Black-Box Optimization Benchmarking: Indicator-Based Selection in Multiobjective Search

Paris Sud University

Amal TARGHI
amal.targhi@u-psud.fr

Anh Khoa NGO HO
anh-khoa.ngo-ho@u-psud.fr

Fella BELKHAM
fella.belkham@u-psud.fr

Mahmut CAVDAR
mahmut.cavdar@u-psud.fr

Trong Bach VU
jsbachvu@gmail.com

ABSTRACT

Evolutionary algorithms are usually used on the multi - objective optimization when no a priori parameters are known. The chosen paper "Indicator-Based Selection in Multi objective Search" proposes a general indicator-based evolutionary algorithm 'IBEA' that can define the optimization goal in term of an hyper-volume-indicator, then use this measure in the selection process. The strength of this algorithm compared to the other MOEAs algorithms, lies in the fact that the preference criteria is always incorporated in dominance relations. For this project, we implement the adaptative IBEA algorithm proposed in the paper, using the Python language. For Benchmarking we used COCO platform (for Comparing Continuous Optimizers in a Black-Box Setting) to compare our results with the standard optimizers on the platform and the optimizers done by the other groups implementing the same algorithm.

Keywords

Benchmarking, Black-box optimization, Bi-objective optimization, Indicator-Based Selection, Evolutionary Algorithm

1. INTRODUCTION

The present repport will serve through as our final report for the optimization project. We choose as paper the: Indicator-Based Selection in Multi-objective search, that we implmented using Python langage. In what follows, we'll present the concepts related to the paper and the motivation behind the work. We'll describe the algorithme and the different operators used to implement it, . We'll discuss the results we obtained and compare them to the other groups and the to the dataset of NSGA-II and random search.

2. BACKGROUND

The paper Indicator-Based Selection in Mutiobjective Search disscusts the way preference informations of the decision maker can be integrated into multiobjective search. However, before talking about the motivation of the work, we'll start by introduce the main concepts adressed in the paper which are multiobjective optimization and evolutionaty algorithms.

2.1 Multi-objective Optimization

Optimization problems are common in various disciplines, including mechanics, economics, logistics and biomolecular, that need to be solved. In optimization problems, we have to find best solutions which are optimal or near-optimal from all feasible solutions with respect to some constraints. We can represent an optimization problem with the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq b_i,\ i = 1, \ldots, m \\ & h_i(x) = 0,\ i = 1, \ldots, p. \end{aligned} \tag{1}$$

where f(x) is the objective function over the variable x, $g_i(x)$ and $h_i(x)$ are called constraints. The main difference between multi and mono-objective optimization problem is additional objective functions to be optimized. For representation of multi-objective optimization problem we need to change only first part of form :

$$\underset{x}{\text{minimize}} \quad (f_1(x), f_2(x), f_3(x), ..., f_k(x)) \tag{2}$$

where the integer $k \geq 2$ is the number of objective functions. We cannot apply our mono-objective optimization algorithm to only one objective, when others objectives are also important. All different objectives contain its own optimal solution. A solution can be optimal solution for an objective, on the other hand, it may require compromise for another objective. If we need to express it in a mathematical expression:

$$\underset{x}{\text{minimize}} \quad f_1(x) \quad \neq \quad \underset{x}{\text{minimize}} \quad f_2(x) \tag{3}$$

where $f_1(x)$ and $f_2(x)$ are two different objectives of same problem. And we can not say which solution is the best with respect to both objectives. Because each objective corresponds to different optimal solution. If that is the case, we

ACM ISBN .
DOI:

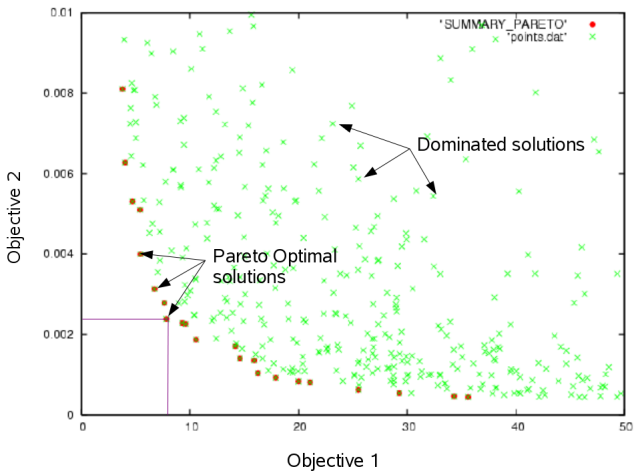

Figure 1: Example of a Pareto frontier[Cenaero]

have to keep both these solutions. So, solution set is composed by all optimal solution. No solution f rom the final set can be said to be better than others. This set of optimal solutions leads us to Pareto optimal solutions notion;that is, solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives. (Figure1)

2.2 Evolutionary algorithms

Evolutionary algorithms attempt to solve complex problems by simulating the processes of Darwin. They develop a set of solutions to a problem with the aim of finding the best results. Since they use random processes iteratively, evolutionary algorithms are never guaranteed to find an optimal solution for any problem, but they will often find a good solution if one exists [1].
The main idea behind evolutionary algorithms is to evolve an initial population in a natural selection process to ensure the rise in the fitness of the population. Based on the fitness, candidates are selected to seed the next generation by applying recombination and/or mutation to them, this selection process favor the solution with the best objective function value. Recombination operator will generate new candidate (children) using two or more candidates (parents), the children inherit the characteristic of the parent. Mutation, in the other side, will be applied to only one candidate and it will generate a new one. Once new candidate are created, and based on the new fitness values, the replacement operator will determine which individuals are to be removed from the population, keeping only the candidate with highest fitness [2].

