

# HNCO

## Empirical cumulative distribution functions of the runtime of various black box optimization algorithms

November 22, 2020

### Abstract

We partly follow the experimental procedure of the COCO framework for the performance assessment of black box optimization algorithms Hansen et al. [2016]. Each algorithm is run independently 20 times on each objective (or fitness) function. The dimension is fixed at  $n = 100$ . Then 50 equally spaced targets are computed for each objective function. For each algorithm and each function we compute the empirical cumulative distribution function (ECDF) of the runtime, that is the proportion of targets reached as a function of the number of evaluations over all 20 runs. We also compute the global ECDF which takes into account the targets of all functions. The results are listed by function. For clarity reasons only 8 algorithms (hence 8 colors) are included in the study. It should be noted that the linear scale of targets does not fit the function EqualProducts.

## Contents

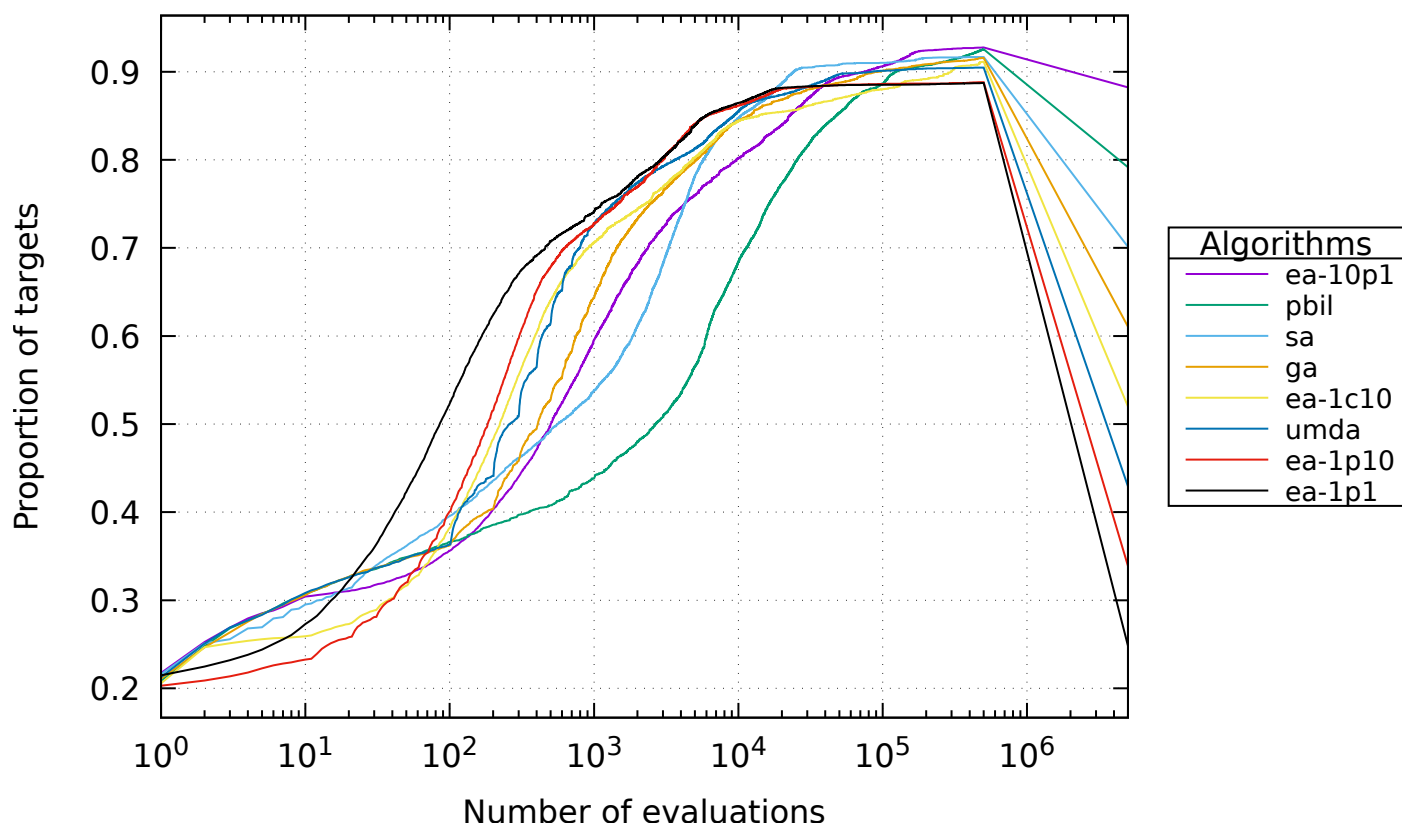
<b>1</b>	<b>Global results</b>	<b>3</b>
1.1	All algorithms . . . . .	3
1.2	Groups . . . . .	4
1.2.1	ec . . . . .	4
1.2.2	eda . . . . .	4
<b>2</b>	<b>Results for one-max</b>	<b>5</b>
2.1	All algorithms . . . . .	5
2.2	Groups . . . . .	5
2.2.1	ec . . . . .	5
2.2.2	eda . . . . .	6
<b>3</b>	<b>Results for lin</b>	<b>6</b>
3.1	All algorithms . . . . .	6
3.2	Groups . . . . .	7
3.2.1	ec . . . . .	7
3.2.2	eda . . . . .	7
<b>4</b>	<b>Results for leading-ones</b>	<b>8</b>
4.1	All algorithms . . . . .	8
4.2	Groups . . . . .	8
4.2.1	ec . . . . .	8
4.2.2	eda . . . . .	9
<b>5</b>	<b>Results for ridge</b>	<b>9</b>
5.1	All algorithms . . . . .	9
5.2	Groups . . . . .	10
5.2.1	ec . . . . .	10
5.2.2	eda . . . . .	10
<b>6</b>	<b>Results for jmp-5</b>	<b>11</b>
6.1	All algorithms . . . . .	11
6.2	Groups . . . . .	11
6.2.1	ec . . . . .	11
6.2.2	eda . . . . .	12

<b>7</b>	<b>Results for jmp-10</b>	<b>12</b>
7.1	All algorithms . . . . .	12
7.2	Groups . . . . .	13
7.2.1	ec . . . . .	13
7.2.2	eda . . . . .	13
<b>8</b>	<b>Results for djmp-5</b>	<b>14</b>
8.1	All algorithms . . . . .	14
8.2	Groups . . . . .	14
8.2.1	ec . . . . .	14
8.2.2	eda . . . . .	15
<b>9</b>	<b>Results for djmp-10</b>	<b>15</b>
9.1	All algorithms . . . . .	15
9.2	Groups . . . . .	16
9.2.1	ec . . . . .	16
9.2.2	eda . . . . .	16
<b>10</b>	<b>Results for fp-5</b>	<b>17</b>
10.1	All algorithms . . . . .	17
10.2	Groups . . . . .	17
10.2.1	ec . . . . .	17
10.2.2	eda . . . . .	18
<b>11</b>	<b>Results for fp-10</b>	<b>18</b>
11.1	All algorithms . . . . .	18
11.2	Groups . . . . .	19
11.2.1	ec . . . . .	19
11.2.2	eda . . . . .	19
<b>12</b>	<b>Results for nk</b>	<b>20</b>
12.1	All algorithms . . . . .	20
12.2	Groups . . . . .	20
12.2.1	ec . . . . .	20
12.2.2	eda . . . . .	21
<b>13</b>	<b>Results for max-sat</b>	<b>21</b>
13.1	All algorithms . . . . .	21
13.2	Groups . . . . .	22
13.2.1	ec . . . . .	22
13.2.2	eda . . . . .	22
<b>14</b>	<b>Results for labs</b>	<b>23</b>
14.1	All algorithms . . . . .	23
14.2	Groups . . . . .	23
14.2.1	ec . . . . .	23
14.2.2	eda . . . . .	24
<b>15</b>	<b>Results for ep</b>	<b>24</b>
15.1	All algorithms . . . . .	24
15.2	Groups . . . . .	25
15.2.1	ec . . . . .	25
15.2.2	eda . . . . .	25
<b>16</b>	<b>Results for cancel</b>	<b>26</b>
16.1	All algorithms . . . . .	26
16.2	Groups . . . . .	26
16.2.1	ec . . . . .	26
16.2.2	eda . . . . .	27
<b>17</b>	<b>Results for trap</b>	<b>27</b>
17.1	All algorithms . . . . .	27
17.2	Groups . . . . .	28
17.2.1	ec . . . . .	28
17.2.2	eda . . . . .	28

<b>18 Results for hiff</b>	<b>29</b>
18.1 All algorithms . . . . .	29
18.2 Groups . . . . .	29
18.2.1 ec . . . . .	29
18.2.2 eda . . . . .	30
<b>19 Results for plateau</b>	<b>30</b>
19.1 All algorithms . . . . .	30
19.2 Groups . . . . .	31
19.2.1 ec . . . . .	31
19.2.2 eda . . . . .	31
<b>20 Results for walsh2</b>	<b>32</b>
20.1 All algorithms . . . . .	32
20.2 Groups . . . . .	32
20.2.1 ec . . . . .	32
20.2.2 eda . . . . .	33
<b>A Plan</b>	<b>33</b>
<b>B Default parameters</b>	<b>36</b>

