(177)	494(456)	1897(2863)	4734(4684)	0/5
Rand	**2.0**(2)	5.0e5(2e6)	1.3e6(5e6)	1.3e6(2e6)	2/5
NSGA	126(274)	**467**(316)	**1162**(1025)	1.1e4(2e4)	2/5
f13					
IBEA	9712(8000)	∞	∞	∞ *2000*	0/5
IBEA	416(542)	1448(2318)	**1534**(890)	**2639**(1060)	0/5
IBEA	296(329)	774(686)	1966(1099)	5007(2082)	0/5
Rand	**92**(190)	1109(1372)	1.3e4(3e4)	2.9e5(7e5)	4/5
NSGA	242(287)	**514**(268)	1.5e4(4e4)	1.6e4(2e4)	4/5
f14					
IBEA	2161(3824)	∞	∞	∞ *2000*	0/5
IBEA	939(1428)	2680(5570)	∞	∞ *2063*	0/5
IBEA	150(158)	896(525)	∞	∞ *1041*	0/5
Rand	**6.2**(4)	**203**(249)	5.2e5(1e6)	1.9e6(4e6)	0/5
NSGA	139(158)	579(111)	**1209**(299)	**2.3e4**(3e4)	0/5
f15					
IBEA	2337(3166)	∞	∞	∞ *2000*	0/5
IBEA	632(886)	4082(1623)	∞	∞ *2063*	0/5
IBEA	349(412)	1270(2200)	4985(8328)	∞ *1041*	0/5
Rand	**16**(2)	**198**(222)	5.1e5(8406)	3.2e5(1e6)	1/5
NSGA	181(224)	483(156)	**791**(244)	**4111**(6926)	2/5
f16					
IBEA	624(440)	4598(3042)	∞	∞ *2000*	0/5
IBEA	33(34)	2055(3094)	∞	∞ *2063*	0/5
IBEA	98(114)	1328(2265)	5111(4684)	∞ *1041*	0/5
Rand	**5.6**(7)	**139**(241)	2850(2914)	7.3e5(1e6)	0/5
NSGA	54(98)	503(121)	**1372**(757)	**1.3e4**(2e4)	4/5
f17					
IBEA	4194(3739)	∞	∞	∞ *2000*	0/5
IBEA	1159(511)	∞	∞	∞ *2063*	0/5
IBEA	1179(1048)	∞	∞	∞ *1041*	0/5
Rand	**19**(3)	1185(1623)	2.9e4(3e4)	6.5e5(6e5)	0/5
NSGA	460(126)	**877**(300)	**1859**(1270)	**5376**(5994)	1/5
f18					
IBEA	3408(4154)	∞	∞	∞ *2000*	0/5
IBEA	390(292)	1115(652)	4274(6705)	∞ *2063*	0/5
IBEA	619(1399)	2048(1896)	∞	∞ *1041*	0/5
Rand	**15**(28)	**146**(306)	1449(1274)	1.4e6(2e6)	1/5
NSGA	170(224)	490(149)	**726**(145)	**1368**(345)	3/5
f19					
IBEA	3890(4000)	∞	∞	∞ *2000*	0/5
IBEA	2247(1692)	∞	∞	∞ *2063*	0/5
IBEA	441(347)	4397(4424)	**4524**(2863)	**4569**(4424)	0/5
Rand	**137**(330)	**851**(880)	5009(4836)	8.1e4(9e4)	1/5
NSGA	334(251)	1908(2540)	8294(9040)	9109(1e4)	4/5

Δf	1e0	1e-1	1e-2	1e-3	#succ
f20					
IBEA	3025(4508)	∞	∞	∞ *2000*	0/5
IBEA	517(516)	3435(4812)	∞	∞ *2063*	0/5
IBEA	404(520)	1688(3060)	4305(4164)	**4777**(8328)	0/5
Rand	**134**(140)	1837(2225)	1.2e5(3e5)	8.9e5(1e6)	0/5
NSGA	371(431)	**693**(537)	**1228**(311)	1.1e4(1e4)	1/5