Evolutionary algorithm have a number of components, procedures or operators [2]:

- Representation : definition of an individual;
- Evaluation function (fitness function);
- Population;
- Parent selection mechanism;
- Variation operators, recombination and mutation;

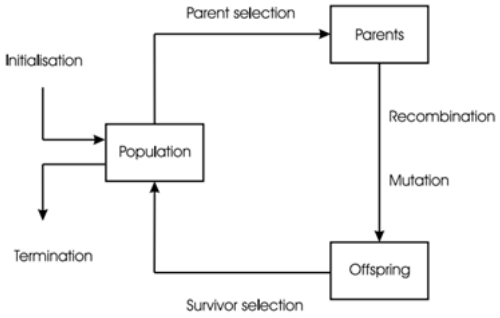

Figure 2: Components of EAs [2]

- Survivor selection mechanism (replacement).

2.3 Evolutionary Algorithms for Multiobjective Optimization

The Evolutionary algorithms ■ EA ■ are well adapted to optimization problems with multiple objectives for many reasons For starter, the initial population is selected randomly and they have the potential of finding multiple Pareto-optimal solutions in a single simulation run, due to their inherent parallelism. However, it is not awlways possible to generate noninferior solutions, much less the entire Pareto-optimal set when it comps to complex applications. Therefore, the optimization goal for an multiobjective problems may be reformulated in a more general fashion based on three objectives [3]:

- The distance of the resulting nondominated front to the Pareto-optimal front should be minimized (i);
- A good (in most cases uniform) distribution of the solutions found is desirable;
- The spread of the obtained nondominated front should be maximized, i.e., for each objective a wide range of values should be covered by the nondominated solutions (ii).

2.4 Motivation

The autors, in our paper consider the two objectives (i) and (ii) are in conflict since, we quote, "there exists no formal definition of two separate objectives, one for convergence and one for diversity, that is compliant with the Pareto dominance relation".
From here, the main question for the work is: How to design MOEAs with respect to arbitrary preference infromation using a Pareto-compliant formalizations of the decision maker's preferences? The main idea behind The paper 'Indicator-Based Selection in Multi-objective Search' is to propose a general indicator-based evolutionary algorithm <IBEA> that can define the optimization goal in term of an indicator (measure) then use this measure in the selection process which is one of the main steps of evolutionary algorithms. IBEA extends the idea of flexible integration of preference information presented by Fonseca and Fleming and Knowles.
To sum up, our paper is an IBEA where the indicator is a hyper-volume indicator (quantitaive measure which gives the volume between the estimated Parteo and the reference

point R) and where the preference criteria is always incorporated based-on dominance relations.

3. DESCRIPTION OF THE ALGORITHM

This section describes Adaptive Hyper-Volume Indicator Based Evolutionary algorithm which is separated into six main steps. It performs fitness assignment with hyper-volume concept, environmental selection removing the worst individuals, binary tournament selecting the best "parent" solutions and line crossover recombination with gaussian mutation creating new offspring generations.

The inputs of this algorithm are the number fixed α of solutions returned population size, maximum number of generation, fitness scaling factor used in fitness assignment steps. It returns a pareto set approximation being the best solutions found after a number fixed of looping.

The first step is initializing a population with size predefined whose individuals are multi-dimension vectors and assigned later a fitness value which is used in comparison.

The second step is scaling objective values of each individual and then calculating its fitness values with hyper-volume indicator. In this step, for each objective, its values are normalized in the interval [0,1] by finding the maximum and the minimum namely upper bound and lower bound respectively. Moreover, indicator value between two individuals is calculated by using hyper-volume concept. In fact, it is the space volume representing the domination relationship of individuals. For each individual, fitness value is calculated by this indicator values and its maximum absolute.

The third step is selecting the best individuals. In this case, individual having the smallest fitness value is removed from the population until its size does not exceed α and then the fitness of its rest is updated.

The fourth step is returning the final best population found pareto set approximation including non-dominated individuals if stopping conditions is satisfied. For instance, one condition is that the number of generation exceeds the threshold permitted.

The fifth step is binary tournament selecting the best "parents" for the next step. In detail, two competitors are randomly chosen in the population and a winner is decided by its fitness value.

The sixth step is recombination of two parents and mutation of its offspring. For the line crossover recombination, two offspring are born by merging each objective value of its parents with a possibility random, which means that an offspring objective value is sum of two parent objective value multiplied by a random number. For the mutation with Gaussian Mutation Operator, each objective value of an offspring is changed by applying the Gaussian formulas. These new offspring are put into population and then the algorithm runs again from the second step.

Adaptive IBEA

Input: α (Population Size), N (Maximum number of generation), k (fitness scaling factor)

Output: A (Pareto set approximation)

Step 1: Initialization: Generate initial population P of size α; set the generation counter m to 0.

Step 2: Fitness assignment: Assign fitness values.

1. Scale each objective value f_i in the interval $[0,1]$ by determining its lower bound $lb_i = min_{x \epsilon P} f_i(x)$ and its upper bound $ub_i = max_{x \epsilon P} f_i(x)$. The new objective value f':

$$f_i'(x) = \frac{f_i(x) - lb_i}{ub_i - lb_i}$$

2. Calculate hyper-volume indicator value $I(X^1, X^2)$ and determine the maximum absolute indicator value $c = max_{X^1, X^2 \epsilon P}(|I(X^1, X^2)|)$. In COCO case, it is calculated with two dimensions $X^1(x_1^1, x_2^1)$, $X^2(x_1^2, x_2^2)$ and reference point $Z(z_1, z_2)$:
$I(X^1) = (|z_1 - x_1^1| * |z_2 - x_2^1|)$; $I(X^2) = (|z_1 - x_1^2| * |z_2 - x_2^2|)$
$I(X^1 + X^2) = |z_1 - min(x_1^1, x_1^2)| * |z_2 - min(x_2^1, x_2^2)| - (max(x_1^1, x_1^2) - min(x_1^1, x_1^2)) * (max(x_2^1, x_2^2) - min(x_2^1, x_2^2))$

if
$((x_1^1 < x_1^2)$ *and* $(x_2^1 < x_2^2))$ *or* $((x_1^1 > x_1^2)$ *and* $(x_2^1 > x_2^2))$
then
  $I(X^2) - I(X^1)$;
end
else
  $I(X^1 + X^2) - I(X^1)$;
end