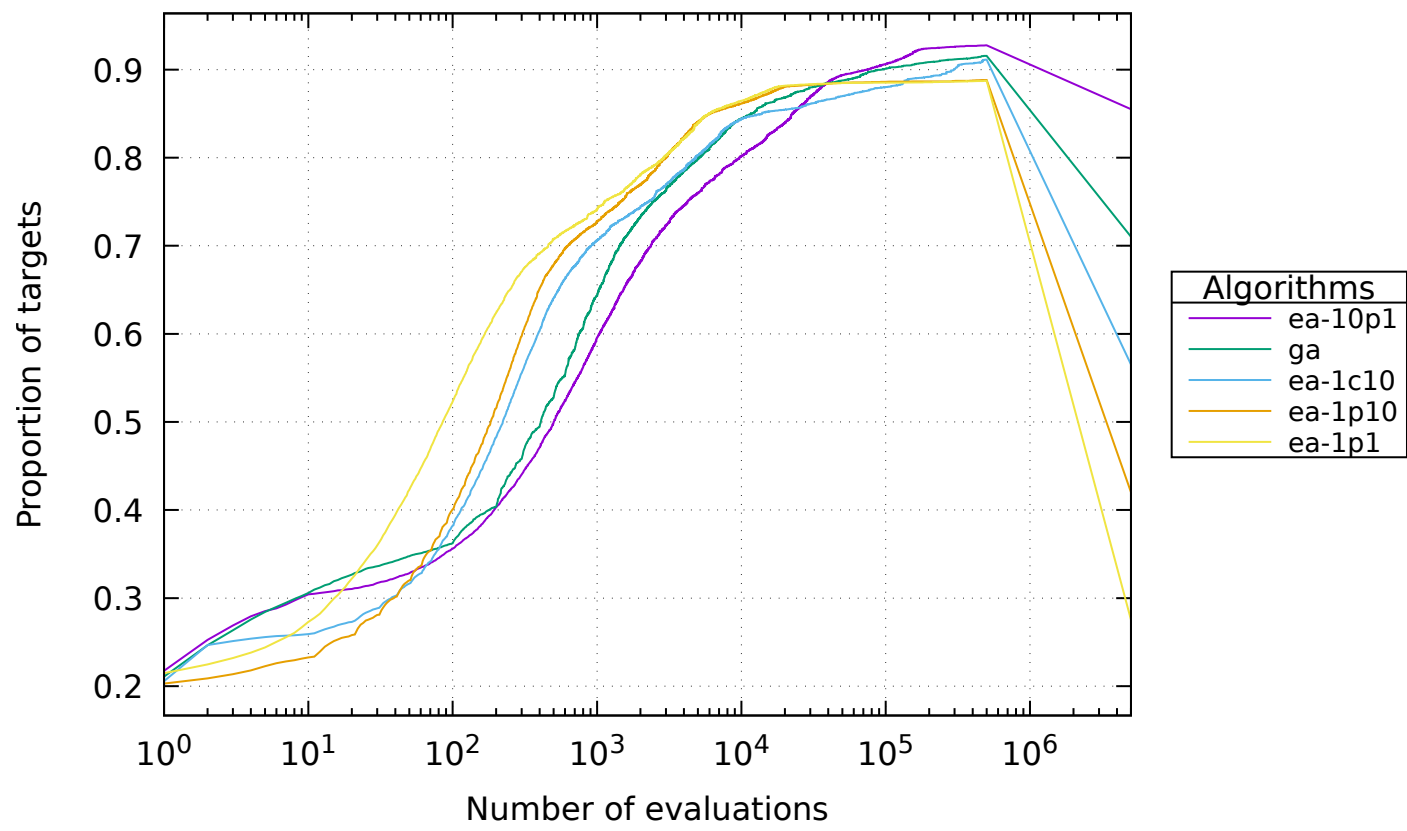
# 1 Global results

## 1.1 All algorithms

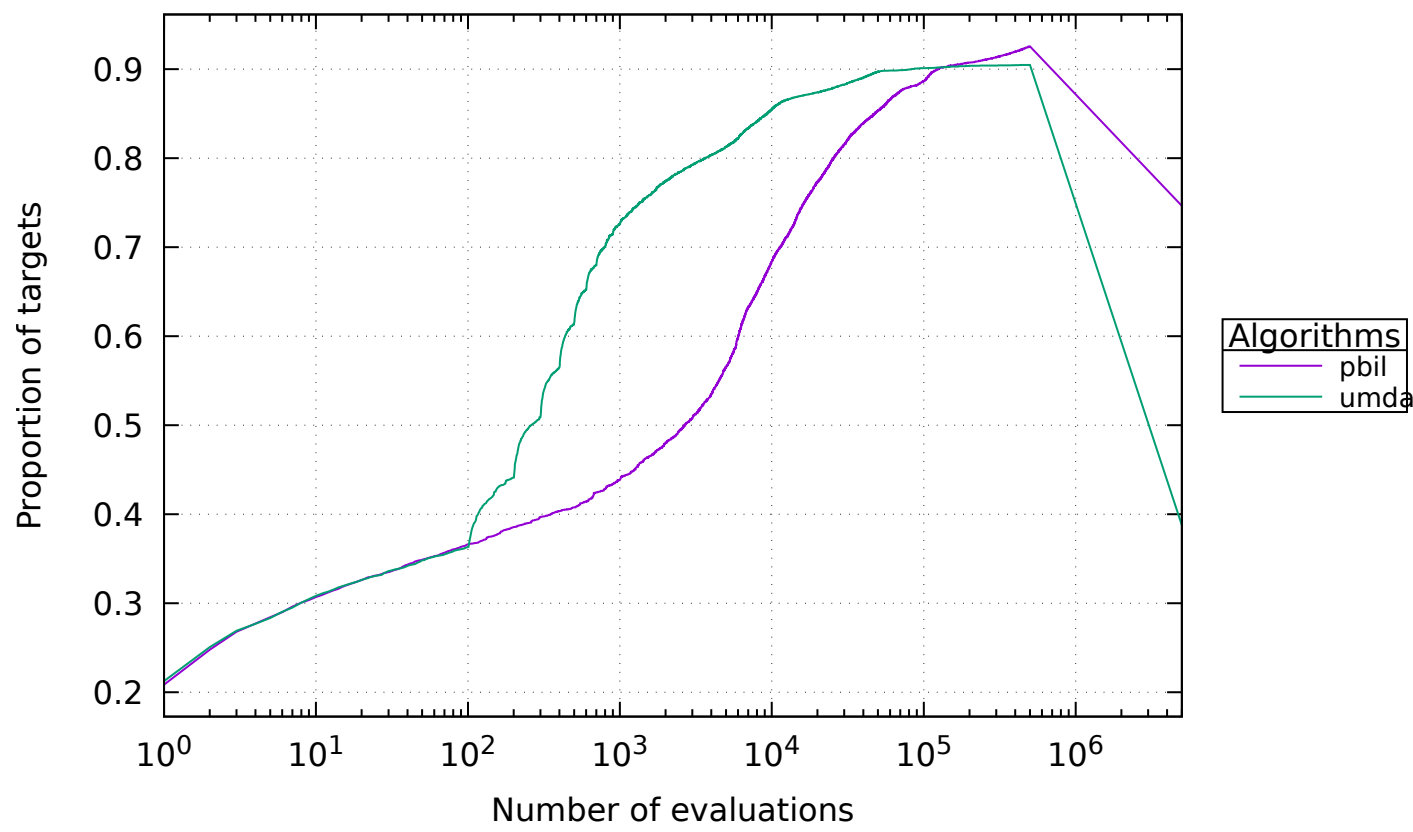


1.2 Groups

1.2.1 ec

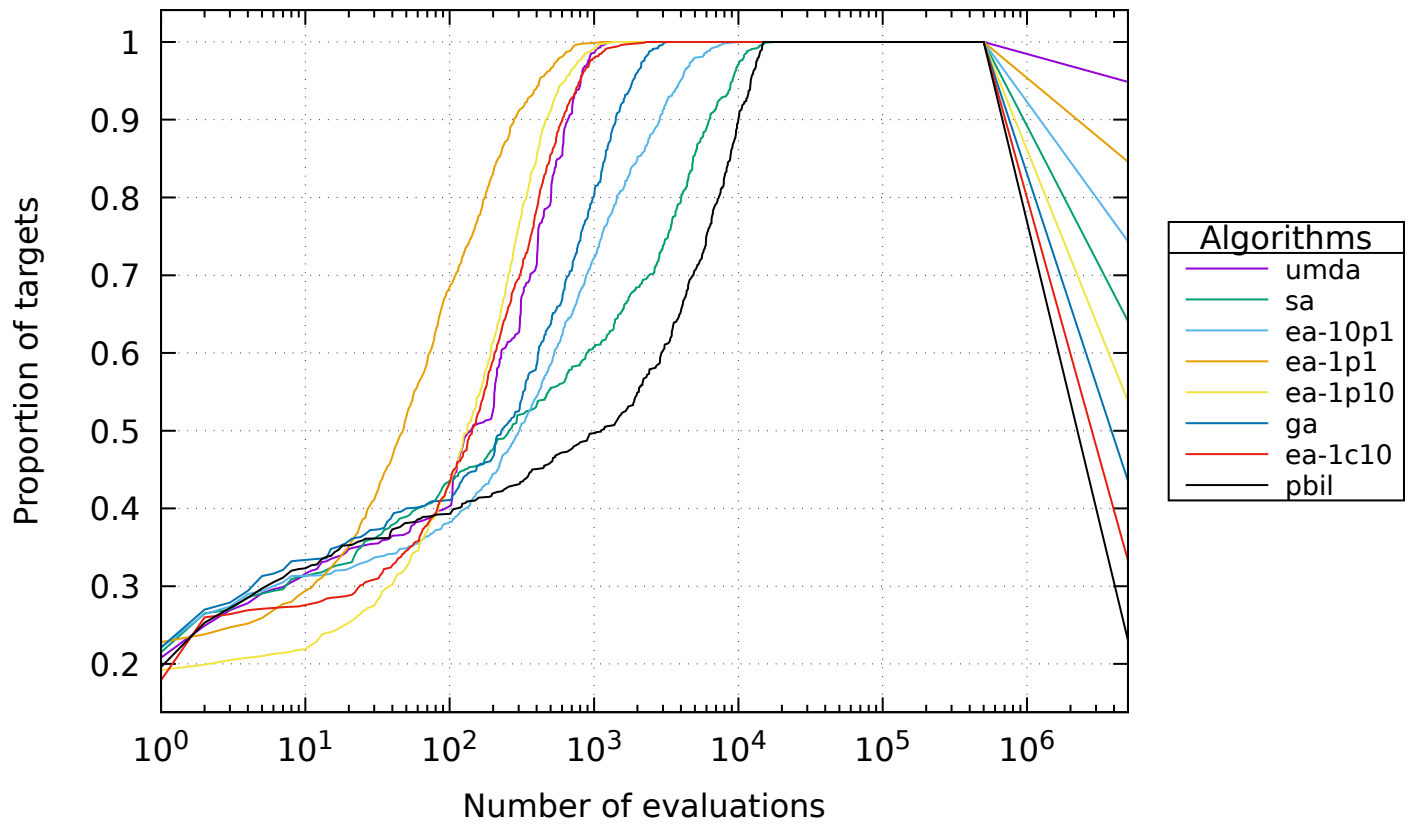


1.2.2 eda



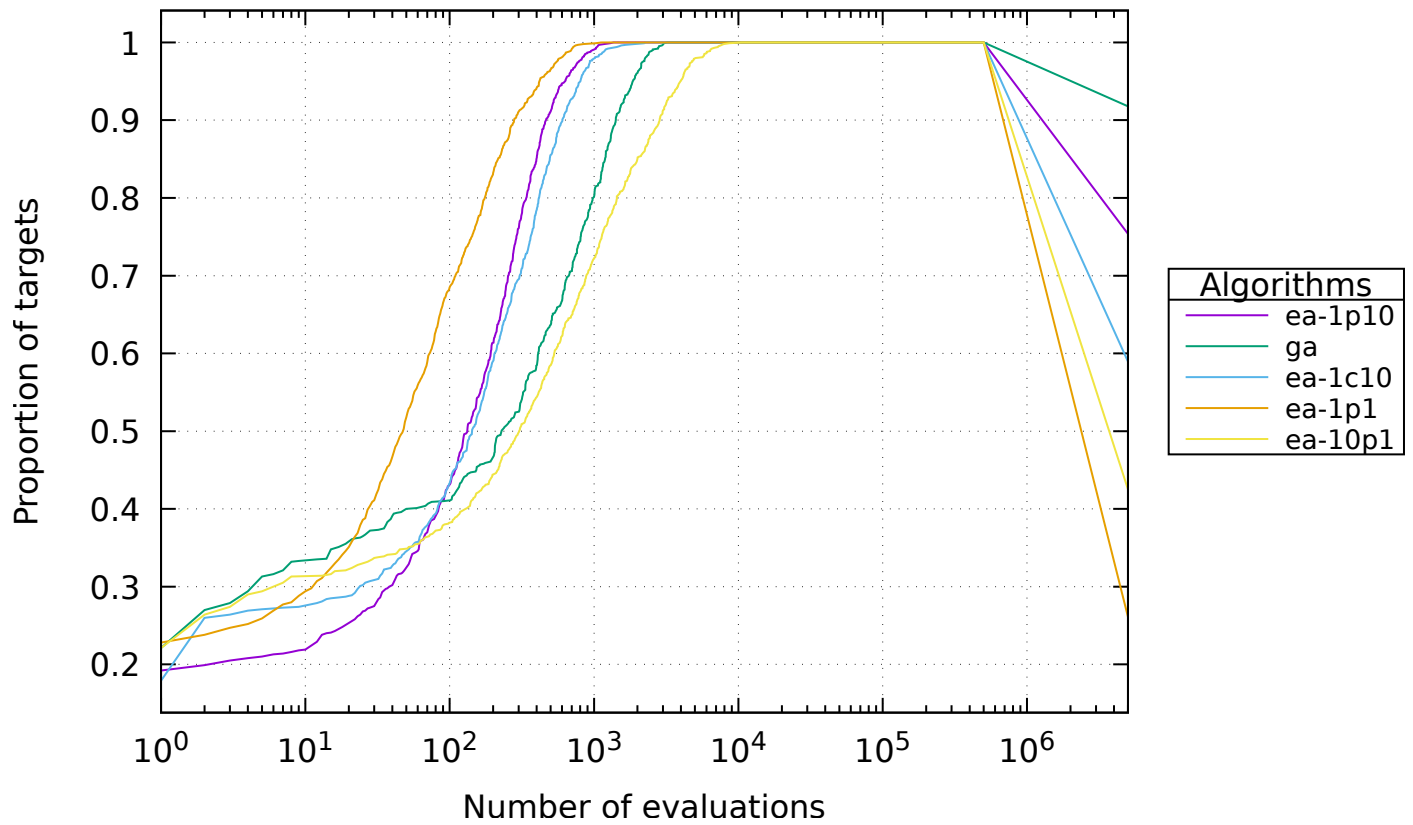
## 2 Results for one-max

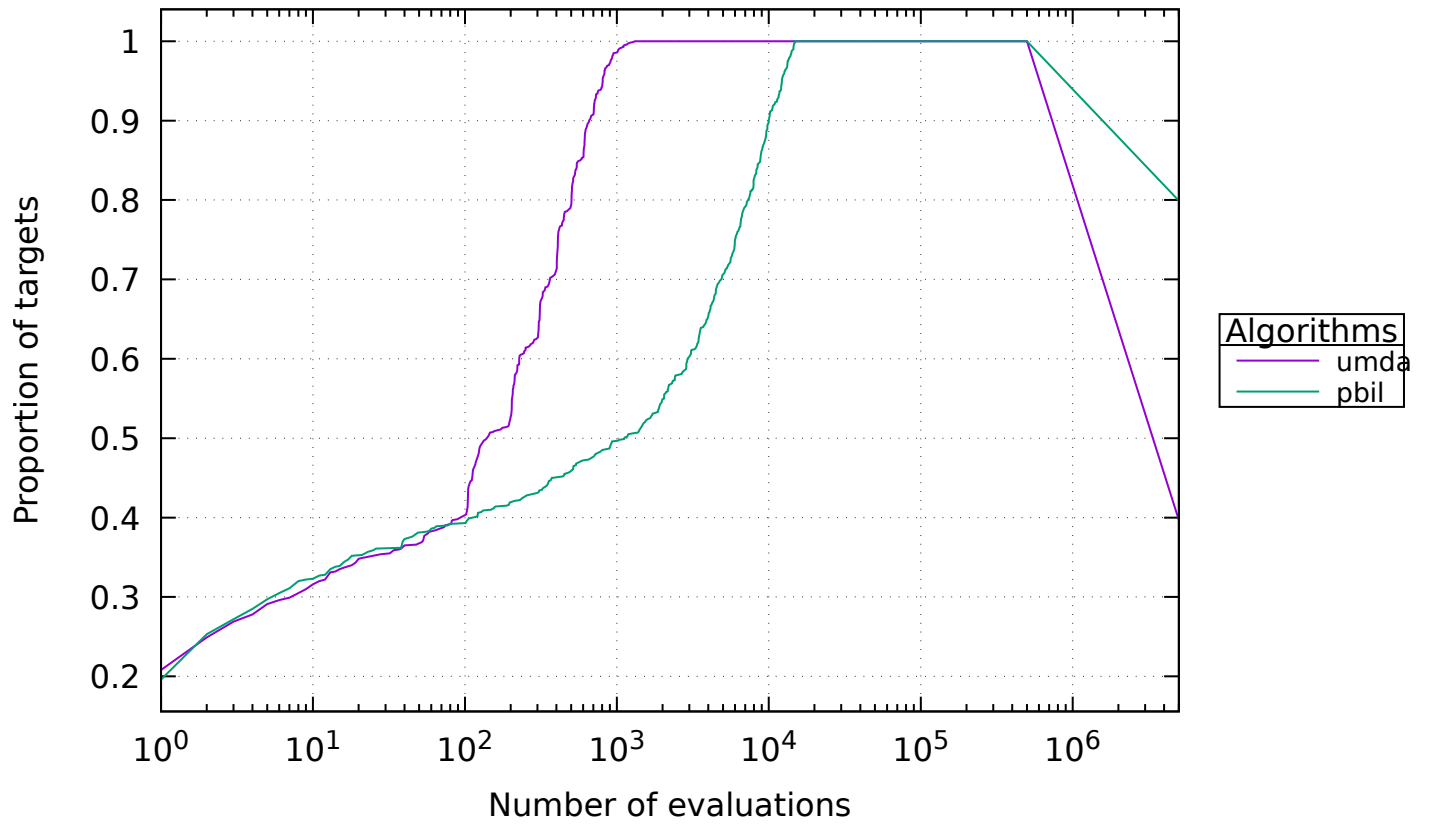
### 2.1 All algorithms



### 2.2 Groups

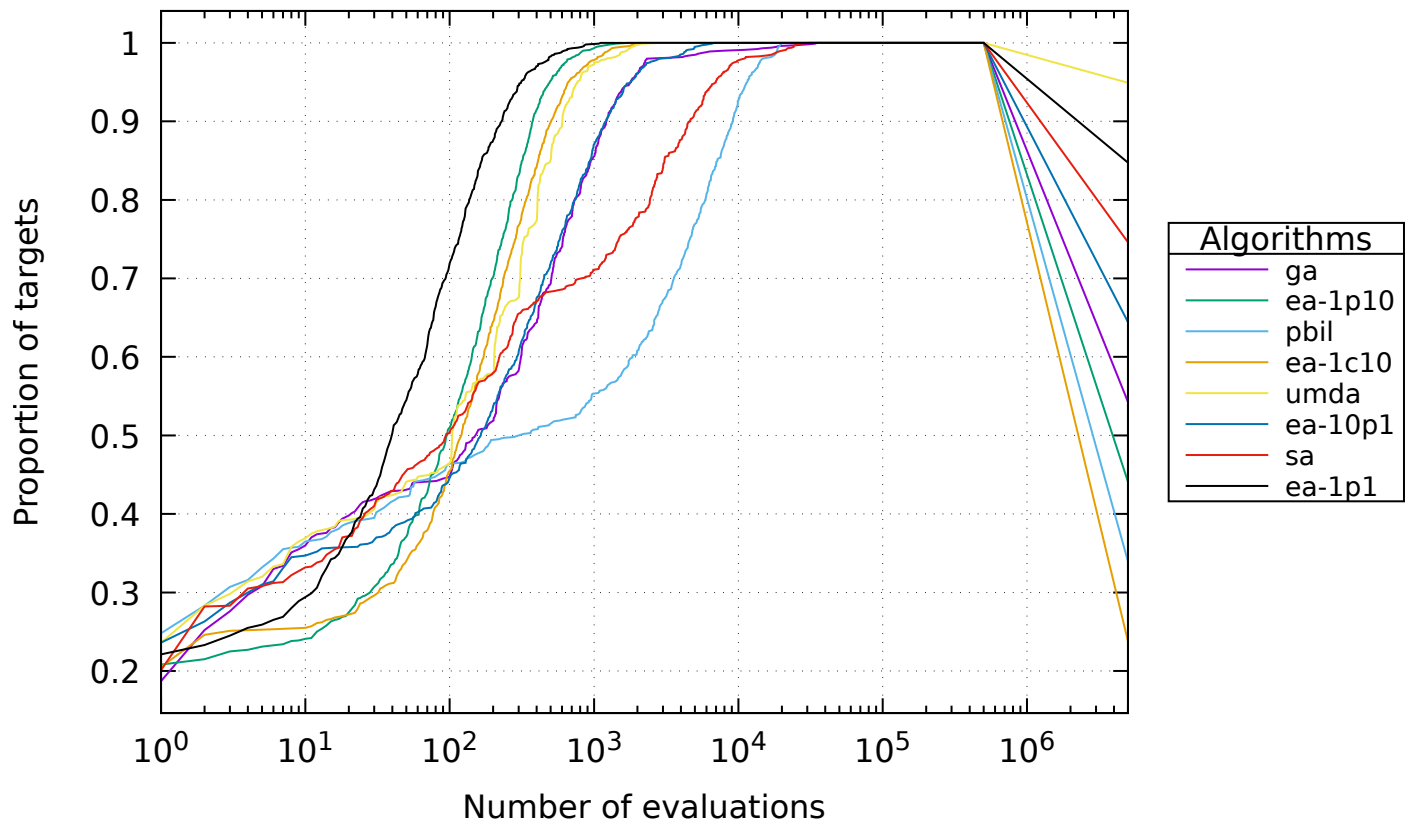
#### 2.2.1 ec





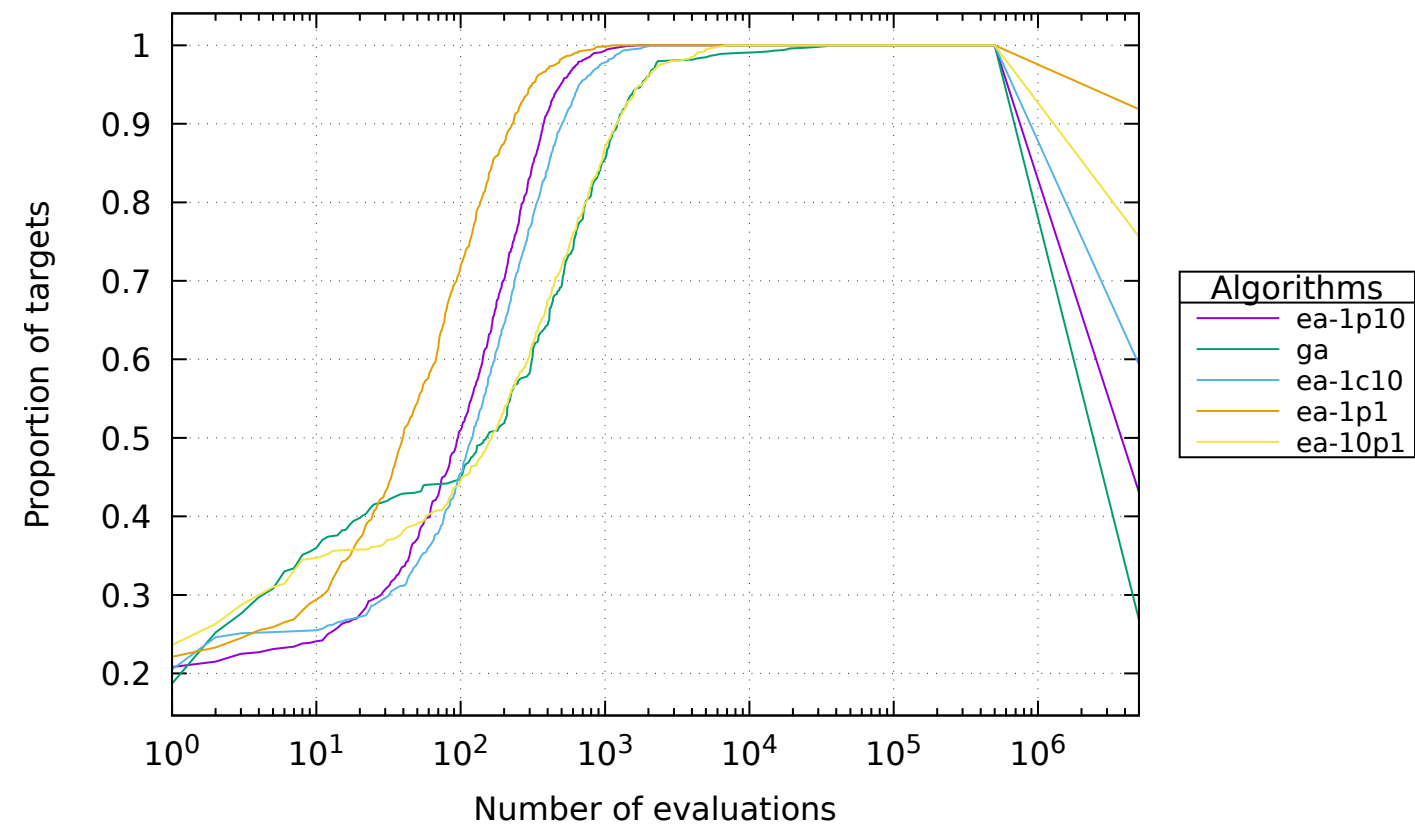
### 3 Results for lin

#### 3.1 All algorithms

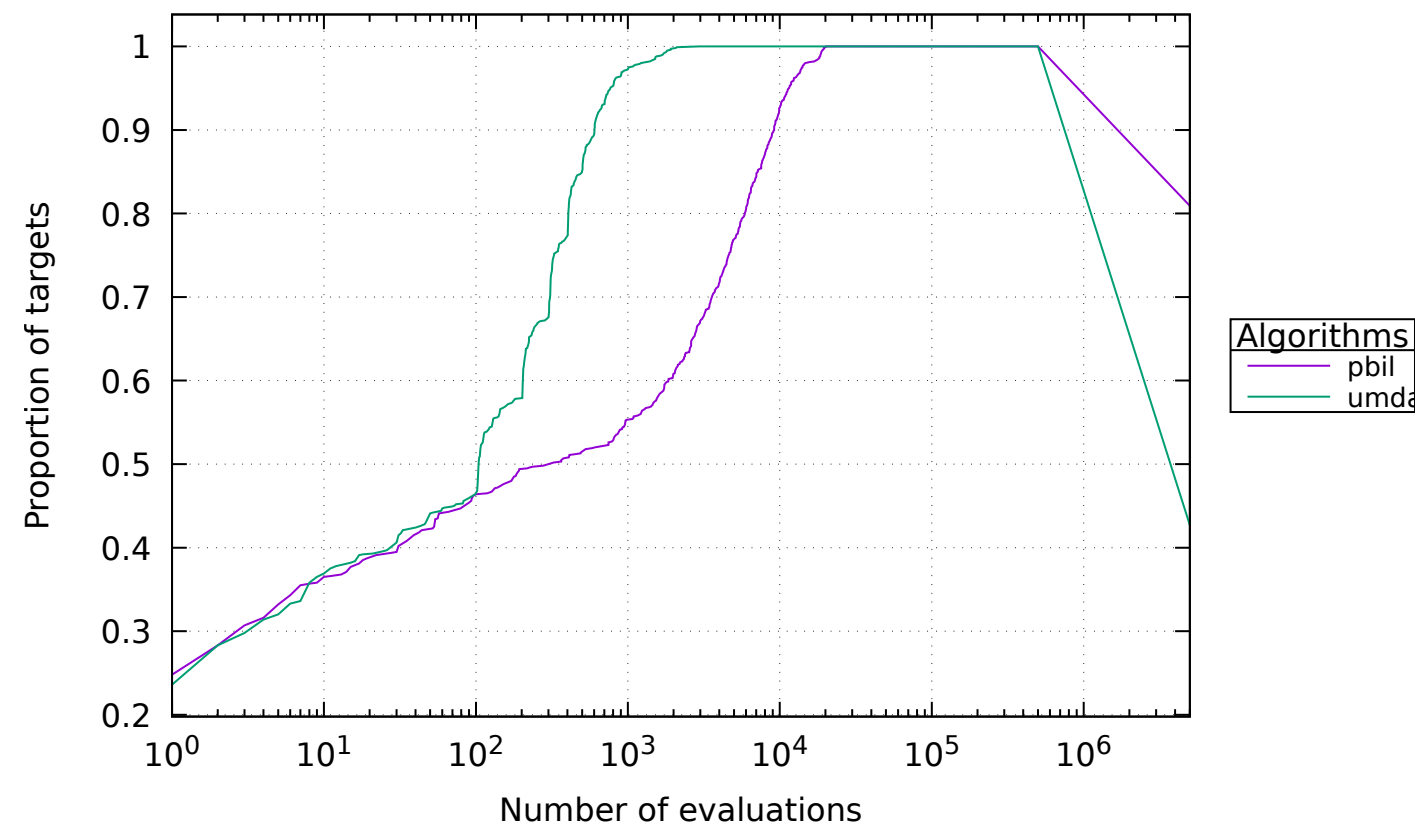


3.2 Groups

3.2.1 ec

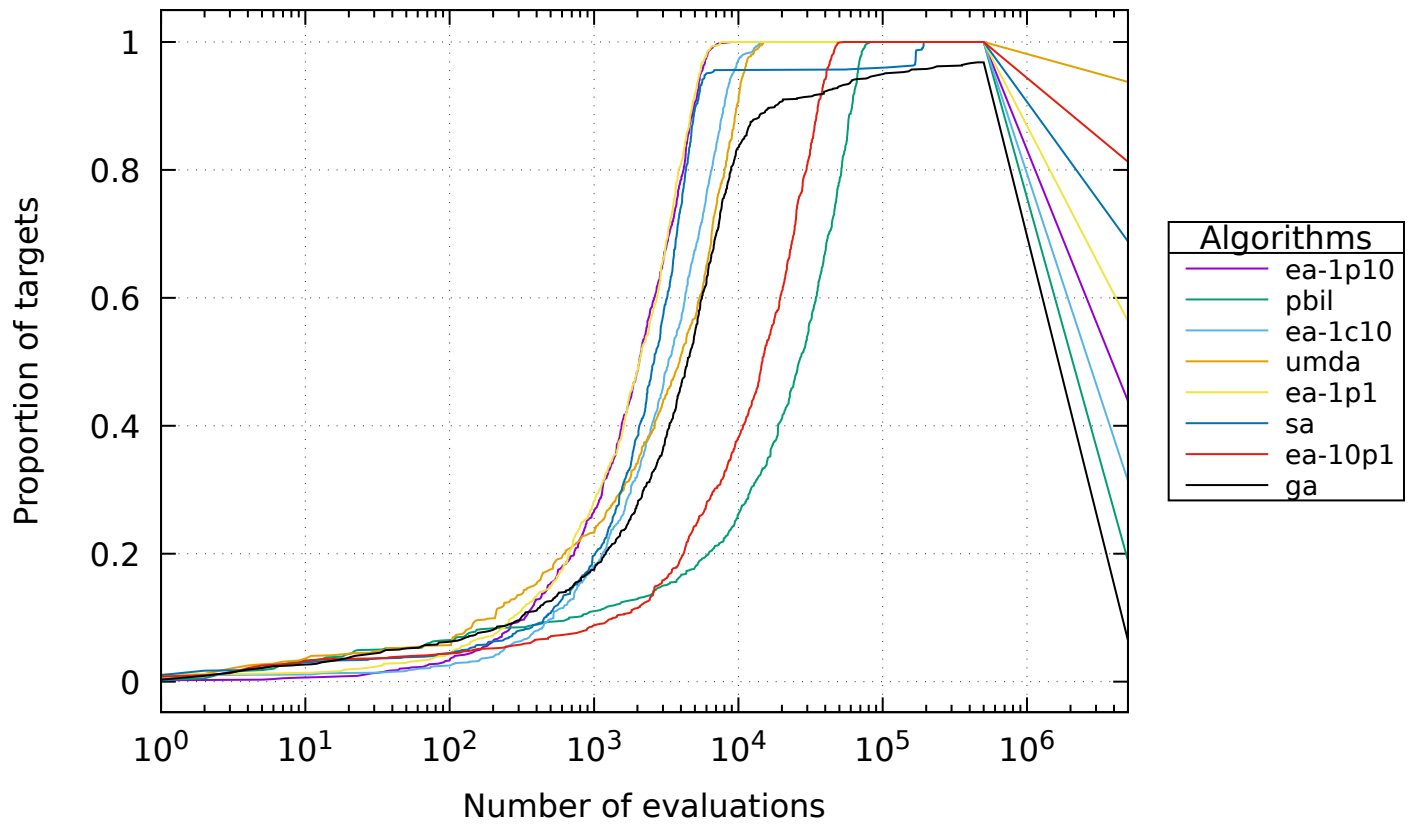


3.2.2 eda



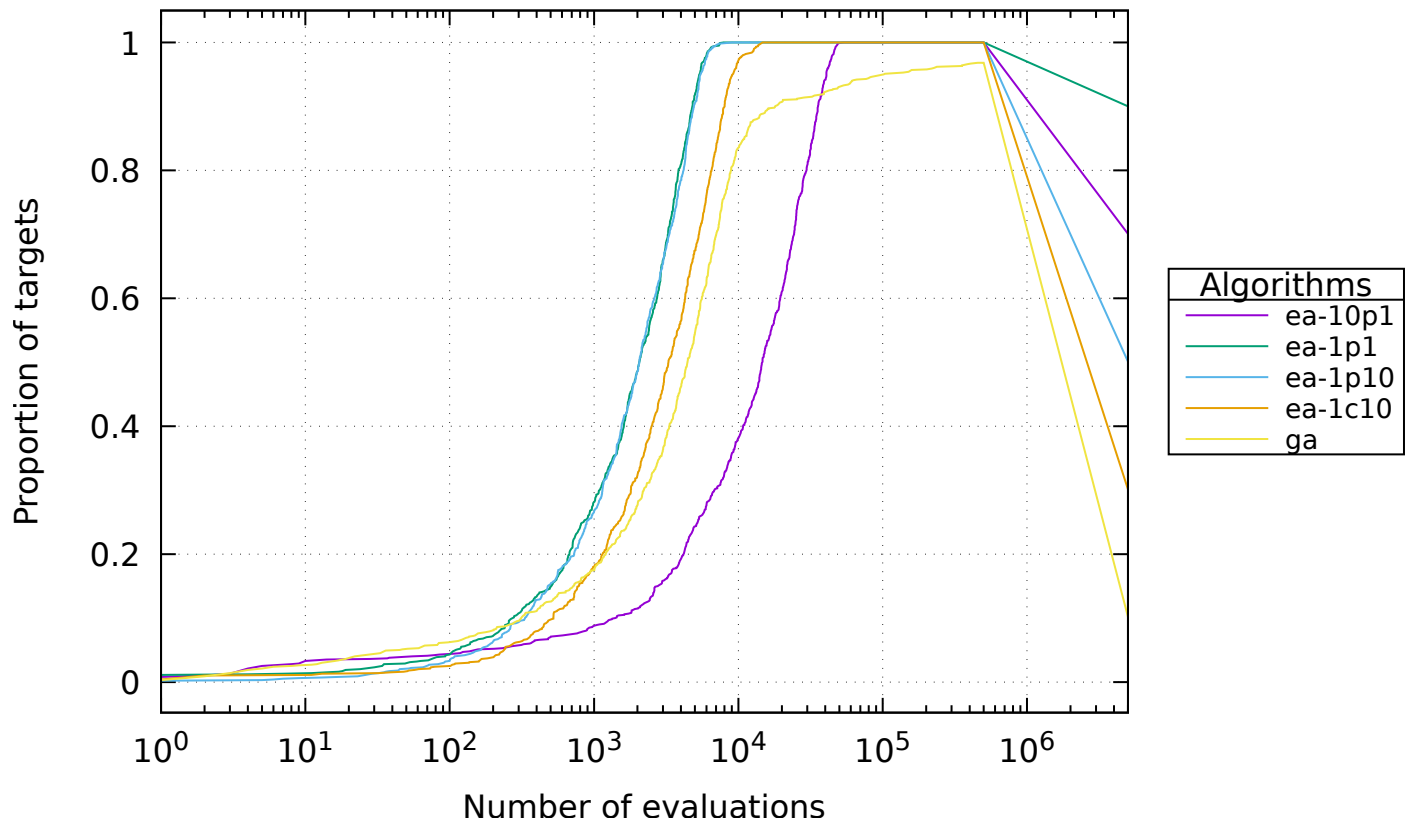
## 4 Results for leading-ones

### 4.1 All algorithms

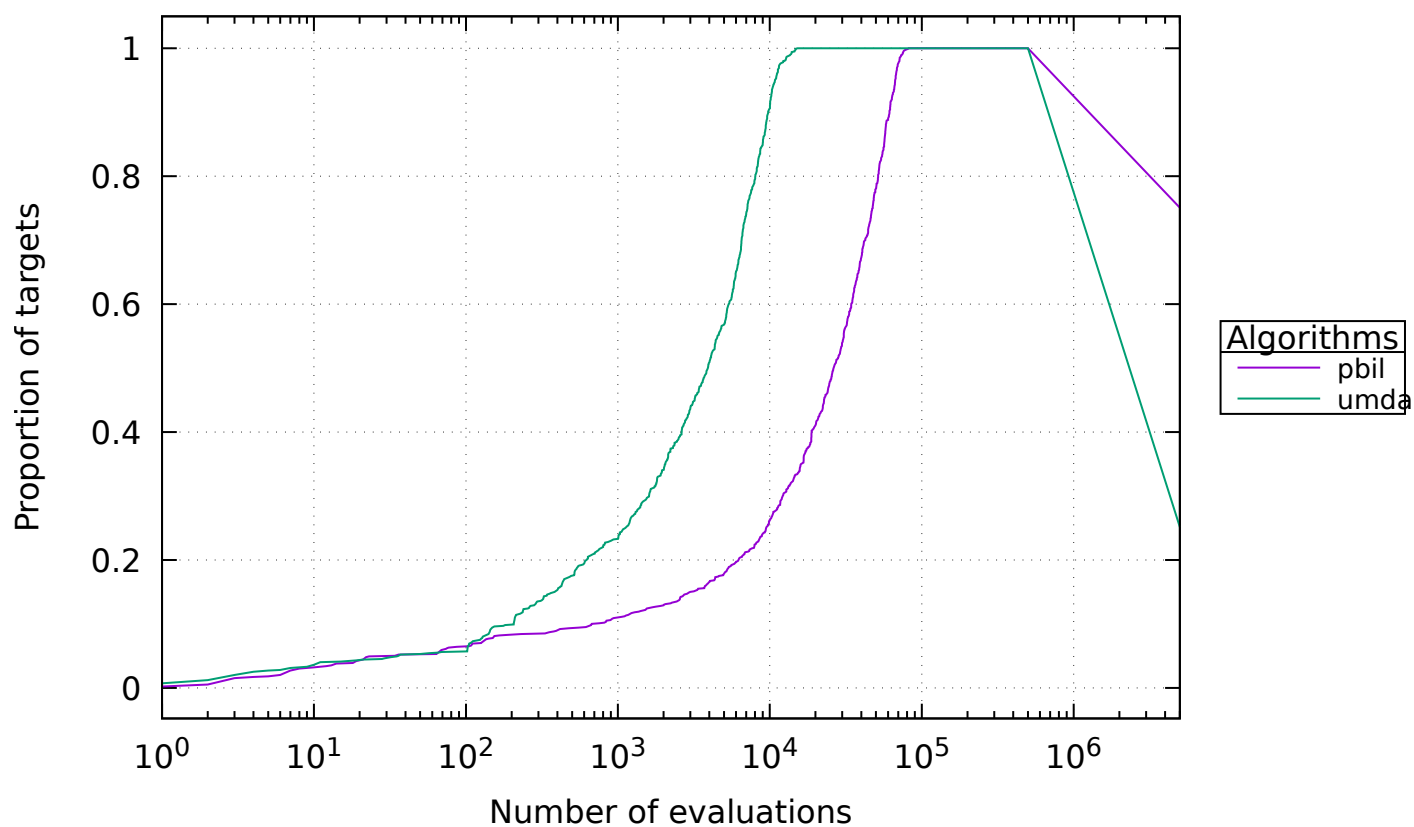


### 4.2 Groups

#### 4.2.1 ec

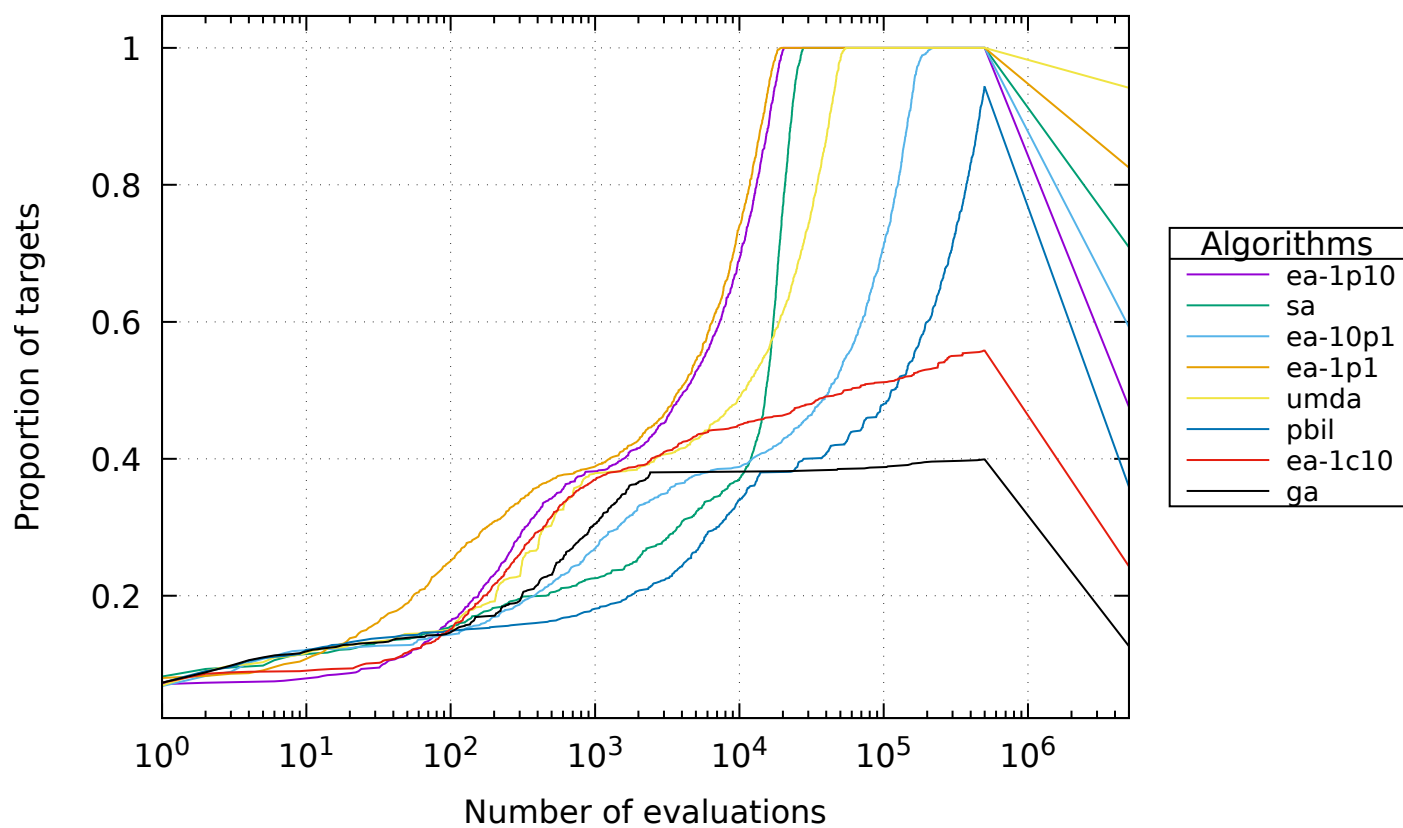






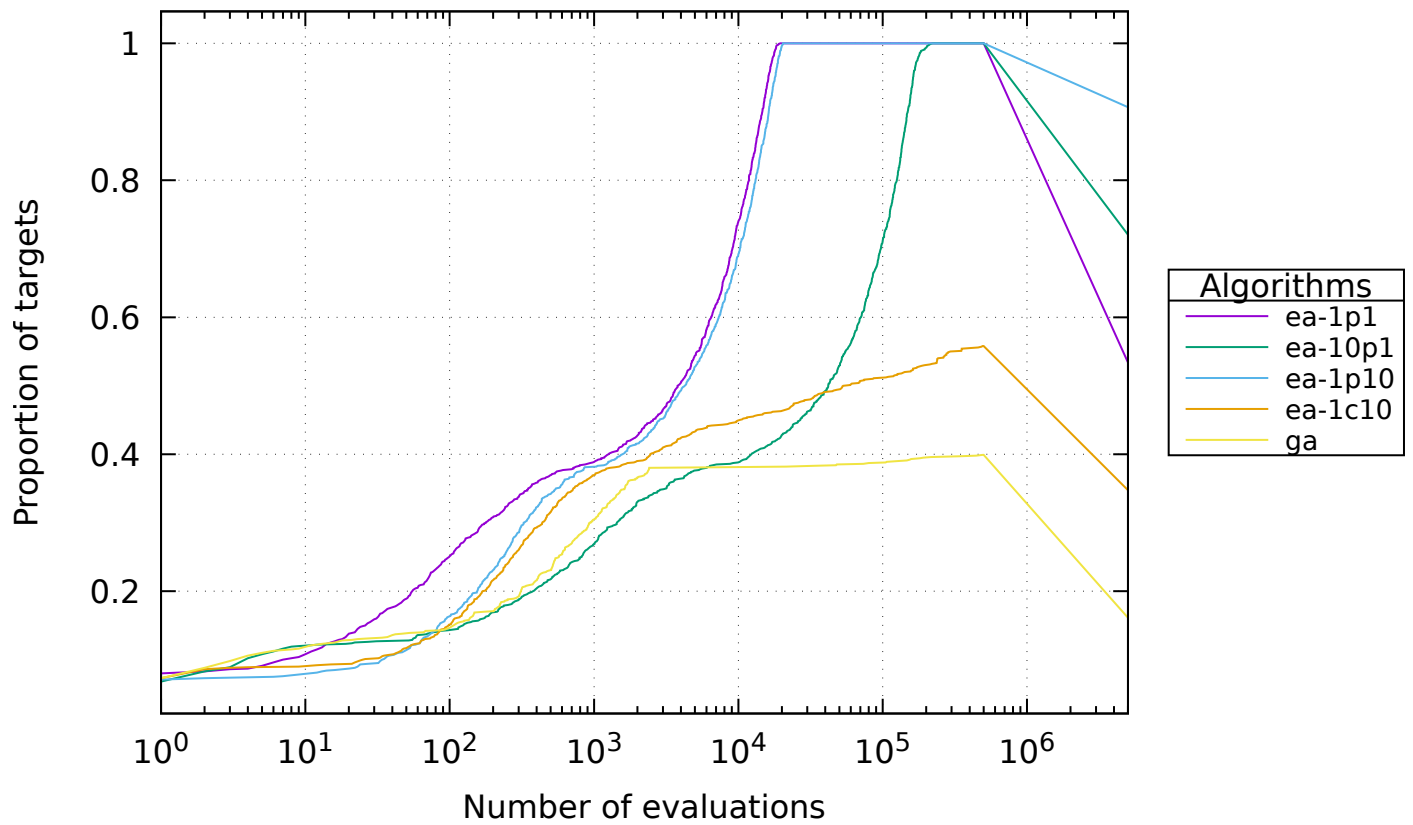
## 5 Results for ridge

### 5.1 All algorithms

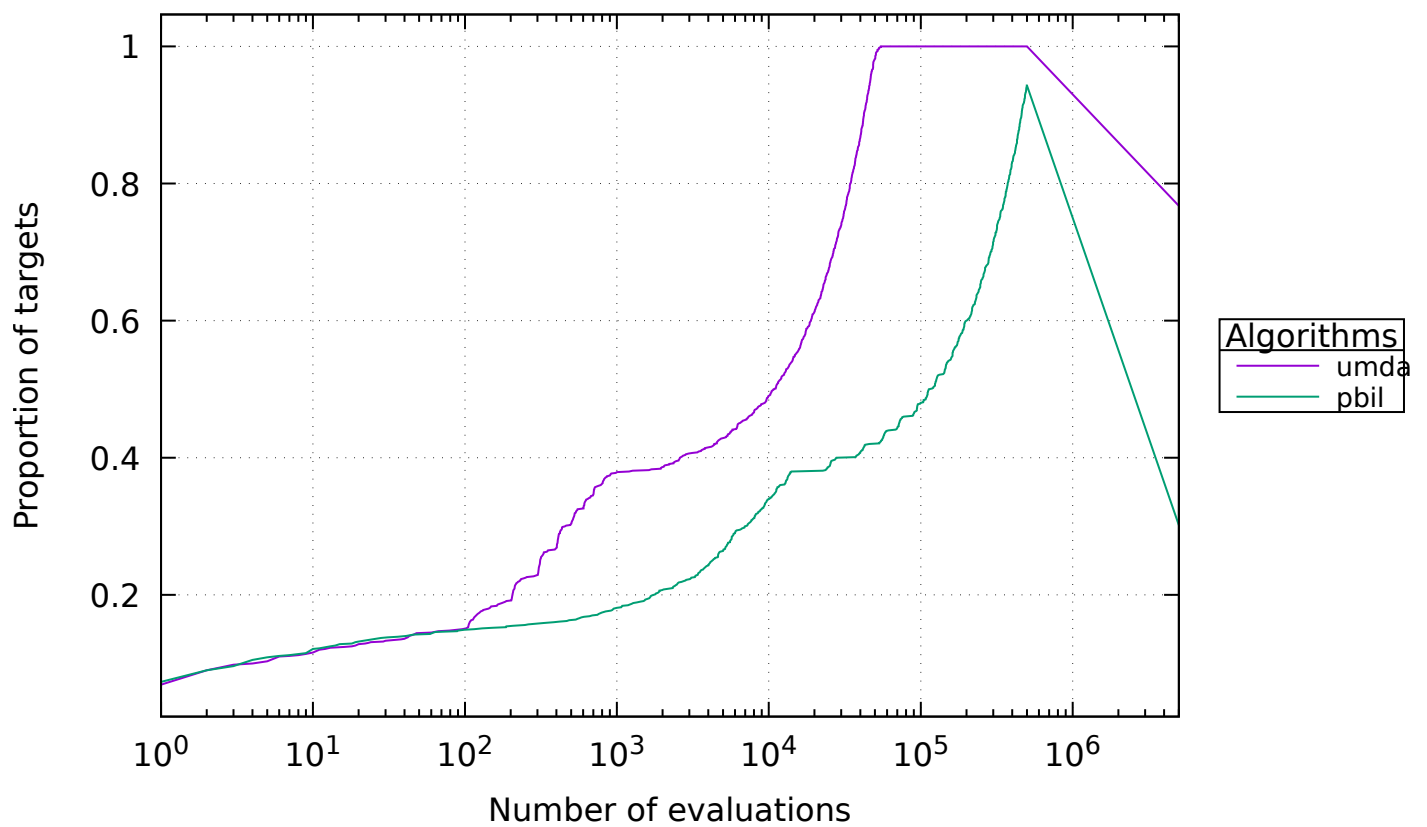


## 5.2 Groups

### 5.2.1 ec

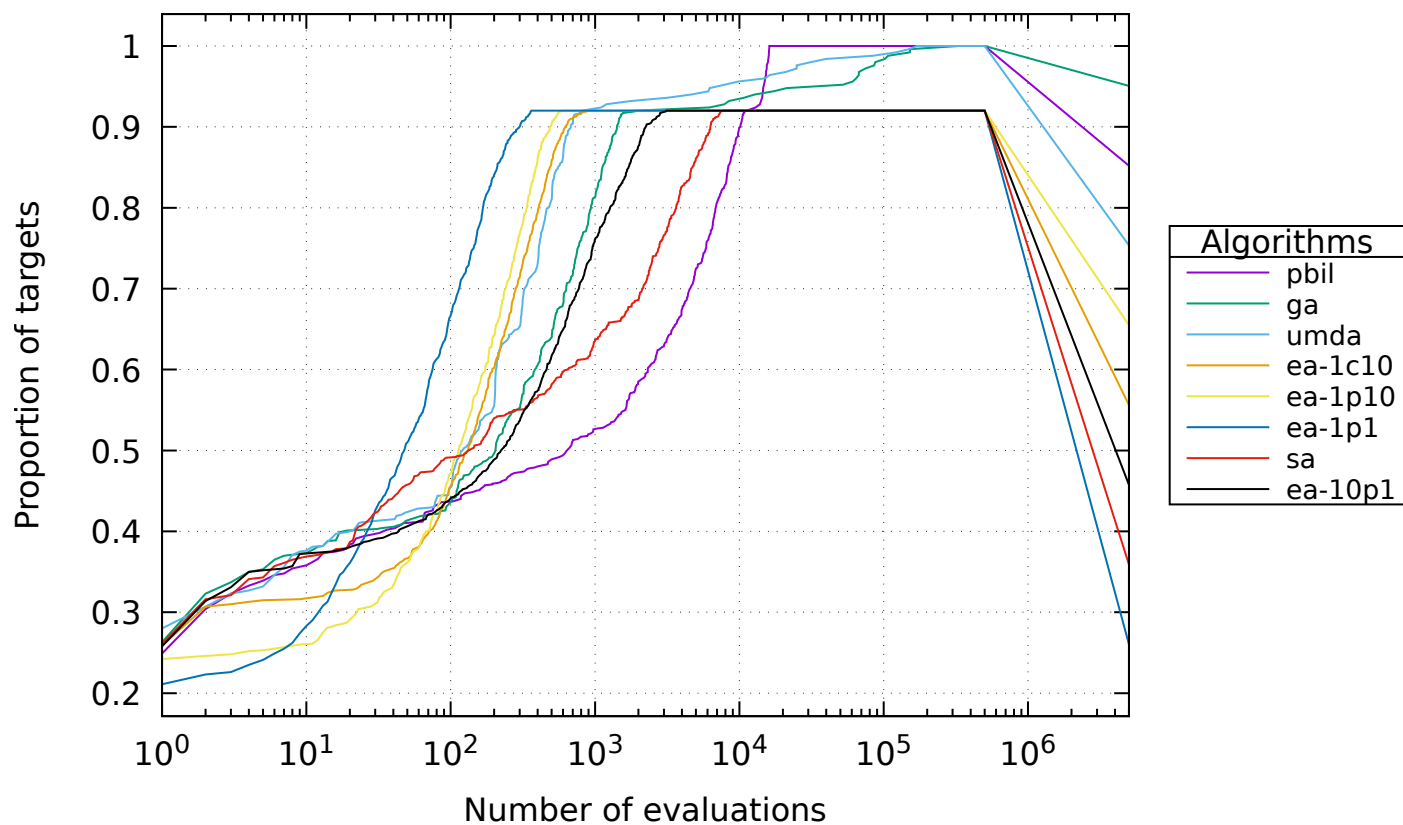


### 5.2.2 eda



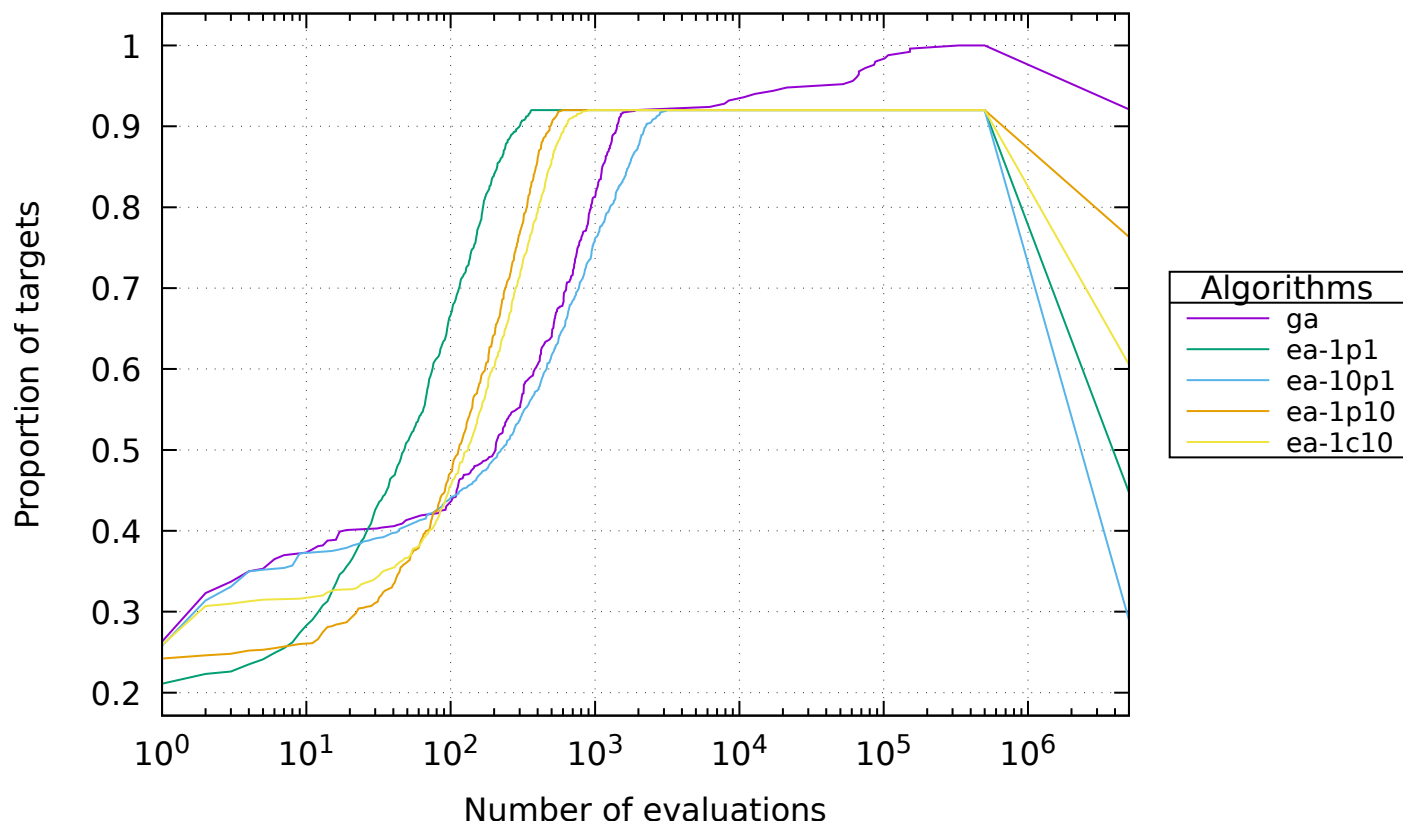
## 6 Results for jmp-5

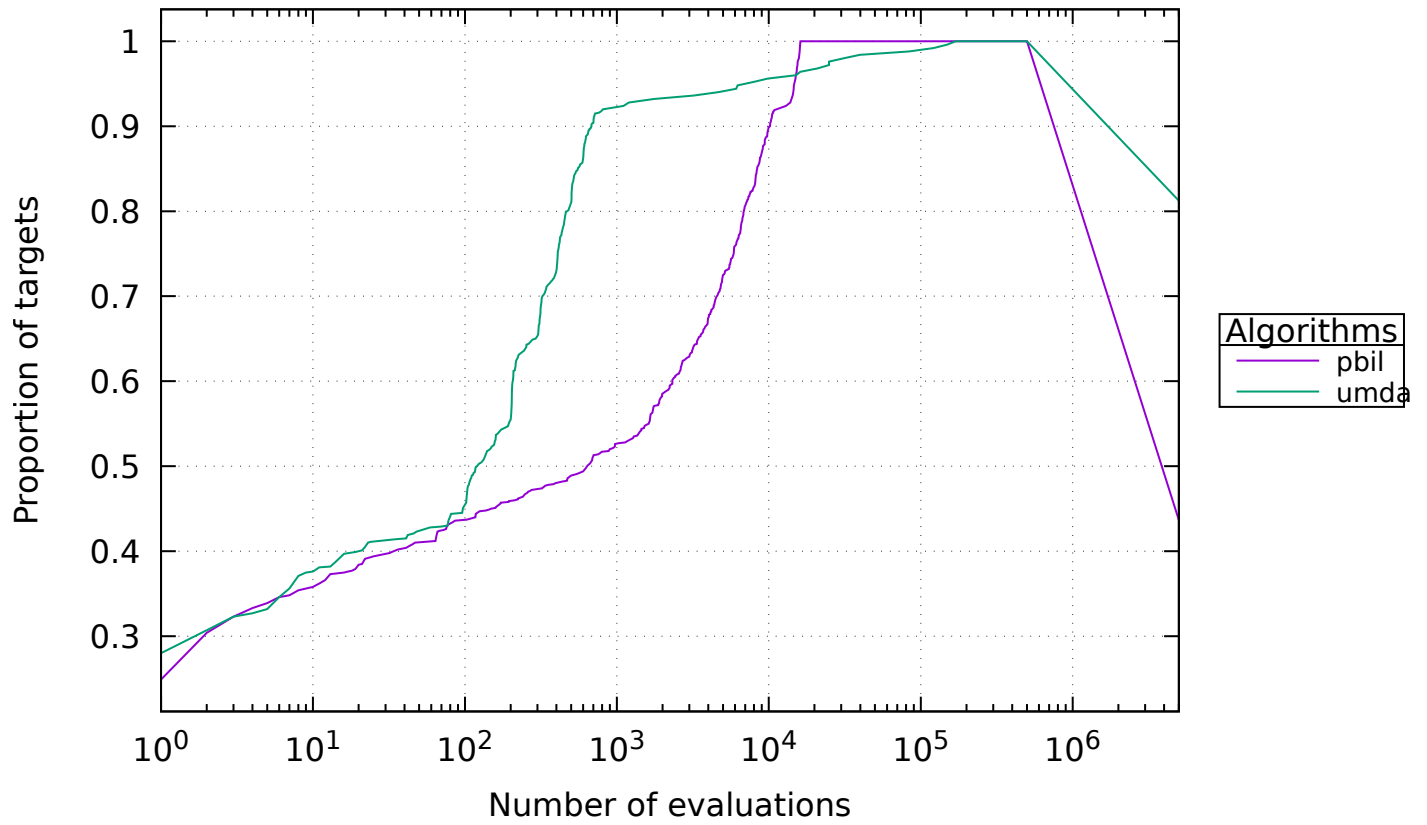
