findBestHyperparam

July 19, 2022

```
[NbConvertApp] WARNING | pattern '#' matched no files
[NbConvertApp] WARNING | pattern 'pdf,' matched no files
[NbConvertApp] WARNING | pattern 'html,' matched no files
[NbConvertApp] WARNING | pattern 'latex' matched no files
[NbConvertApp] Converting notebook findBestHyperparam.ipynb to html
[NbConvertApp] Writing 1743033 bytes to findBestHyperparam.html
```

0.1 1. Datasets insight

We used a corpus made of 16 datasets: 2 public (pX datasets), 4 synthetic sampled for agile projects (aX datasets), and 10 synthetic sampled for plan-driven projects (cX and dX datasets): - *pX datasets*: 2 small datasets which have been used in several works, as in 'Ant Colony Optimization for the Next Release Problem'. P1 has very few requirements (20) and also very few dependencies (7). P2 makes a bit more effort in providing more realistic project datasets, althouth dependencies are requirements are not too large.

- *aX datasets*: commonly agile projects have a lower number of stakeholders actively managed, althouth their buy-in in the development of the product if more constant. Thus, aX datasets present a lower number of stakeholders. Since requirements are not decided a priori with a long elicitation process, requirements are dependencies among them are not usually large for a given minor o functional release. Thus, we produced datasets with not a ver high number of dependencies and requirements. For the sake of completenees, two aX dataset have a large number of requirements. Effort estimations are computed using a fibonacci scale, similar to some agile estimation techniques.

- *cX datasets*: classic or plan-driven datsets tend to have a large number of requirements and, due to a long and expensive planning, algo a large number of dependencies. Also due to usual processes which deal with management of stakeholders interests, it is also common to identify more stakeholders than in agile datasets. Effort values were simulated by using Function Points values extracted from the 2015 version of the International Software Benchmarking Standards Group (ISBSG) dataset, using {A,B} values for "Unadjusted Function Points rating", "New development" for "Development type" and "IFPUG 4+" for "Count approach". This procedure is used to generate percentile 25,50,75 of total FPs of a classic project, in order to generate a realistic sample of classic estimation of requirements, done by selecting randomly, for a given number of requirements, a list of costs that sums up to the percentile value.

- *dX datasets*: following the same procedure than for *cX datasets*, here we simulate the most complex classical projects, with very large number of requirements (column #PBI) and dependencies (column #(PBI-->[PBI])). In fact, this is the case in which the MONRP might be of greater help for the decision maker.

- *eX|datasets*: again, derived from classical estimation of effort with Function Points from SBSG 2015, these datasets not only contain large number of requirements, and requirements which imply dependencies, but also the cardinality of these dependencies is also large (column Avg_len[PBI]); that is, when a requirement has a dependency X --> [list of requirements], this list is larger in the eX datasets compared to the others. ---> aún no añadidos en la experimentación, pendiente de que termine MIMIC.

```
   Dataset #Stakeholders #PBI #(PBI-->[PBI])  %(PBI-->[PBI])  Avg_len[PBI]
0       p1             5   20              7        0.350000      1.857143
1       p2             5  100             29        0.290000      2.689655
2       a1             5   50             18        0.360000      2.222222
3       a2            15   50             18        0.360000      2.722222
4       a3             5  200             74        0.370000      1.945946
5       a4            15  200             75        0.375000      2.253333
6       c1            15   50             20        0.400000      2.400000
7       c2           100   50             17        0.340000      3.529412
8       c3            15  200             69        0.345000      1.942029
9       c4           100  200             75        0.375000      2.093333
10      d1            15  100             45        0.450000      2.844444
11      d2            50  100             39        0.390000      2.102564
12      d3            15  200             88        0.440000      3.352273
13      d4            50  200             88        0.440000      4.852273
14      d5            50  200             88        0.440000      3.477273
15      d6            15  300            131        0.436667      3.770992
16      d7            50  300            145        0.483333      3.696552
17      e1            50  200             66        0.330000      6.227273
18      e2            15  300             99        0.330000      9.404040
19      e3            50  300             98        0.326667     12.428571
20      e4            50  200             73        0.365000      8.890411
21      e5            15  300            135        0.450000      4.518519
22      e6            50  300            139        0.463333      5.870504
```

0.2 2. FEDA description:

```
Given an initial set of requirements dependencies in the form of
X1-->X2, FEDA uses this knowledge as a prefixed structure.
e.g: we can have an acyclic graph like this: G={0-->2, 1-->2, 3, 2-->4},
 where requirements 0,1,3 do not have parents,
parents(2)={0,1} and parents(4)={2}.

Thus, learning is not structural and only applies to data.
Sampling is always performed following a topological (ancestral) order
 ([3,0,1,2,4] in the example above).

Algorithm is as follows:

1. Sampling of First generation:
```

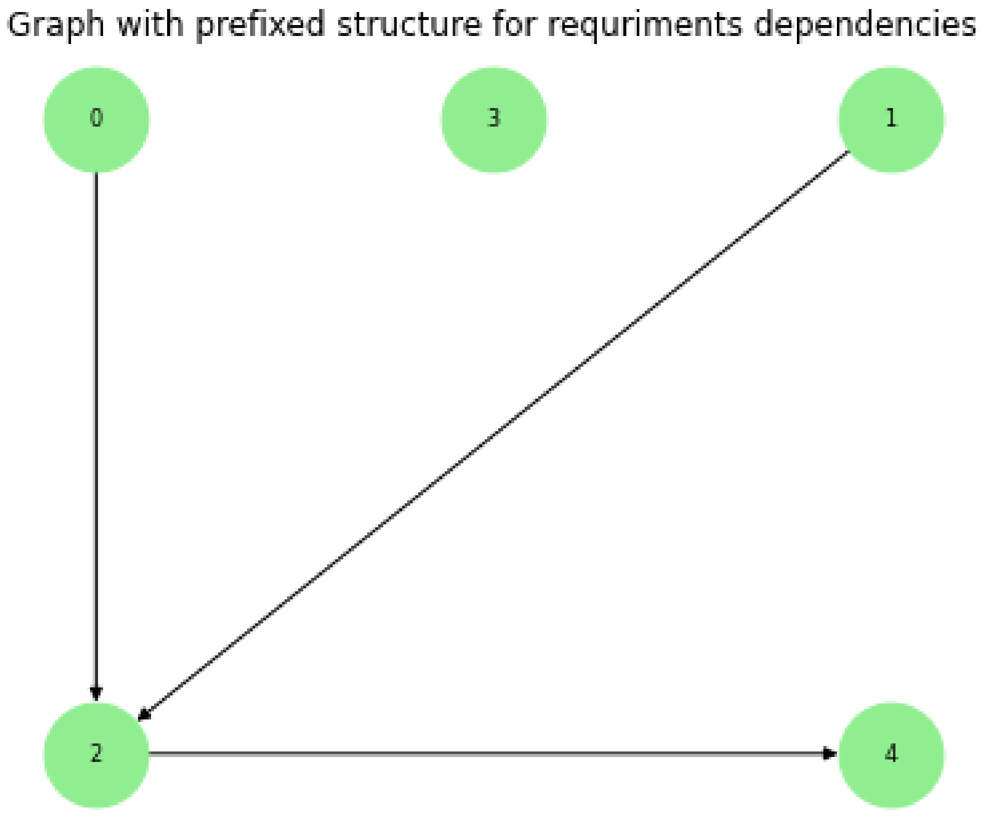
```
-- If X does not have any parents, then sample using  P(X)=1/num_requirements
-- If any Y in parents(X) is set to 1, then X=1, else use P(X)=1/num_requirements

do

    2. Learning
    -- If X does not have any parents in graph structure, then learn
    its marginal probability
    -- If X does have parents in graph structure, learn Conditional:
     P(X| all Y in parents(X)==0) In the example above, P(2| 0==0,1==0).
    Thus, we only need to learn P(X|parents(X)) from requirements
    whose parents Y are not selected.
    That is, we do not need P(X | any parents(X)==1), just the
    all parents(X)==0 case.
    This means that conditional probability can be stored in a
    unidimensional array,
     using the same array to store either marginal or conditional
     probability for each X.

    3. Sampling
    -- In the case of requirements without parents in graph structure,
     use learned marginal probability
    -- In any Y in parents(X) is set to 1, then X=1,
    else use P(X|parents(X)==0)

while(!stop_criterion)
```



Ancestral order: [3,0,1,2,4]

a) Sampling of first generation: In the dependencies structure shown above, when sampling the first set of solutions, the requirement in each solution is selected given the following probabilites:

P(3) = 1/5

P(0) = 1/5

P(1) = 1/5

P(2) = 1 if requirement 0 or 1 has been selected in current solution; 1/5 otherwise.

P(4) = 1 if requirement 2 has been selected in current soluction; 1/5 otherwise.

Let us assume that we sample 6 individuals with the following result:

solutions =

```
[ [0 0 1 0 1]

  [1 0 1 1 1]

  [0 1 1 0 1]
```

```
    [0 0 0 0 0]

    [0 1 1 0 1]

    [0 1 0 1 1]

    [0 0 0 1 0]
  ]
```

An impossible sampled individual would be, for example: [0 1 0 0 1] because requirement 2 should be selected since requirement 1 is. Thus, the dependencies graph structured is respected.

The whole population is evaluated, and the local NDS in current iteration set is identified. Let us assume this NDS:

nds_local =

```
[
 [1 0 1 1 1]

 [0 0 1 0 1]

 [0 1 0 1 1]

 [0 0 0 1 0]
]
```

nds_global = nds_local

b) Learning This step consists in updating the sampling probability of each requirement from the nds_local population.

P(0) = 1/4

P(1) = 1/4

P(2) = P_nds_local(2|requirement_0=0 and requirement_1=0) = 1/2

P(3) = 2/4

P(4) = P_nds_local(4|requirement_0=0) = 2/3

Thus, our probabilities vector is now:

probabilities =

```
[0.25, 0.25, 0.5, 0.5, 0.66]
```

Sidenote about learning

we could have special cases in which a requirement is never selected. For instance, let us imagine this local NDS:

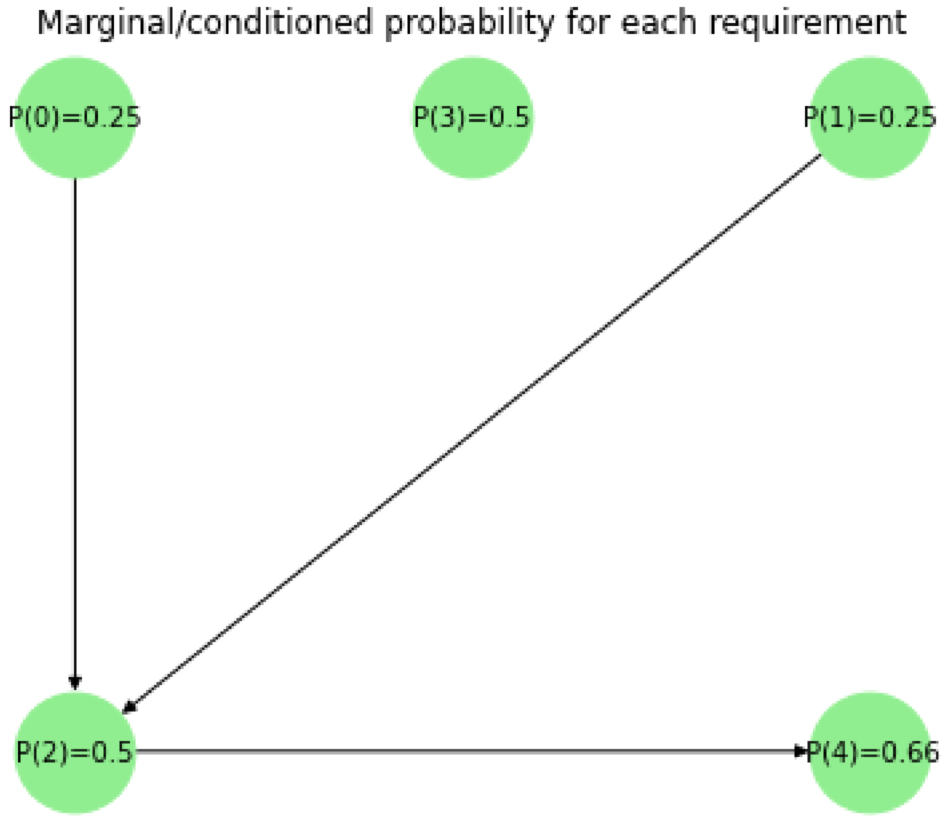
nds_local =

```
[
 [1 0 1 1 1]

 [0 0 1 0 1]

 [0 0 0 1 1]
]
```

P(1) = 0, so sampling of new individuals would not select requirement 1 anymore. In this case, we keep P(1) as its previous stored probability. Another possible alleaviations to this cases could be learning P(1) from the global NDS, or using some smooth.



c) Sampling Each new individual is sampled given the Ancestral order: [3,0,1,2,4], and using the following probabilities:

P(3) = 0.5

P(0) = 0.25

P(1) = 0.25

P(2) = 1 if requirement 0 or 1 is selected. 0.5 otherwise.

P(4) = 1 if requirement 2 is selected. 0.66 otherwise.

Let us assume that we sample 6 individuals with the following result:

new_solutions =

```
        [ [0 0 0 1 1]

          [0 0 1 1 1]

          [1 1 1 0 1]

          [0 0 1 1 1]

          [0 0 0 1 0]

          [0 0 0 0 1]
        ]
```

We evaluate the solutions to: - update the global NDS given the new_solution population - set local NDS to the nds found in new_soluction population

Repeat Learning+Sampling until stop criterion.

0.3 3. Search of the best hyperparameters configuration for each algorithm.

3.1 Best configuration for: FEDA

```
These are the different values used to set hyperparameters in FEDA, for each
dataset:
population_length: [ 100  200  500  700 1000]
max_generations: [ 50 100 200 300 400]
max_evaluations: [0]
selection_scheme: ['nds']
In total, 25 configuration per dataset.

Counts of best configurations found in 23 datasets. Please notice that those
with less than the  maximum possible #iterations or #solutions_per_iteration
have converged sooner, and they obtain the same HV with higher configurations,
thus the least configuration possible is shown. In conclusion: 'best
configuration' can be interpreted as 'minimum configuration to converge'.
```

```
  population_length max_generations max_evaluations selection_scheme  \
0              1000             300               0              nds
1              1000              50               0              nds
2              1000             400               0              nds
3              1000             200               0              nds
4              1000             100               0              nds

                           datasets  \
0  [a1, a2, c2, c3, d2, d4, e4, e6]
1          [a3, a4, d1, d5, d7, e2]
2                      [c1, c4, e3]
```

```
3                       [d3, e5, p1]
4                       [d6, e1, p2]

                                                         HV  wins
0  [0.8728, 0.9435, 0.932, 0.9061, 0.8702, 0.808,…   8.0
1     [0.8244, 0.8047, 0.9126, 0.812, 0.781, 0.8102]   6.0
2                          [0.8999, 0.8521, 0.7821]   3.0
3                          [0.8099, 0.7628, 0.8789]   3.0
4                          [0.8011, 0.7941, 0.7891]   3.0

Best hyperparameter configuration for FEDA is:
population_length:1000
max_generations:300
max_evaluations:0
selection_scheme:nds
```

3.2 Best configuration for: GRASP It takes too long. In the few datasets in which results are available, GRASP's HV is better than all algorithms. However, in one week experiments did not finish for datasets with a large number of PBIs.