3. Calculate fitness value $F(x^i)$ for all individuals in P:

$$F(x^1) = \sum_{x^2 \epsilon P \ without \ x^2} -e^{-I(X^1, X^2)/(c.k)}$$

Step 3: Environmental selection:

while *population size* $< \alpha$ **do**
  Remove the smallest fitness individual x^* *from P;*
  Update the fitness value of the remaining individuals for all $x \in P$*:*

$$F(x) = F(x) + -e^{-I(X^*, X)/k}$$

end

Step 4: Termination:

if $m \leq N$ *and others stopping conditions are satisfied* **then**
  Set the newest generation P into A - Pareto set approximation;
  Stop the algorithm;
end

Step 5: Mating selection: Perform binary tournament selection for having the α best parents.

while *parent population size* $< \alpha$ **do**
  Select randomly two parents from population P;
  Select a parent having a higher fitness value;
end

Step 6: Variation: Create new generations with Line Crossover Recombination and Gaussian Mutation Operator.

1. Line Crossover Recombination: Two parents are randomly selected for having two offspring and then removed from parent population.

while *parent population size* ≥ 2 **do**
  Select randomly two parents from parent population;
  Select randomly two possibilities a and b in the interval [-0.25, 1.25];
  for *each objective value* f_i *of an offspring* **do**
    f_i*(First Offspring)* = f_i*(First Parent)*a +* f_i*(Second Parent)*(1 - a);*
    f_i*(Second Offspring)* = f_i*(Second Parent)*b +* f_i*(First Parent)*(1 - b);*
  end
  Remove its parents from population P;
end

2. Gaussian Mutation Operator: Select randomly a number of offspring for mutation.

for *each offspring with M objectives in offspring population* **do**
  Select randomly Mutation Possibility in the interval [0,1];
  if *Muatation Possibility < 0.01* **then**
    Select randomly a normally distributed one-dimensional random number with mean zero and standard deviation one N(0,1);
    Select randomly $N_i(0,1)$ for each objective value;
    Assign initial σ value as 10^{-2} for each objective value;
    $T_1 = \frac{1}{\sqrt{2*M}}$;
    $T_2 = \frac{1}{\sqrt{2*\sqrt{M}}}$;
    for *each objective value* f_i **do**
      $\sigma_i = \sigma_i * e^{T_1*N(0,1)+T_2*N_i(0,1)}$;
      $f_i = f_i + \sigma_i * N_i$;
    end
  end
end

3. Put a half of offspring selected randomly into population P and m++.

4. DESCRIPTION OF THE IMPLEMENTATION

For the implementation we integrated our code to the COCO plateform using Python. Since, in our paper, there were no details about which recombinaison and mutation to use, we compared two methods in each operators that we'll describe in this section.

4.1 Step's implementation

Step 1: Initialization

We generate initial population P with size Alpha - population size and set Max Generation counter to 0. An array F is generated by Problem in bbob-biobj suite (fun()). We used F = fun(P) with two objectives to calculate Pareto set.

Step 2 - Fitness assignment

In this function we calculates fitness values of individuals in P. Fitness value is calculated by using Hypervolume Indicator with sub-function "indicator_value(x1 , x2, reference point Z)".

- Input: population (The population what we want to calculate the fitness), referencePointZ(point Z(2,2)), fitness scaling factor: 0.05.
- Output: fitness (Fitness values of each individual), indicator (hypervolume indicator values), max_indicator (maximum absolute indicator value C).

****SubFunction: Indicator value***

This function helps to calculate indicator value I for 2 points x1 and x2.

- Input: X1, X2 (individual's values), referencePointZ (point Z (2,2)).
- Output: hypervolume indicator value.

Step 3 : Environmental selection

The environmental selection function helps to remove individuals having smallest fitness value until Alpha-Value individuals chosen.

- Input: population (The population what we want to remove individuals having smallest fitness value from), FN (F normalisation), fitness(Fitness array values), indicator(hypervolume indicator values), max_indicator(The maximum value of indicators).
- Output: population(The population containing Alpha-Value best individuals), fitness(Fitness values of new population), indicator(hypervolume indicator values of new population).

Step 4 : Mating selection

In this function selects two individuals for binary tournament, where the individual with the best fitness value is placed in the temporary mating pool P' until Alpha-Value individuals are chosen.

- Input: population(The population containing Alpha-Value individuals),fitness(Fitness array values).
- Output: Parent Population (Population with the best individual with replacements).

Step 5 : Variation

The function "variation(initial Population P, parent Population)" has two sub-functions "recombination(parent Population)" and "mutation(baby Population)".
We first apply the recombinaison operators on the parents in P'; then, we mutate the resulting children. After that, we'll go back and repeate step 2.

- Since there are no indication in our paper on what recombinaison algorithm to use, we comparend two methods.

1.1. Cycle Crossover Algorithm: This algorithm is known to preserves the information in both parents. It define the

cycle defined by the parent's chromosomes. Theses cycles are used to create the children, the 1st one is copied into the 1st child, the 2nd cycle is copied into the 2nd child and so on.

1.2. Line Crossover: this algorithm consist of producing offspring that are somewhere around and between the variable values of the parents. For this, a two dimentional variable is generated randomly between [-0.25, 1.5] and using the following rule :

$$Individu_i^{C_i} = Individu_i^{P_1} * a_i + Individu_i^{P_2} * (1 - a_i)$$

In our case, the Cycle Crossover turned out not to be effective since it does not suit our data, cycles can rarely be found when it come to real values.

- Same thing is to be said about the mutation, no further details are giving in the paper, so again, we implmented two methods to be compared.

2.1. Swap Mutation Algorithm: Each baby mutates by swapping randomly its two gene values. The swapping possibility depends on a threshold chosen swapping possibility (parameter).