### 6.1 All algorithms



### 6.2 Groups

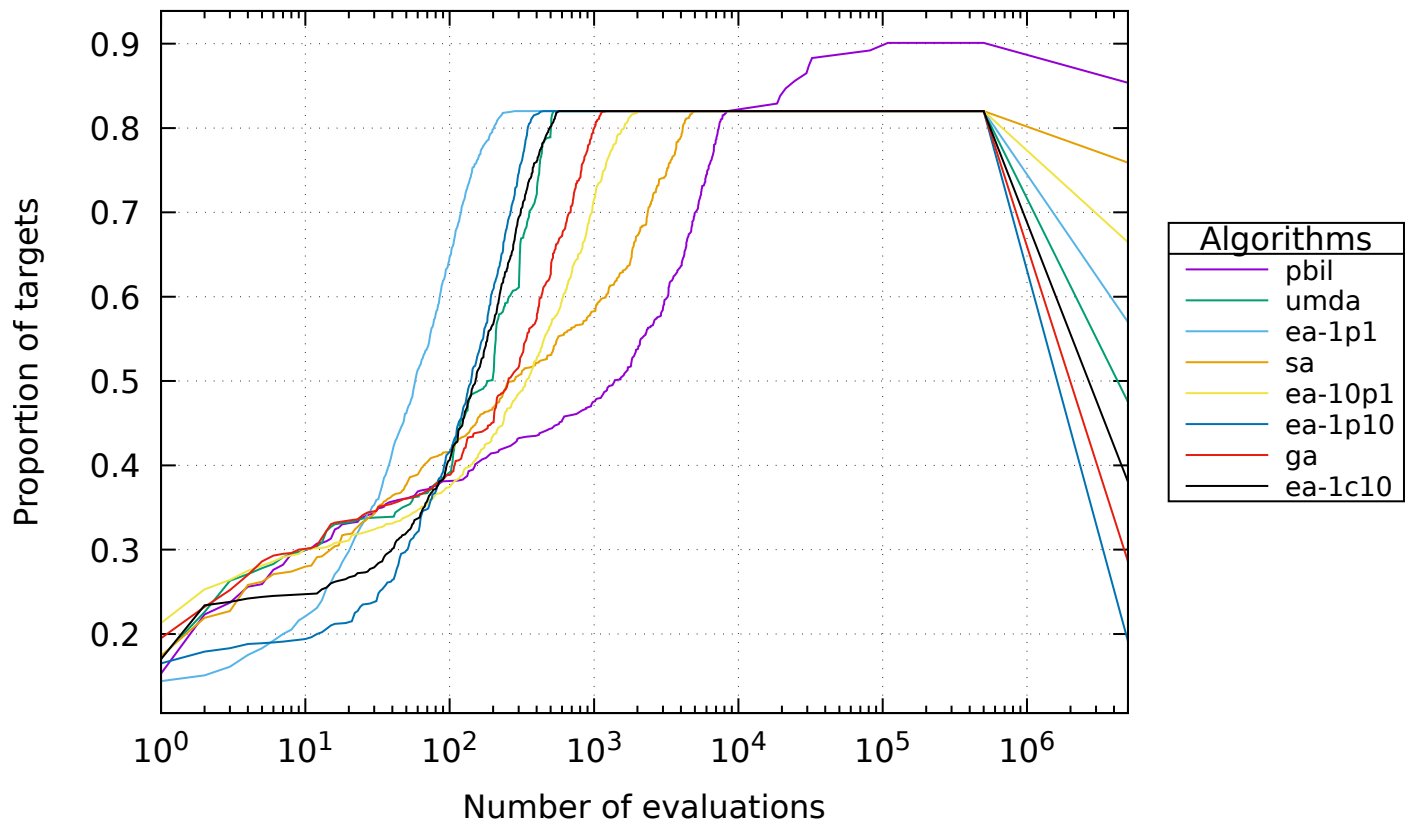
#### 6.2.1 ec





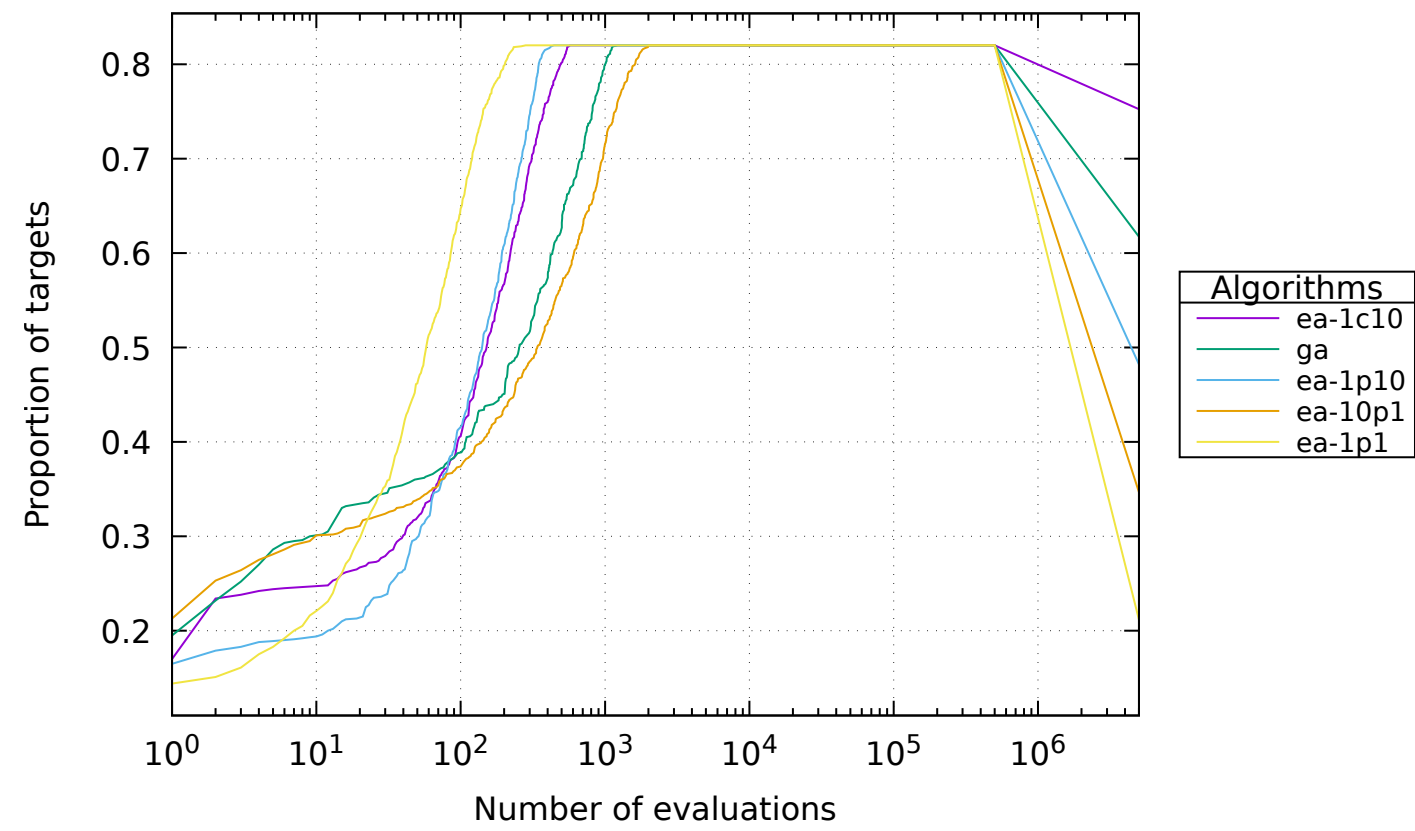
## 7 Results for jmp-10

### 7.1 All algorithms

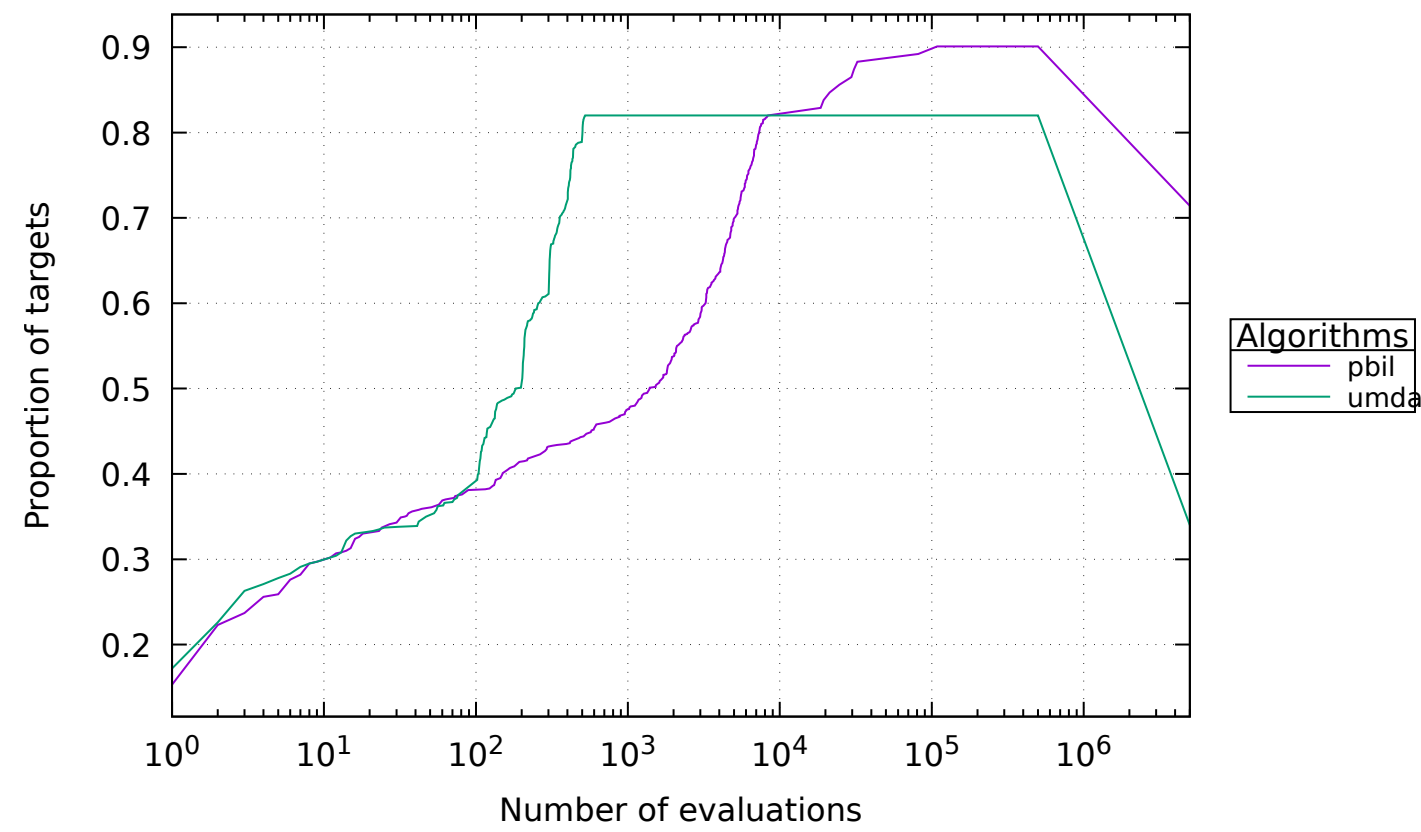


7.2 Groups

7.2.1 ec

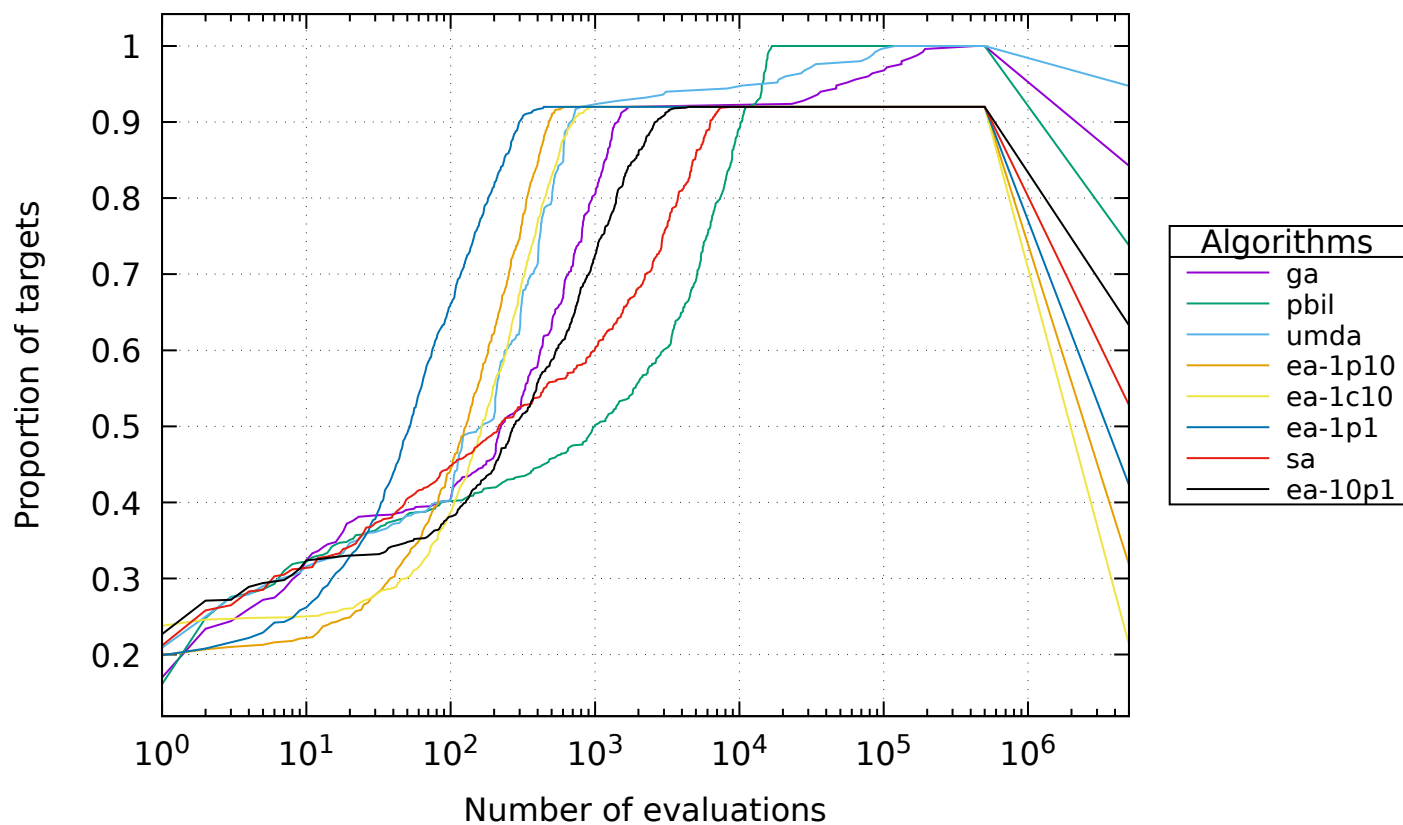


7.2.2 eda



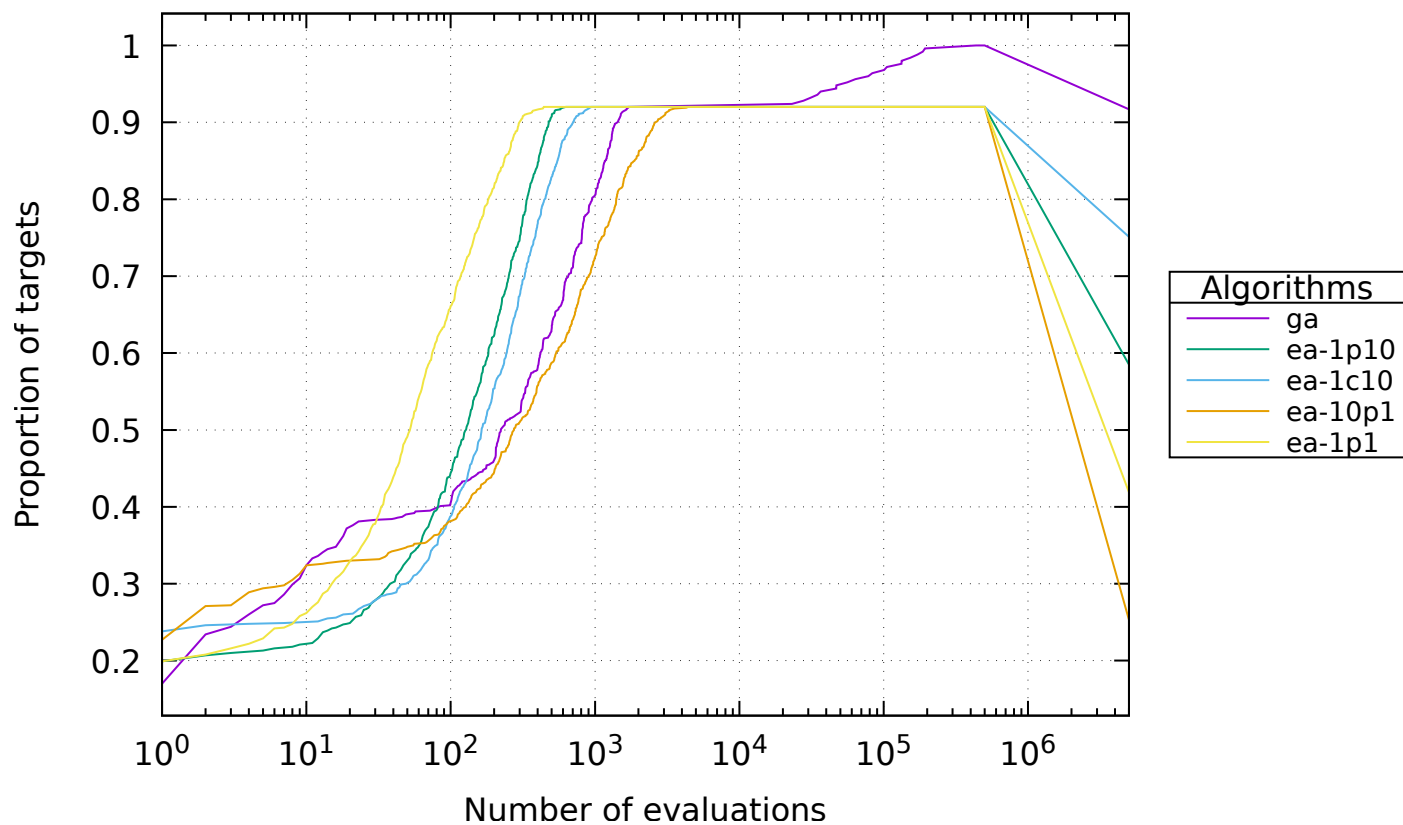
## 8 Results for djmp-5

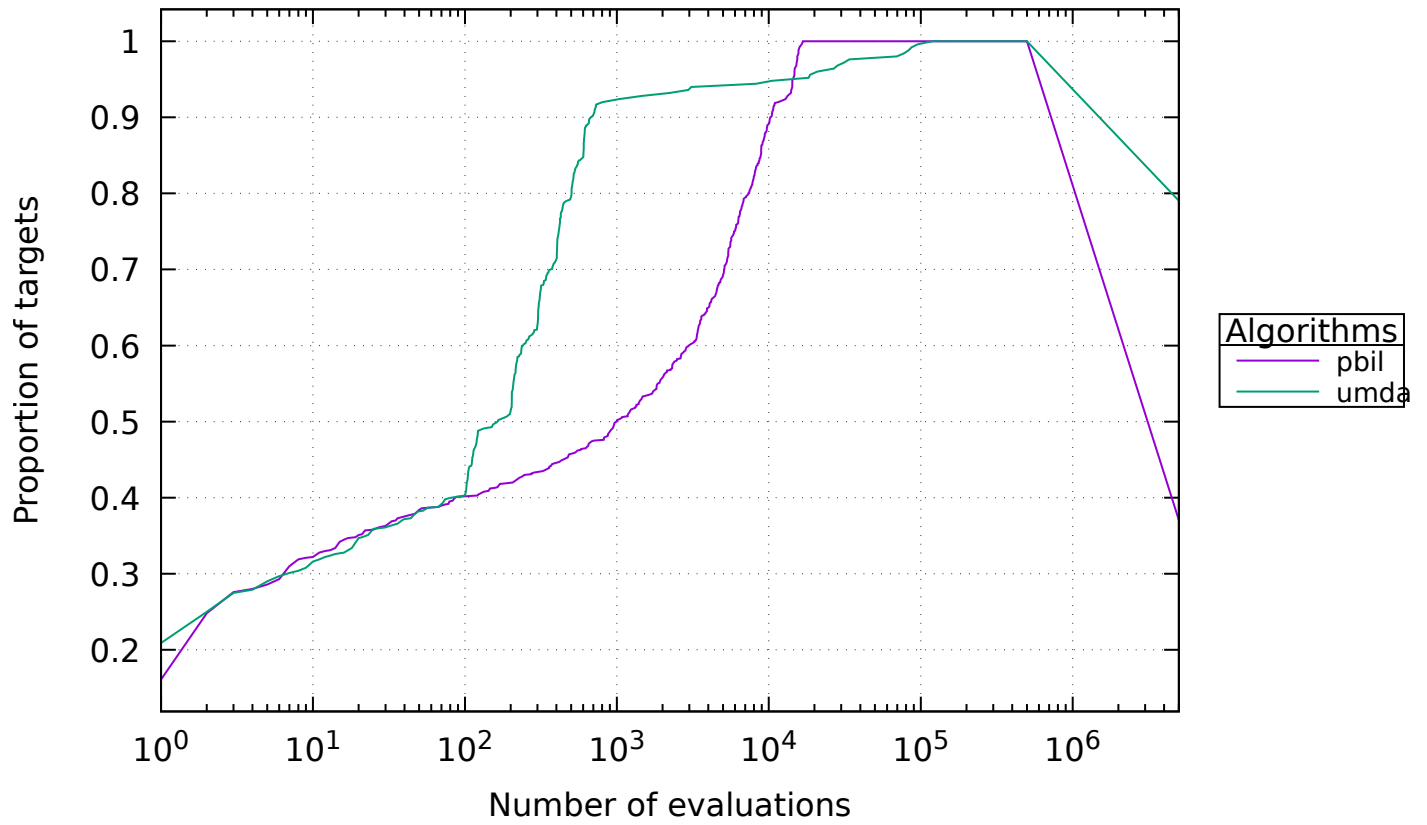
### 8.1 All algorithms



### 8.2 Groups

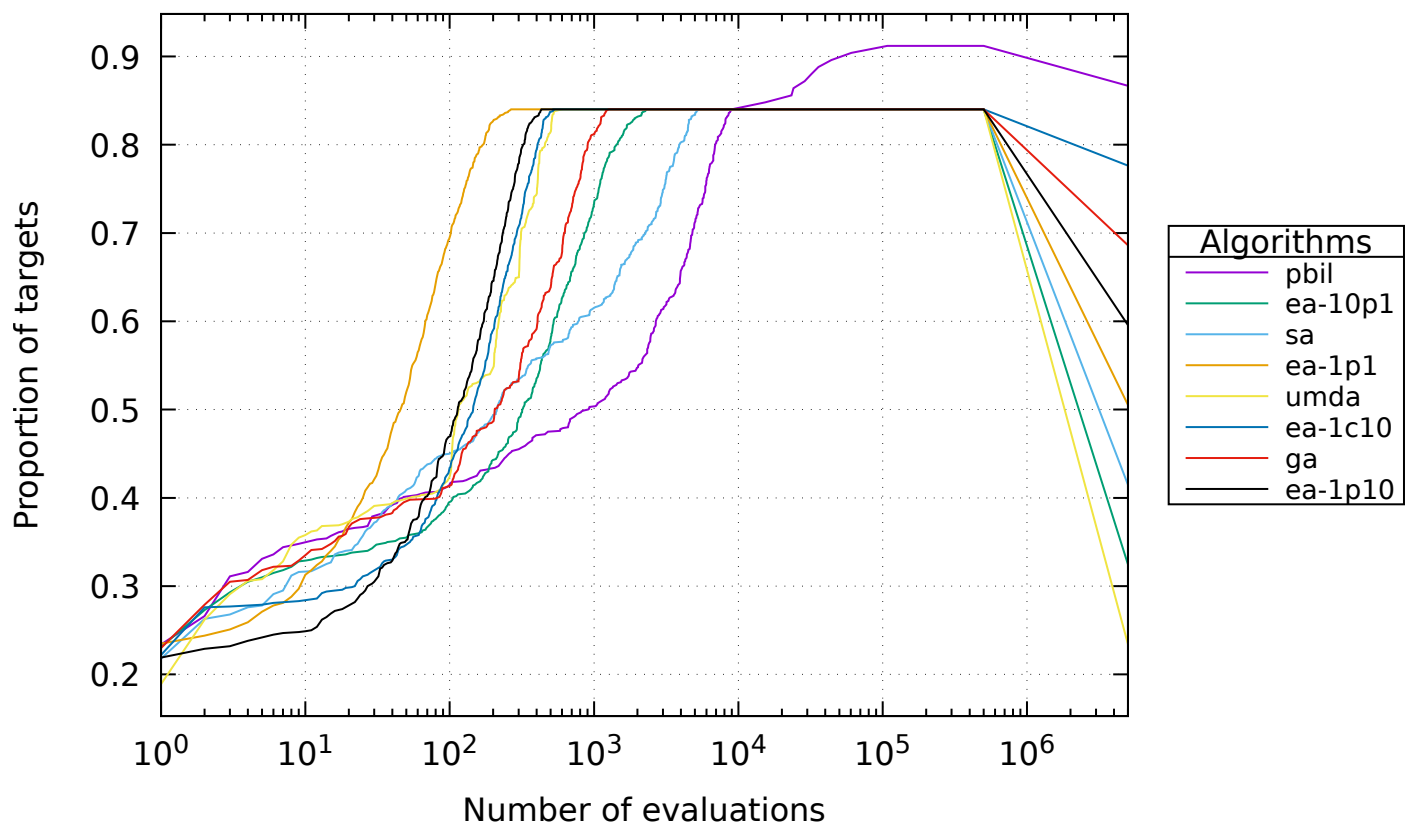
#### 8.2.1 ec





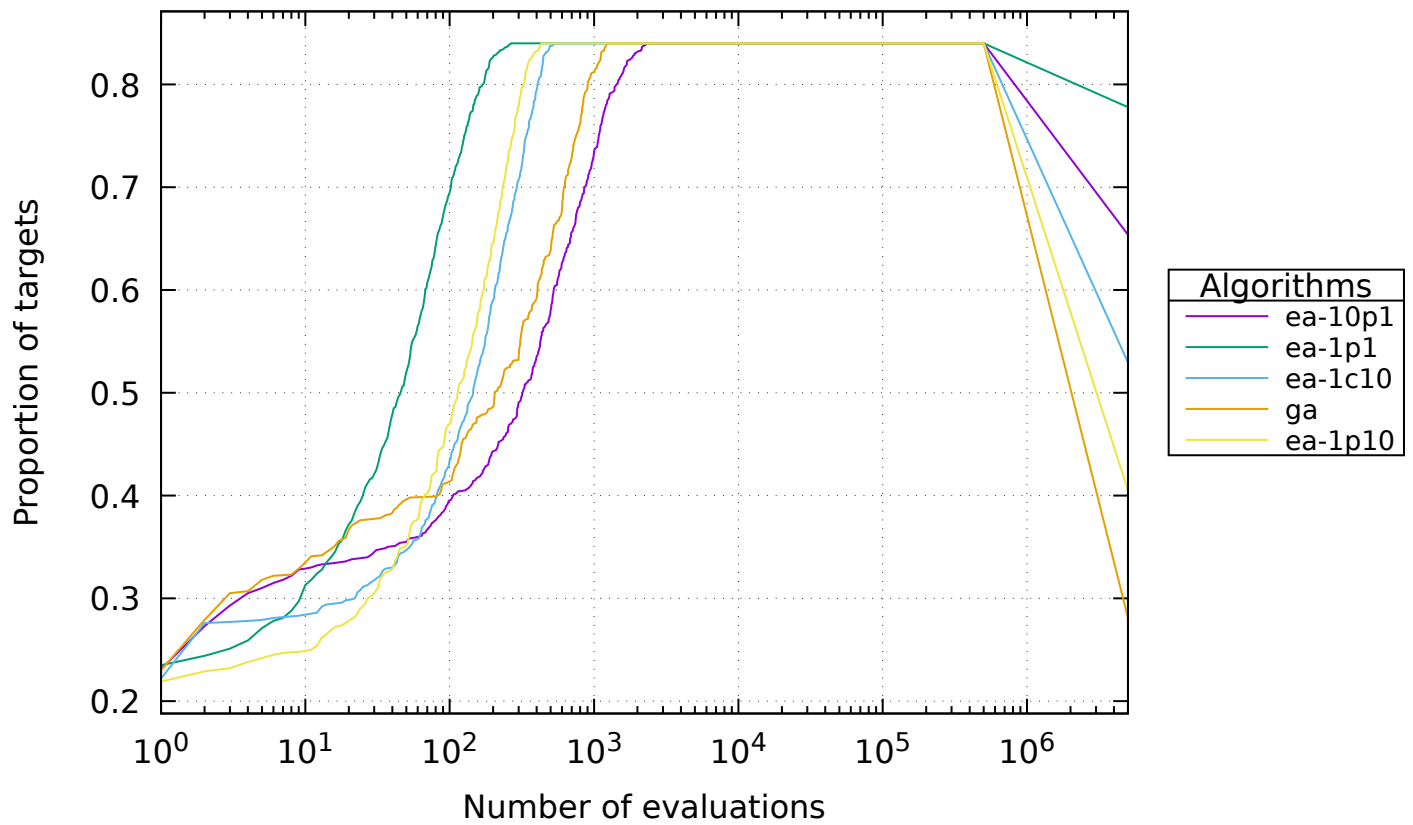
## 9 Results for djmp-10

### 9.1 All algorithms

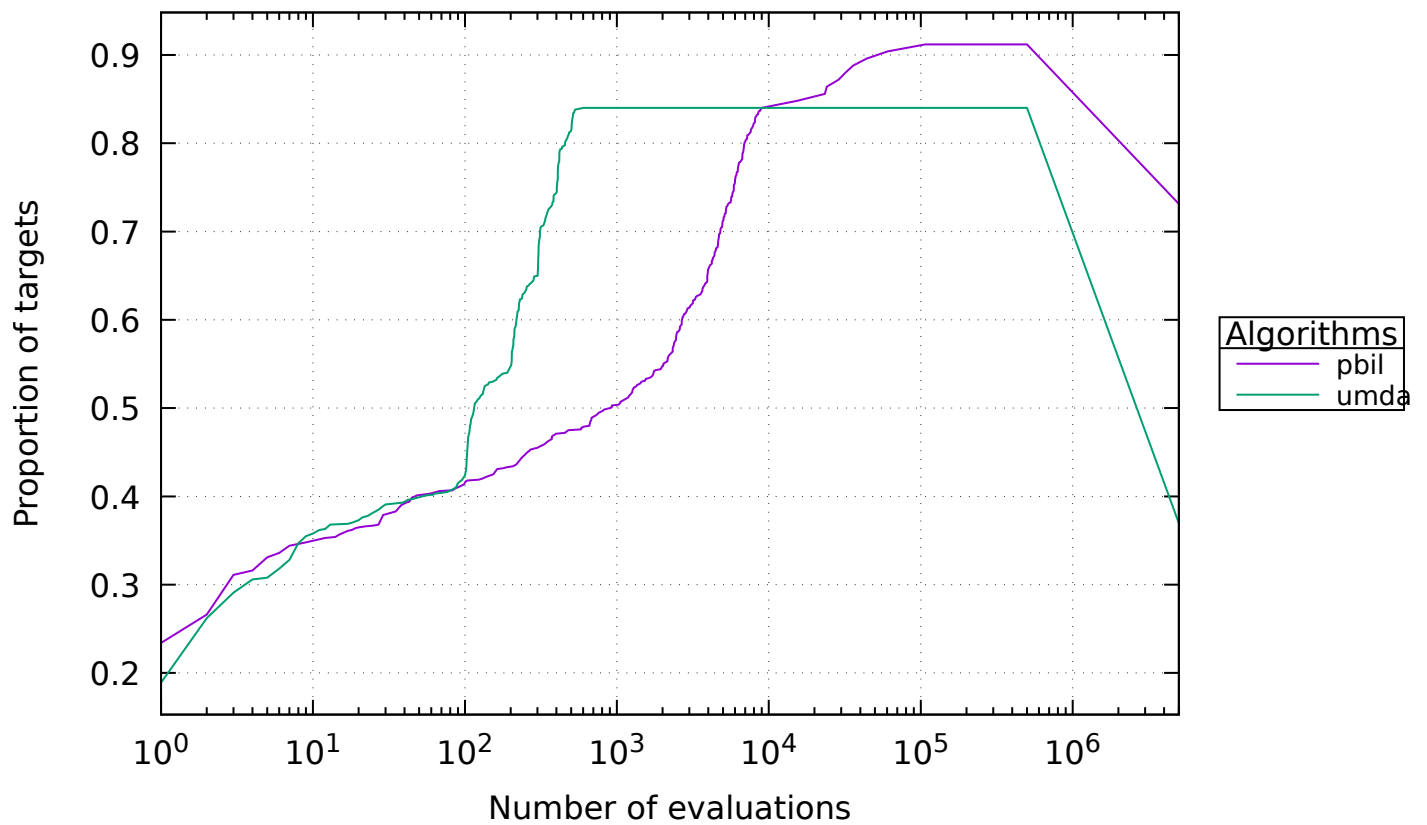


## 9.2 Groups

### 9.2.1 ec



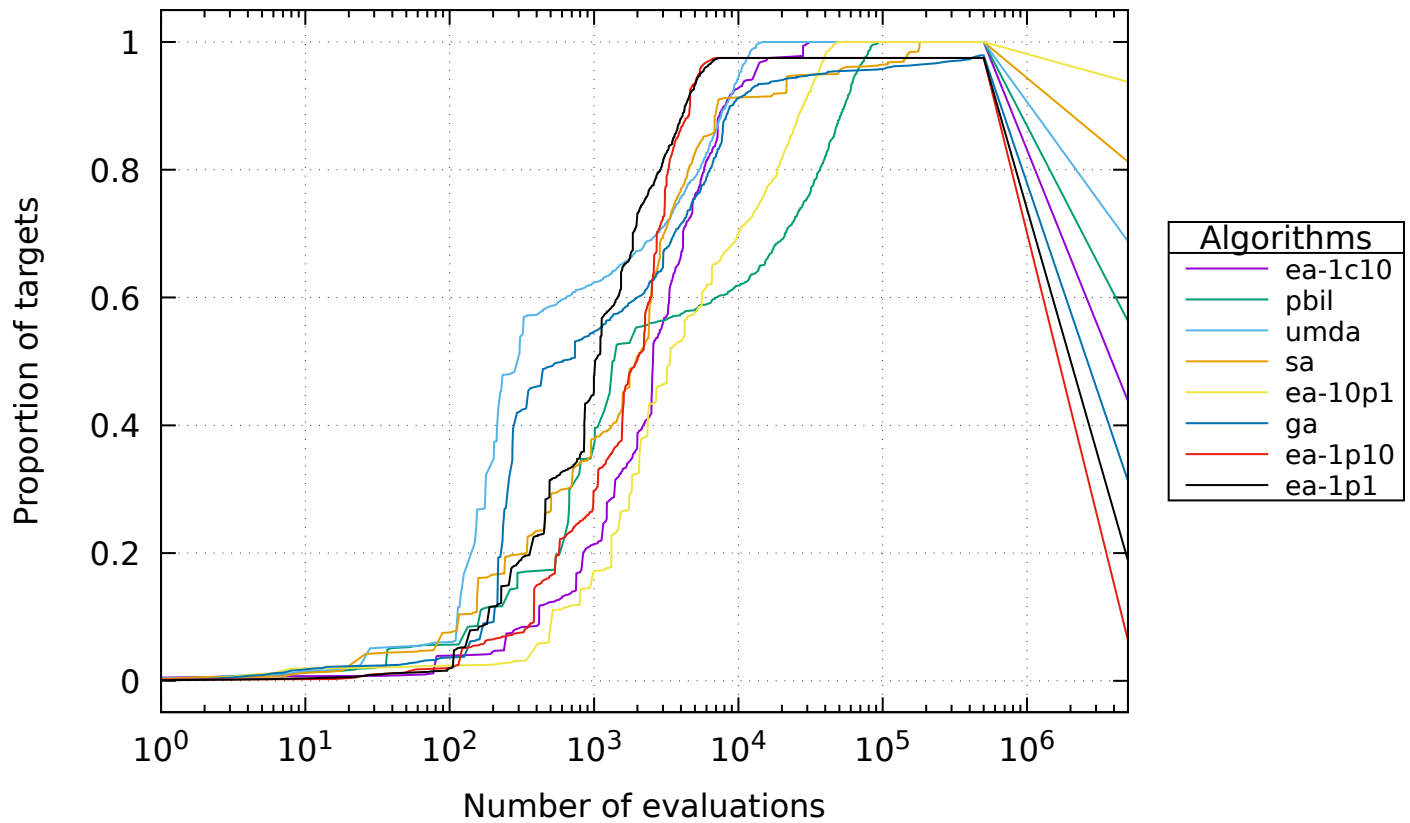
### 9.2.2 eda





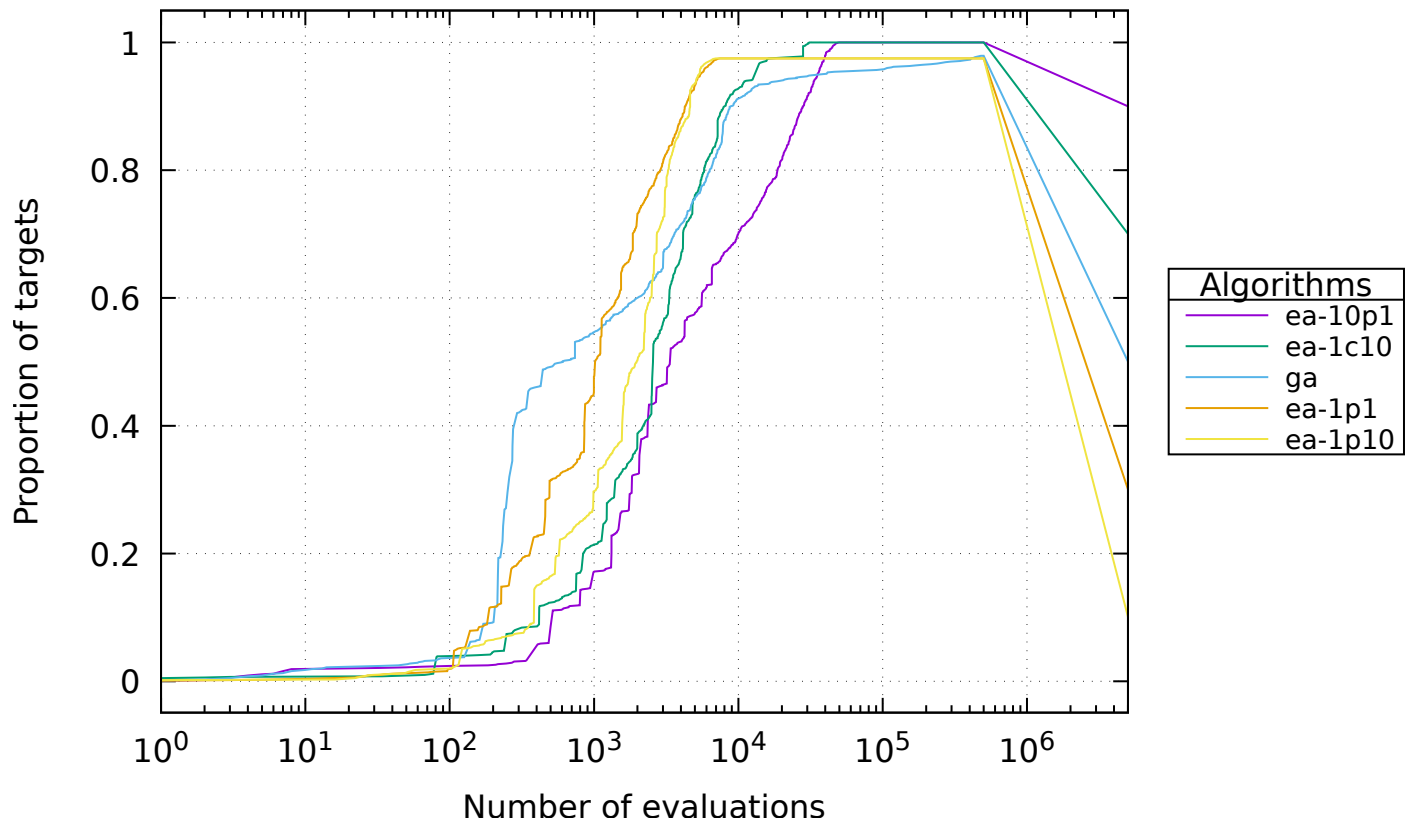
## 10 Results for fp-5

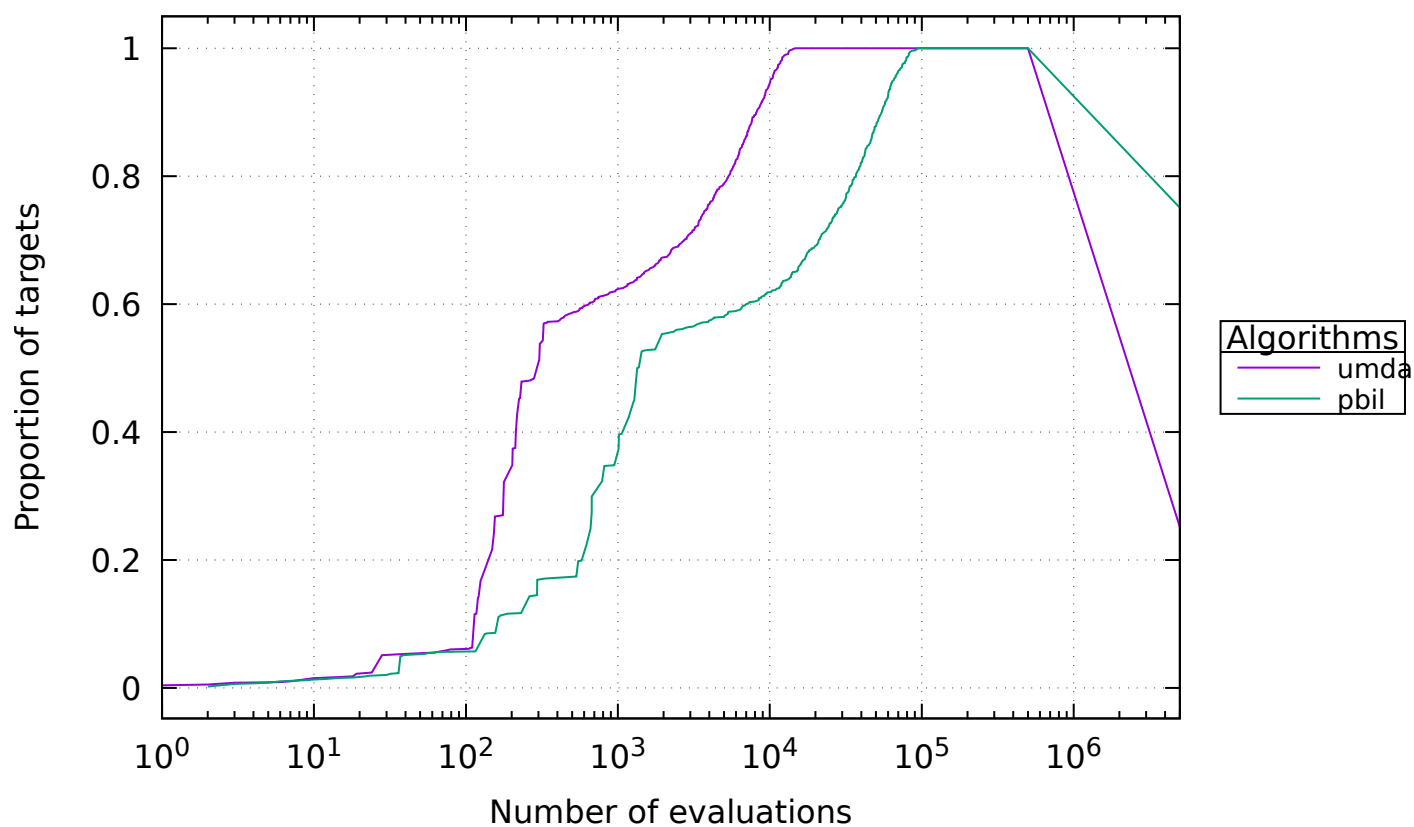
### 10.1 All algorithms



### 10.2 Groups

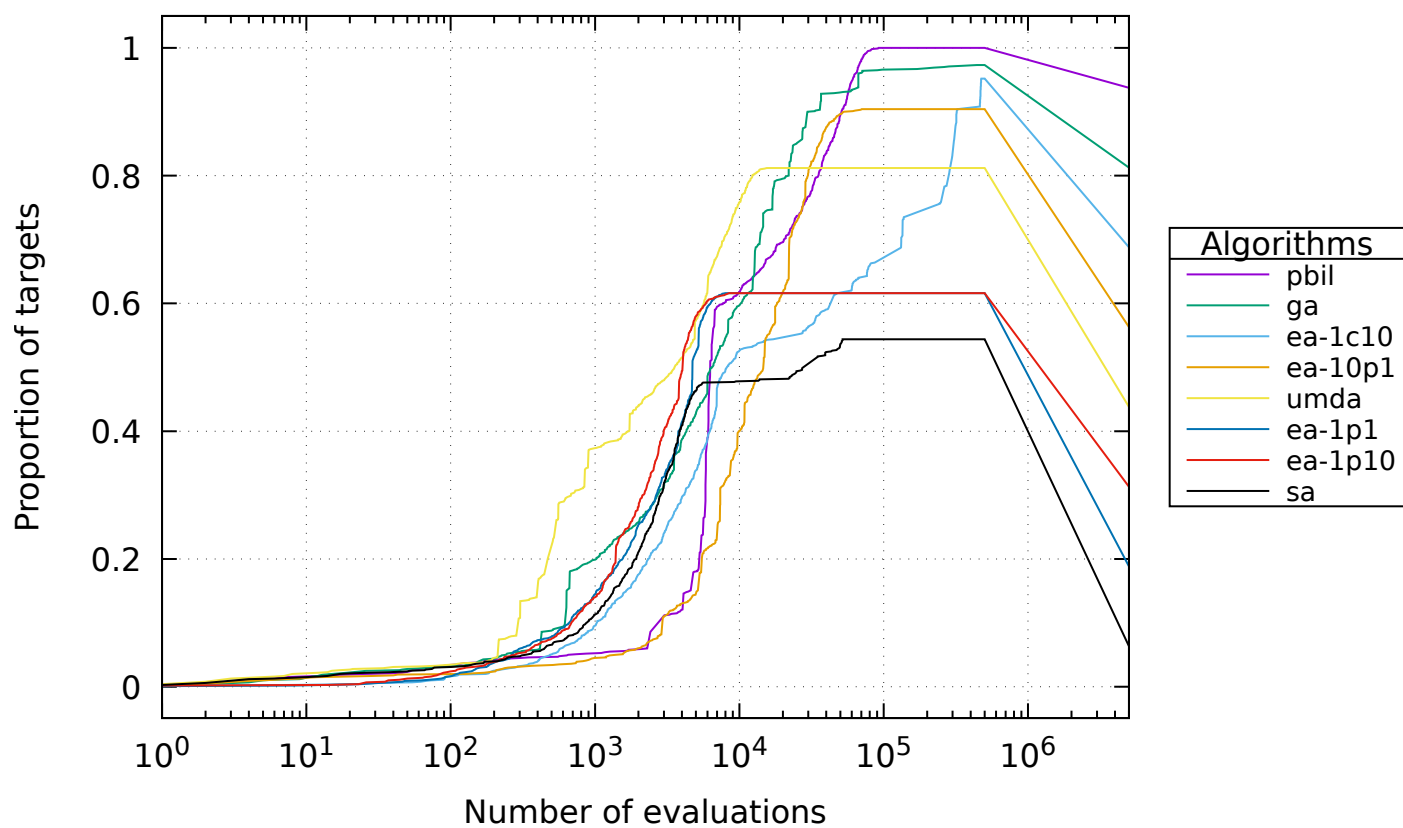
#### 10.2.1 ec





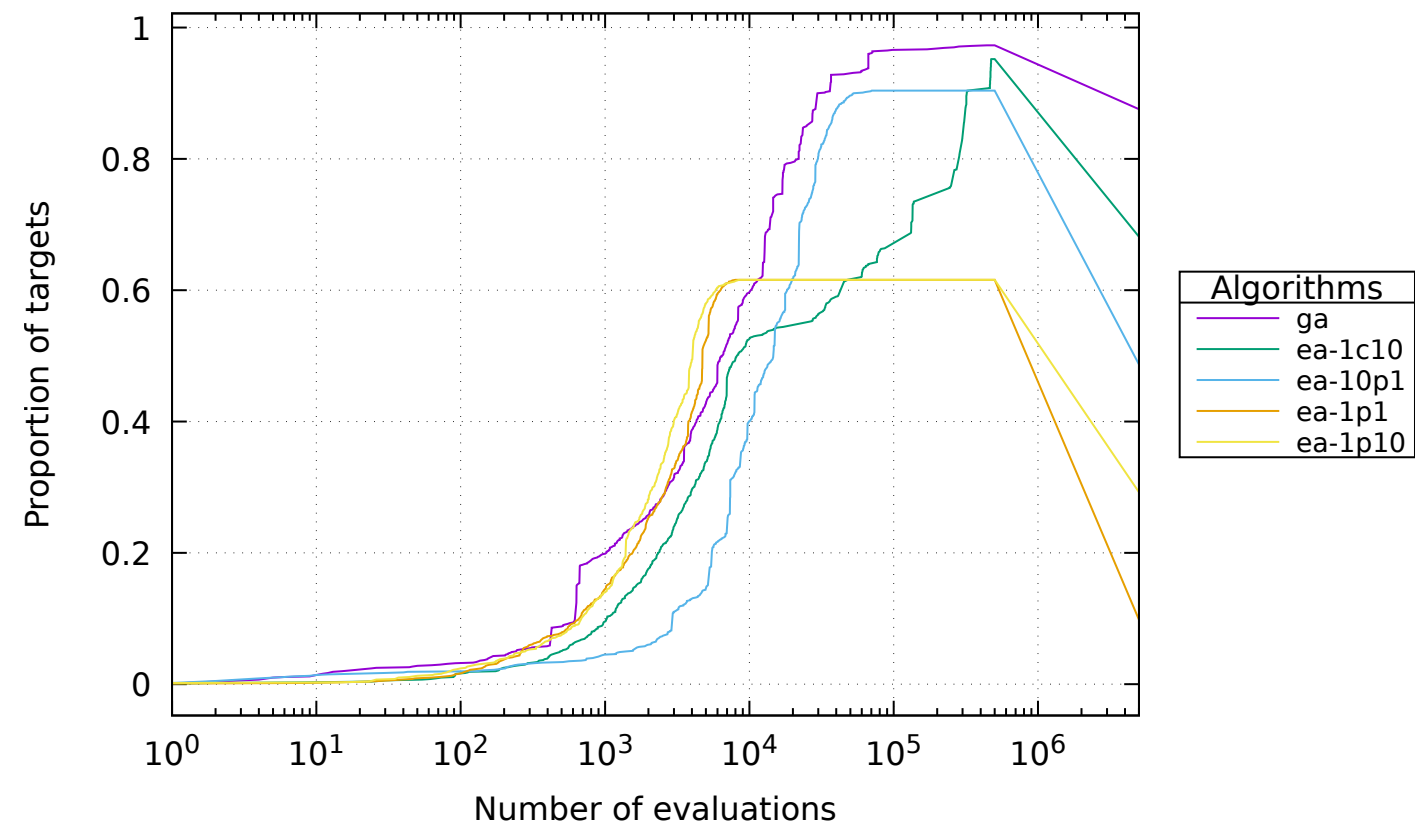
## 11 Results for fp-10

### 11.1 All algorithms

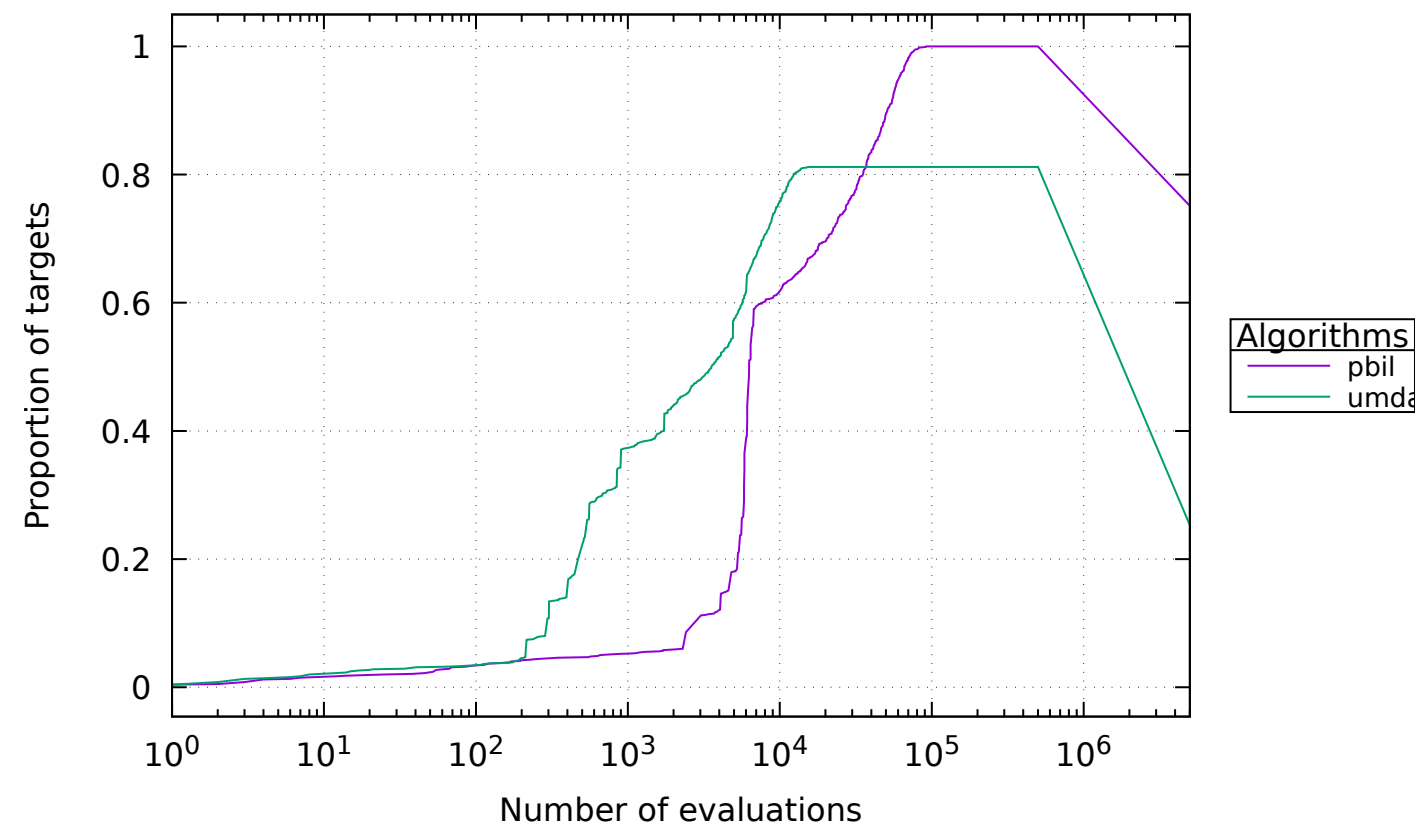


11.2 Groups

11.2.1 ec

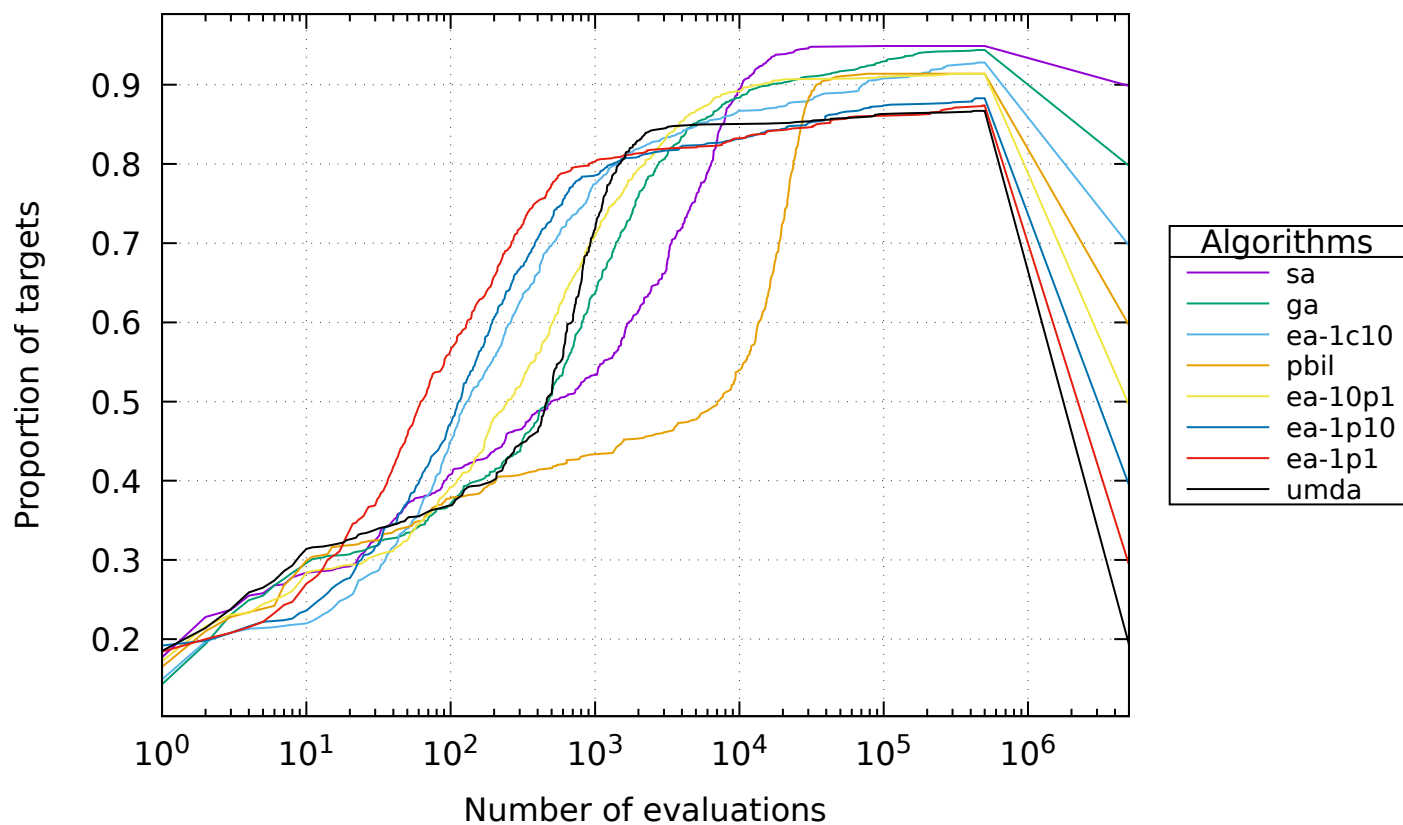


11.2.2 eda



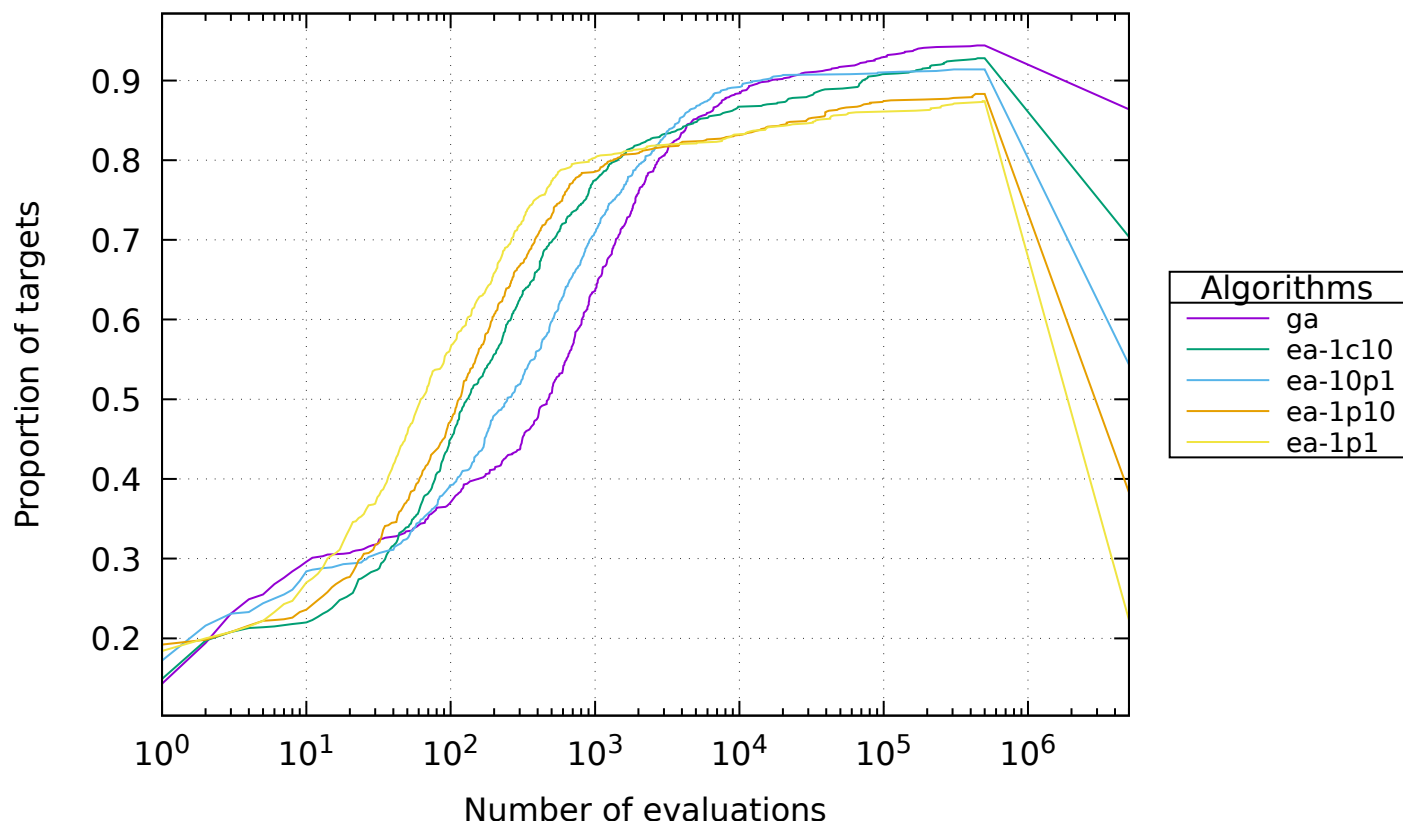