3.3 Best configuration for: GeneticNDS

```
These are the different values used to set hyperparameters in GeneticNDS, for
each dataset:
population_length: [ 100  200  500  700 1000]
max_generations: [ 50 100 200 300 400]
selection_candidates: [2]
crossover_prob: [0.8]
mutation_prob: [0.1 0.3]
mutation: ['flip1bit']
replacement: ['elitismnds']
In total, 50 configuration per dataset.

Counts of best configurations found in 17 datasets. Please notice that those
with less than the  maximum possible #iterations or #solutions_per_iteration
have converged sooner, and they obtain the same HV with higher configurations,
thus the least configuration possible is shown. In conclusion: 'best
configuration' can be interpreted as 'minimum configuration to converge'.

  population_length max_generations selection_candidates crossover_prob  \
0              1000              50                    2            0.8
1              1000             300                    2            0.8
2              1000             400                    2            0.8
3              1000             100                    2            0.8
4              1000             200                    2            0.8
5              1000             400                    2            0.8

  mutation_prob  mutation replacement                  datasets  \
0          0.3  flip1bit  elitismnds                [a1, d1]
1          0.3  flip1bit  elitismnds            [a2, c4, d7]
2          0.3  flip1bit  elitismnds  [a3, a4, c3, d4, d5, d6]
3          0.3  flip1bit  elitismnds                [c1, p2]
4          0.3  flip1bit  elitismnds            [c2, d2, d3]
5          0.1  flip1bit  elitismnds                    [p1]

                                                HV  wins
0                              [0.8163, 0.7937]   2.0
1                       [0.8916, 0.724, 0.6546]   3.0
2  [0.7022, 0.6978, 0.808, 0.6969, 0.7019, 0.6686]   6.0
3                               [0.8493, 0.685]   2.0
4                      [0.8285, 0.7663, 0.7074]   3.0
5                                      [0.8837]   1.0


Best hyperparameter configuration for GeneticNDS is:
population_length:1000
max_generations:400
selection_candidates:2
crossover_prob:0.8
mutation_prob:0.3
mutation:flip1bit
replacement:elitismnds
```

3.4 Best configuration for: UMDA

```
These are the different values used to set hyperparameters in UMDA, for each
dataset:
population_length: [ 100  200  500  700 1000]
max_generations: [ 50 100 200 300 400]
selection_scheme: ['nds']
replacement_scheme: ['elitism']
In total, 25 configuration per dataset.

Counts of best configurations found in 22 datasets. Please notice that those
with less than the  maximum possible #iterations or #solutions_per_iteration
have converged sooner, and they obtain the same HV with higher configurations,
thus the least configuration possible is shown. In conclusion: 'best
configuration' can be interpreted as 'minimum configuration to converge'.
```

```
  population_length max_generations selection_scheme replacement_scheme  \
0              1000              50              nds            elitism
1              1000             400              nds            elitism
2              1000             300              nds            elitism
3              1000             200              nds            elitism
4              1000             100              nds            elitism
5               200              50              nds            elitism
6               500              50              nds            elitism
```

```
                  datasets                                                  HV  \
0  [a1, a3, c3, c4, d3, p1]  [0.8639, 0.8031, 0.8779, 0.8204, 0.7945, 0.8783]
1          [a2, d2, d7, e6]                    [0.9303, 0.831, 0.7516, 0.7581]
2      [a4, c1, d1, e1, e5]          [0.7942, 0.8819, 0.865, 0.7706, 0.7464]
3                  [c2, d4]                                   [0.8631, 0.802]
4              [d5, d6, p2]                          [0.7844, 0.7648, 0.7713]
5                      [e2]                                          [0.7636]
6                      [e3]                                          [0.7569]

   wins
0   6.0
1   4.0
2   5.0
3   2.0
4   3.0
5   1.0
6   1.0


Best hyperparameter configuration for UMDA is:
population_length:1000
max_generations:50
selection_scheme:nds
replacement_scheme:elitism
```

3.5 Best configuration for: PBIL

```
These are the different values used to set hyperparameters in PBIL, for each
dataset:
population_length: [ 100  200  500  700 1000]
max_generations: [ 50 100 200 300 400]
max_evaluations: [0]
learning_rate: [0.1]
mutation_prob: [0.1]
mutation_shift: [0.1]
In total, 25 configuration per dataset.

Counts of best configurations found in 23 datasets. Please notice that those
with less than the  maximum possible #iterations or #solutions_per_iteration
have converged sooner, and they obtain the same HV with higher configurations,
thus the least configuration possible is shown. In conclusion: 'best
configuration' can be interpreted as 'minimum configuration to converge'.
```

```
  population_length max_generations max_evaluations learning_rate  \
0            1000.0           400.0             0.0           0.1
1             700.0           400.0             0.0           0.1
2             500.0           400.0             0.0           0.1
3             200.0           300.0             0.0           0.1
```

```
  mutation_prob mutation_shift  \
0           0.1            0.1  
1           0.1            0.1  
2           0.1            0.1  
3           0.1            0.1  

                                          datasets  \
0  [a1, c1, c2, c3, c4, d1, d2, d3, d4, d5, d6, d…   
1                                     [a2, a3, a4]   
2                                         [e5, e6]   
3                                             [p1]   

                                                HV  wins  
0  [0.8968, 0.903, 0.9421, 0.7414, 0.7336, 0.9033…  17.0  
1                         [0.9736, 0.6568, 0.6583]   3.0  
2                                 [0.6407, 0.6467]   2.0  
3                                         [0.8948]   1.0  

Best hyperparameter configuration for PBIL is:
population_length:1000.0
max_generations:400.0
max_evaluations:0.0
learning_rate:0.1
mutation_prob:0.1
mutation_shift:0.1
```

3.6 Best configuracion for: MIMIC

```
These are the different values used to set hyperparameters in MIMIC, for each
dataset:
population_length: [ 100  200  500  700 1000]
max_generations: [ 50 100 200 300 400]
max_evaluations: [0]
selection_scheme: ['nds']
selected_individuals: [ 50 100]
In total, 50 configuration per dataset.

Counts of best configurations found in 23 datasets. Please notice that those
with less than the  maximum possible #iterations or #solutions_per_iteration
have converged sooner, and they obtain the same HV with higher configurations,
thus the least configuration possible is shown. In conclusion: 'best
configuration' can be interpreted as 'minimum configuration to converge'.
```

```
  population_length max_generations max_evaluations selection_scheme  \
0              1000             200               0              nds   
1              1000             300               0              nds   
2              1000             400               0              nds   
```

```
3              1000              50               0               nds

  selected_individuals                                          datasets  \
0                   50                                              [a1]
1                   50                                      [a2, c1, e4]
2                   50  [a3, a4, c2, c3, c4, d1, d2, d3, d4, d5, d6, d…
3                   50                                              [p1]

                                                  HV  wins
0                                           [0.9106]   1.0
1                           [0.9701, 0.9313, 0.8107]   3.0
2  [0.8458, 0.8355, 0.9412, 0.9203, 0.8679, 0.912…  18.0
3                                           [0.9134]   1.0
```

```
Best hyperparameter configuration for MIMIC is:
population_length:1000
max_generations:400
max_evaluations:0
selection_scheme:nds
selected_individuals:50
```

All algorithms find their best results when using a Population Size = 1000, the maximum value among the 5 given for this hyperparameter.

Respect to the number of generations, all algorithms but UMDA are slow to converge, needing the maximum (400) almost the maximum (300 in FEDA) value among the possible values. UMDA seems to converge really soon, since in 6 out of 17 datasets it finds its best results with just 50 generations, in both agile and classic projects.

0.4 4. Pareto plots for each dataset, setting each algorithm with its best configuration found (wins among all datasets).

Given the most frequently best configuration (over all datasets), we plot the pareto for each dataset given that configuration. That is, the configuration for a given algorithm is the same acrooss all datasets, concretely the one which performed the best more times (more wins).

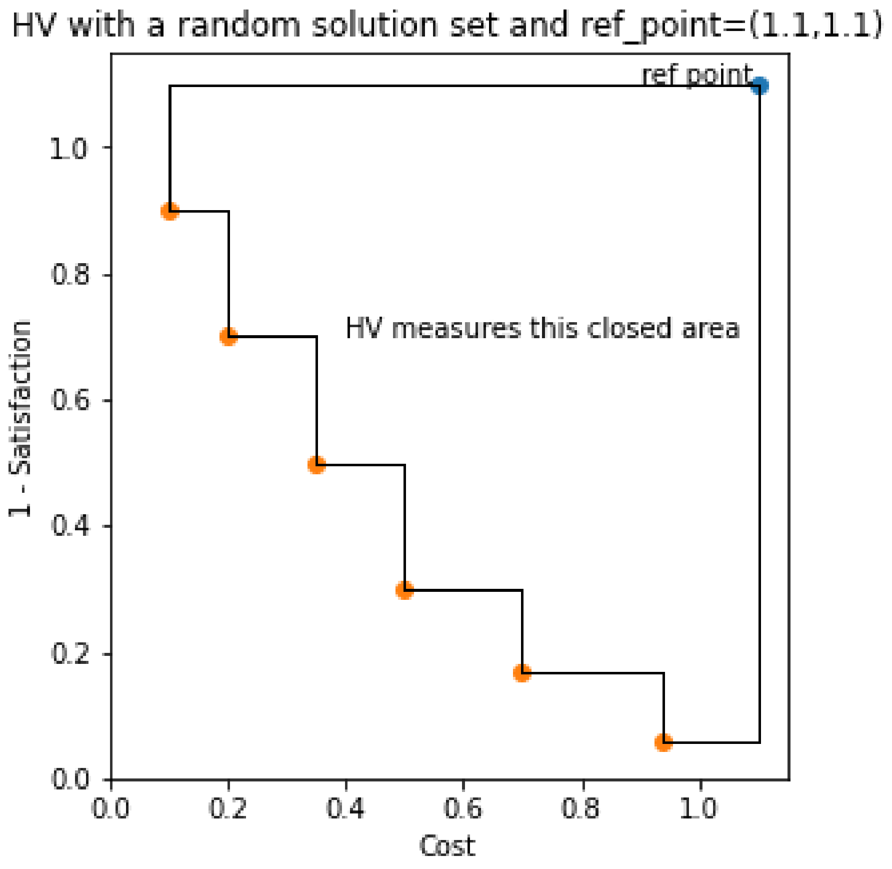
We show a plot for each dataset. In each plot, for each algorithm, we show the pareto front found in all the executions (commonly 30). Since the solutions subset size is commonly 10, thus for each algorithm we plot 300 points. Since each algorithm has 30 paretos, please note that such paretos are not non-dominated among them, which can be seen in the shapes they create.

4.1 Metrics We show the average results over 30 executions for each algorithm. Each execution produces a NDS, from wich we keep the subset of 10 solutions which maximize HV, as suggested in 'Difficulties in Fair Performance Comparison of Multi-Objective Evolutionary Algorithms'. Such a subset is constructed by following an incremental forward greedy search.

Metrics shown are: 4 Quality Indiciators (HV, gd+, UNFR, spread), time and cardinality of the global NDS found by the algorithm during execution.

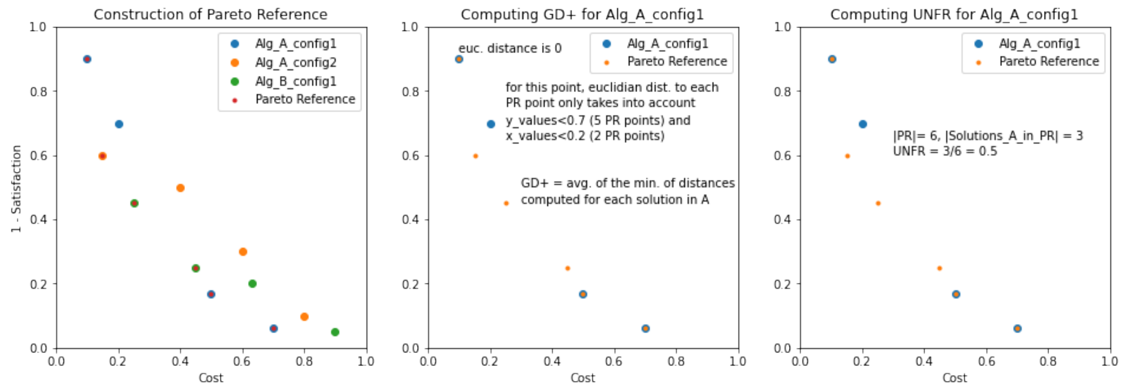
- HV (Hypervolume): this is the most widely used metric to assess paretos solutions for multi-objective problems and, concretly, in SBSE. It summarizes the four aspects of a solution set (convergence, spread, uniformity and cardinality); that is, this metric can be used as a compliance metric representing how good a Pareto front is. HV tends to be greater as knee points in the pareto are neareer the optimal point (0,0), thus it is preferable when Decision Makers prefer balanced solutions. In order to compute it, a reference point is needed, and this should be the same for all algorithms under comparison. Following the results and suggestions in (”How to evaluate solutions in pareto-based search-bases software engin.”), we set the following reference point in our bi-objective problem: $ref_x = nadir_x + range_x/10$ $ref_y = nadir_y + range_y/10$ The nadir point is the worst point found by algorithms during search. Since we normalize both cost and satisfaction, our worst point is 1 for both metrics (satisfaction is plotted as $1 - satisfaction$). Range is the difference between the best and worst point found. Best point is 0, so clearly the value of the reference point for both goals is $1+(1-0)/10 = 1.1$. HV is pareto compliance, so $HV_a > HV_b$ means that, visually, the pareto front of algorithm a dominates algorithm b. A great advantage of HV is that it does not need an ideal Pareto Reference, so its computation and fair comparison with other algorithm only needs a shared reference point which, thanks to goals normalization, is known a priori.

```
Text(0.4, 0.7, 'HV measures this closed area')
```



- GD+: General Distance (GD) covers the convergence aspect of the quality of solution set, measuring the Euclidian distance of such solution set to the ideal Pareto Reference. For each solution, its GD is the minimum of the distances to each point in the PF. In order to become GD compliant with Pareto dominance, GD+ enhances GD by measuring distances between points using only the goal coordinates which are superior in the Pareto Reference than those from the solutions set being measured. This metric is to be minimized, and a key point is the computation of the Pareto Reference, which needs to be done after execution of search algorithms. In our experiments, Pareto Reference is constructed by finding the Non Dominated Solutions set among all solutions sets found by all algorithm, under all hyperparameters configurations explained in Section 2.
- UNFR: Unique Non Dominated Front Ratio. It measures the ratio of solutio points in the PR which belong to the solution set of the evaluated algorithm. That is, it measures the contribution (from to 0 to 1) of an algorithm to the PR. Of curse, a point in the PR might be present in the solution sets of several algorithms. In our case, since the PR is constructed from such a large number of algorithms and configuration combinations, it presents a high cardinality of solutions. Furthermore, since each algorithm is evaluated using only using a selected subset of 10 points from its solution set, the UNFR value for each algorithm tends to be quite low, and the maximum possible is never 1, since the PR contais further more than 10 points. Anyway, greater UNFR values is desiderable.

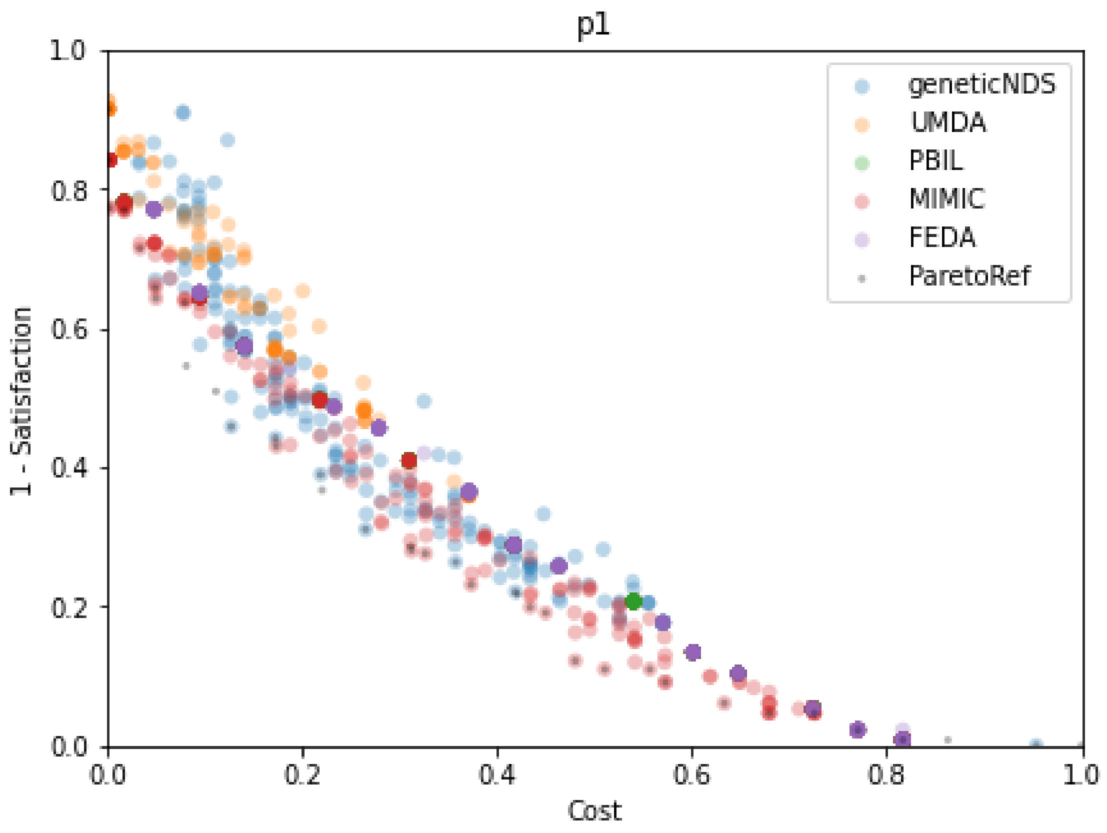
```
Text(0.3, 0.6, 'UNFR = 3/6 = 0.5')
```



4.2 Results Here we show one plot pero datasets, containing the result of all algorithms with their best configuration. As mentioned before, each algorithm is run 30 times; afeter each execution, we keep a subset of the NDS constructed. This subset contains the 10 solutions from NDS which maximize HV in a forward greedy search over the NDS. Thus, for each algorithm, we plot 10*30=300 points. This way, it is easy to identify the common pareto shape relative to the algorithm. After each plot, all metrics are tabbed to show the average over the 30 executions in the dataset.