2.2. Gaussian Mutation: in this algorithm, we add to each verctor entry of the individual a randomly produced number from a Gaussian distribution with mean equal to zero. In this case, the variance of the distribution is a parameters, which is also the case for the swapping possibility.

Step 6 : Termination

We recheck if the Max Generation counter = Max Generation value or budget < 0. If one of the conditions is satisfied, we set the output of function "non_dominated_selection(initial Population P, indicator Array)" as Pareto set approximation, which means that our output are non-dominated individuals.

- Input: population(Population), indicator(Hypervolume indicator values).
- Output: paretoSetApproximation (The Pareto Set).

4.2 Parameters

Each operator of an evolutionary algorithm may give better or worst results according to the parameters used. Along the implmentation, we used multiple parameters with different intialization in order to compare the results obtained in every simulation.

- Population size : we varied the size of the popultation during the test, mainly keeping it between 50 and 200.
- Maximum number of generation.

Other parameters are related to our algorithm IBEA and were fixed at precise values:

- K factor : fixed at 0,05.
- Reference point : (2,2).

The Budget parameter is related to the COCO plateform, we strated with small budget in order to fix the problem related to our code and incresed it everytime, for the purpose of this report we used budget at 4000.

4.3 Stopping criteria

Our stoping cretaria in the algorithm are mainly reaching the maximum number of generation, or the COCO budget being inferior to zero.

5. DISCUSSION

5.1 Evolution of Algorithm

First implementation of algorithm had a lot of problem. One of them was about recombination operators to the maiting pool. Discrete recombination doesn't perform well for this kind of problem because of float objective function values. Line recombination approach is used for resolve this problem. Also the choise of parent recombination algorithm is not clear in the paper. Generation from combination of all parents seems good point for to start. But size of new population($n^2 + n$) produces a time consuming algorithm. To tackle the problem of combination we generate babies as much as population size. The hardest problem we face is huge execution time of the indicator value function. As already author said, I_{HD}(A,B) values is computationally very expensive and essential part of algorithm.

5.2 CPU Timing

In order to evaluate the CPU timing of the algorithm, we have run the with restarts on the entire bbob-biobj test suite for $2D$ function evaluations. The code was run on a Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz with 1 processor and 4 cores. The time per function evaluation for dimensions 2, 3 equals 13h05:32 and 16h46:22 respectively. For dimension more three we still have not received yet solution because the expansive computation of indicator value. An additional example to show in more detail how the concept need a lot computation: the time of 3th function evaluation for dimension 2 with 10000 budget equals 2h05:55.

6. RESULTS

The indicator based approach have been tested with a lot of combination. In general, size of population, budget, maximum generation number and mutation probability are used. If we examine our result carefully we can see that our algorithm runs well for some of function than the others. For example on function 13(Ellipsoid separable/Rosenbrock original) it reaches 0.9 proportion, as can be seen in Figure 8. For example on function 25(Attractive sector/Schaffer) the result remains stable for a long period, than increases moderately to about 0.2 and remains constant in the rest of period. As can be seen in Figure 10, the proposed algorithm performs better for seperable and moderate functions. In contrast, the indicator based algorithm doesn't show good performance for the multimodal functions.

The result of Hyper-Volume Indicator Based Evolutionary Algorithm (HV-IBEA) is compared with NSGA-II and Random Search Algorithm (RS), which is based on runtime distribution per function provided by COCO. In this case, there are 55 bi-objective functions with the dimension being two. The graphs show the number of function evaluations needed to reach the target function value.

As can be seen from the Figure3, the three functions has similar upward trends. In fact, NSGA-II remains steady in the interval $0 - 2$ and rises sharply to about 0.8 at 7. In addition, Random Search has an upward trend of about 0.7. HV-IBEA reaches the highest value, namely 0.6 in 5 and unchanges in the rest of evaluation period. As a result, at the end of the period, NSGA-II takes the first place, followed by RS and HV-IBEA.

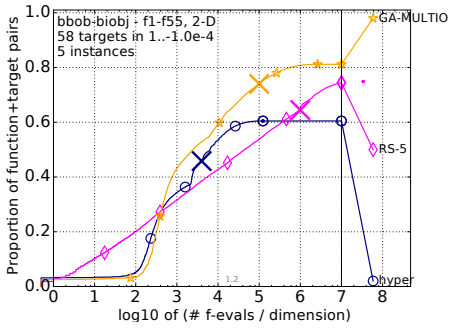

Figure 3: All functions

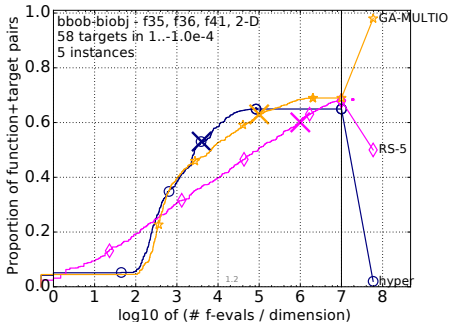

Figure 4: Graph (f35, f36, f41)

With regards to the sub-graph of functions ill-condictioned Figure4, the numbers of three algorithms have similar levels ranging between 6.0 and 6.5 in 7. HV-IBEA gets the lowest number at 0.4 in the graph moderate – multimodal Figure5 and the peak at 0.75 in the graph separable – moderate Figure6.

In detail for the charts in which HV-IBEA has the improvements significant, the graphs of function f2, f14 and f39 witnesses that HV-IBEA ranks a higher place than RS and shoot up to about 0.8 in 5. Moreover, the algorithm HV-IBEA, in the interval 4 – 5 of the f12 and f13 graphs, gets the higher values (about under 0.9) than the two other algorithm. It is expected that HV-IBEA will increase whereas RS and NSGA-II will drop.

To summarize, HV-IBEA does not have in general a better trend than NSGA-II and RS with the peak of 0.6 (All function Graph). However in the group of function separable (f2, f12, f13, f14)Figure7 and function ill-conditioned - weakly-structured (f39), it shows the significant runtime distribution.