## 12 Results for nk

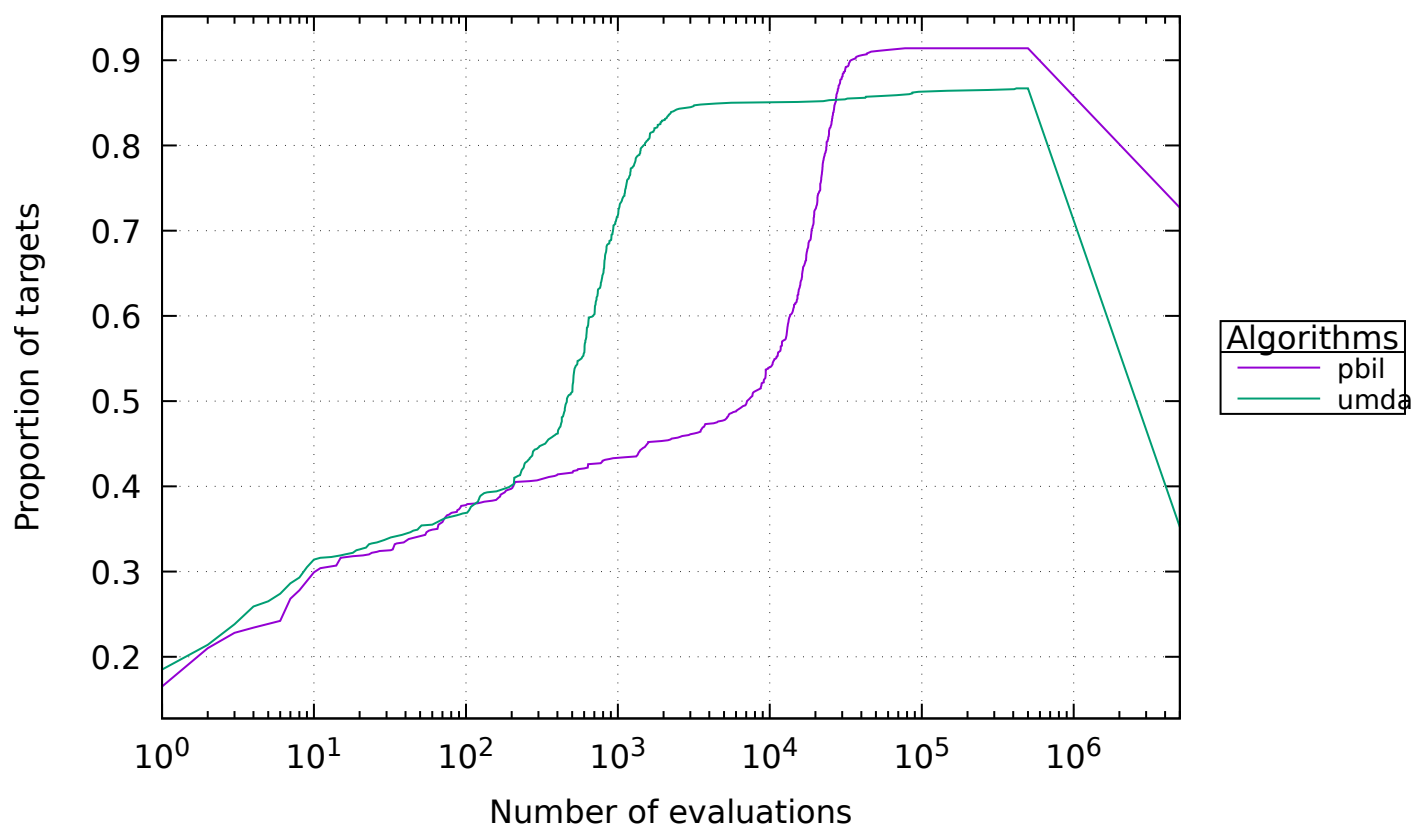
### 12.1 All algorithms



### 12.2 Groups

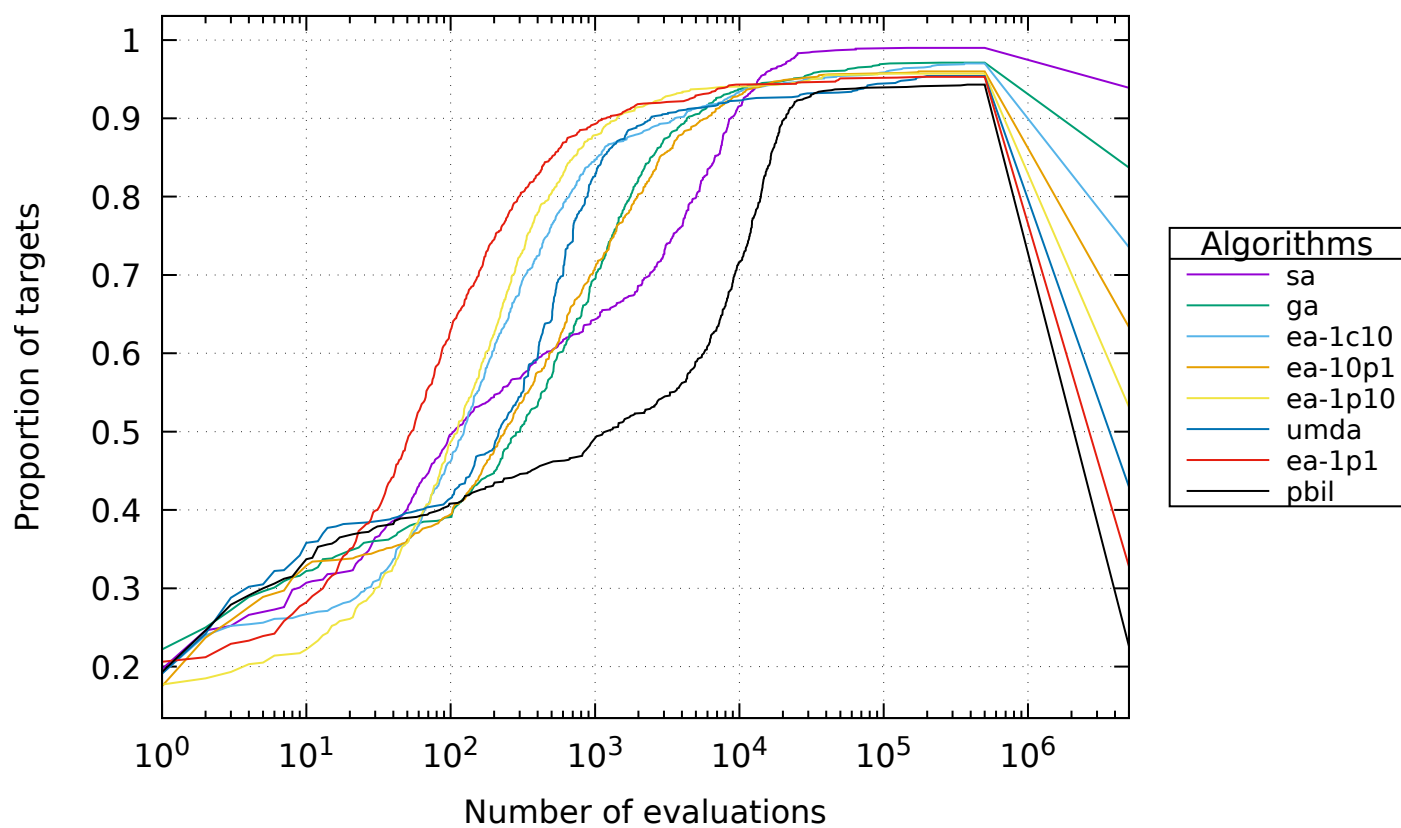
#### 12.2.1 ec





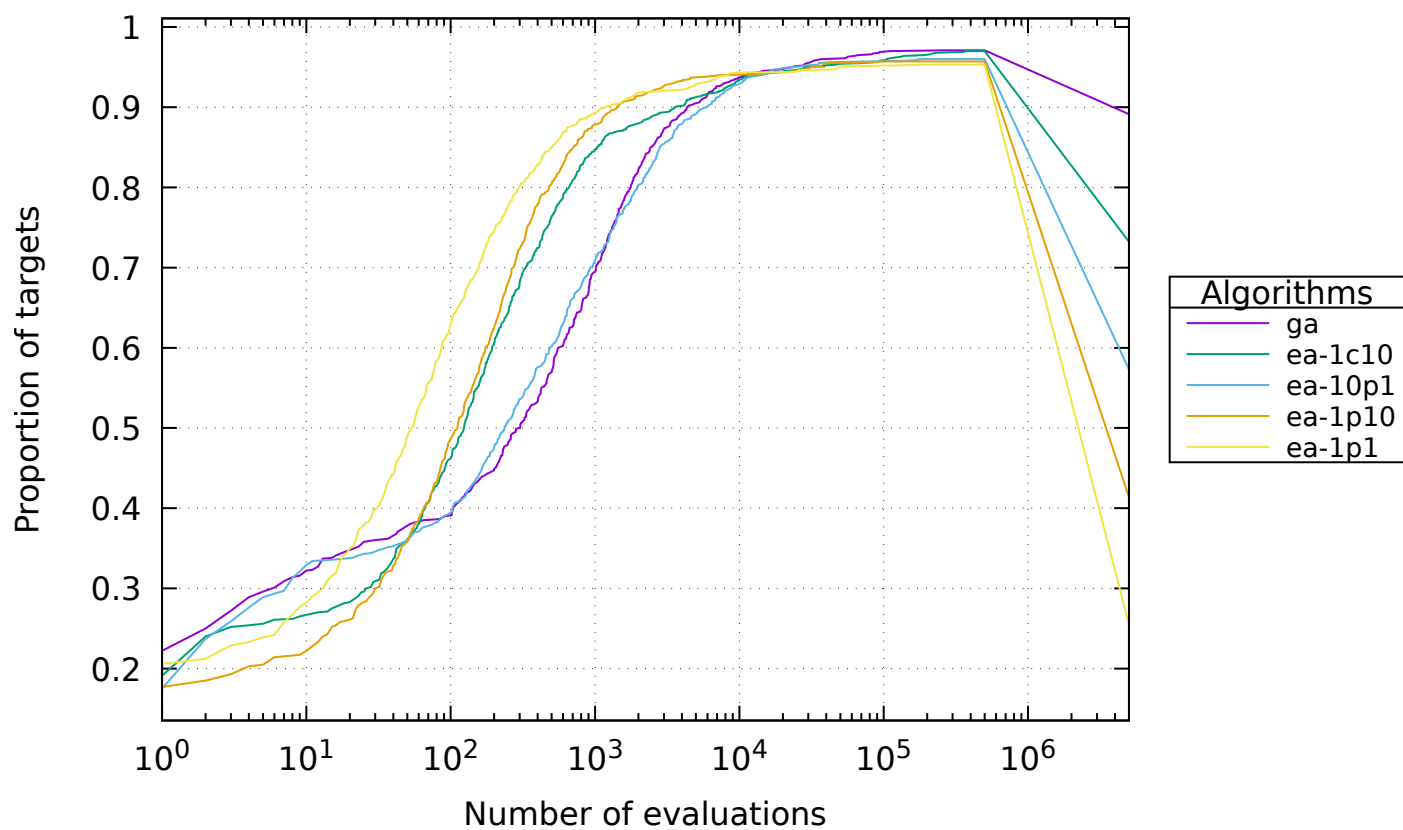
## 13 Results for max-sat

### 13.1 All algorithms

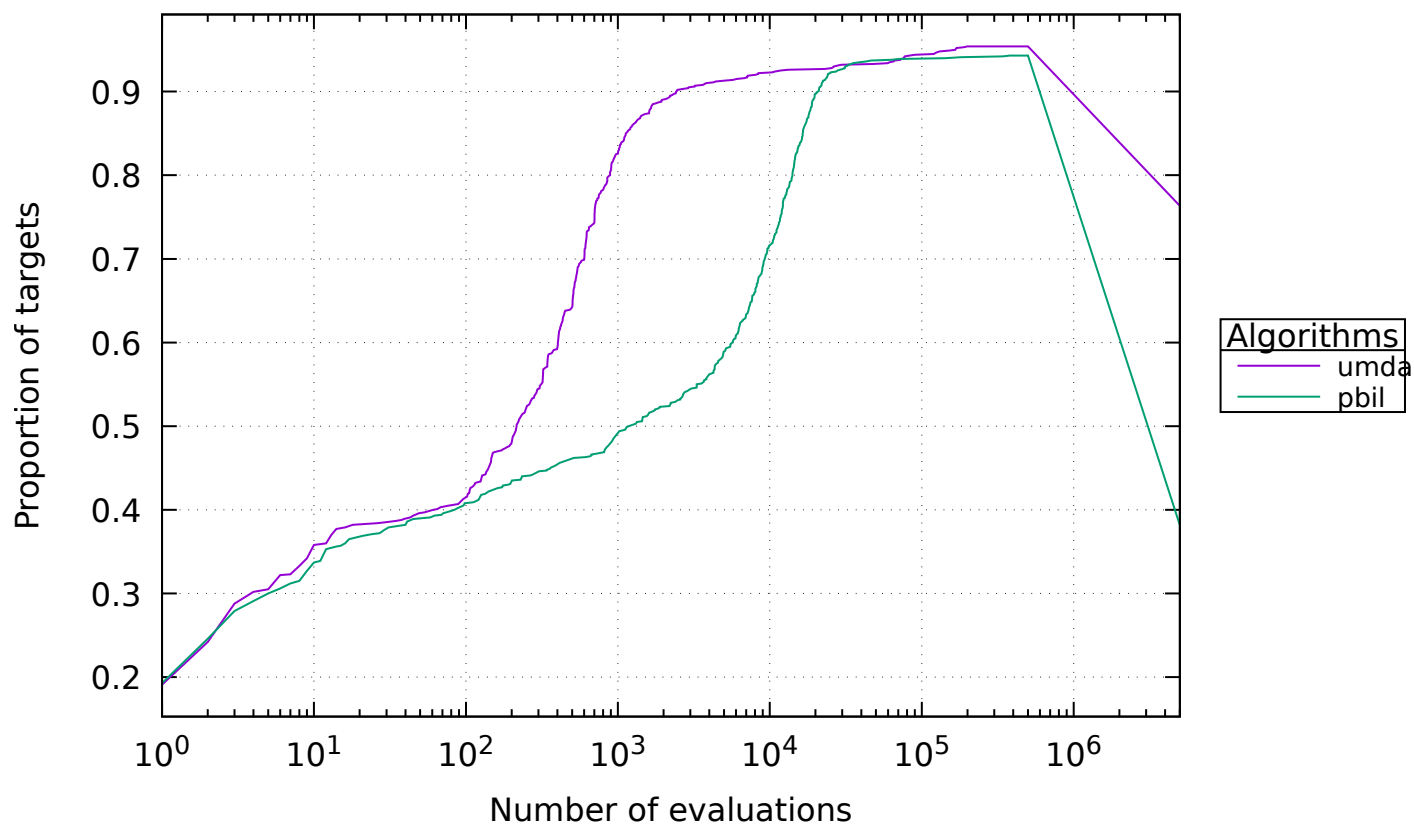


## 13.2 Groups

### 13.2.1 ec

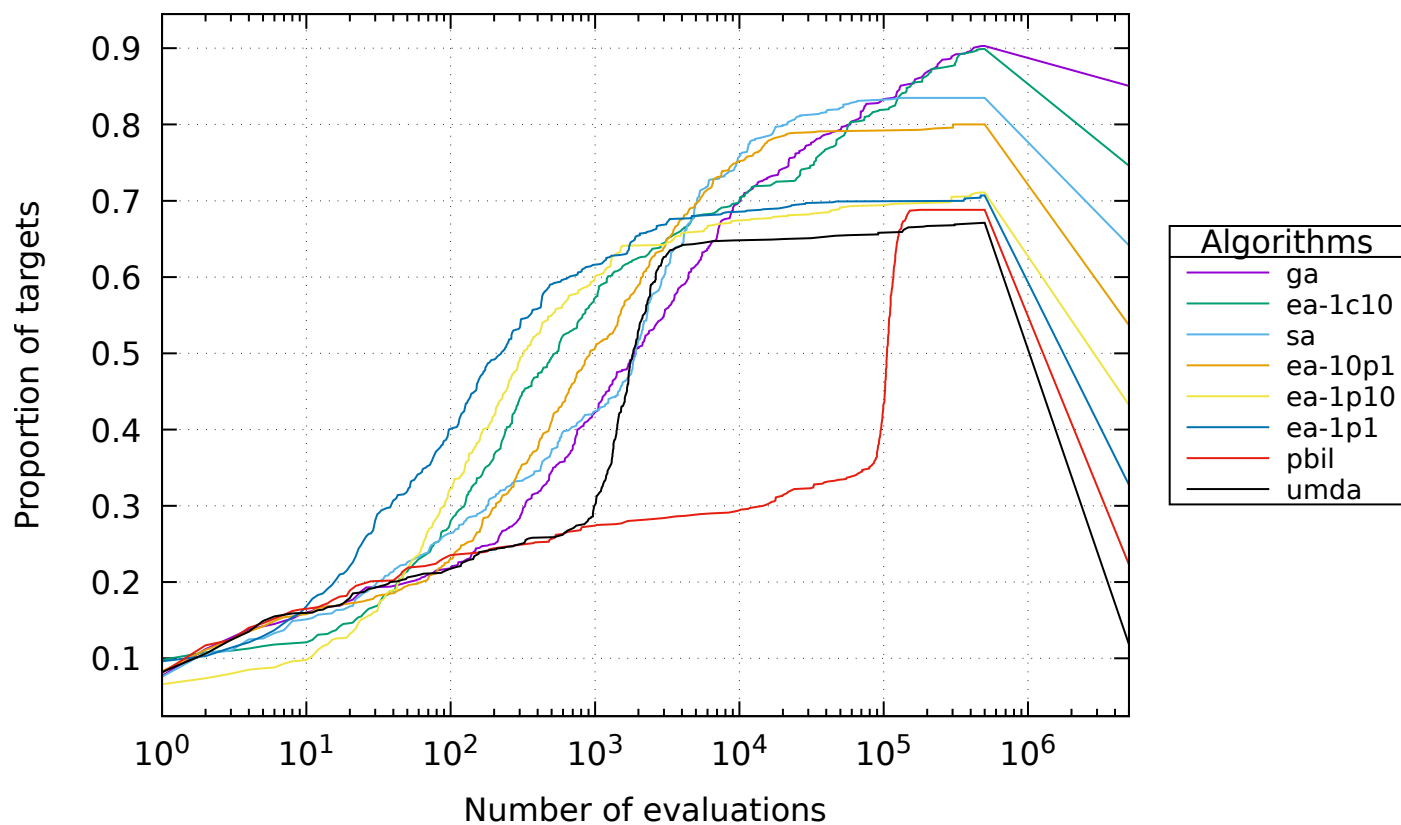


### 13.2.2 eda



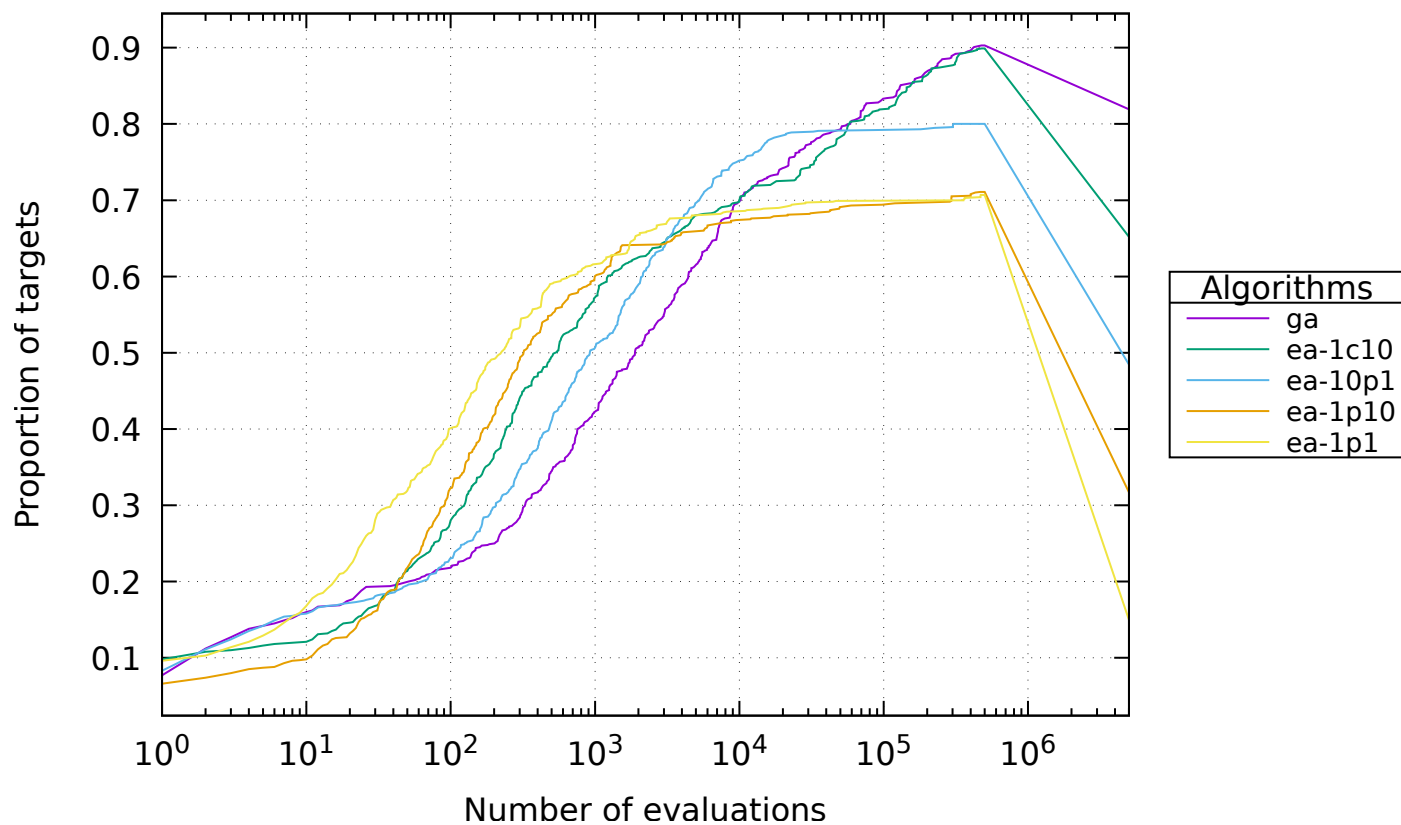
## 14 Results for labs

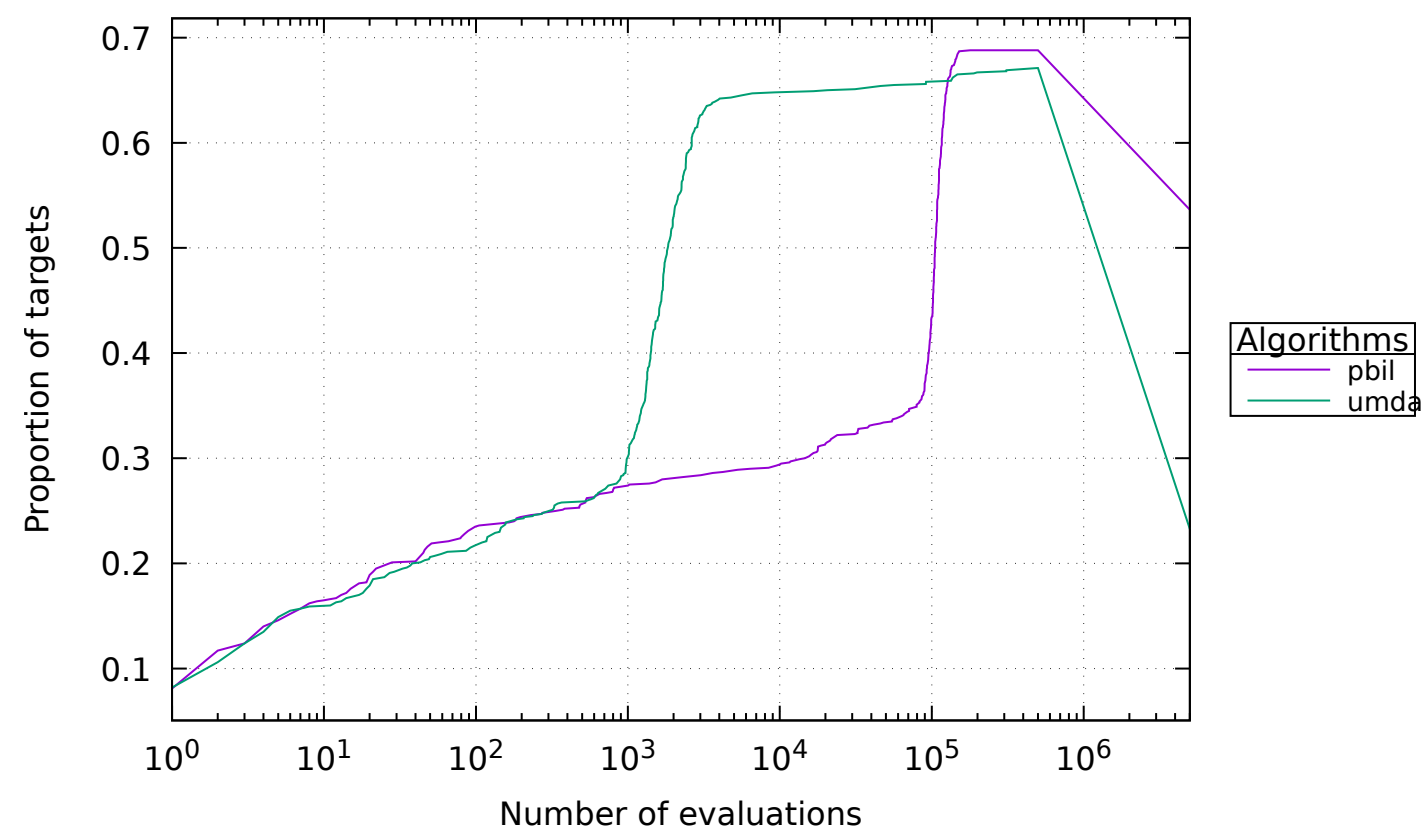
### 14.1 All algorithms



### 14.2 Groups

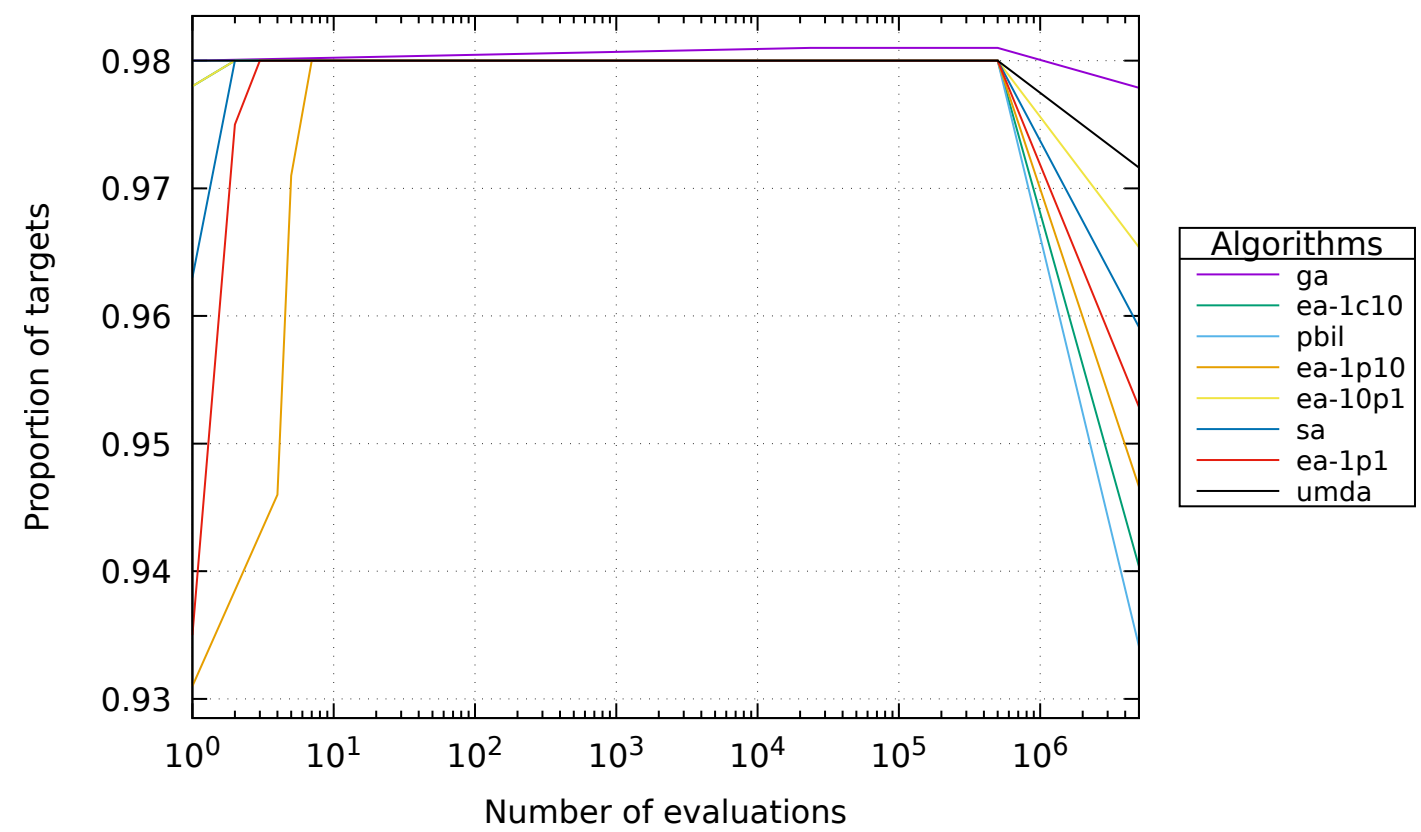
#### 14.2.1 ec





15 Results for ep

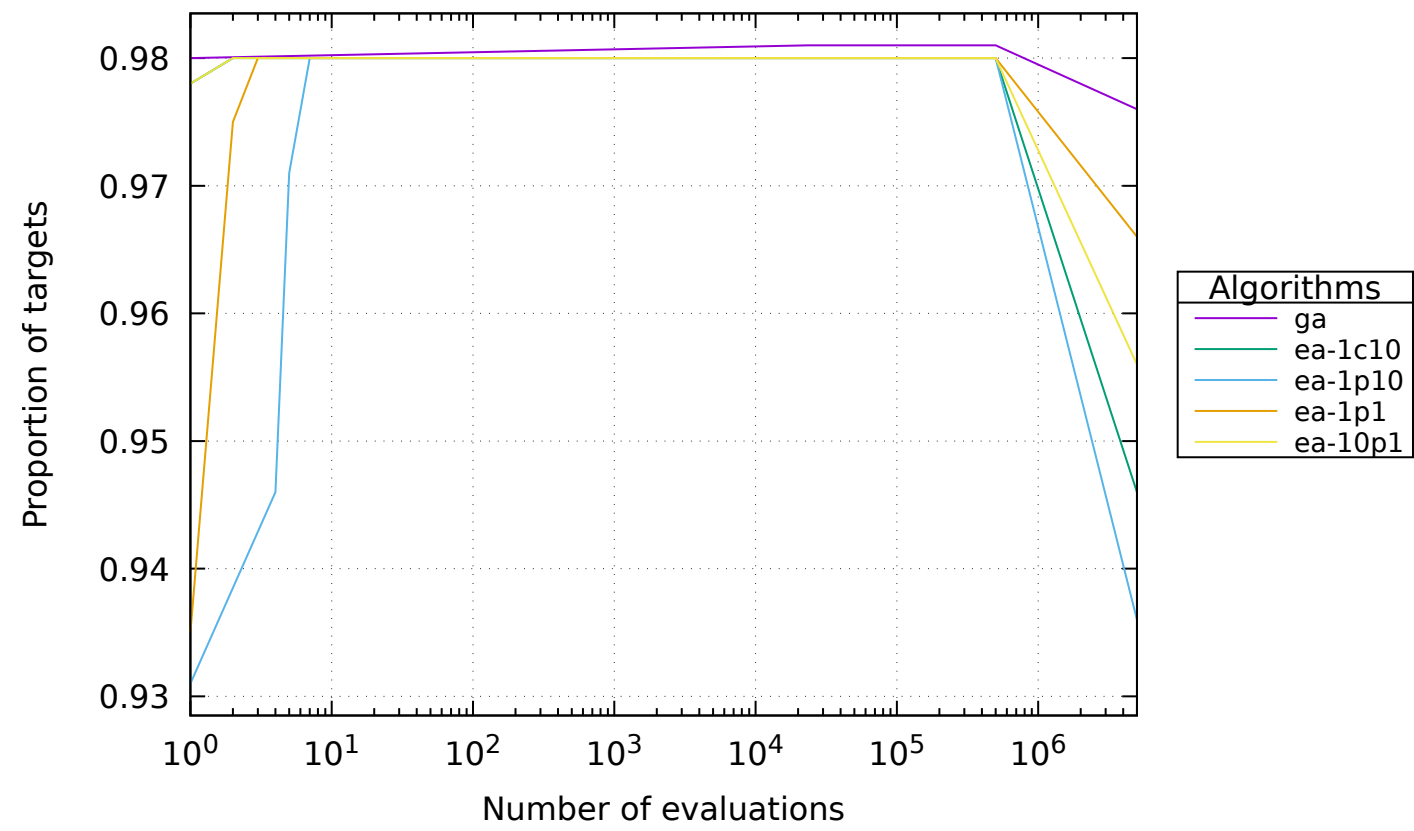
15.1 All algorithms



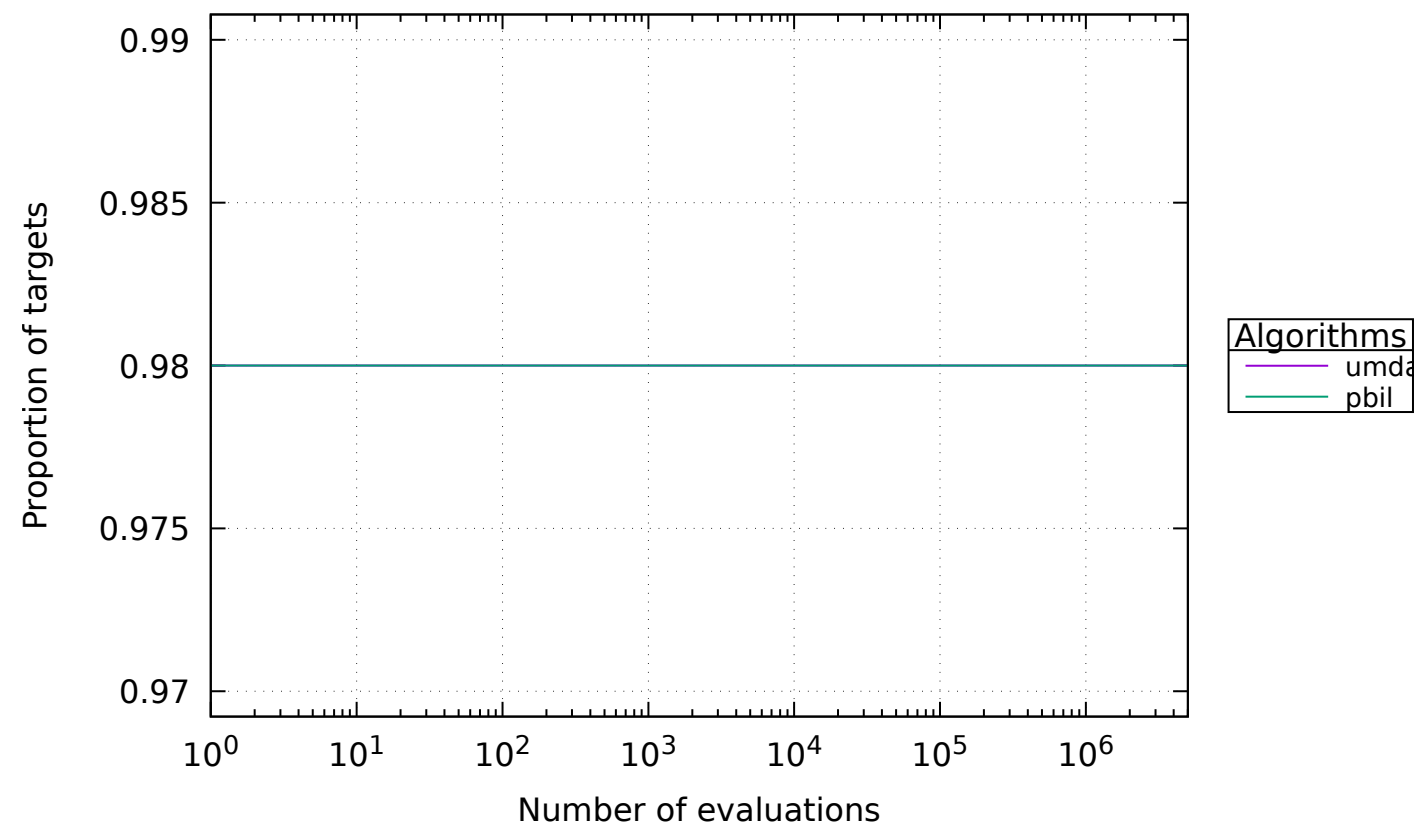


15.2 Groups

15.2.1 ec

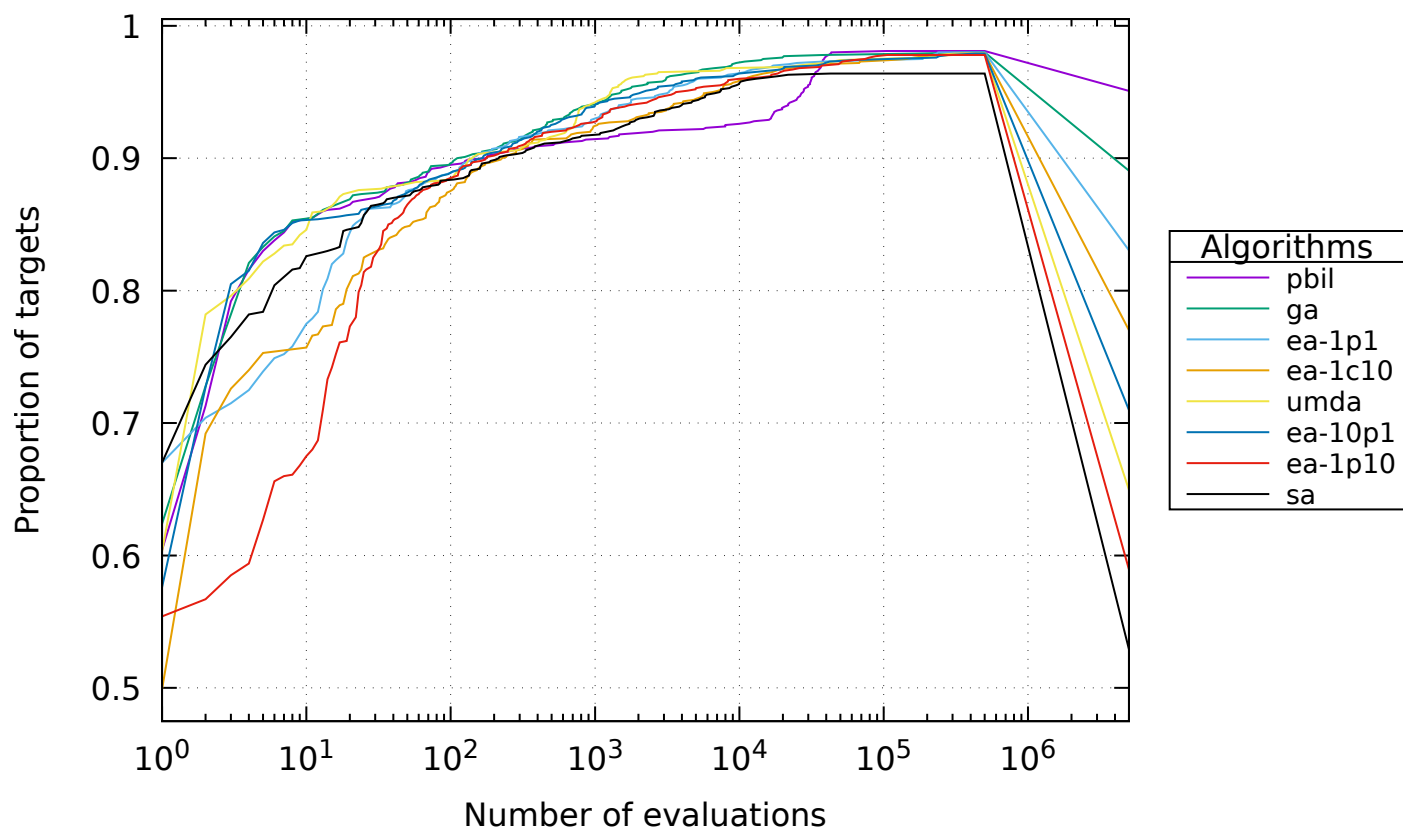


15.2.2 eda



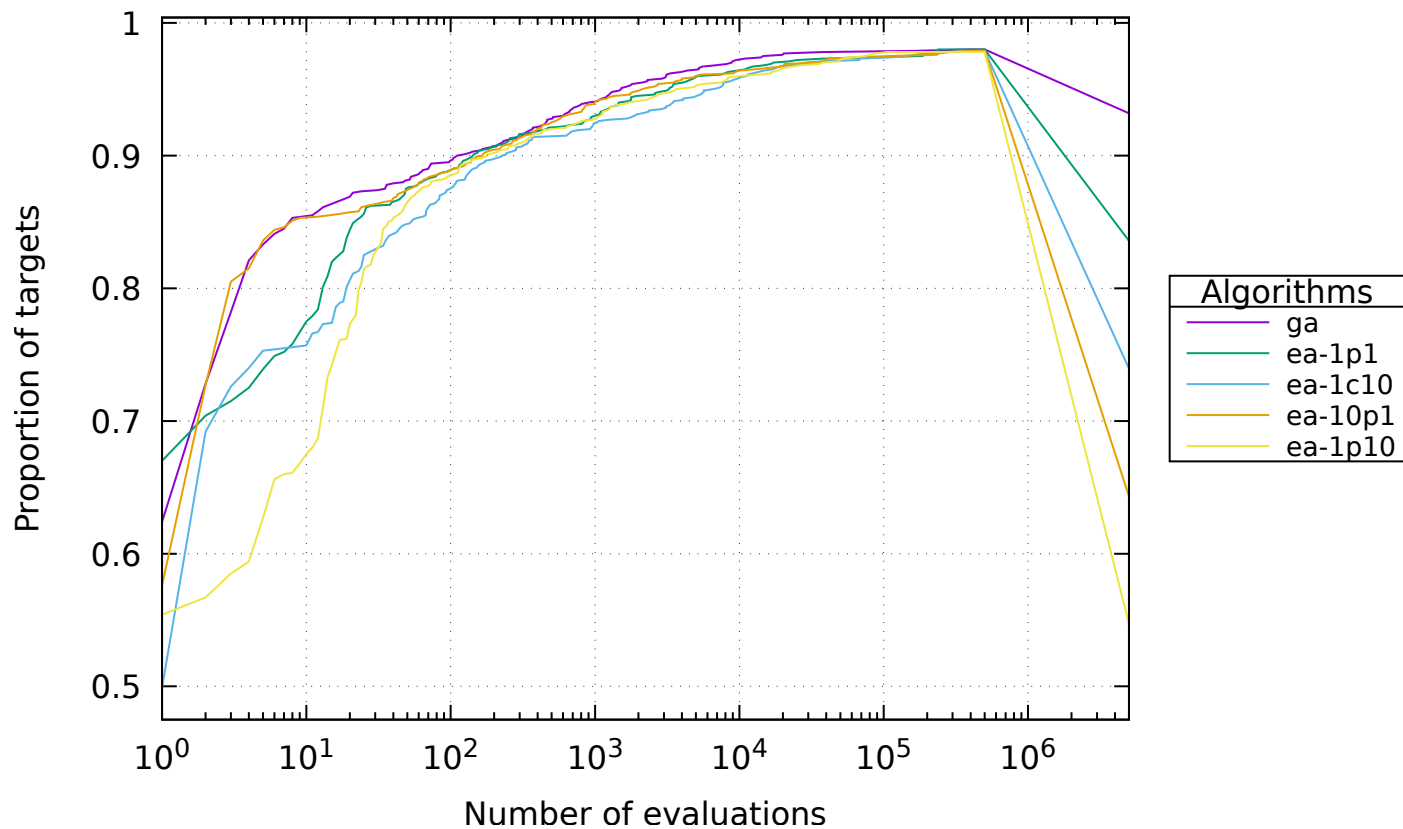
## 16 Results for cancel

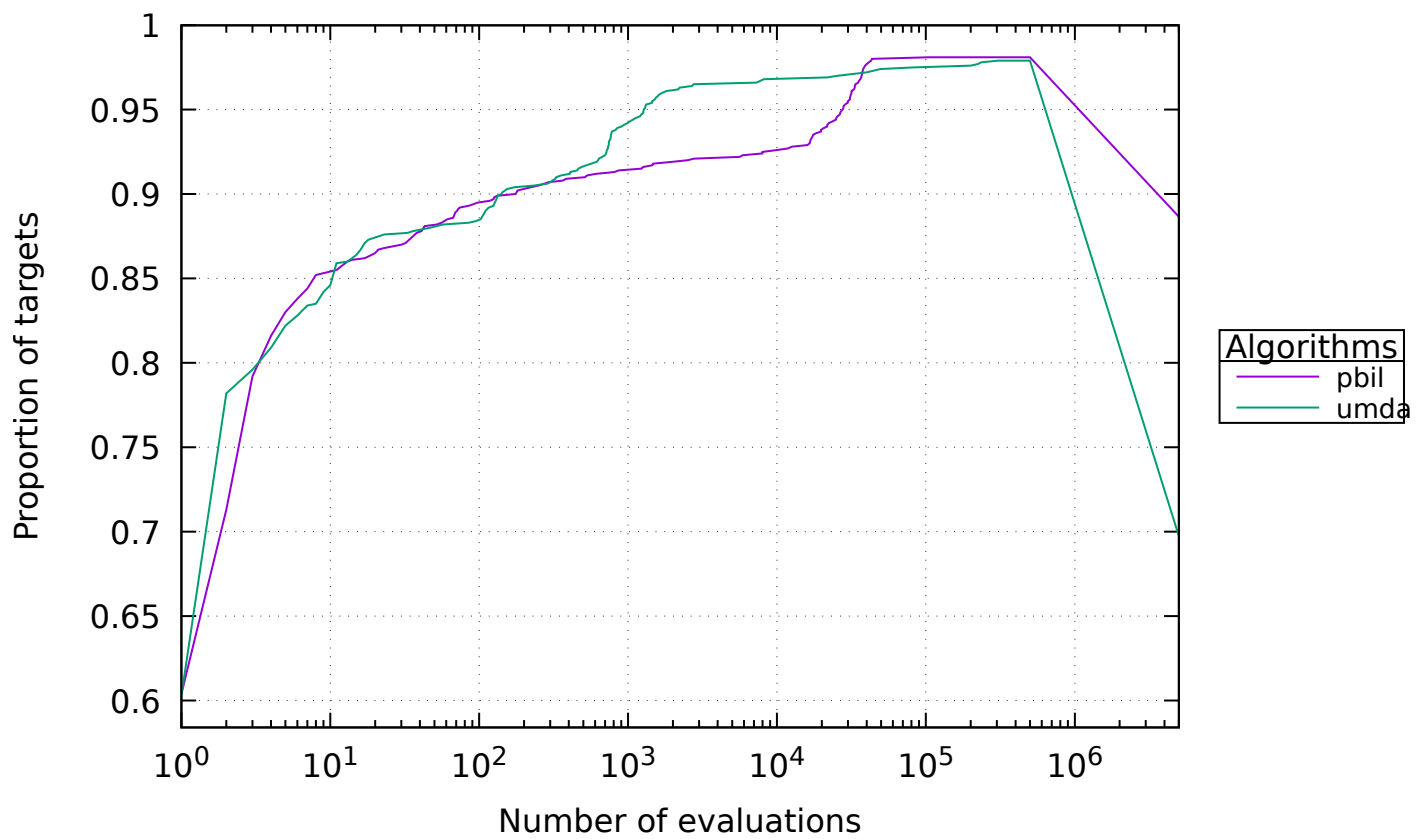
### 16.1 All algorithms



### 16.2 Groups

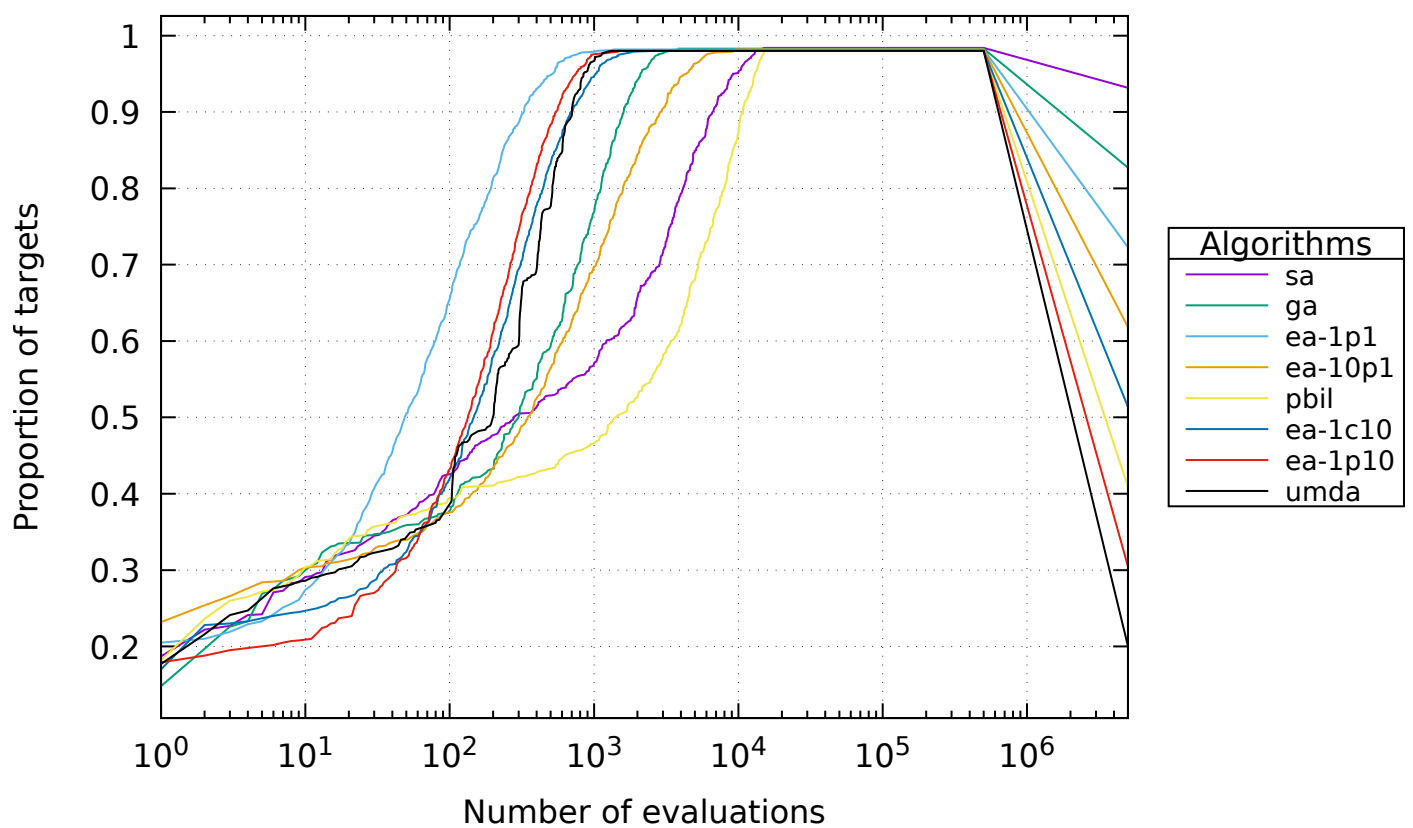
#### 16.2.1 ec





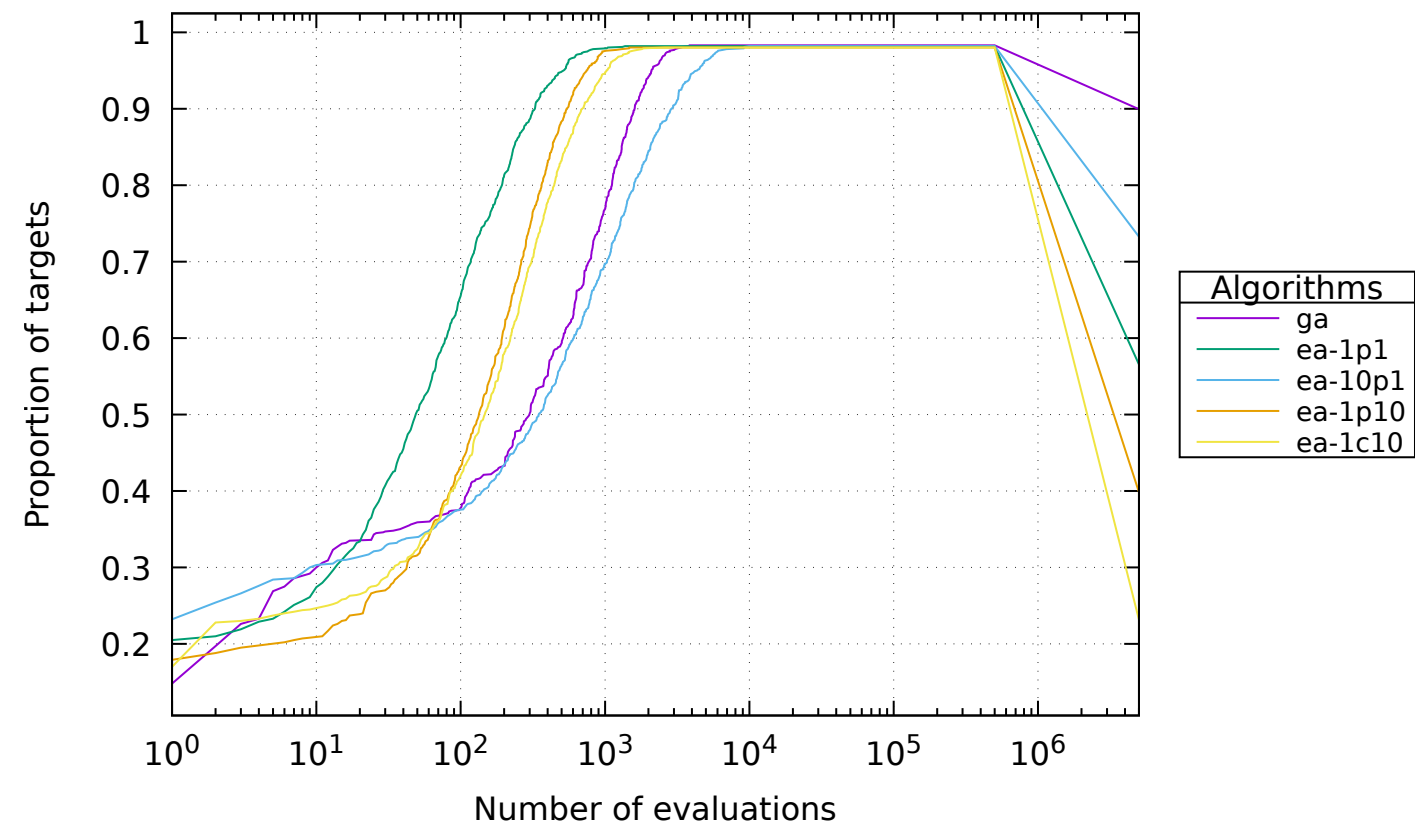
## 17 Results for trap

### 17.1 All algorithms

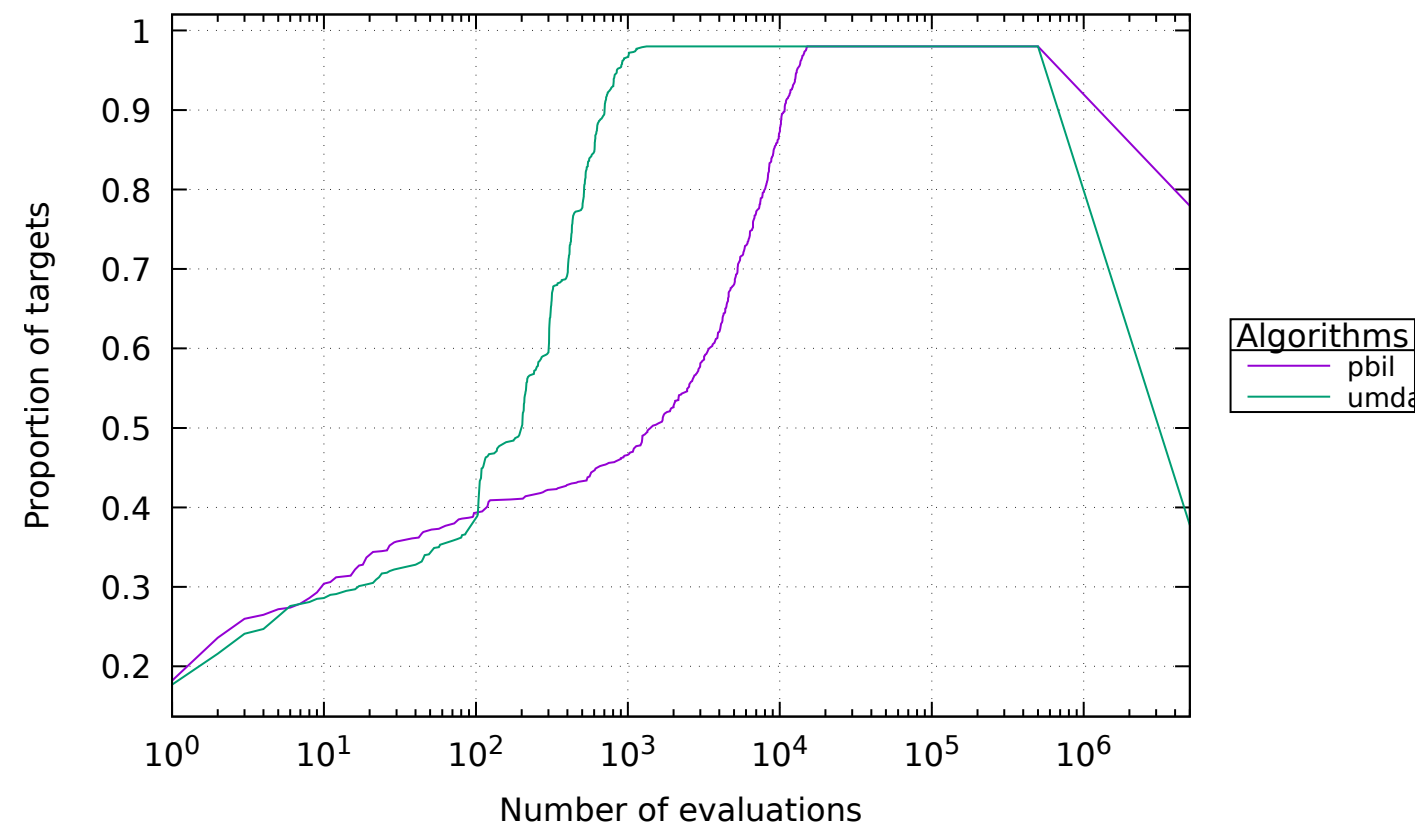