f21					
IBEA	3694(5588)	9894(9500)	∞	∞ *2000*	0/5
IBEA	1713(3094)	2505(2047)	9554(2e4)	∞ *2063*	0/5
IBEA	309(190)	1988(1822)	**4752**(9109)	**4969**(3644)	0/5
Rand	**8.6**(6)	**227**(112)	7747(8586)	1.8e5(2e5)	2/5
NSGA	338(214)	5220(1e4)	5763(1e4)	1.5e4(3e4)	2/5
f22					
IBEA	1738(1064)	8006(1e4)	∞	∞ *2000*	0/5
IBEA	424(756)	9508(9799)	9881(1e4)	∞ *2063*	0/5
IBEA	89(80)	2262(2181)	∞	∞ *1041*	0/5
Rand	**3.6**(2)	**660**(431)	3.0e4(3e4)	3.2e6(4e6)	0/5
NSGA	242(189)	670(145)	**1697**(1170)	**3.6e4**(4e4)	0/5
f23					
IBEA	556(1994)	1328(858)	∞	∞ *2000*	0/5
IBEA	29(36)	1046(617)	9641(1e4)	∞ *2063*	0/5
IBEA	40(48)	310(386)	∞	∞ *1041*	0/5
Rand	**2.6**(2)	**57**(88)	1.3e6(2e6)	1.4e6(4e6)	0/5
NSGA	50(61)	423(160)	**857**(268)	**7093**(7127)	1/5
f24					
IBEA	∞	∞	∞	∞ *2000*	0/5
IBEA	615(1032)	9773(7736)	∞	∞ *2063*	0/5
IBEA	136(195)	5079(4164)	∞	∞ *1041*	0/5
Rand	**9.0**(14)	1264(1210)	2.5e4(2e4)	1.2e6(6e6)	0/5
NSGA	80(190)	**856**(204)	**4719**(4220)	**2.0e4**(2e4)	4/5
f25					
IBEA	1878(866)	∞	∞	∞ *2000*	0/5
IBEA	742(1042)	9595(7736)	∞	∞ *2063*	0/5
IBEA	281(380)	5197(6506)	∞	∞ *1041*	0/5
Rand	**6.0**(4)	**403**(394)	2.8e4(3e4)	5.7e5(9e5)	0/5
NSGA	190(220)	687(182)	**1350**(322)	**3026**(1604)	1/5
f26					
IBEA	3287(5723)	3848(5500)	9136(1e4)	9136(6000)	0/5
IBEA	1979(754)	4832(4568)	∞	∞ *2063*	0/5
IBEA	377(316)	1837(2478)	**2162**(2478)	**4729**(2863)	0/5
Rand	**12**(8)	**274**(329)	1.8e4(4e4)	3.0e5(4e5)	1/5
NSGA	327(252)	782(547)	6305(1e4)	1.3e4(3e4)	1/5
f27					
IBEA	3084(3504)	∞	∞	∞ *2000*	0/5
IBEA	606(716)	4558(5244)	∞	∞ *2063*	0/5
IBEA	166(184)	4968(5465)	∞	∞ *1041*	0/5
Rand	**18**(15)	**299**(250)	**5428**(6275)	1.2e5(9e4)	0/5
NSGA	215(212)	1094(956)	7403(9920)	**1.1e4**(1e4)	1/5
f28					
IBEA	3400(2500)	8707(4500)	∞	∞ *2000*	0/5
IBEA	1991(3610)	3192(6608)	8752(9799)	8949(1e4)	0/5
IBEA	221(296)	352(404)	1062(1220)	**4822**(2342)	0/5
Rand	**5.4**(8)	**64**(142)	1256(1684)	3.2e4(5e4)	3/5
NSGA	291(154)	502(214)	**851**(495)	4970(4814)	3/5
f29					
IBEA	9253(5500)	∞	∞	∞ *2000*	0/5
IBEA	693(1724)	∞	∞	∞ *2063*	0/5