Results from experiments according to $IBEAHypervolume$, $IBEA\epsilon Python$ and $IBEA\epsilon C$ on the benchmark functions given in 'biobj2016func' are presented in Figure8.The experiments were performed with COCO 2016. Our implementation $IBEA_HD$ performs significantly better than $IBEA_\epsilon$ implementations almost for all group functions in two dimensions.

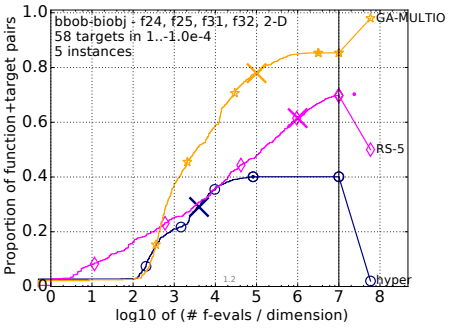

Figure 5: Graph (f24, f25, f31, f32)

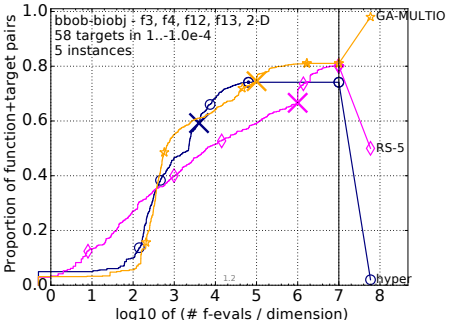

Figure 6: Graph (f3, f4, f12, f13)

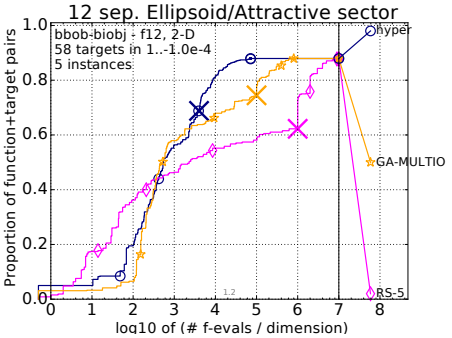

Figure 7: Graph (f12)

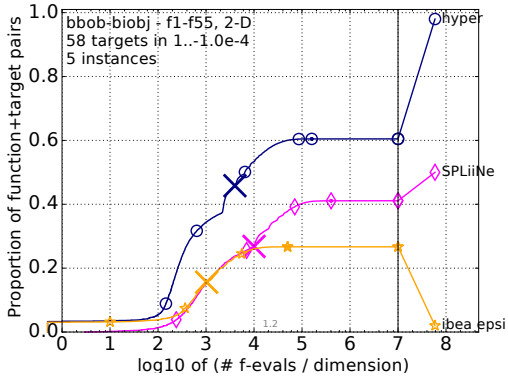

Figure 8: All functions

7. CONCLUSION

During this project we studied and implemented the adaptative Indicator-Based Selection in Multiobjective Search' algortithm , IBEA . The main idea behind IBEA is to to propose a general indicator-based evolutionary algorithm that can define the optimization goal in term of an indicator (measure) then use this measure in the selection processs. This indicator, in our case, is a hypervolume indicator which is a space volume representing the domination relationship of individuals, for each individual, fitness value is calculated by this indicator values and its maximum absolute.

Only problem with $IBEA_{HV}$ approach is execution time.It requires high-level computational capacity.Over all, the main difficulties we faced where choosing the right operators to implments so our implmentation performs as well as the results mentionned in the paper, and also choosing and testing the parameters. Since the algorithm give better results when the COCO budget is very hight, we only tasted large budget for some function because it's time consuming.

References

1-Mishra, B. S. P., Dehuri, S., Kim, E. (Eds.). (2016). Techniques and Environments for Big Data Analysis: Parallel, Cloud, and Grid Computing (Vol. 17). Springer.

2-Eiben, Agoston E., and James E. Smith. Introduction to evolutionary computing. Vol. 53. Heidelberg: springer, 2003.

3-Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications.

4-N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockho . COCO: A platform for comparing continuous optimizers in a black-box setting, 2016.

5-T. Tusar, D. Brockho , N. Hansen, and A. Auger. Coco: The bi-objective black box optimization benchmarking (bbob-biobj) test suite, 2016.