17.2 Groups

17.2.1 ec

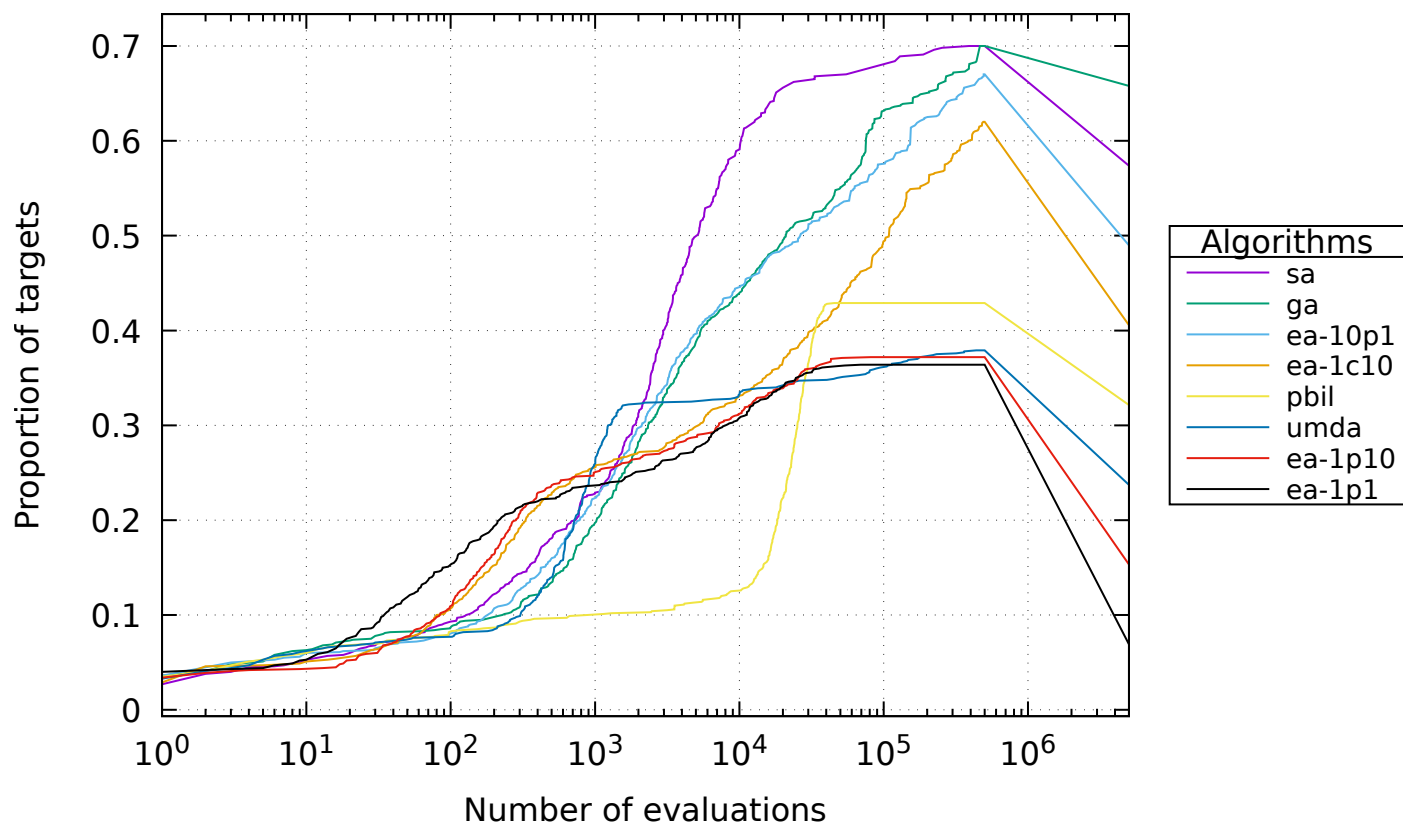


17.2.2 eda



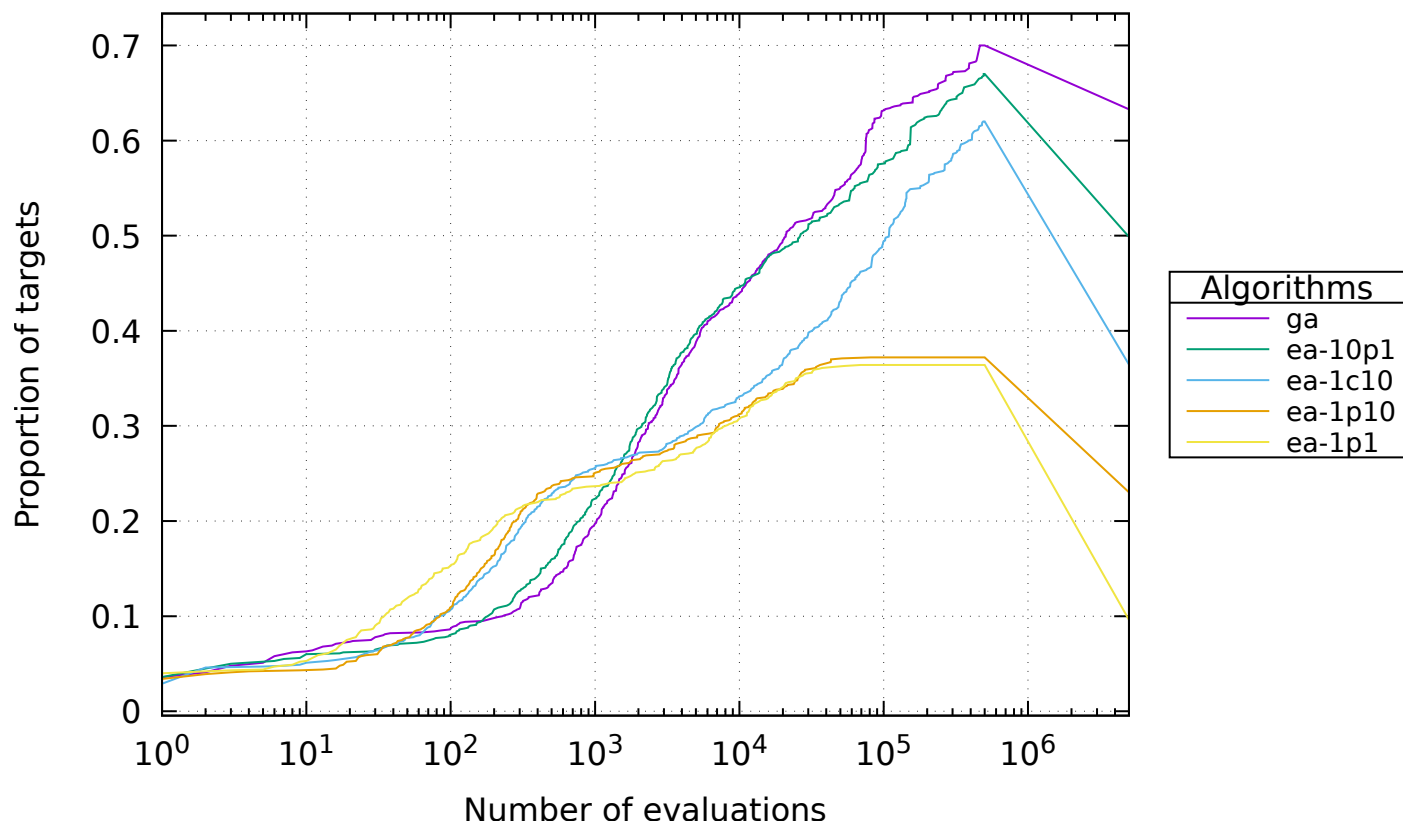
## 18 Results for hiff

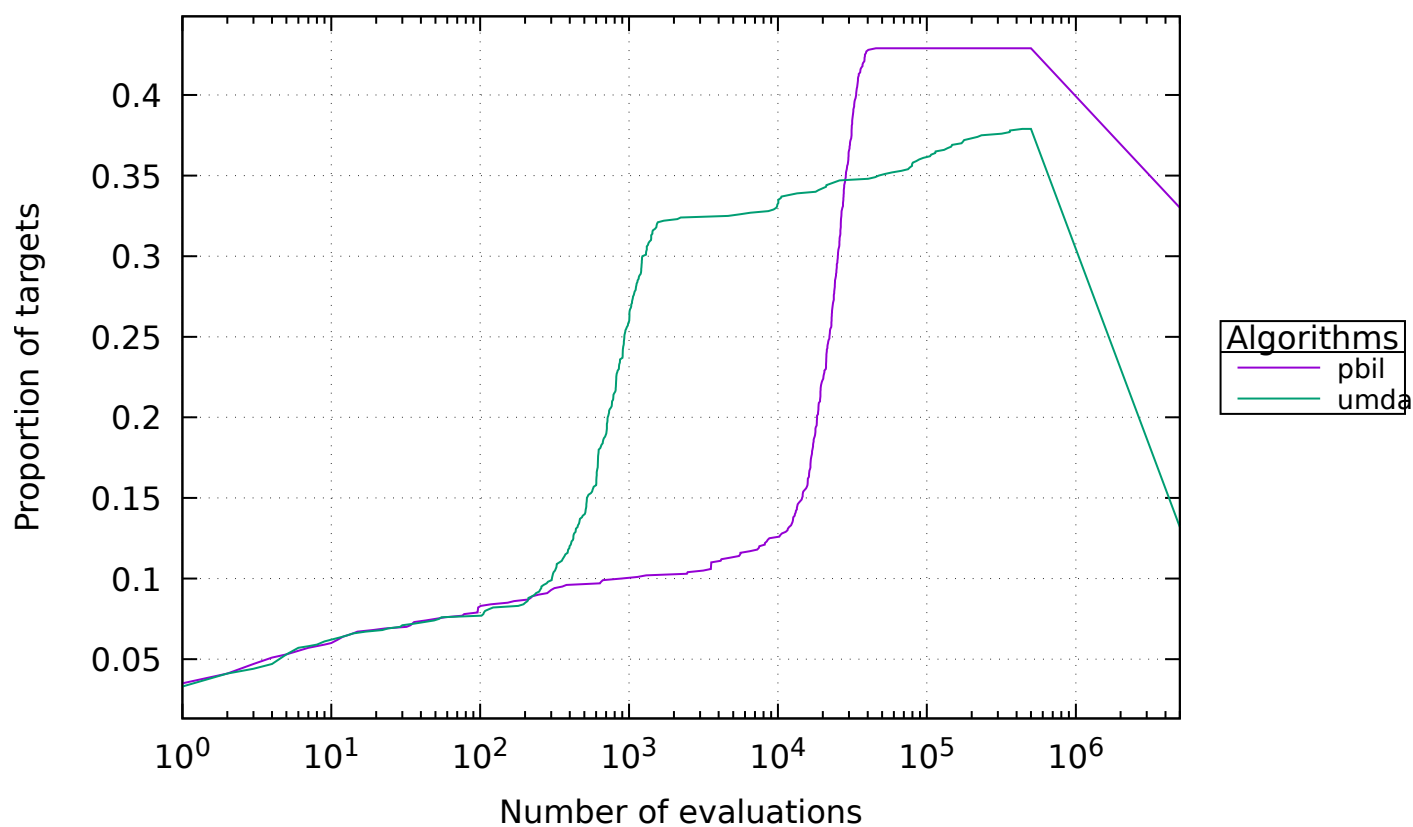
### 18.1 All algorithms



### 18.2 Groups

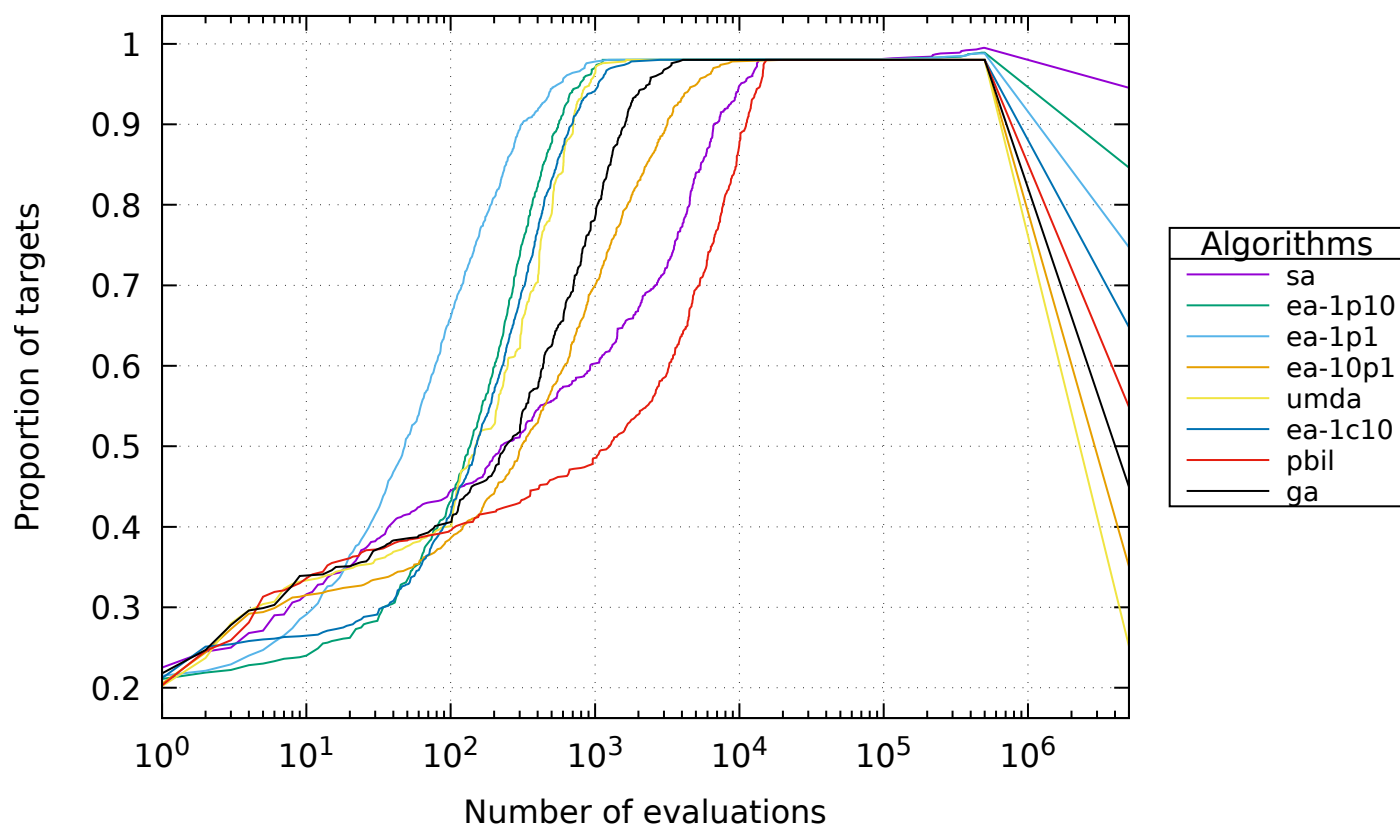
#### 18.2.1 ec





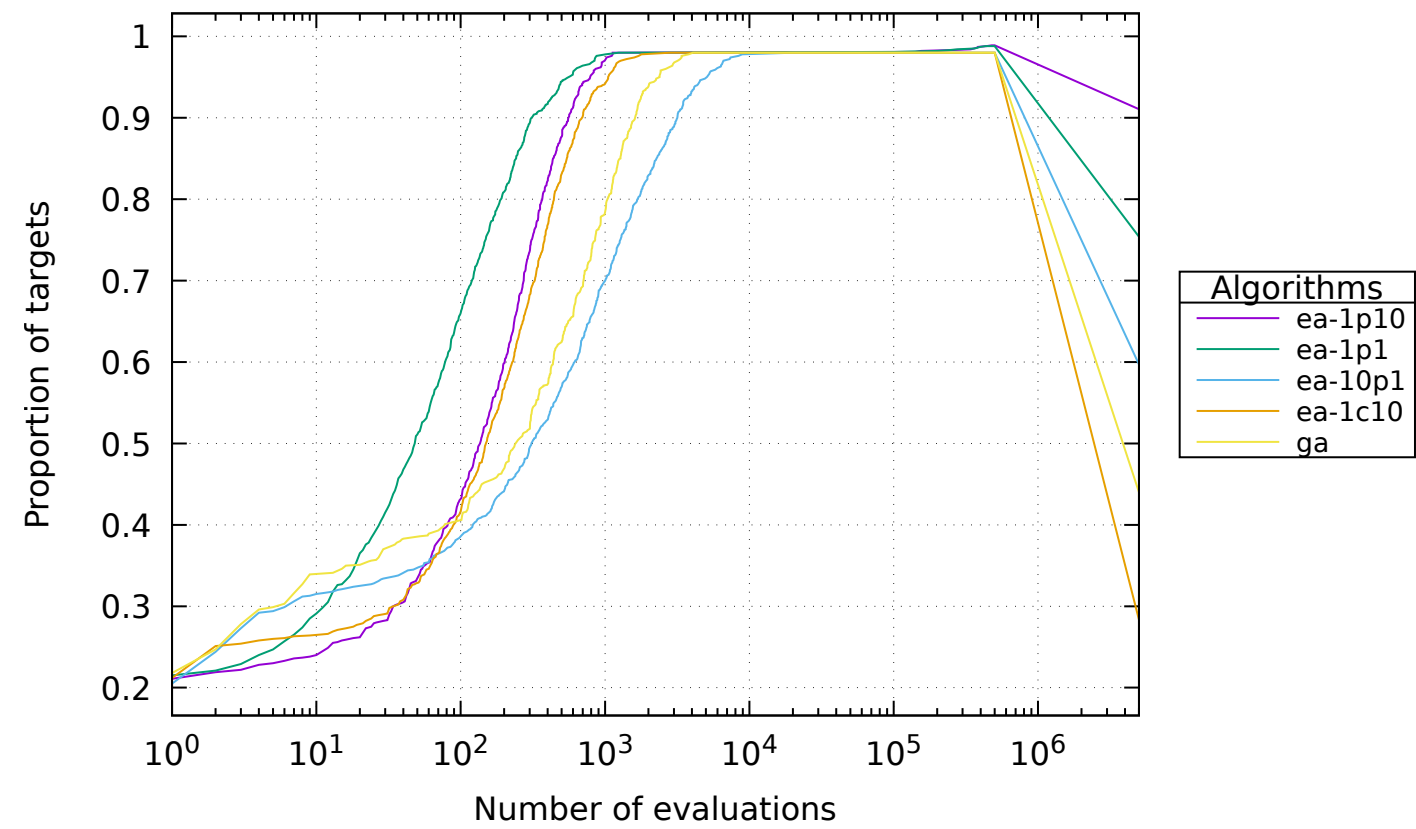
## 19 Results for plateau

### 19.1 All algorithms

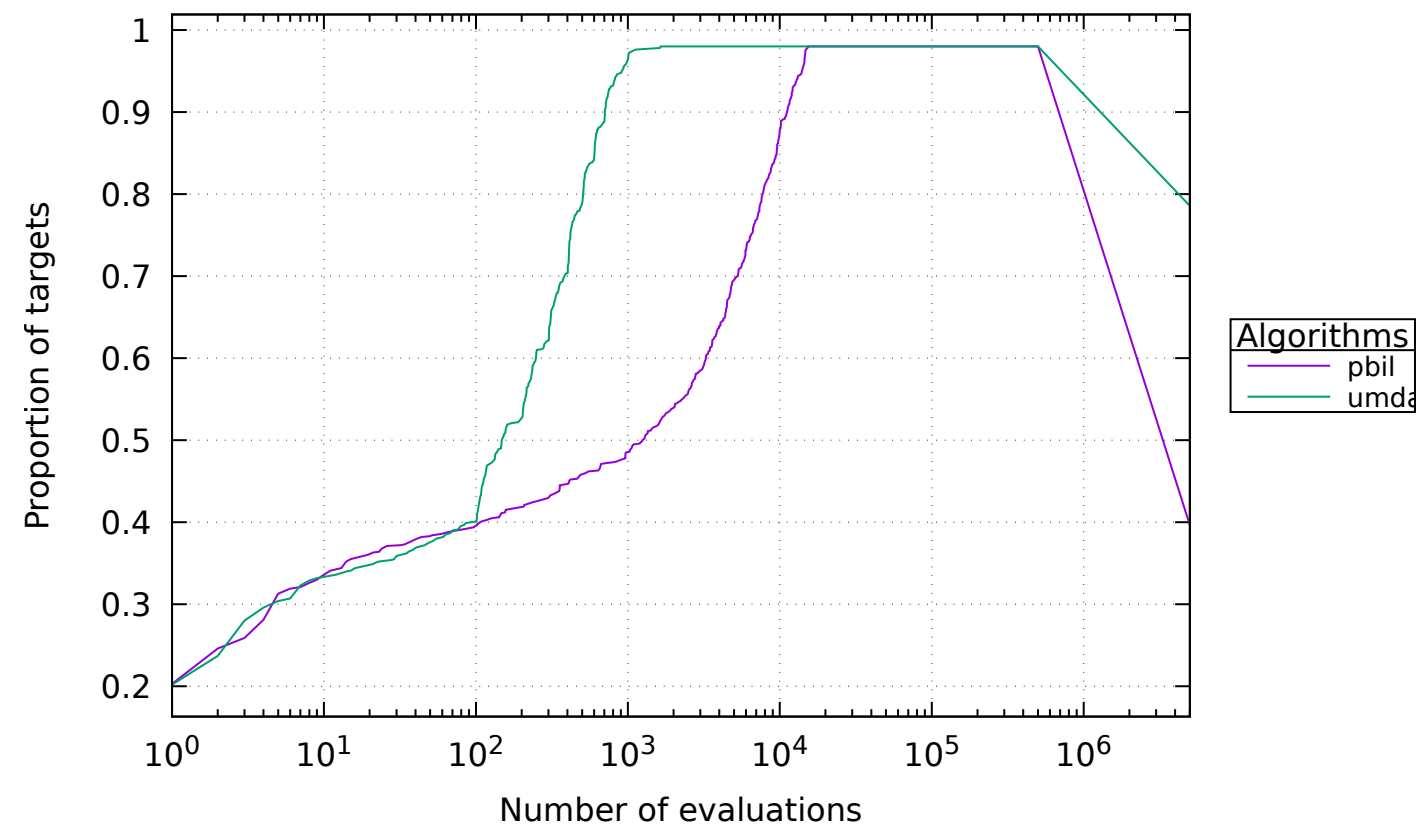


19.2 Groups

19.2.1 ec

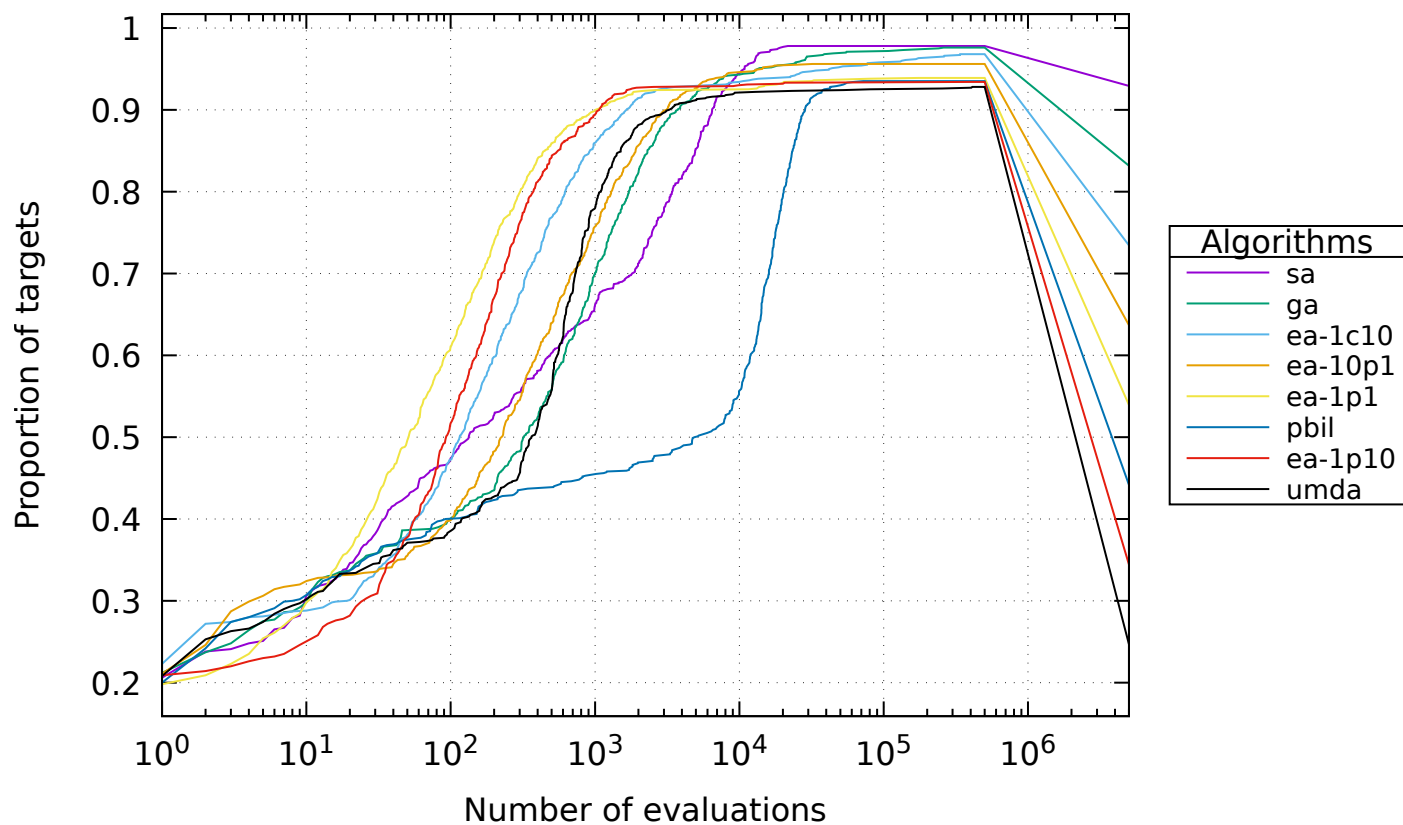


19.2.2 eda



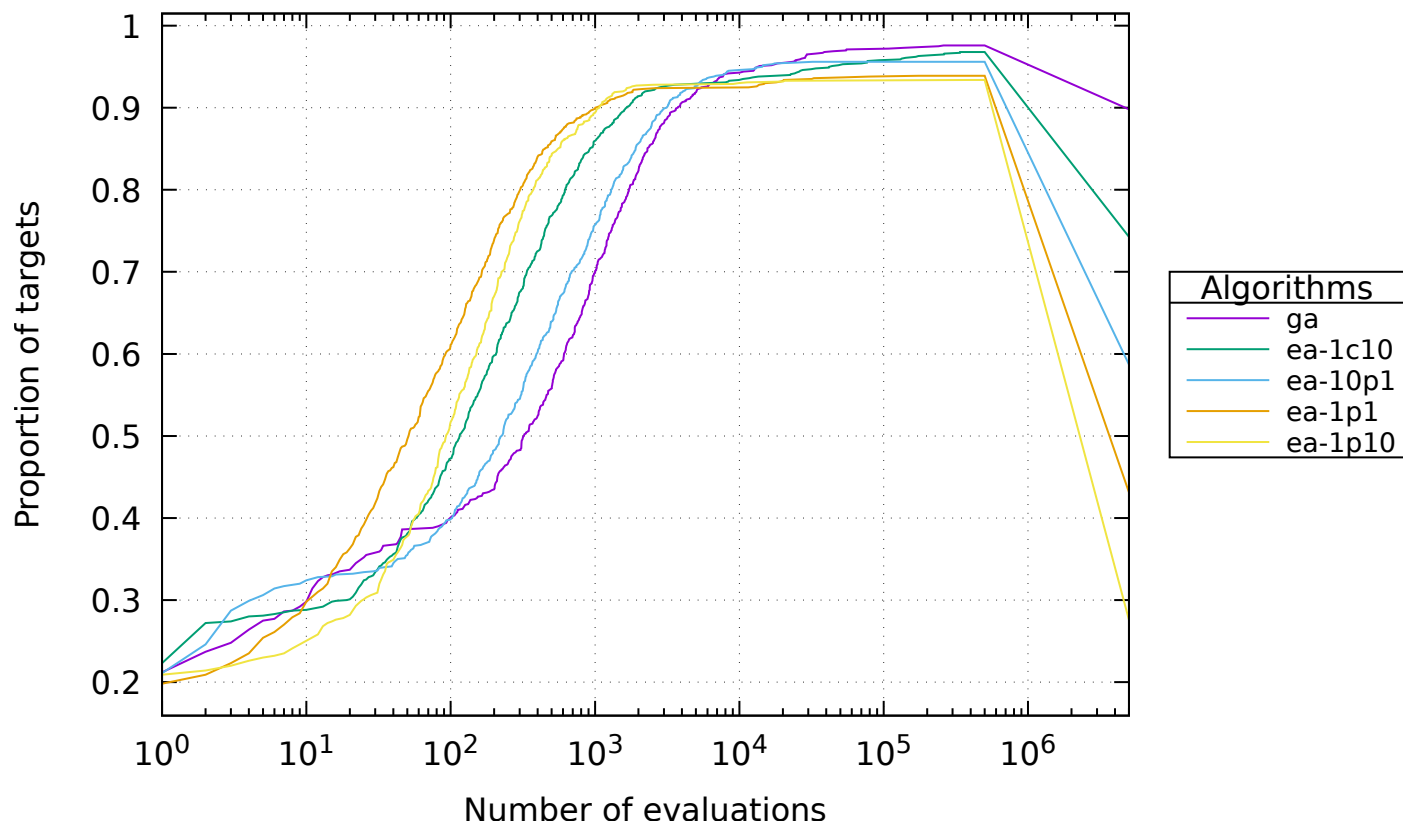
## 20 Results for walsh2

### 20.1 All algorithms

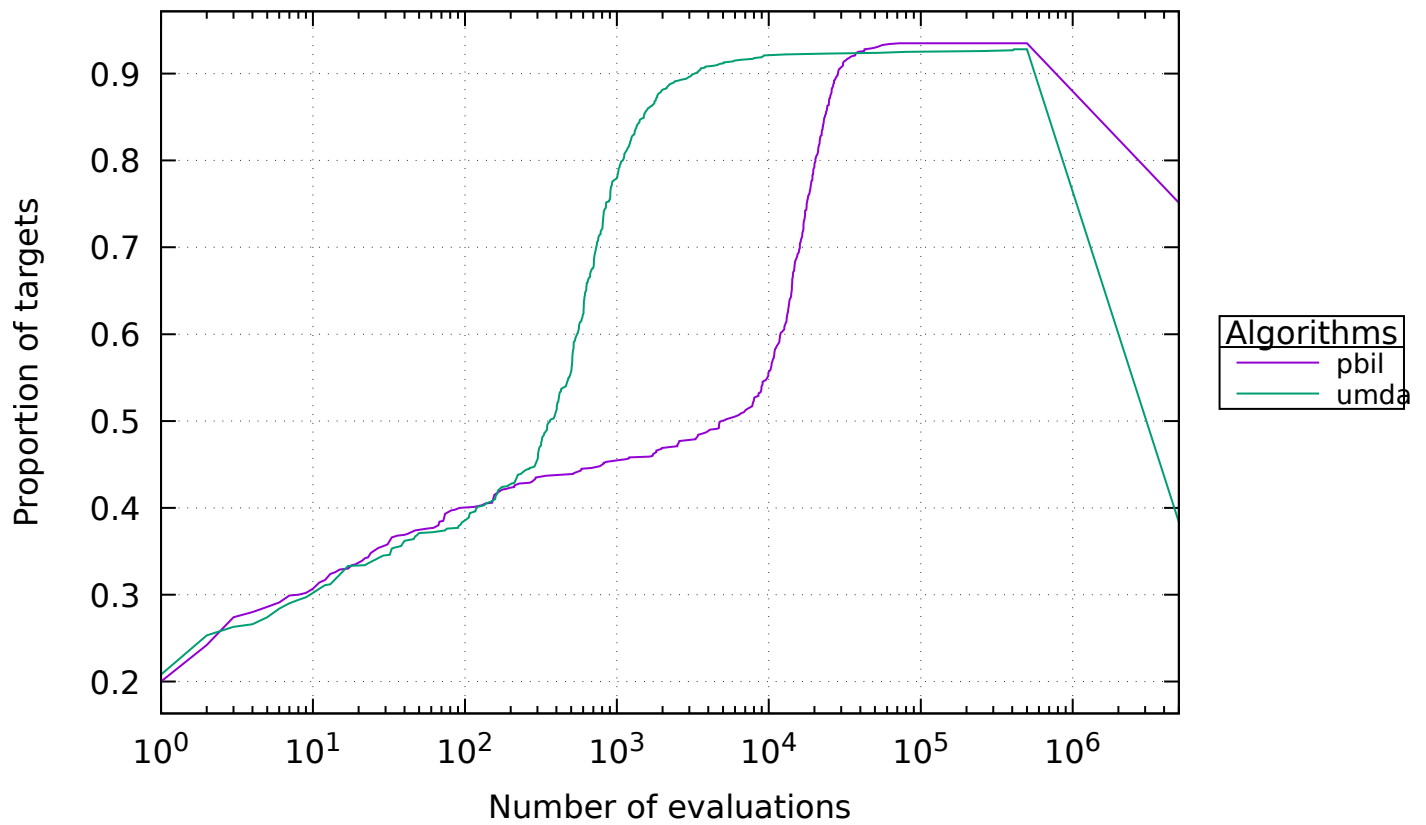


### 20.2 Groups

#### 20.2.1 ec







## References

Nikolaus Hansen, Anne Auger, Dimo Brockhoff, Dejan Tutar, and Tea Tutar. COCO: performance assessment. *CoRR*, abs/1605.03560, 2016. URL <http://arxiv.org/abs/1605.03560>.

## A Plan

```
{
  "exec": "hnco",
  "opt": "--log-improvement --map 1 --map-random -s 100",
  "budget": 500000,
  "num_runs": 20,
  "num_targets": 50,
  "parallel": true,
  "graphics": {
    "global": {
      "all": {
        "helper": true
      },
      "groups": {
        "helper": true
      }
    },
    "function": {