IBEA	191(426)	∞	∞	∞ *1041*	0/5
Rand	**34**(80)	**219**(169)	1.5e4(2e4)	1.5e6(1e6)	0/5
NSGA	197(114)	616(171)	**1193**(354)	**1.5e4**(1e4)	0/5
f30					
IBEA	263(292)	832(968)	∞	∞ *2000*	0/5
IBEA	121(300)	1454(1117)	∞	∞ *2063*	0/5
IBEA	31(76)	704(531)	2284(1562)	∞ *1041*	0/5
Rand	**1.4**(0.5)	**13**(9)	**664**(1237)	2.6e4(4e4)	0/5
NSGA	119(148)	409(106)	757(113)	**1526**(500)	1/5
f31					
IBEA	1650(1271)	9945(1e4)	∞	∞ *2000*	0/5
IBEA	538(1053)	9503(8768)	∞	∞ *2063*	0/5
IBEA	76(62)	632(912)	∞	∞ *1041*	0/5
Rand	**2.4**(2)	**498**(568)	6.7e4(1e5)	3.3e5(3e5)	0/5
NSGA	241(262)	774(253)	**6499**(1e4)	**1.4e4**(1e4)	0/5
f32					
IBEA	3684(4105)	8965(5000)	∞	∞ *2000*	0/5
IBEA	4300(5016)	∞	∞	∞ *2063*	0/5
IBEA	160(124)	4855(5205)	∞	∞ *1041*	0/5
Rand	**28**(32)	2598(3418)	3.9e4(3e4)	1.7e6(2e6)	0/5
NSGA	433(68)	**877**(124)	**3874**(6062)	**7327**(4648)	0/5
f33					
IBEA	4186(3500)	∞	∞	∞ *2000*	0/5
IBEA	803(393)	2674(2646)	∞	∞ *2063*	0/5
IBEA	226(178)	554(337)	4862(2082)	5020(5205)	0/5
Rand	**3.0**(3)	**49**(73)	7563(9050)	2.9e5(1e4)	1/5
NSGA	297(200)	568(458)	**888**(950)	**4620**(4727)	2/5
f34					
IBEA	1960(4224)	8407(5000)	∞	∞ *2000*	0/5
IBEA	176(437)	2536(1688)	∞	∞ *2063*	0/5
IBEA	19(45)	884(1103)	5004(8068)	∞ *1041*	0/5
Rand	**4.0**(3)	**297**(253)	2059(1433)	3.6e4(2e4)	0/5
NSGA	161(210)	611(212)	**1138**(56)	**4375**(4103)	1/5
f35					
IBEA	1262(1706)	9135(8500)	∞	∞ *2000*	0/5
IBEA	29(35)	3781(2893)	∞	∞ *2063*	0/5
IBEA	74(182)	1218(1906)	∞	∞ *1041*	0/5
Rand	**3.4**(5)	**285**(292)	2.3e4(3e4)	1.7e6(2e6)	0/5
NSGA	111(151)	570(177)	**1929**(1378)	**7.0e4**(5e4)	0/5
f36					
IBEA	986(1500)	8520(9500)	∞	∞ *2000*	0/5
IBEA	292(728)	2232(5213)	∞	∞ *2063*	0/5
IBEA	261(781)	644(517)	∞	∞ *1041*	0/5
Rand	**4.2**(4)	**387**(446)	2.1e4(5e4)	6.8e5(2e6)	0/5
NSGA	107(265)	630(152)	**1372**(247)	**2.6e4**(2e4)	4/5
f37					
IBEA	3289(6267)	∞	∞	∞ *2000*	0/5
IBEA	283(542)	∞	∞	∞ *2063*	0/5
IBEA	94(49)	4652(8068)	∞	∞ *1041*	0/5
Rand	**2.0**(2)	2391(526)	8.4e4(2e4)	1.4e6(1e6)	0/5
NSGA	192(336)	**760**(243)	**3917**(3156)	**4.8e4**(3e4)	0/5

Δf	1e0	1e-1	1e-2	1e-3	#succ
f38					