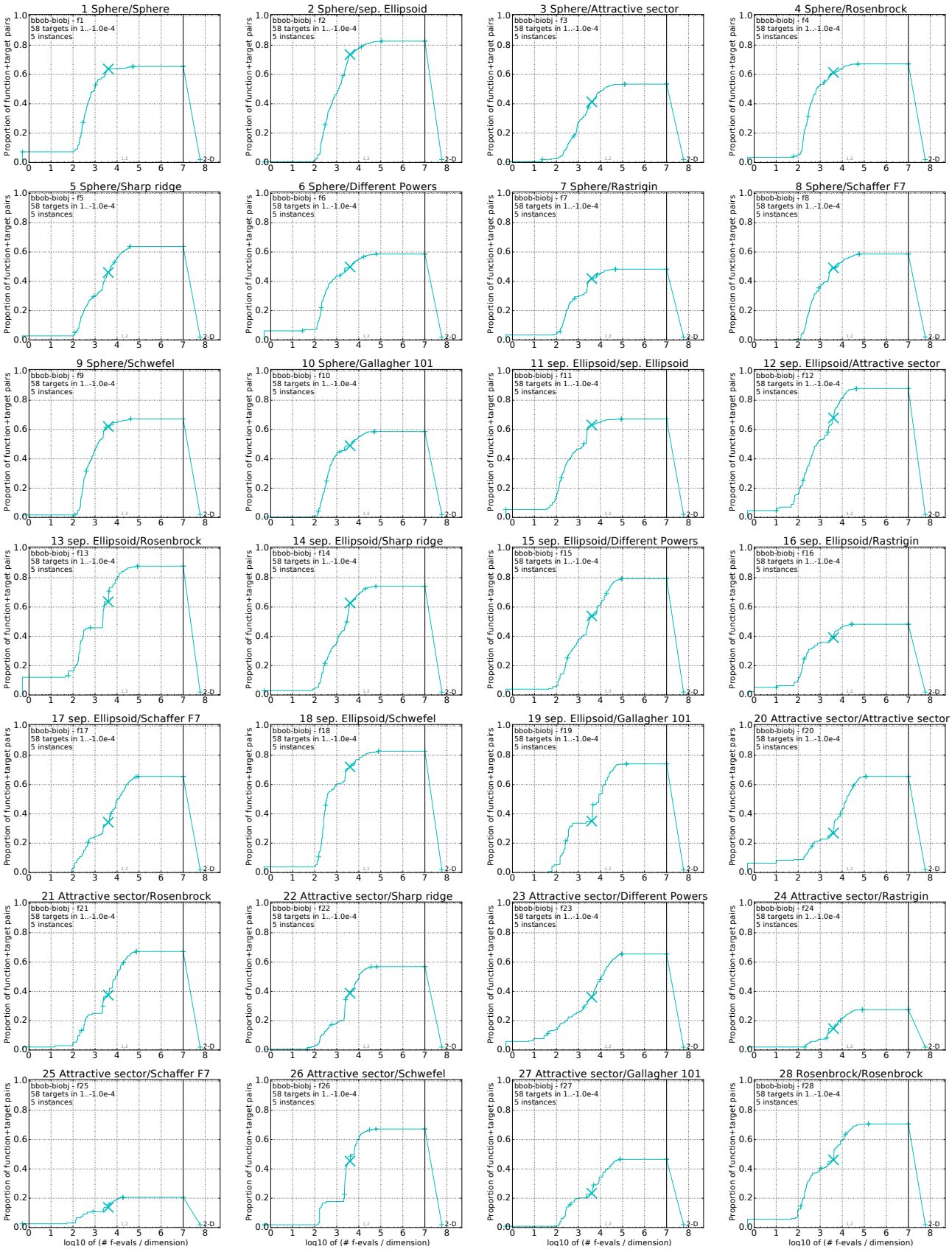

Figure 9: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for 58 targets with target precision in $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10^{-4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$ for each single function f_1 to f_{28} in 10-D.

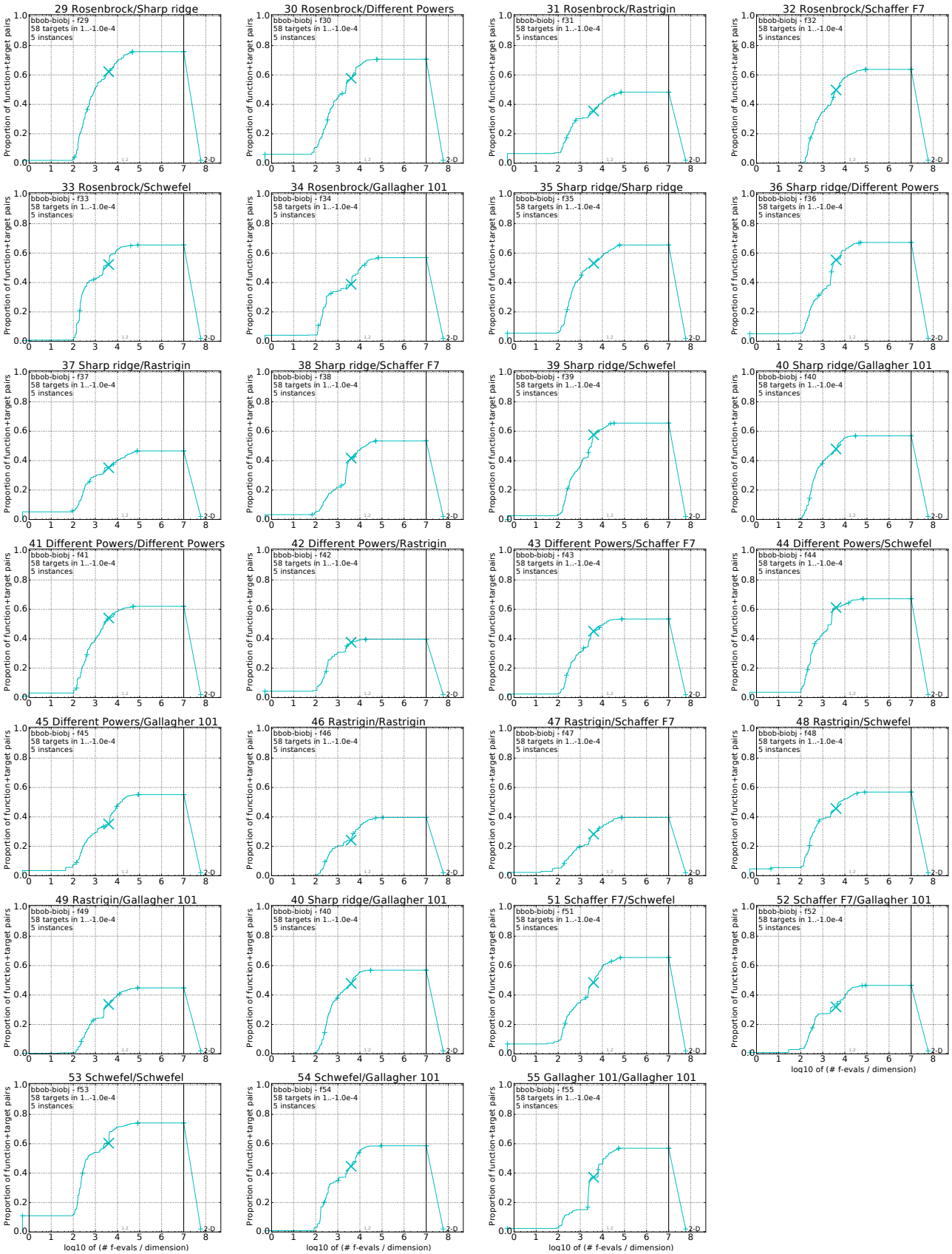

Figure 10: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) as in Fig. 9 but for functions f_{29} to f_{55} in 10-D.