      "all": {
        "helper": true
      },
      "groups": {
        "helper": true
      }
    }
  }
},
```

```

"groups": [
  {
    "id": "ec",
    "algorithms": [ "ea-1p1", "ea-1p10", "ea-10p1", "ea-1c10", "ga" ]
  },
  {
    "id": "eda",
    "algorithms": [ "pbil", "umda" ]
  }
],
"functions": [
  {
    "id": "one-max",
    "opt": "-F 0 --stop-on-maximum"
  },
  {
    "id": "lin",
    "opt": "-F 1 -p instances/lin.100"
  },
  {
    "id": "leading-ones",
    "opt": "-F 10 --stop-on-maximum"
  },
  {
    "id": "ridge",
    "opt": "-F 11 --stop-on-maximum"
  },
  {
    "id": "jump-5",
    "opt": "-F 30 --stop-on-maximum -t 5"
  },
  {
    "id": "jump-10",
    "opt": "-F 30 --stop-on-maximum -t 10"
  },
  {
    "id": "djump-5",
    "opt": "-F 31 --stop-on-maximum -t 5"
  },
  {
    "id": "djump-10",
    "opt": "-F 31 --stop-on-maximum -t 10"
  },
  {
    "id": "fp-5",
    "opt": "-F 40 --stop-on-maximum -t 5"
  },
  {
    "id": "fp-10",
    "opt": "-F 40 --stop-on-maximum -t 10"
  },
  {
    "id": "nk",
    "opt": "-F 60 -p instances/nk.100.4"
  },
  {
    "id": "max-sat",
    "opt": "-F 70 -p instances/ms.100.3.1000"
  },
  {
    "id": "labs",