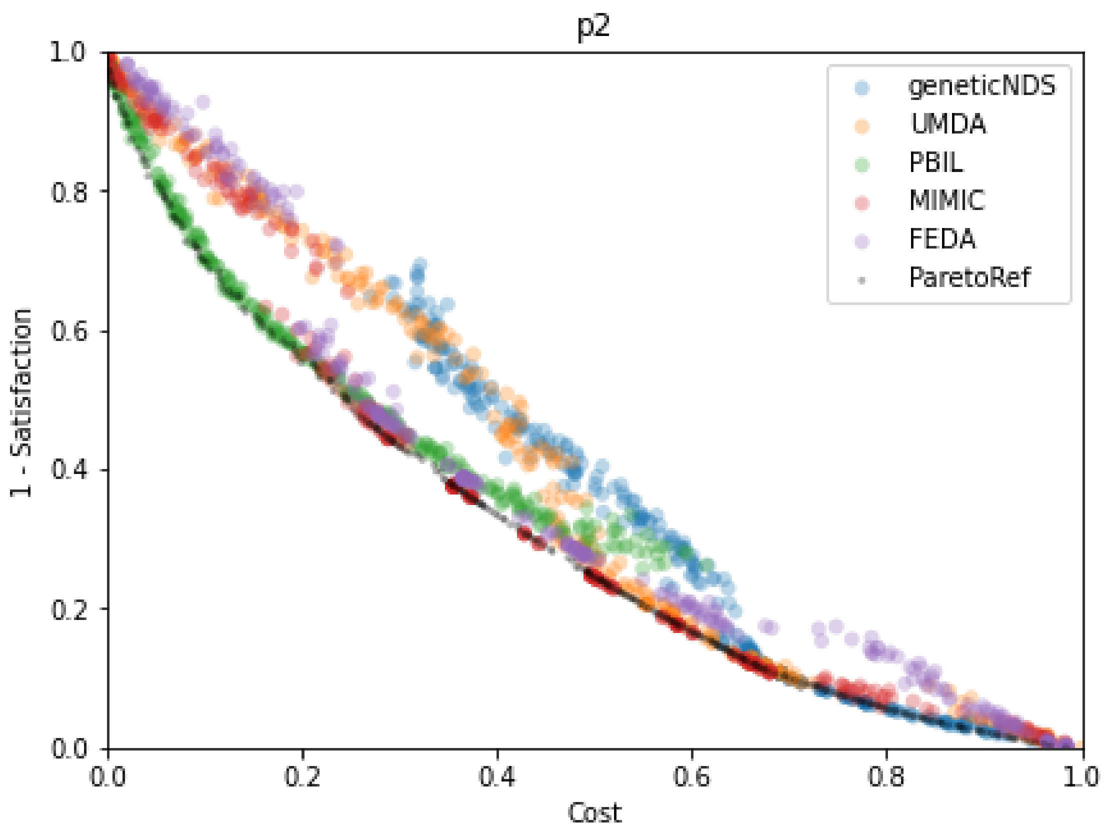
It is worth to mention that half of the execution time in algorithms is due to updating NDS after each iteration. Thus, although after each execution we select a subset of 10 solutions in the final NDS, more time consuming algorithms are usually those which tend to develop a NDS with higher cardinality suring their search.

```
Pareto Reference has 38 points
Maximum UNFR possible is 10/38=0.2632
```



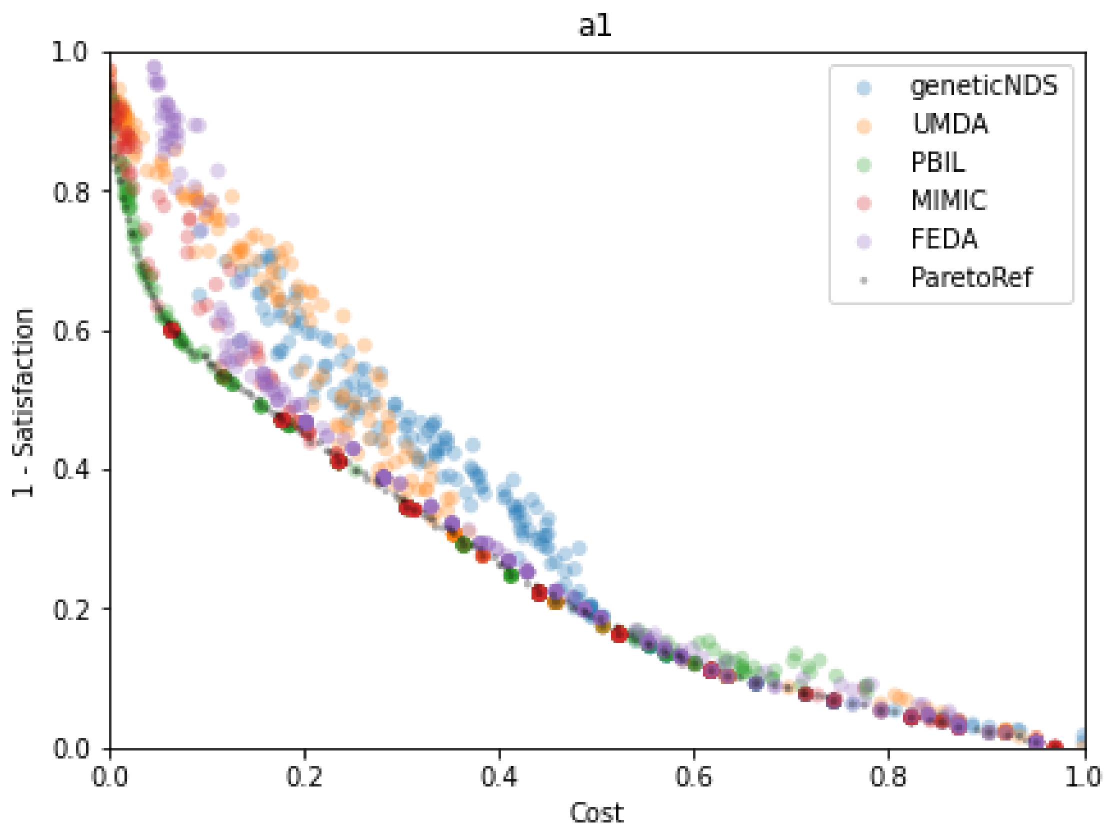
```
       Method      HV    UNFR     gd+   spread   time(s)    |NDS|
0  geneticNDS 0.8793 0.0386 0.0439  0.6453 800.8662 47.0667
1        UMDA 0.8783 0.0456 0.0527  0.6236  84.4473 34.8667
2        PBIL 0.8948 0.0263 0.0468  0.6038 452.2194 42.8333
3       MIMIC 0.9134 0.0772 0.0311  0.5853 765.0339 39.8667
4        FEDA 0.8787 0.0377 0.0500  0.6795 554.5325 31.9667

---------------------------------------------------------------
Pareto Reference has 309 points
Maximum UNFR possible is 10/309=0.0324
```



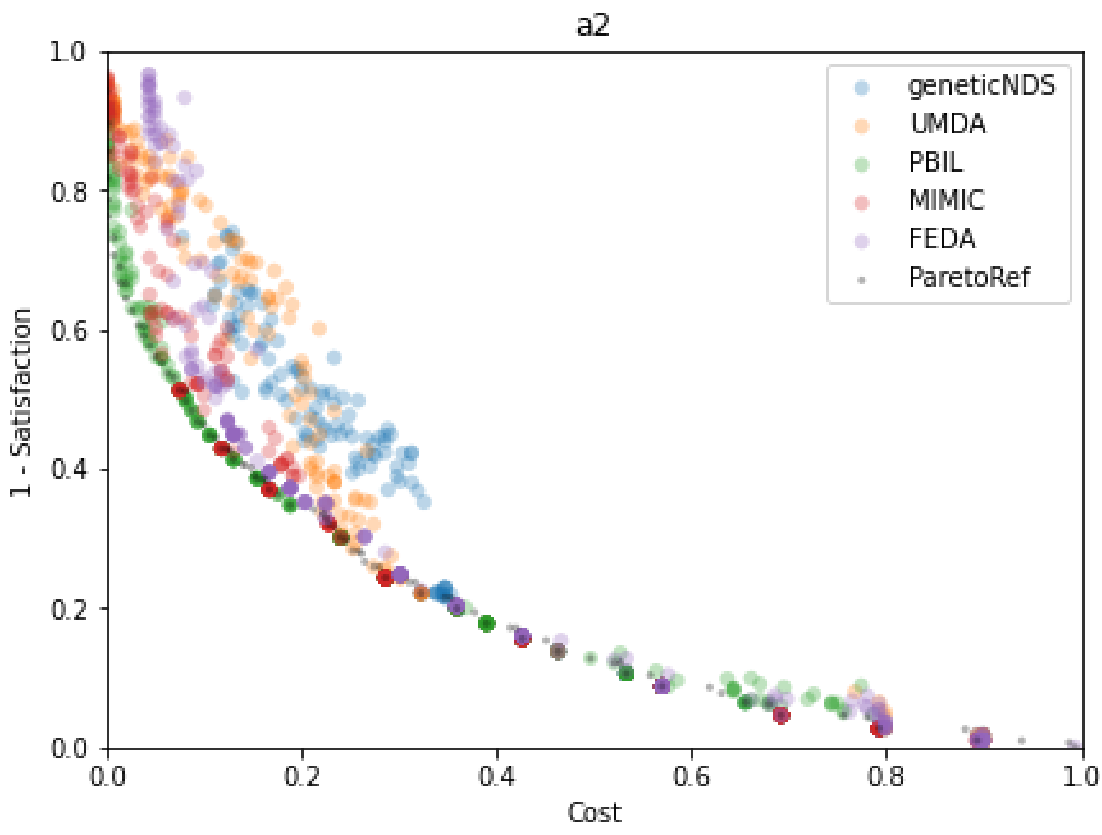
```
        Method      HV    UNFR     gd+  spread   time(s)     |NDS|
0  geneticNDS 0.6844 0.0047 0.0610  0.6608 2213.7621 114.0000
1        UMDA 0.7712 0.0016 0.0490  0.5759  335.5445 151.1667
2        PBIL 0.7390 0.0026 0.0159  0.5571 1024.6246 107.2667
3       MIMIC 0.8206 0.0132 0.0152  0.5886 5083.6006 321.0333
4        FEDA 0.7886 0.0010 0.0338  0.6018 3037.3493 217.2667

---------------------------------------------------------------
Pareto Reference has 148 points
Maximum UNFR possible is 10/148=0.0676
```



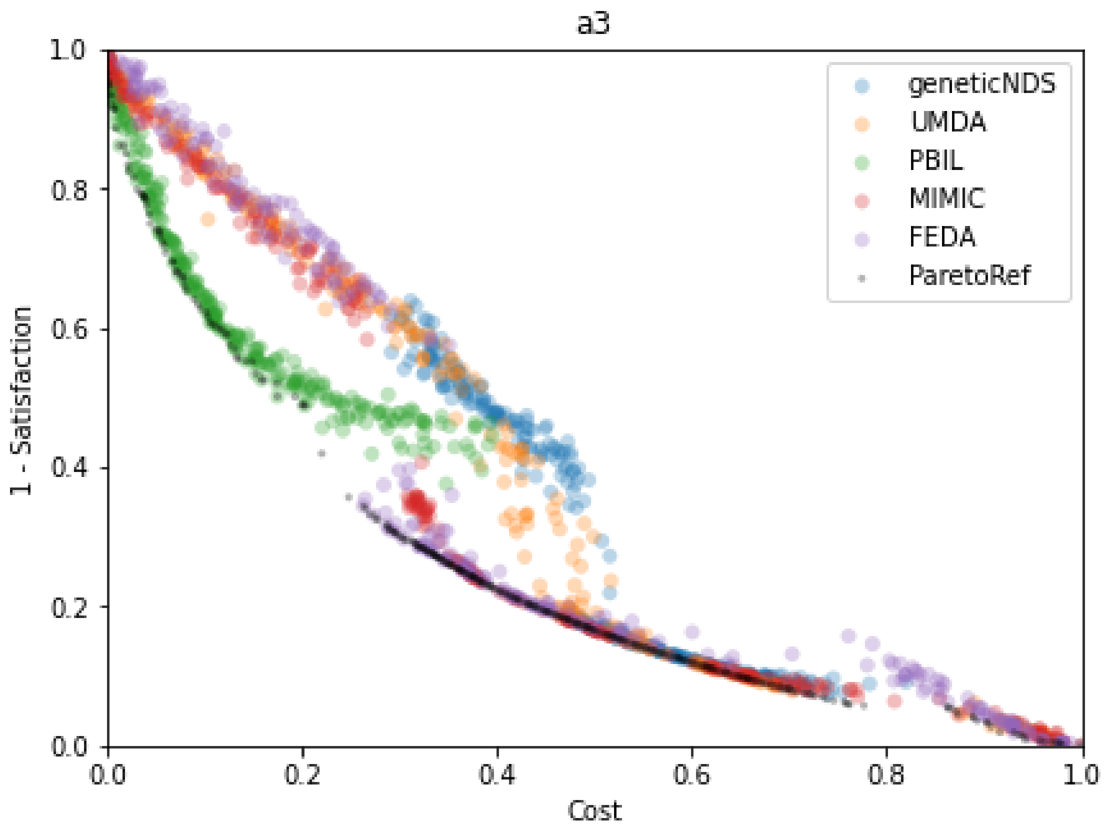
```
       Method      HV    UNFR     gd+   spread    time(s)     |NDS|
0  geneticNDS  0.8150  0.0268  0.0370   0.6649  1193.1049   64.5667
1        UMDA  0.8639  0.0266  0.0300   0.5756   174.7565   67.0000
2        PBIL  0.8968  0.0471  0.0031   0.6078   698.1781  110.7667
3       MIMIC  0.9096  0.0518  0.0043   0.5510  1859.5326  127.8000
4        FEDA  0.8728  0.0115  0.0179   0.5812  1427.4162   94.9333

---------------------------------------------------------------
Pareto Reference has 90 points
Maximum UNFR possible is 10/90=0.1111
```



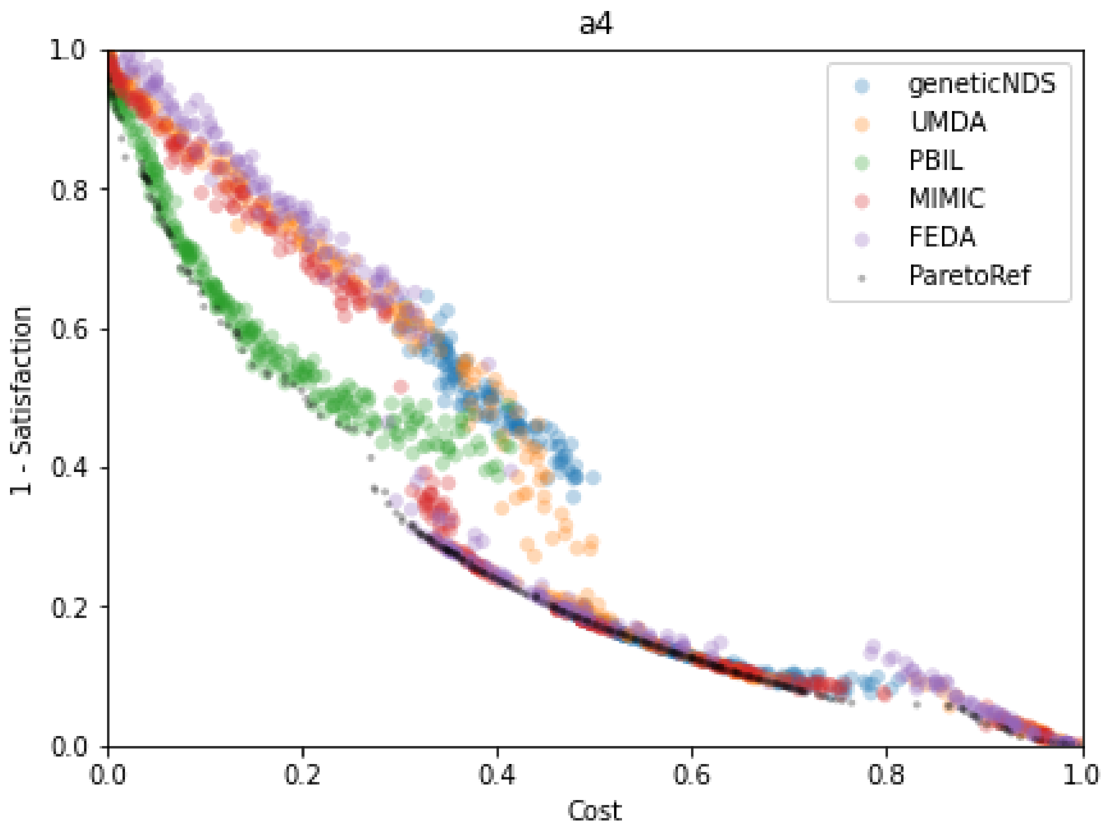
```
      Method      HV    UNFR     gd+   spread    time(s)    |NDS|
0  geneticNDS 0.8877 0.0685 0.0338  0.6420   806.1419 50.5000
1        UMDA 0.9290 0.0467 0.0275  0.5923   136.8646 47.1667
2        PBIL 0.9736 0.0759 0.0027  0.6097   548.6406 73.6000
3       MIMIC 0.9696 0.0770 0.0079  0.6071  1241.4517 78.4000
4        FEDA 0.9435 0.0137 0.0174  0.6313   662.6650 50.9667

---------------------------------------------------------------
Pareto Reference has 320 points
Maximum UNFR possible is 10/320=0.0312
```