IBEA	1382(1515)	∞	∞	∞ *2000*	0/5
IBEA	182(234)	∞	∞	∞ *2063*	0/5
IBEA	287(781)	1403(1401)	∞	∞ *1041*	0/5
Rand	**1.6**(0.5)	**379**(394)	8465(6593)	3.3e5(3e5)	0/5
NSGA	93(122)	676(22)	**1668**(378)	**2.1e4**(1e4)	0/5
f39					
IBEA	1808(1207)	3972(3244)	∞	∞ *2000*	0/5
IBEA	837(1119)	2397(1762)	∞	∞ *2063*	0/5
IBEA	805(603)	∞	∞	∞ *1041*	0/5
Rand	**18**(28)	**420**(299)	2.9e4(2e4)	3.8e6(4e6)	0/5
NSGA	264(268)	690(78)	**1431**(702)	**2.5e4**(3e4)	0/5
f40					
IBEA	9877(7000)	∞	∞	∞ *2000*	0/5
IBEA	677(454)	8943(7736)	∞	∞ *2063*	0/5
IBEA	177(61)	2406(2042)	∞	∞ *1041*	0/5
Rand	**2.0**(0.5)	**761**(539)	1.8e4(2e4)	5.6e5(6e5)	0/5
NSGA	279(245)	906(378)	**6671**(1e4)	**1.9e4**(1e4)	0/5
f41					
IBEA	4242(3606)	∞	∞	∞ *2000*	0/5
IBEA	522(930)	4158(4658)	∞	∞ *2063*	0/5
IBEA	198(298)	852(643)	∞	∞ *1041*	0/5
Rand	**3.0**(0.5)	**219**(58)	6677(2794)	2.4e5(1e5)	0/5
NSGA	246(208)	598(39)	**1090**(57)	**8817**(5514)	0/5
f42					
IBEA	8755(7000)	∞	∞	∞ *2000*	0/5
IBEA	517(1547)	9776(2e4)	∞	∞ *2063*	0/5
IBEA	68(166)	1140(1272)	5135(6506)	∞ *1041*	0/5
Rand	**6.2**(2)	2497(3062)	2.3e4(2e4)	8.8e5(3e5)	0/5
NSGA	107(152)	**677**(174)	**1857**(1102)	**2.5e4**(1e4)	0/5
f43					
IBEA	3510(5968)	∞	∞	∞ *2000*	0/5
IBEA	1376(4642)	9800(6189)	∞	∞ *2063*	0/5
IBEA	101(209)	4610(6766)	∞	∞ *1041*	0/5
Rand	**2.8**(2)	**573**(465)	1.6e4(1e4)	2.3e5(2e5)	0/5
NSGA	185(154)	715(123)	**1523**(440)	**9809**(9991)	0/5
f44					
IBEA	2168(3372)	∞	∞	∞ *2000*	0/5
IBEA	619(872)	2246(3249)	∞	∞ *2063*	0/5
IBEA	239(339)	614(855)	2312(3676)	∞ *1041*	0/5
Rand	**4.6**(3)	**357**(834)	3447(5184)	2.3e5(4e5)	0/5
NSGA	195(268)	532(138)	**849**(188)	**1627**(690)	0/5
f45					
IBEA	1394(1215)	9132(1e4)	∞	∞ *2000*	0/5
IBEA	245(360)	∞	∞	∞ *2063*	0/5
IBEA	52(78)	1371(1080)	∞	∞ *1041*	0/5
Rand	**7.8**(5)	**217**(151)	1.1e4(1e4)	3.3e5(4e5)	0/5
NSGA	152(171)	652(110)	**4602**(4401)	**2.1e4**(2e4)	0/5
f46					
IBEA	1996(4602)	∞	∞	∞ *2000*	0/5
IBEA	1643(2184)	∞	∞	∞ *2063*	0/5
IBEA	332(190)	5197(4424)	∞	∞ *1041*	0/5
Rand	**30**(11)	7120(1e4)	2.1e5(3e5)	3.8e6(6e6)	0/5
NSGA	248(225)	**1072**(325)	**6613**(4863)	**5.9e4**(6e4)	0/5