    "opt": "-F 81"
  }
],

```

```

{
  "id": "ep",
  "opt": "-F 90 -p instances/ep.100",
  "reverse": true,
  "logscale": true
},
{
  "id": "cancel",
  "opt": "-F 100 -s 99",
  "reverse": true
},
{
  "id": "trap",
  "opt": "-F 110 --stop-on-maximum --fn-num-traps 10"
},
{
  "id": "hiff",
  "opt": "-F 120 --stop-on-maximum -s 128"
},
{
  "id": "plateau",
  "opt": "-F 130 --stop-on-maximum"
},
{
  "id": "walsh2",
  "opt": "-F 162 -p instances/walsh2.100"
}
],
"algorithms": [
{
  "id": "sa",
  "opt": "-A 200 --sa-beta-ratio 1.05 --sa-num-trials 10"
},
{
  "id": "ea-1p1",
  "opt": "-A 300"
},
{
  "id": "ea-1p10",
  "opt": "-A 310 --ea-mu 1 --ea-lambda 10"
},
{
  "id": "ea-10p1",
  "opt": "-A 310 --ea-mu 10 --ea-lambda 1"
},
{
  "id": "ea-1c10",
  "opt": "-A 320 --ea-mu 1 --ea-lambda 10 --allow-no-mutation"
},
{
  "id": "ga",
  "opt": "-A 400 --ea-mu 100"
},
{
  "id": "pbil",
  "opt": "-A 500 -l 5e-3"
},
{
  "id": "umda",
  "opt": "-A 600 -x 100 -y 10"
}
]
}

```

## B Default parameters

```
# algorithm = 100
# bm_mc_reset_strategy = 1
# bm_num_gs_cycles = 1
# bm_num_gs_steps = 100
# bm_sampling = 1
# budget = 10000
# bv_size = 100
# description_path = description.txt