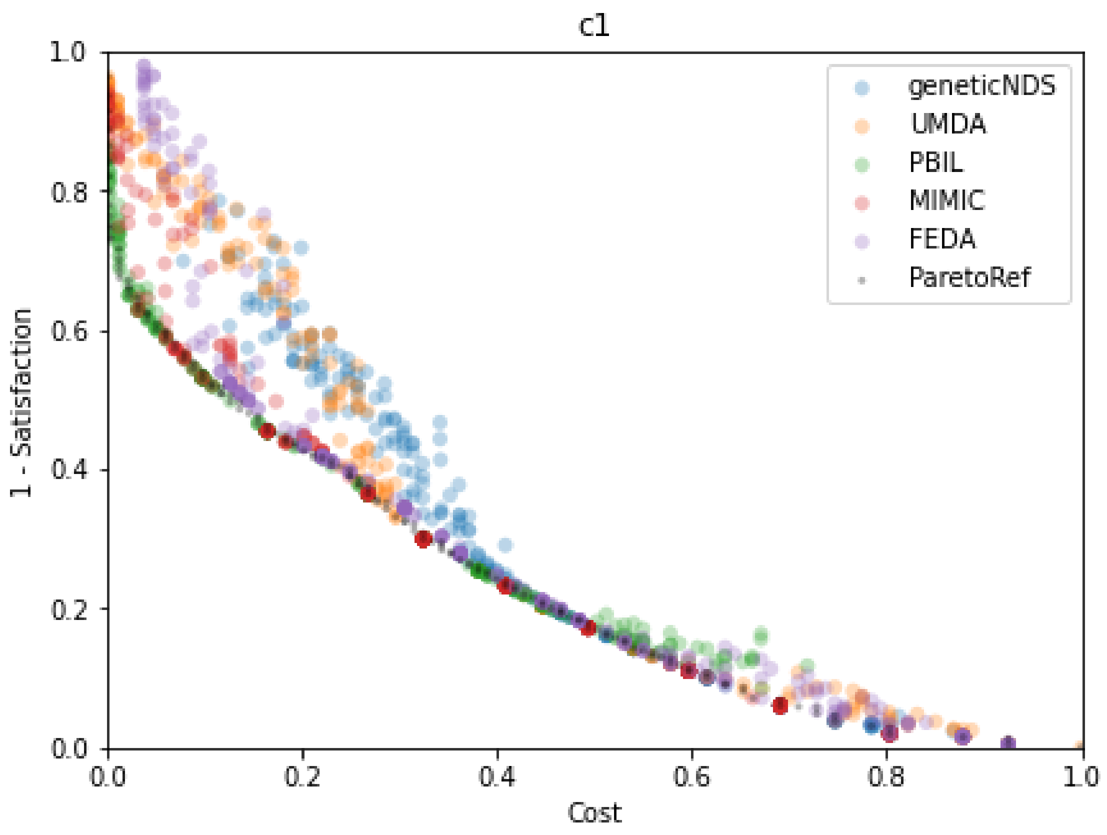
```
       Method      HV    UNFR     gd+  spread    time(s)     |NDS|
0  geneticNDS 0.7022 0.0017 0.0805  0.7520 1504.1177  86.9333
1        UMDA 0.8031 0.0030 0.0663  0.5809  344.3830 126.4667
2        PBIL 0.6558 0.0021 0.0275  0.5703  828.3615  62.8667
3       MIMIC 0.8458 0.0052 0.0354  0.5726 2980.0839 178.7000
4        FEDA 0.8238 0.0034 0.0454  0.5921 2268.7561 153.9000

---------------------------------------------------------------
Pareto Reference has 287 points
Maximum UNFR possible is 10/287=0.0348
```



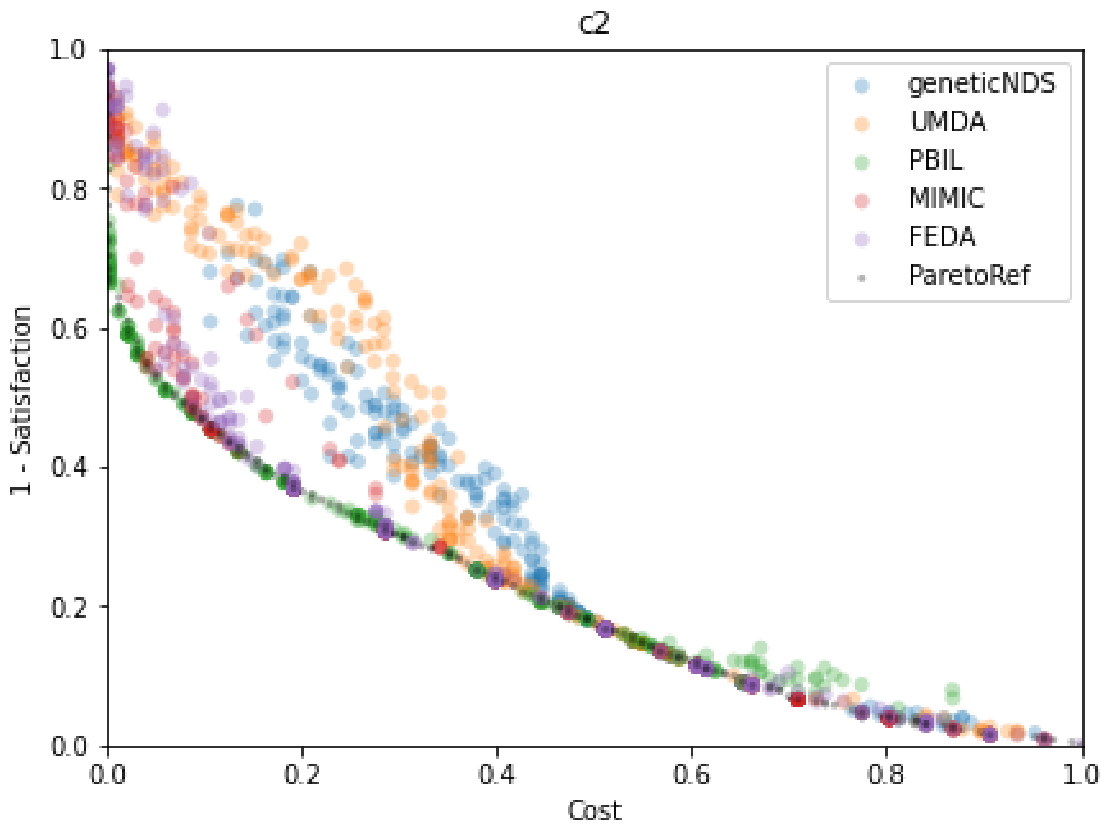
```
       Method     HV   UNFR    gd+  spread   time(s)     |NDS|
0  geneticNDS 0.6978 0.0053 0.0638  0.7291 1761.2607  99.8333
1        UMDA 0.7923 0.0023 0.0638  0.5875  375.8887 143.3667
2        PBIL 0.6576 0.0014 0.0237  0.5685  874.9545  64.6333
3       MIMIC 0.8355 0.0057 0.0339  0.5777 2973.7487 158.9000
4        FEDA 0.8022 0.0024 0.0497  0.6169 2775.3945 170.3000

--------------------------------------------------------------
Pareto Reference has 156 points
Maximum UNFR possible is 10/156=0.0641
```



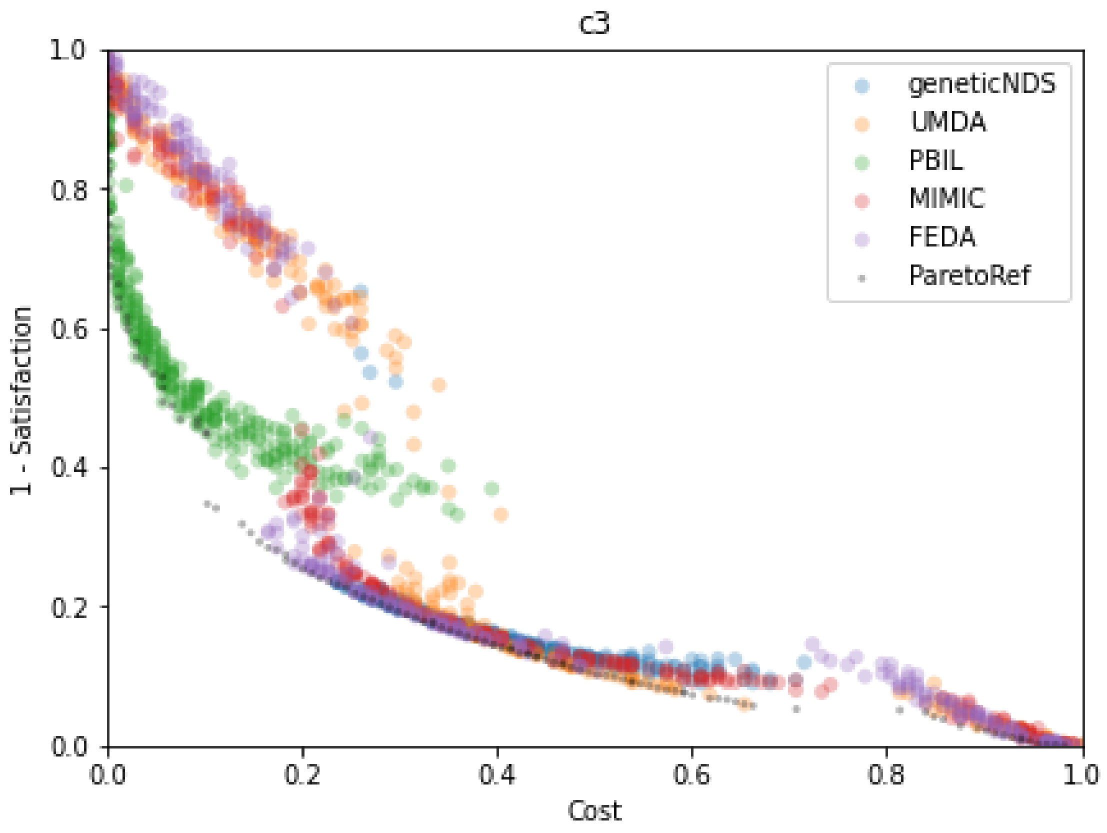
```
        Method      HV    UNFR     gd+  spread    time(s)     |NDS|
0   geneticNDS 0.8468 0.0316 0.0338  0.6418 1369.9759   86.2667
1         UMDA 0.8817 0.0271 0.0301  0.6068  223.7486   79.3000
2         PBIL 0.9030 0.0440 0.0037  0.6150  743.3746 105.7333
3        MIMIC 0.9311 0.0487 0.0051  0.6640 1865.9571 131.8667
4         FEDA 0.8967 0.0154 0.0170  0.6054 1202.8190   90.1667

---------------------------------------------------------------
Pareto Reference has 163 points
Maximum UNFR possible is 10/163=0.0613
```



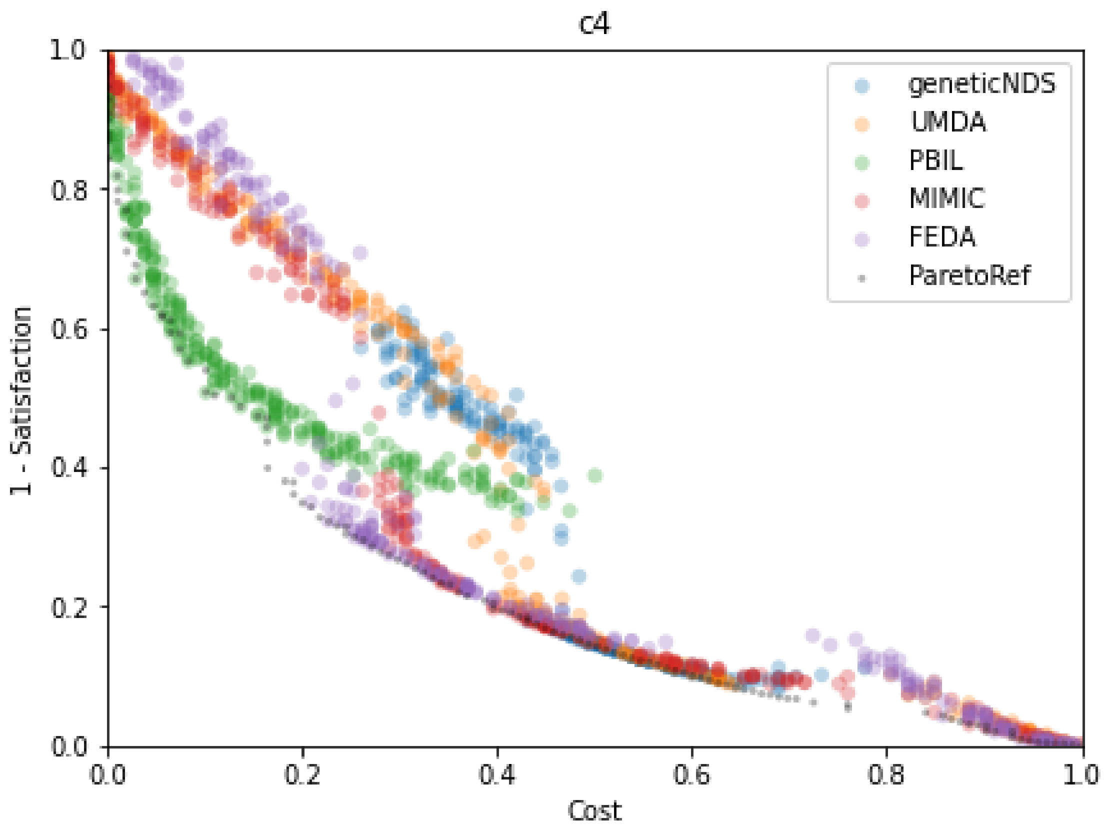
```
       Method     HV   UNFR    gd+  spread   time(s)     |NDS|
0  geneticNDS 0.8252 0.0247 0.0509  0.6707 1099.2675   81.3000
1        UMDA 0.8584 0.0196 0.0505  0.5627  185.9247   69.6333
2        PBIL 0.9421 0.0462 0.0030  0.6031  716.0132  103.4667
3       MIMIC 0.9412 0.0456 0.0064  0.5754 1989.3758  135.7000
4        FEDA 0.9320 0.0380 0.0072  0.5564 1342.6053   98.3000

---------------------------------------------------------------
Pareto Reference has 123 points
Maximum UNFR possible is 10/123=0.0813
```