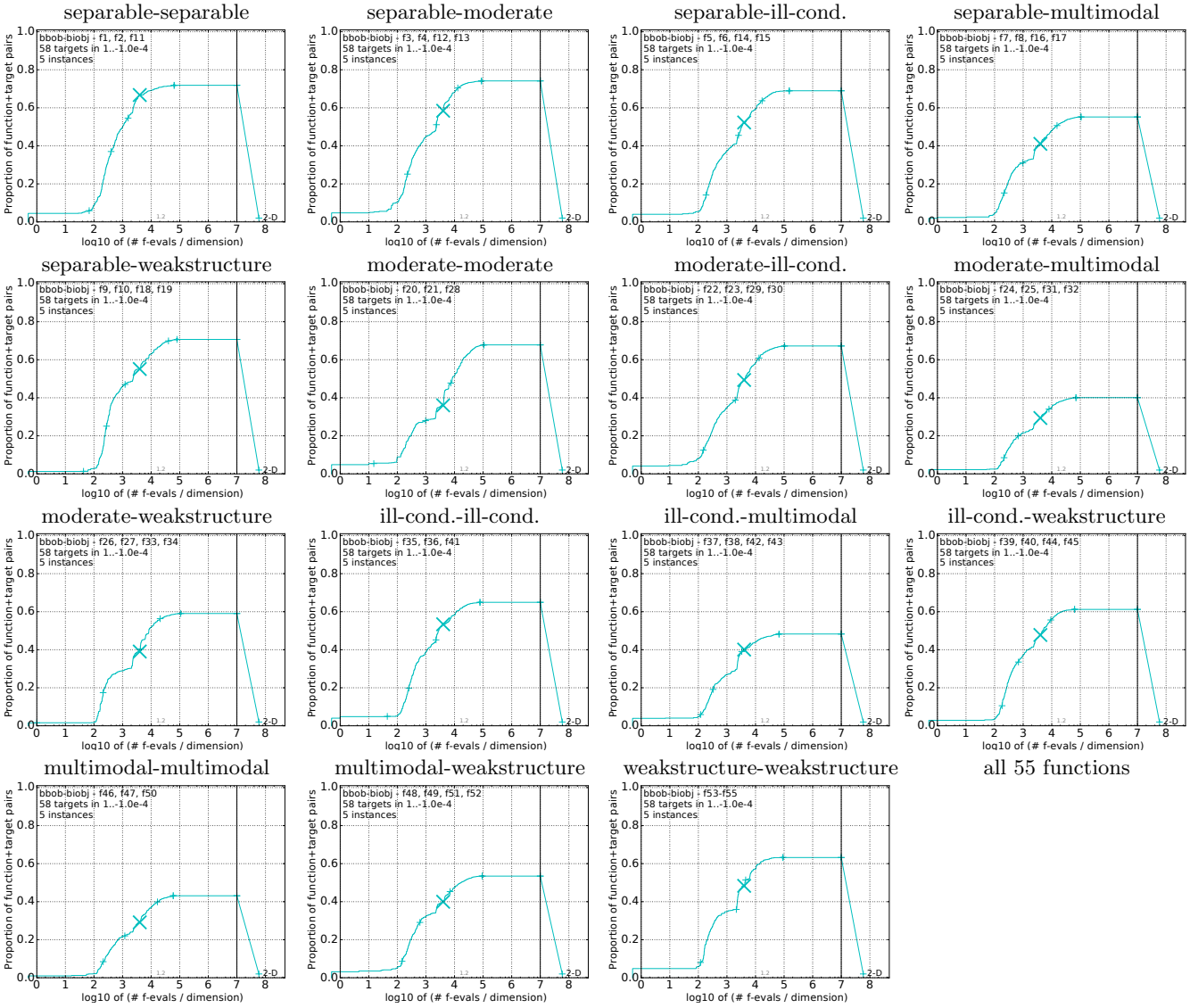

Figure 11: Bootstrapped empirical cumulative distribution per group

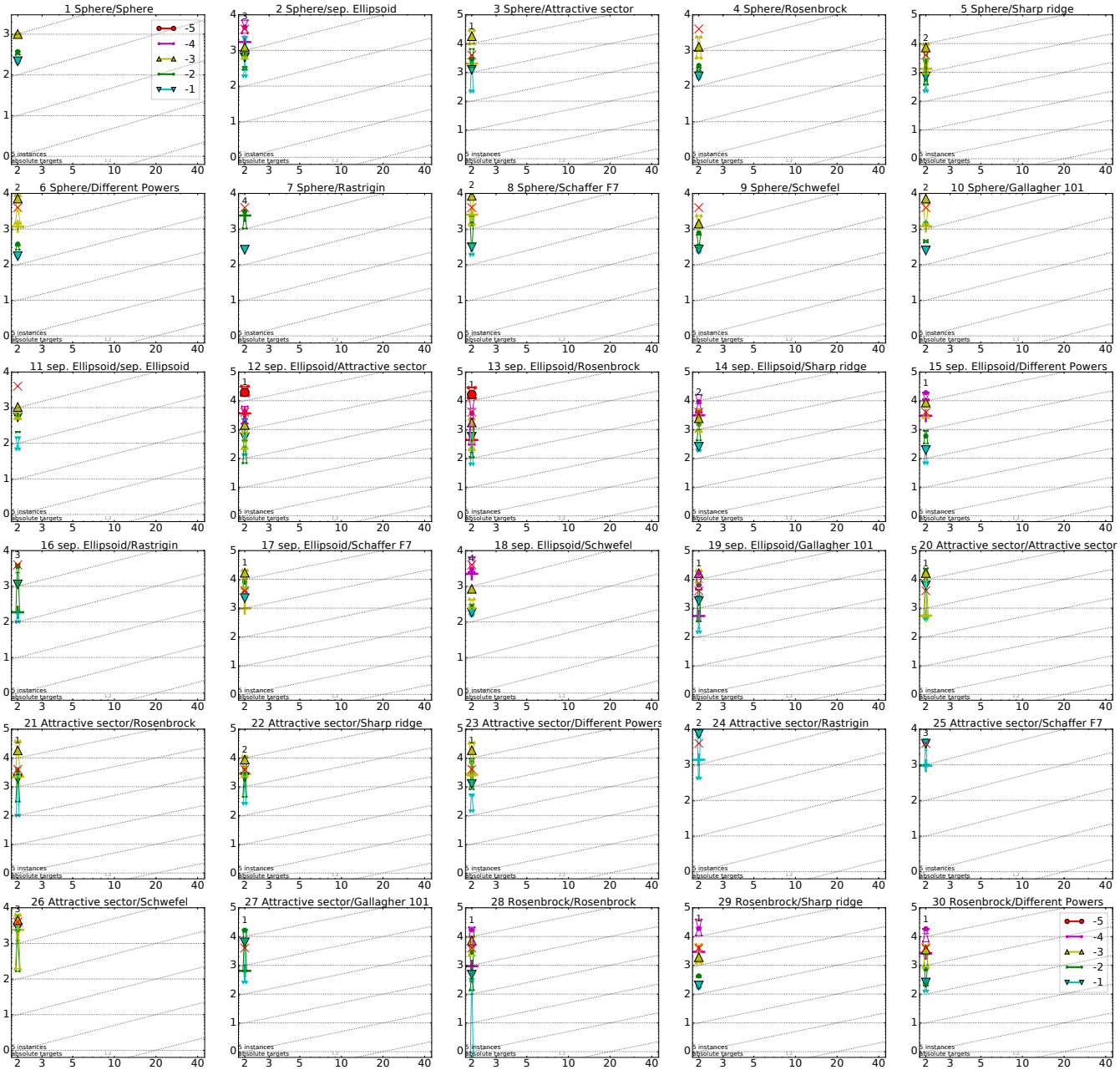