f47					
IBEA	3931(3892)	∞	∞	∞ *2000*	0/5
IBEA	675(1036)	∞	∞	∞ *2063*	0/5
IBEA	197(260)	4806(3904)	∞	∞ *1041*	0/5
Rand	**5.2**(6)	3813(8204)	7.8e4(9e4)	2.2e6(2e6)	0/5
NSGA	112(172)	**862**(200)	**5473**(6272)	**3.6e4**(3e4)	0/5
f48					
IBEA	3360(4252)	∞	∞	∞ *2000*	0/5
IBEA	369(7)	2903(1638)	∞	∞ *2063*	0/5
IBEA	493(492)	1203(1313)	∞	∞ *1041*	0/5
Rand	**8.0**(8)	3856(4761)	2.1e5(5e5)	6.6e5(6e5)	0/5
NSGA	285(284)	**715**(258)	**1292**(818)	**1.4e4**(2e4)	0/5
f49					
IBEA	9450(1e4)	∞	∞	∞ *2000*	0/5
IBEA	877(740)	9765(1e4)	∞	∞ *2063*	0/5
IBEA	480(446)	∞	∞	∞ *1041*	0/5
Rand	**22**(36)	3900(5215)	1.4e5(1e5)	2.4e6(2e6)	0/5
NSGA	284(259)	**1151**(760)	**6833**(6736)	**4.1e4**(4e4)	0/5
f50					
IBEA	4287(2656)	∞	∞	∞ *2000*	0/5
IBEA	263(327)	∞	∞	∞ *2063*	0/5
IBEA	69(120)	2294(1380)	∞	∞ *1041*	0/5
Rand	**3.8**(2)	758(924)	2.0e4(1e4)	5.8e5(2e5)	0/5
NSGA	230(186)	**743**(272)	**1755**(256)	**7294**(5086)	0/5
f51					
IBEA	1493(1543)	∞	∞	∞ *2000*	0/5
IBEA	77(190)	2174(2834)	∞	∞ *2063*	0/5
IBEA	26(63)	459(273)	∞	∞ *1041*	0/5
Rand	**2.8**(1)	**65**(72)	1976(1133)	4.0e4(5e4)	0/5
NSGA	62(152)	482(52)	**919**(142)	**2385**(755)	2/5
f52					
IBEA	2320(3153)	∞	∞	∞ *2000*	0/5
IBEA	547(1026)	9681(1e4)	∞	∞ *2063*	0/5
IBEA	96(152)	∞	∞	∞ *1041*	0/5
Rand	**9.4**(13)	**572**(502)	2.0e4(2e4)	3.1e5(2e5)	0/5
NSGA	328(218)	4415(4688)	**1.1e4**(1e4)	**1.8e4**(7e4)	0/5
f53					
IBEA	2202(1431)	∞	∞	∞ *2000*	0/5
IBEA	142(176)	2212(1578)	9474(2e4)	∞ *2063*	0/5
IBEA	70(173)	**467**(337)	**1400**(1895)	∞ *1041*	0/5
Rand	**6.8**(14)	4187(1e4)	1.1e5(10)	5.0e5(5e4)	0/5
NSGA	63(155)	652(552)	1.0e4(1e4)	**1.3e4**(5e4)	0/5
f54					
IBEA	1374(1012)	∞	∞	∞ *2000*	0/5
IBEA	1979(2360)	3890(3206)	∞	∞ *2063*	0/5
IBEA	282(308)	1267(1041)	∞	∞ *1041*	0/5
Rand	**6.4**(8)	**258**(229)	**3091**(2347)	6.6e4(6e4)	0/5
NSGA	185(214)	572(172)	8069(9188)	**1.0e4**(8e4)	0/5
f55					
IBEA	2064(2443)	∞	∞	∞ *2000*	0/5
IBEA	250(470)	∞	∞	∞ *2063*	0/5
IBEA	52(83)	2356(2558)	∞	∞ *1041*	0/5
Rand	**3.2**(4)	**177**(140)	**4754**(3428)	1.4e5(1e5)	0/5
NSGA	290(18)	1020(746)	1.3e4(2e4)	**4.0e4**(2e4)	0/5