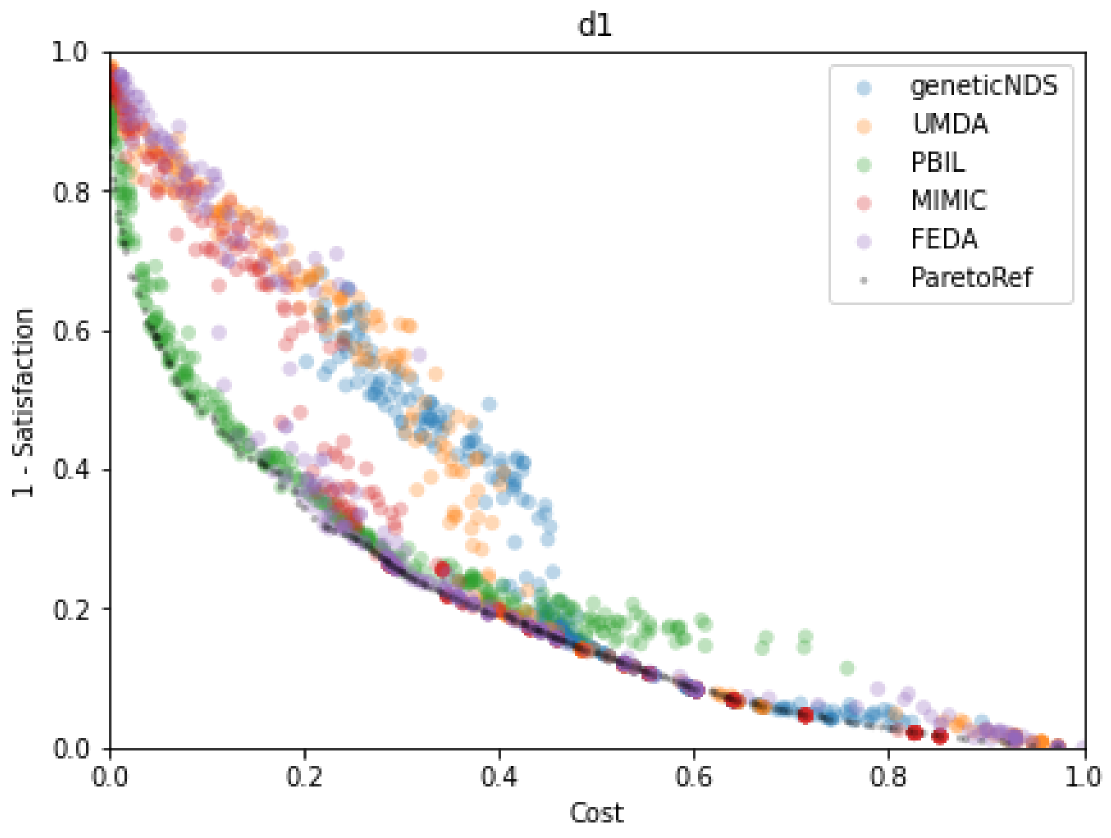
```
       Method      HV   UNFR     gd+  spread    time(s)   |NDS|
0  geneticNDS 0.8080 0.0016 0.0173  0.7739   874.2606 38.1333
1        UMDA 0.8779 0.0027 0.0612  0.5699   197.7094 65.7333
2        PBIL 0.7414 0.0043 0.0391  0.6520   531.1763 36.9000
3       MIMIC 0.9203 0.0011 0.0377  0.5956 2064.1445 64.8000
4        FEDA 0.9061 0.0035 0.0407  0.6381 1115.9604 57.4333

---------------------------------------------------------------
Pareto Reference has 121 points
Maximum UNFR possible is 10/121=0.0826
```



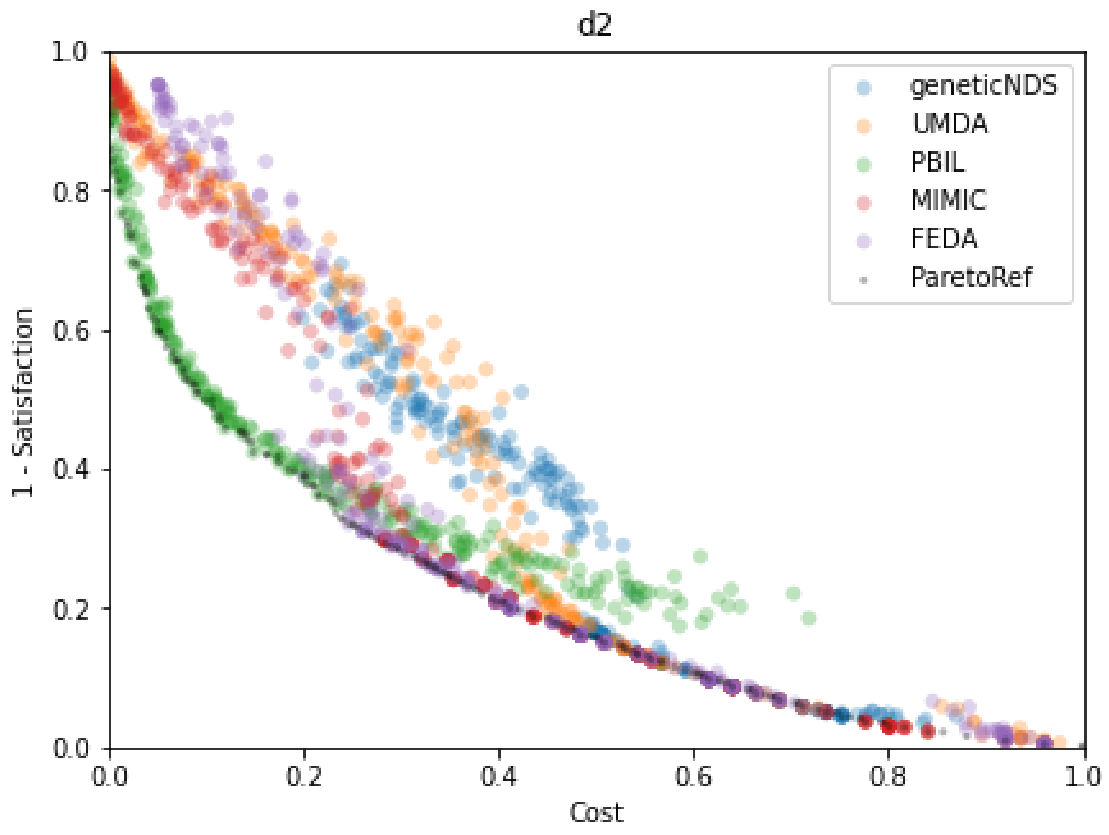
```
       Method     HV   UNFR    gd+  spread   time(s)   |NDS|
0  geneticNDS 0.7231 0.0025 0.0772  0.7743  820.1618 32.4667
1        UMDA 0.8204 0.0011 0.0755  0.5832  204.6724 61.0000
2        PBIL 0.7336 0.0047 0.0408  0.6020  656.5295 48.4333
3       MIMIC 0.8679 0.0050 0.0410  0.5706 2075.8709 73.1000
4        FEDA 0.8497 0.0050 0.0479  0.6148 1172.4405 59.4667

---------------------------------------------------------------
Pareto Reference has 209 points
Maximum UNFR possible is 10/209=0.0478
```



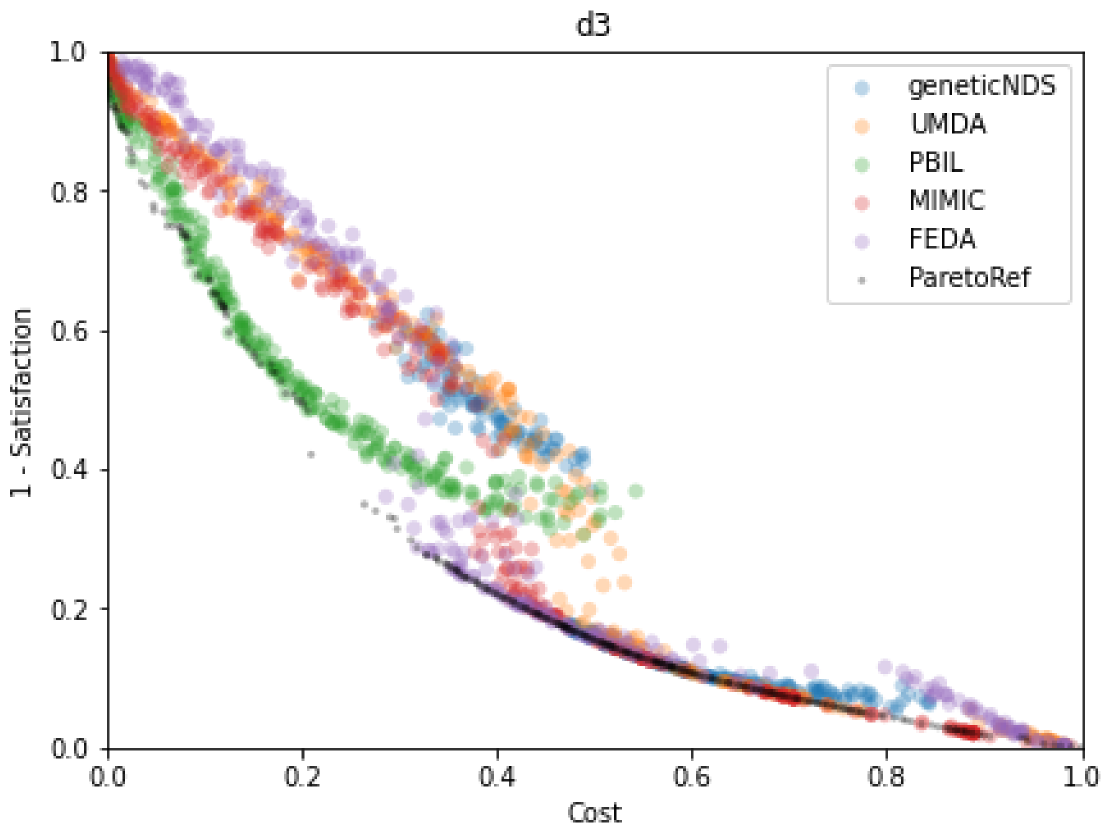
```
      Method     HV   UNFR    gd+  spread   time(s)     |NDS|
0  geneticNDS 0.7906 0.0129 0.0699  0.7176 1346.1592  74.6667
1        UMDA 0.8612 0.0099 0.0667  0.5579  239.4590  90.2667
2        PBIL 0.9033 0.0026 0.0201  0.6027  662.4540  72.0667
3       MIMIC 0.9129 0.0254 0.0280  0.6096 2864.2902 178.2333
4        FEDA 0.9060 0.0128 0.0273  0.6000 1907.2975 121.5333

---------------------------------------------------------------
Pareto Reference has 174 points
Maximum UNFR possible is 10/174=0.0575
```



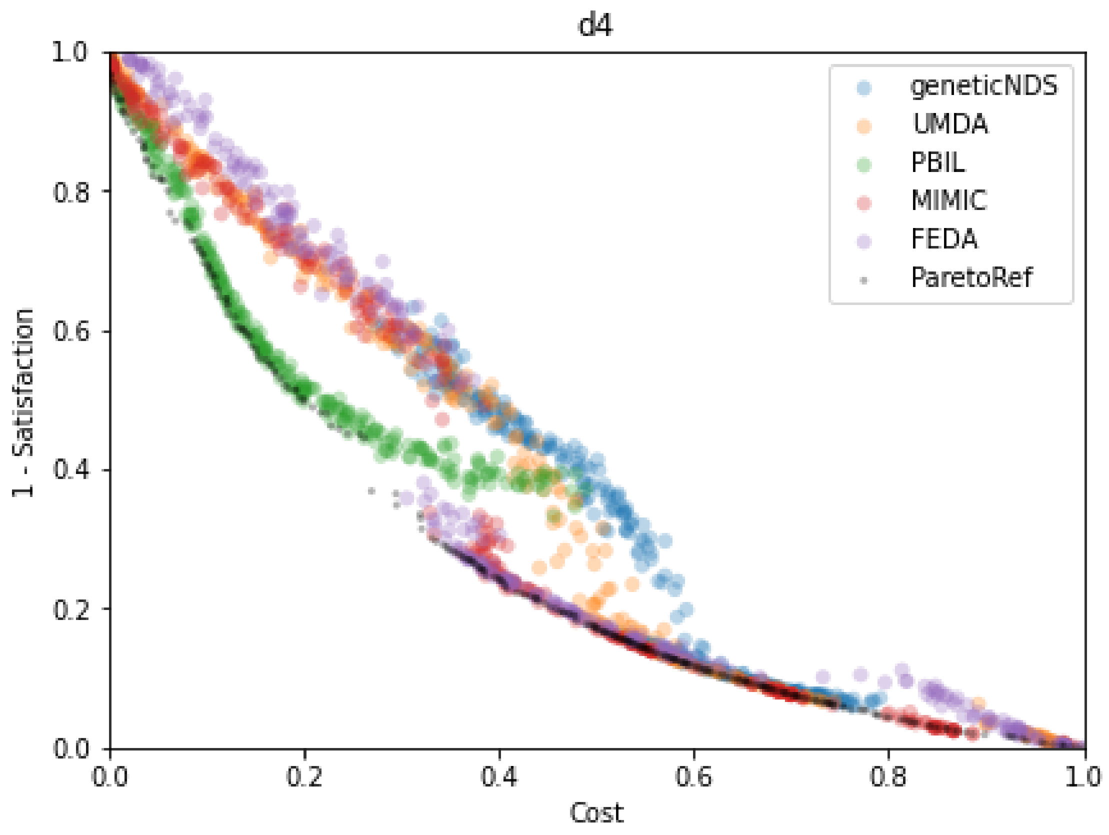
```
       Method     HV   UNFR    gd+   spread   time(s)     |NDS|
0  geneticNDS 0.7608 0.0211 0.0757  0.7147  939.1621  65.0000
1        UMDA 0.8304 0.0130 0.0732  0.5560  209.9717  71.2333
2        PBIL 0.8559 0.0042 0.0269  0.5932  671.5176  70.5667
3       MIMIC 0.8922 0.0236 0.0319  0.5895 2144.3830 138.6667
4        FEDA 0.8702 0.0220 0.0340  0.6145 1414.7278  86.5000

---------------------------------------------------------------
Pareto Reference has 315 points
Maximum UNFR possible is 10/315=0.0317
```



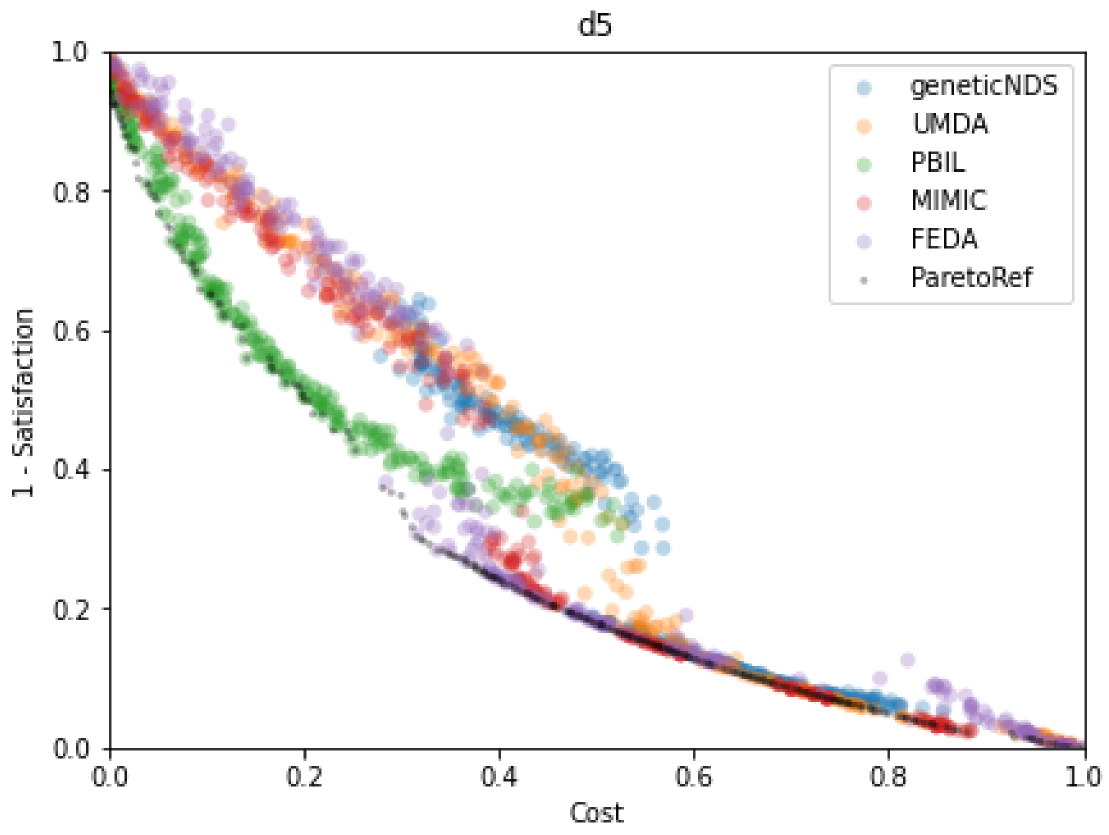
```
       Method      HV    UNFR     gd+   spread    time(s)     |NDS|
0  geneticNDS  0.7073  0.0062  0.0630   0.7581  1813.7526   86.8000
1        UMDA  0.7945  0.0033  0.0720   0.5627   338.0281  123.7667
2        PBIL  0.7197  0.0020  0.0388   0.5822   979.6848   71.1667
3       MIMIC  0.8298  0.0099  0.0451   0.5693  6314.2968  300.5333
4        FEDA  0.8098  0.0036  0.0523   0.5935  2108.9927  141.5000

---------------------------------------------------------------
Pareto Reference has 296 points
Maximum UNFR possible is 10/296=0.0338
```