Figure 12: Average runtime f_1 and f_{30}

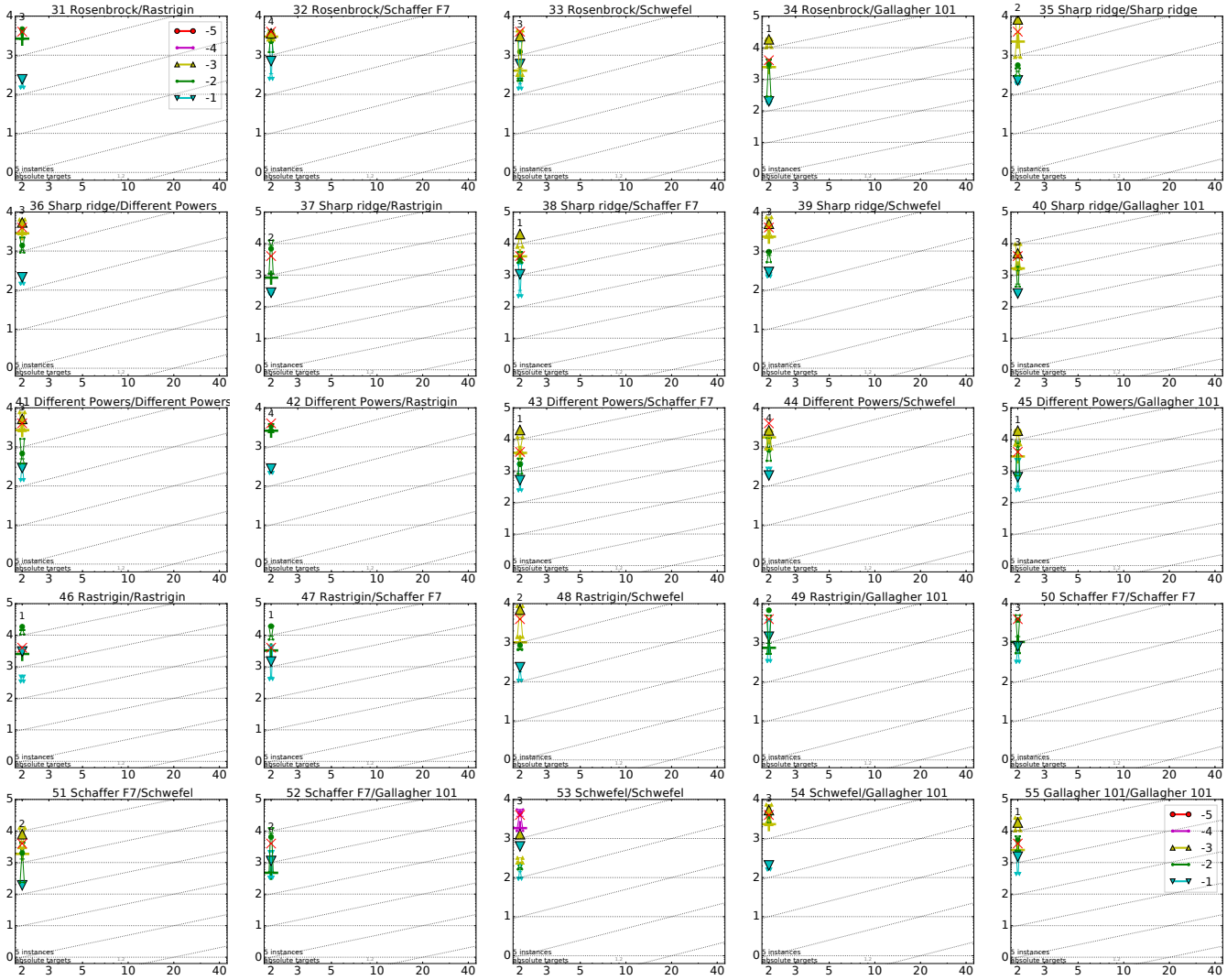

Figure 13: Average runtime f_{31} and f_{55}