# ea_lambda = 100
# ea_mu = 10
# expression = x
# fn_name = noname
# fn_num_traps = 10
# fn_prefix_length = 2
# fn_threshold = 10
# fp_expression = (1-x)^2+100*(y-x^2)^2
# fp_lower_bound = -2
# fp_num_bits = 8
# fp_upper_bound = 2
# function = 0
# ga_crossover_bias = 0.5
# ga_crossover_probability = 0.5
# ga_tournament_size = 10
# hea_bit_herding = 0
# hea_num_seq_updates = 100
# hea_reset_period = 0
# hea_sampling_method = 0
# hea_weight = 1
# learning_rate = 0.001
# map = 0
# map_input_size = 100
# map_path = map.txt
# map_ts_length = 10
# map_ts_sampling_mode = 0
# mutation_rate = 1
# neighborhood = 0
# neighborhood_iterator = 0
# noise_stddev = 1
# num_iterations = 0
# num_threads = 1
# path = function.txt
# pn_mutation_rate = 1
# pn_neighborhood = 0
# pn_radius = 2
# population_size = 10
# pv_log_num_components = 5
# radius = 2
# results_path = results.json
# rls_patience = 50
# sa_beta_ratio = 1.2
# sa_initial_acceptance_probability = 0.6
# sa_num_transitions = 50
# sa_num_trials = 100
# seed = 0
# selection_size = 1
# solution_path = solution.txt
# target = 100
# print_defaults
# last_parameter
# exec_name = hnco
# version = 0.15
```

# Generated from hnco.json