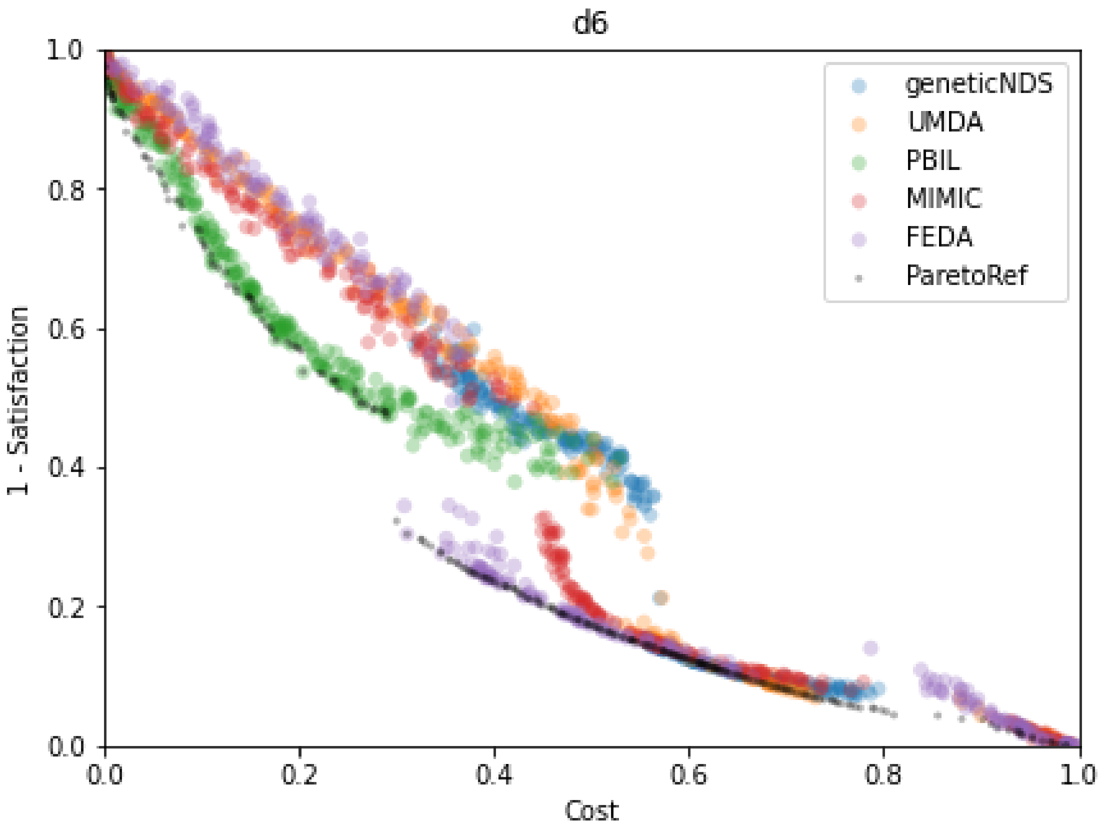
```
       Method     HV   UNFR    gd+  spread   time(s)     |NDS|
0  geneticNDS 0.6969 0.0016 0.0823  0.7401 1775.7885  89.2667
1        UMDA 0.8006 0.0037 0.0562  0.5709  397.6439 147.8667
2        PBIL 0.6934 0.0015 0.0263  0.5751 1184.9404  79.5667
3       MIMIC 0.8312 0.0110 0.0362  0.5627 7403.3030 358.2333
4        FEDA 0.8080 0.0019 0.0473  0.5963 3208.9304 200.3000

---------------------------------------------------------------
Pareto Reference has 278 points
Maximum UNFR possible is 10/278=0.0360
```



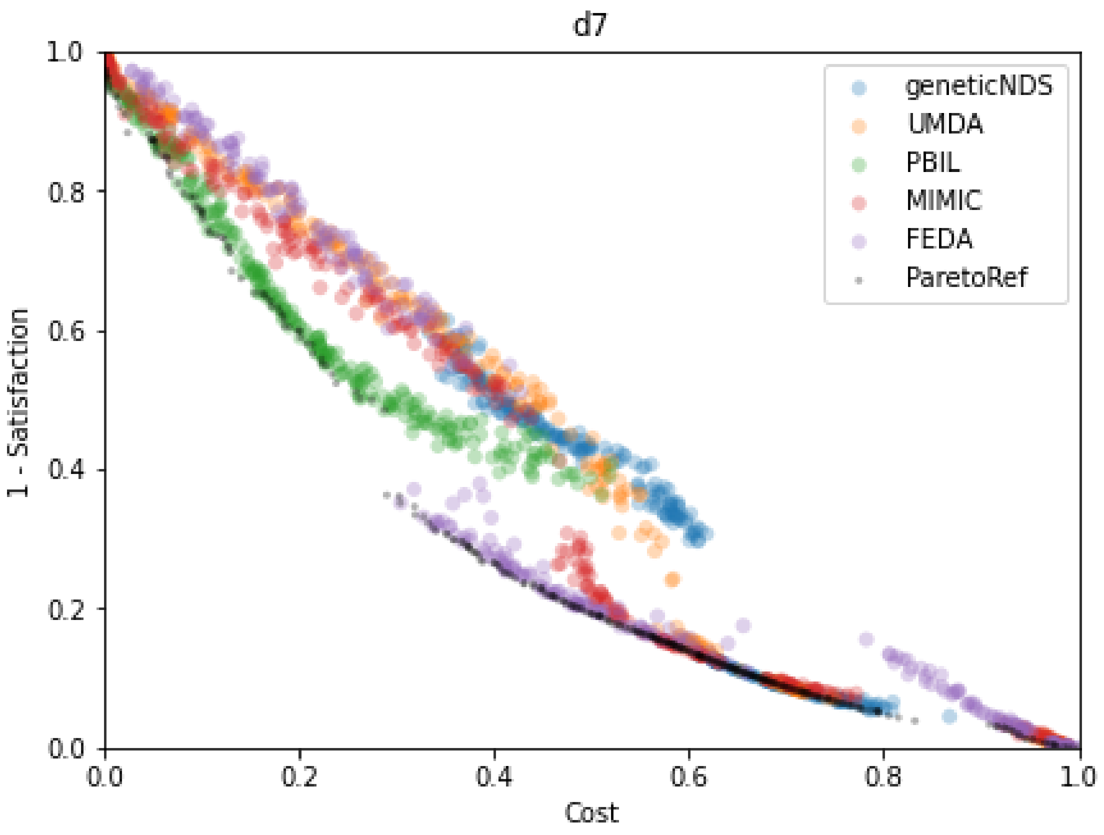
```
       Method     HV   UNFR    gd+  spread   time(s)     |NDS|
0  geneticNDS 0.7019 0.0002 0.0723  0.7259 1916.8350  94.4667
1        UMDA 0.7842 0.0031 0.0669  0.5800  328.0819 122.6667
2        PBIL 0.7197 0.0026 0.0307  0.5789  885.8085  67.7000
3       MIMIC 0.8227 0.0070 0.0447  0.5614 5207.5402 252.0333
4        FEDA 0.8090 0.0030 0.0493  0.5953 2356.9579 149.2667

---------------------------------------------------------------
Pareto Reference has 275 points
Maximum UNFR possible is 10/275=0.0364
```



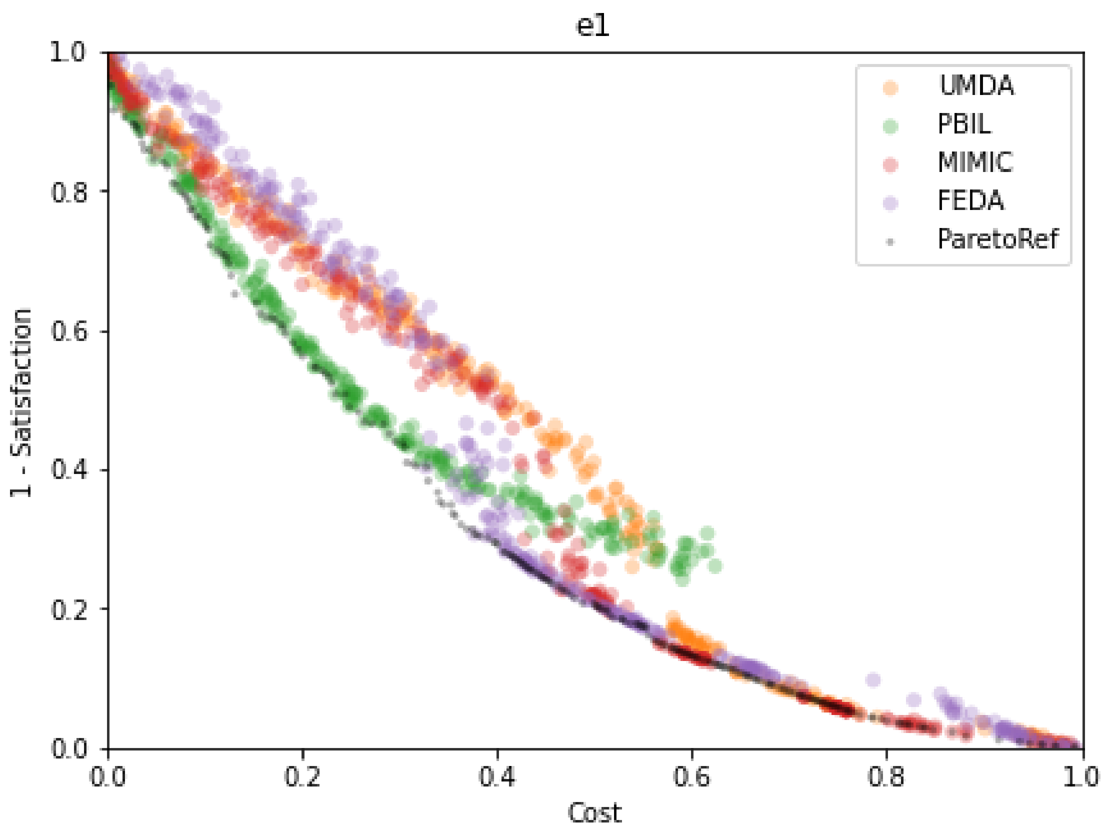
```
       Method     HV   UNFR    gd+  spread   time(s)    |NDS|
0  geneticNDS 0.6686 0.0030 0.0786  0.7611 1568.3957  78.3333
1        UMDA 0.7632 0.0015 0.0691  0.5703  359.8537 124.0667
2        PBIL 0.6441 0.0029 0.0327  0.5626 1200.1904  70.9667
3       MIMIC 0.7926 0.0004 0.0445  0.5459 3944.9460 103.8667
4        FEDA 0.7980 0.0025 0.0429  0.6004 2437.1066 166.1000

---------------------------------------------------------------
Pareto Reference has 336 points
Maximum UNFR possible is 10/336=0.0298
```



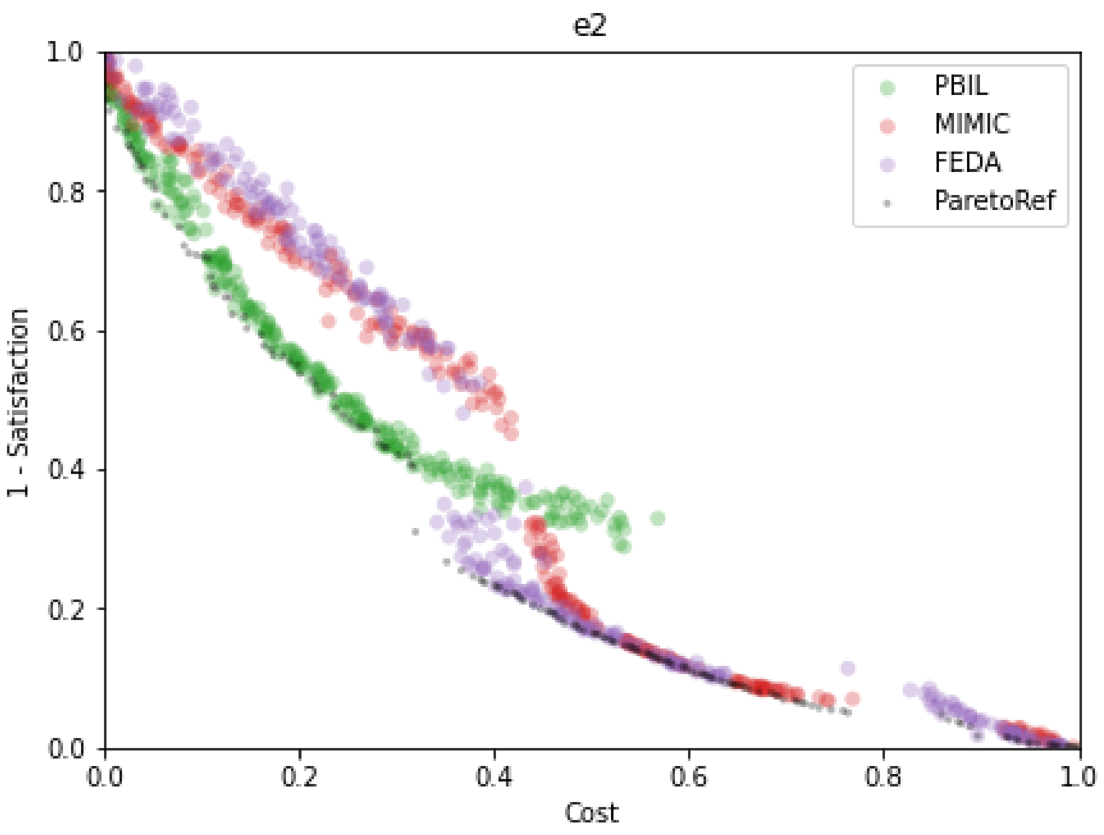
```
       Method      HV    UNFR     gd+  spread   time(s)     |NDS|
0  geneticNDS 0.6545 0.0061 0.0897  0.7401 2144.0206 104.4333
1        UMDA 0.7503 0.0012 0.0658  0.5694  369.0416 148.0333
2        PBIL 0.6451 0.0022 0.0324  0.5723 1221.0485  80.3000
3       MIMIC 0.7832 0.0016 0.0393  0.5483 4608.8875 127.4433
4        FEDA 0.7791 0.0018 0.0432  0.6063 3178.8110 192.3667

---------------------------------------------------------------
../output/geneticnds/geneticNDSTruee15101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
Pareto Reference has 242 points
Maximum UNFR possible is 10/242=0.0413
```



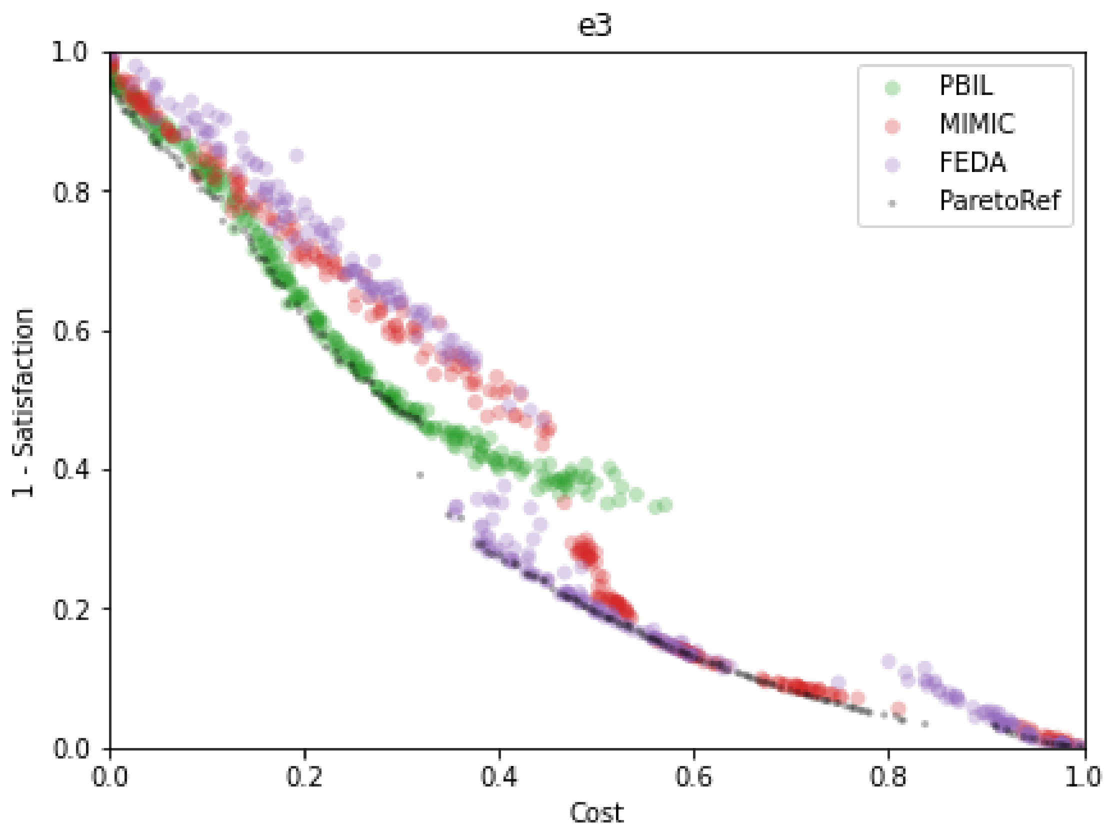
```
  Method      HV    UNFR     gd+  spread   time(s)     |NDS|
0   UMDA 0.7700 0.0025 0.0601  0.5679  333.5748 107.4000
1   PBIL 0.7345 0.0023 0.0306  0.5764 1103.2115  84.3000
2  MIMIC 0.7999 0.0091 0.0385  0.5808 5132.9078 184.7333
3   FEDA 0.7882 0.0019 0.0432  0.6007 2085.4413 127.7000
```

```
-----------------------------------------------------------------
../output/geneticnds/geneticNDSTruee25101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee25101000500ndselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
Pareto Reference has 168 points
Maximum UNFR possible is 10/168=0.0595
```

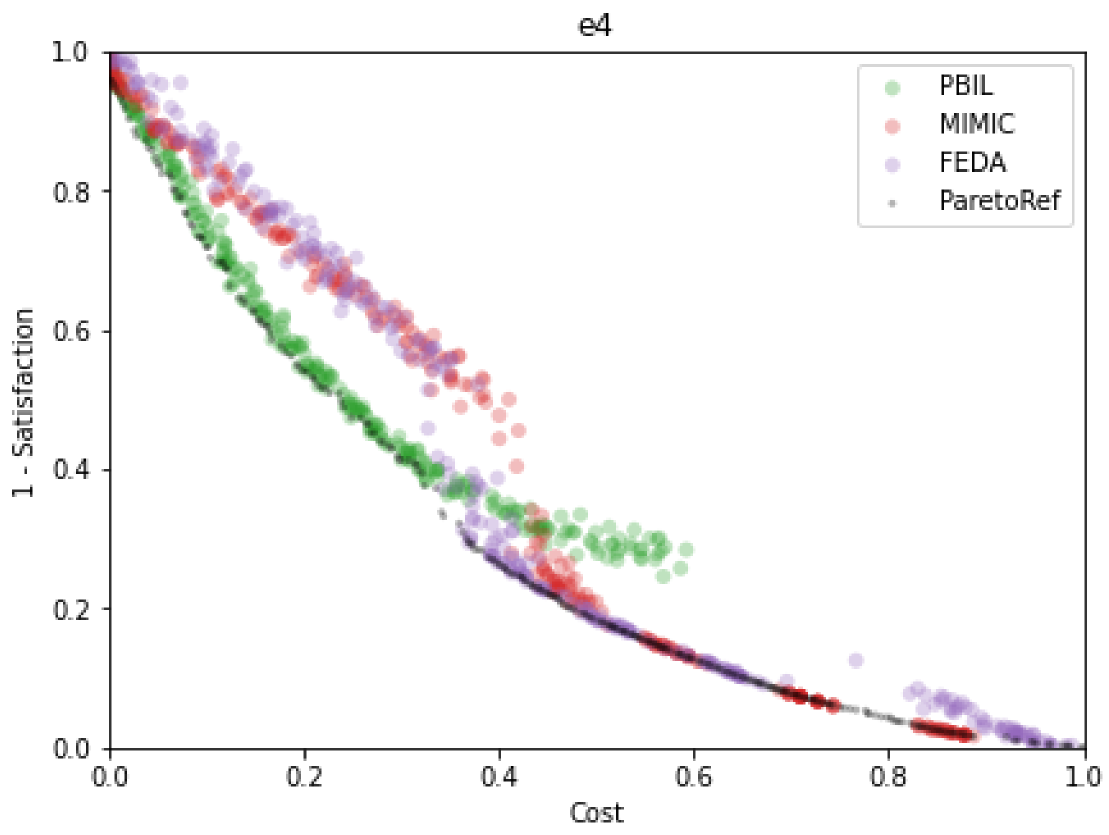


```
  Method     HV   UNFR    gd+  spread   time(s)    |NDS|
0   PBIL 0.7186 0.0036 0.0315  0.5760  988.6748  68.6000
1  MIMIC 0.8054 0.0010 0.0477  0.5462 4315.7328  94.6333
2   FEDA 0.8076 0.0036 0.0480  0.5869 1973.8371 102.5333
```

```
----------------------------------------------------------------
../output/geneticnds/geneticNDSTruee35101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee35101000500ndselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
Pareto Reference has 200 points
Maximum UNFR possible is 10/200=0.0500
```

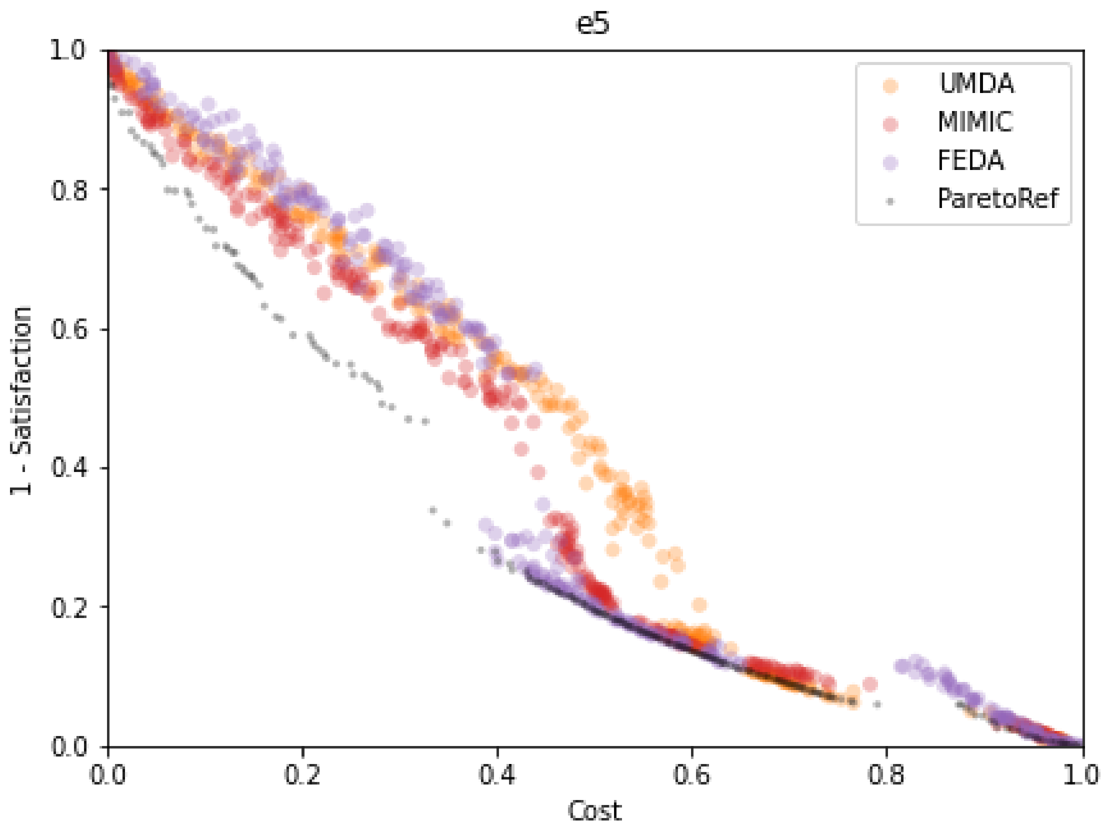


```
  Method      HV   UNFR    gd+  spread   time(s)     |NDS|
0   PBIL  0.6622 0.0030 0.0313  0.5878 1230.0837   77.4333
1  MIMIC  0.7852 0.0020 0.0337  0.5509 4410.1522  101.5000
2   FEDA  0.7781 0.0070 0.0356  0.5904 1829.5486   97.0000
--------------------------------------------------------------
../output/geneticnds/geneticNDSTruee45101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee45101000500ndselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
Pareto Reference has 286 points
Maximum UNFR possible is 10/286=0.0350
```



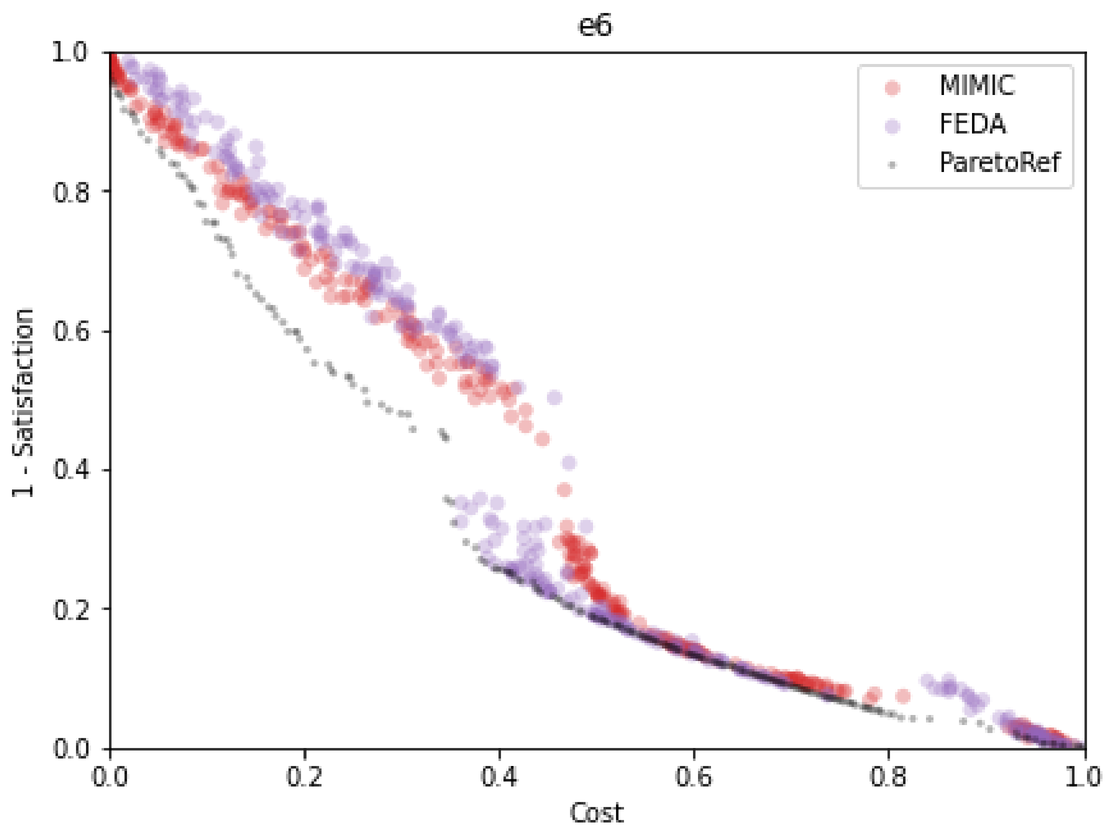
```
  Method     HV   UNFR    gd+  spread   time(s)    |NDS|
0   PBIL 0.7460 0.0023 0.0263  0.5803 1089.7584  82.8667
1  MIMIC 0.8104 0.0077 0.0403  0.5424 5929.8505 225.5667
2   FEDA 0.8080 0.0041 0.0390  0.6149 2404.1686 132.8667
```

```
--------------------------------------------------------------
../output/geneticnds/geneticNDSTruee55101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/pbil/pbilTruee5510100040000.10.10.130.json tried to be used due to
PBIL best configuration in this dataset, but file is not available yet
Pareto Reference has 201 points
Maximum UNFR possible is 10/201=0.0498
```



```
  Method     HV   UNFR    gd+  spread   time(s)   |NDS|
0   UMDA 0.7458 0.0022 0.0653  0.5659  288.6172 94.0667
1  MIMIC 0.7828 0.0003 0.0409  0.5765 4131.0095 92.6333
2   FEDA 0.7595 0.0032 0.0510  0.5889 1725.0309 87.7000

-------------------------------------------------------------
../output/geneticnds/geneticNDSTruee65101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee6510100050Ondselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
../output/pbil/pbilTruee6510100040000.10.10.130.json tried to be used due to
PBIL best configuration in this dataset, but file is not available yet
Pareto Reference has 215 points
Maximum UNFR possible is 10/215=0.0465
```



```
  Method     HV   UNFR    gd+  spread   time(s)    |NDS|
0  MIMIC 0.7854 0.0003 0.0382  0.5294 3892.7625  99.8667
1   FEDA 0.7814 0.0045 0.0456  0.5876 2061.4621 105.0333

---------------------------------------------------------------
Wins Counts:
{'geneticNDS': 0, 'UMDA': 0, 'PBIL': 2, 'MIMIC': 19, 'FEDA': 2}
Wins in datasets:
{'geneticNDS': [], 'UMDA': [], 'PBIL': ['a2', 'c2'], 'MIMIC': ['p1', 'p2', 'a1',
'a3', 'a4', 'c1', 'c3', 'c4', 'd1', 'd2', 'd3', 'd4', 'd5', 'd7', 'e1', 'e3',
'e4', 'e5', 'e6'], 'FEDA': ['d6', 'e2']}
```

Given the results, we see that PBIL behaves really well in datasets wich not a large number of requirements (aX and cX datasets). In the case dX datasets, with hundreds of requirements, FEDA obtains greater Hypervolumes than PBIL and the rest of algorithms. In some cases, FEDA obtains a very similar HV value than PBIL or UMDA; in such cases, in order to be sure that FEDA performs better we can take into account the UNFR value, which is pareto compliant, and when FEDA's HV is just slightly better, UNFR is clearly better than the other algorithm. \

A withdraw FEDA presents is that its execution time is much worse than the other algorithms. This is mostly due to the large number of Non Dominated Solutions it finds. \

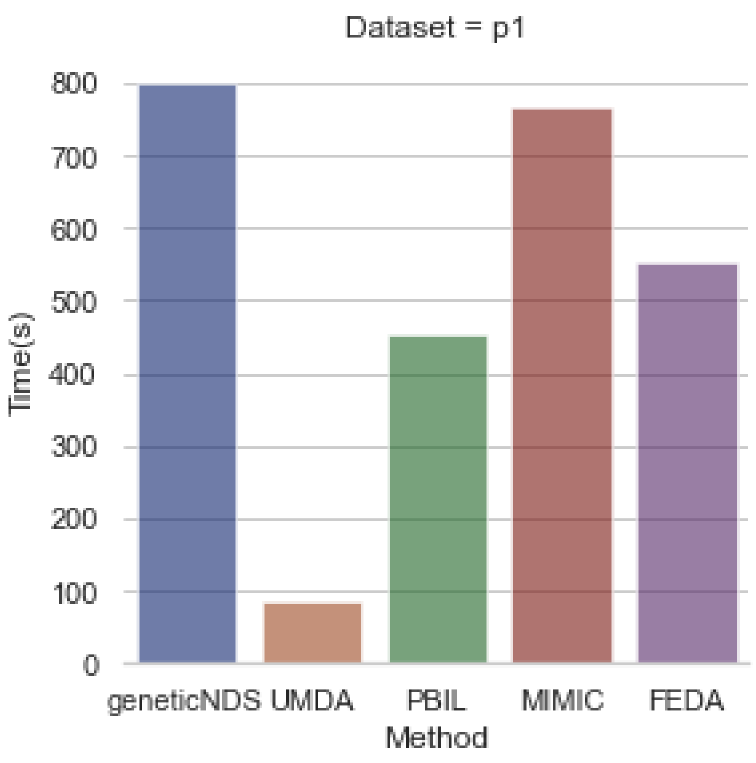
Respect to gd+, FEDA is usually the second algorithm with best (lowest) mean general distance to the Pareto Reference, while PBIL commonly finds the closer solutions to the PR, although its HV covered is lower, as said, when the project presents hundreds of requirements.

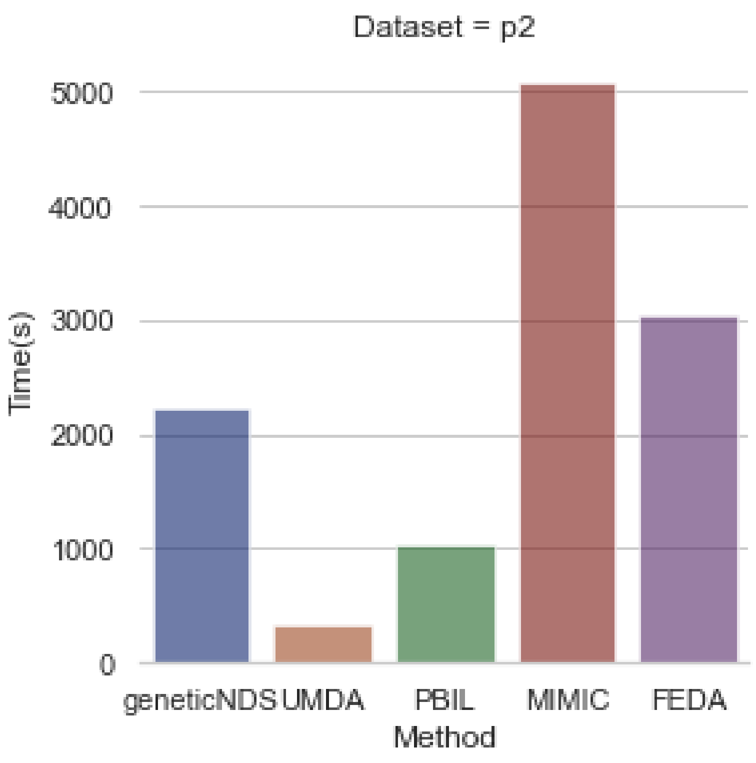
4.3 Execution times

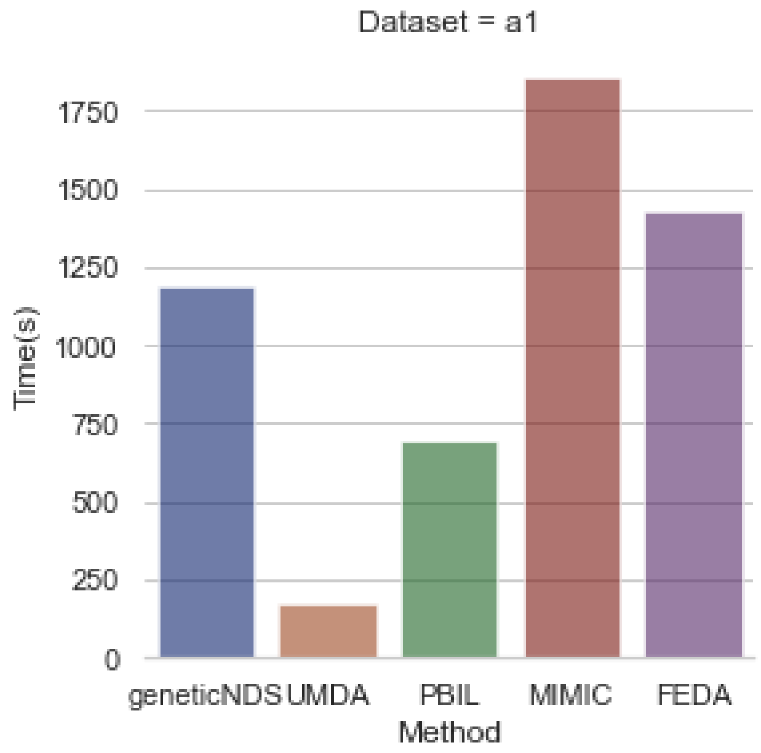
```
../output/geneticnds/geneticNDSTruee15101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/geneticnds/geneticNDSTruee25101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee2510100050Ondselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
../output/geneticnds/geneticNDSTruee35101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee3510100050Ondselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
../output/geneticnds/geneticNDSTruee45101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee4510100050Ondselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
```

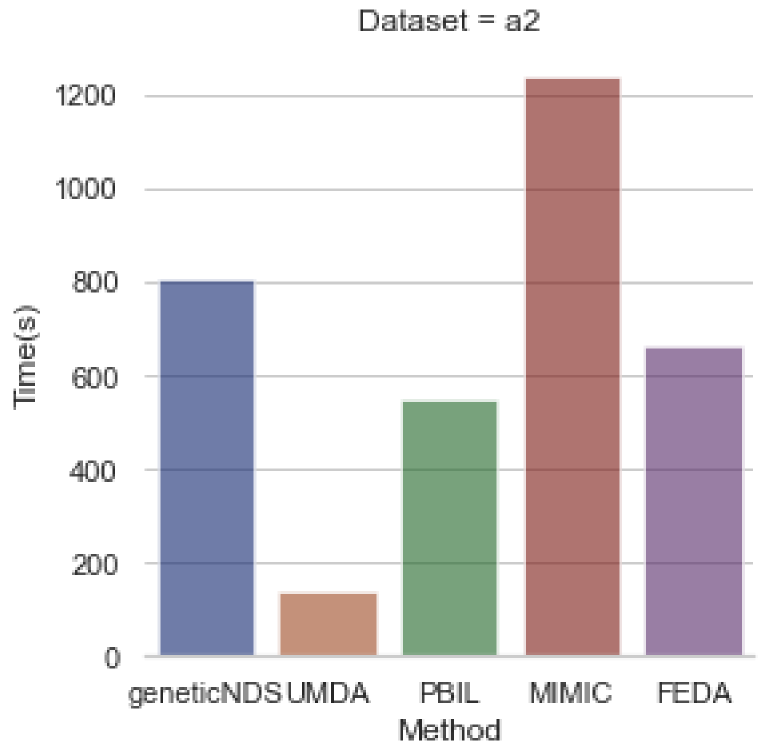
```
c:\Users\Pablo.Bermejo\AppData\Local\Programs\Python\Python39\lib\site-
packages\seaborn\axisgrid.py:392: RuntimeWarning: More than 20 figures have been
opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may
consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`).
  fig, axes = plt.subplots(nrow, ncol, **kwargs)
```

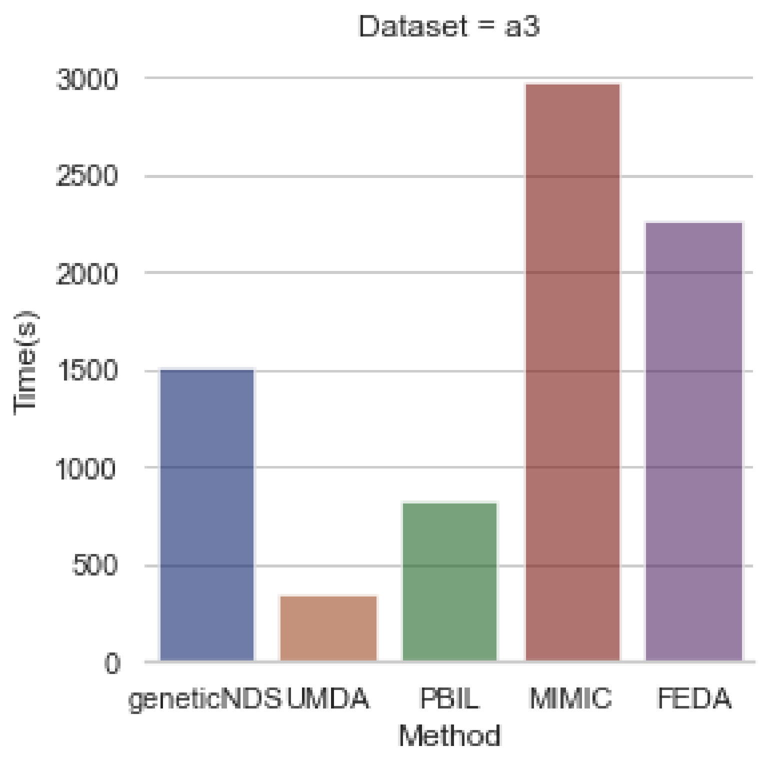
```
../output/geneticnds/geneticNDSTruee55101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/pbil/pbilTruee5510100040000.10.10.130.json tried to be used due to
PBIL best configuration in this dataset, but file is not available yet
../output/geneticnds/geneticNDSTruee65101000400020.80.3tournamentonepointflip1bi
telitismnds30.json tried to be used due to geneticNDS best configuration in this
dataset, but file is not available yet
../output/umda/umdaTruee6510100050Ondselitism30.json tried to be used due to
UMDA best configuration in this dataset, but file is not available yet
../output/pbil/pbilTruee6510100040000.10.10.130.json tried to be used due to
PBIL best configuration in this dataset, but file is not available yet
```

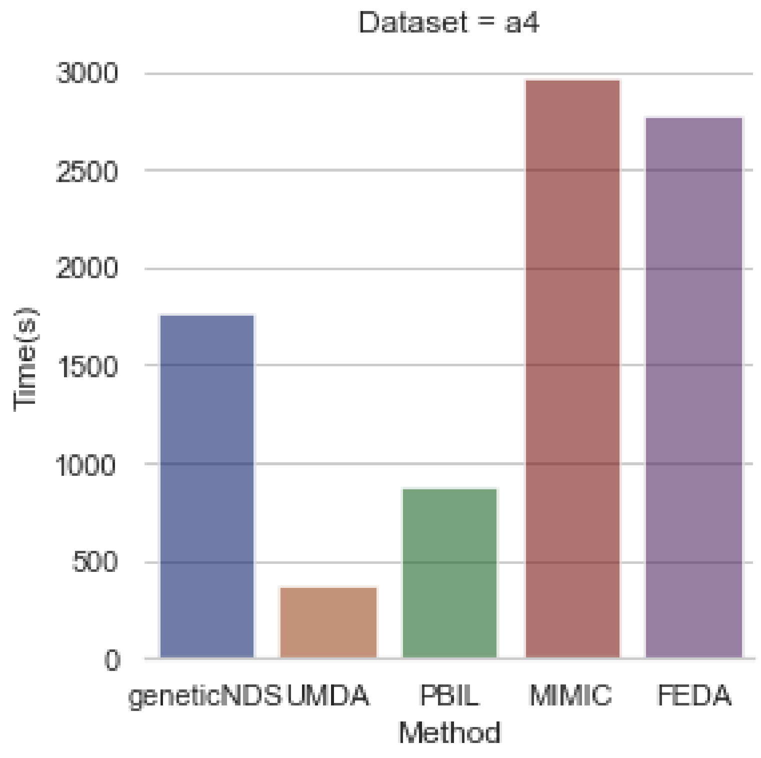

Dataset = p1
800
700
600
500
400
300
200
100
0
Time(s)
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

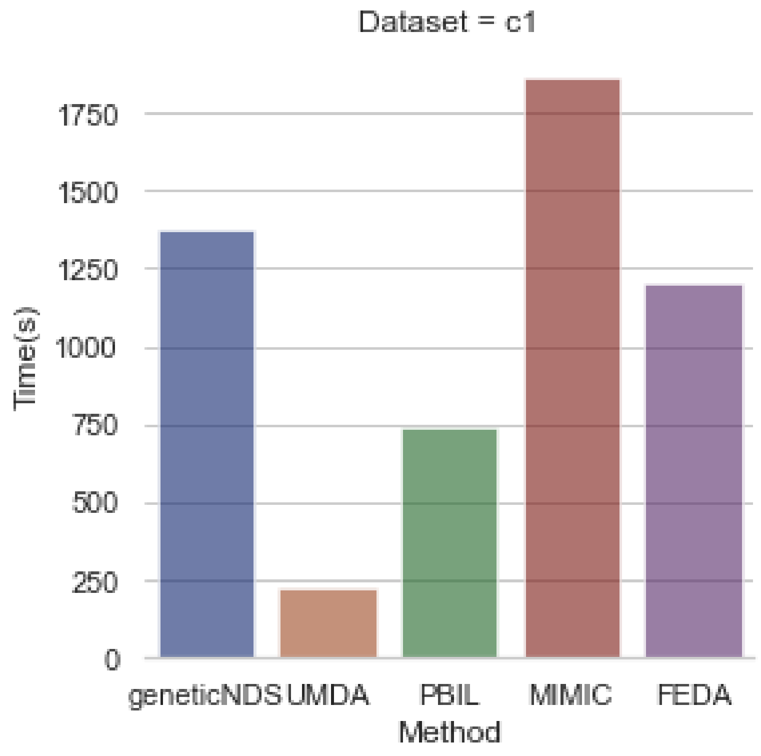

Dataset = p2
5000
4000
3000
2000
1000
0
Time(s)
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

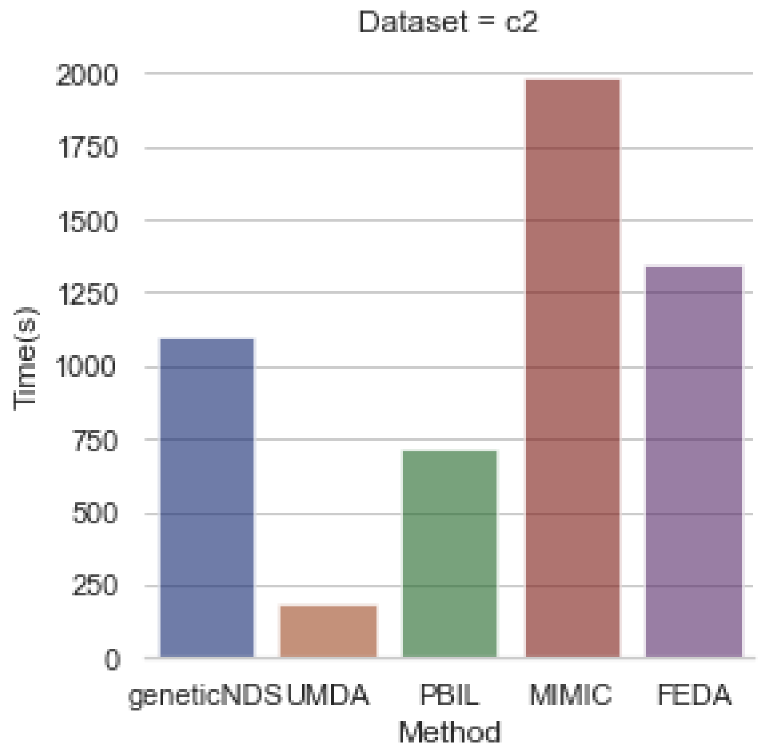

Dataset = a1
Time(s)
0
250
500
750
1000
1250
1500
1750
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

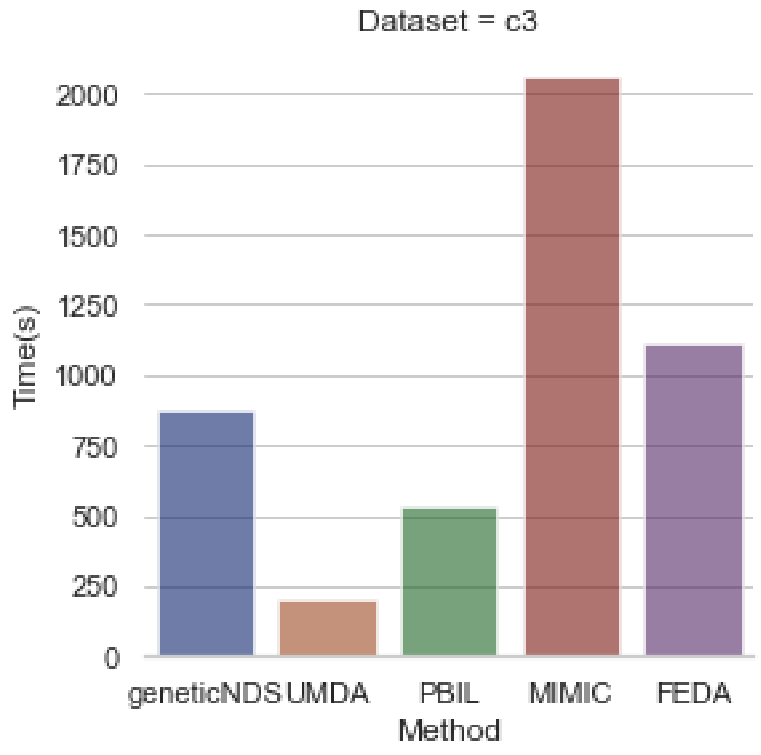

Dataset = a2
Time(s)
0
200
400
600
800
1000
1200
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

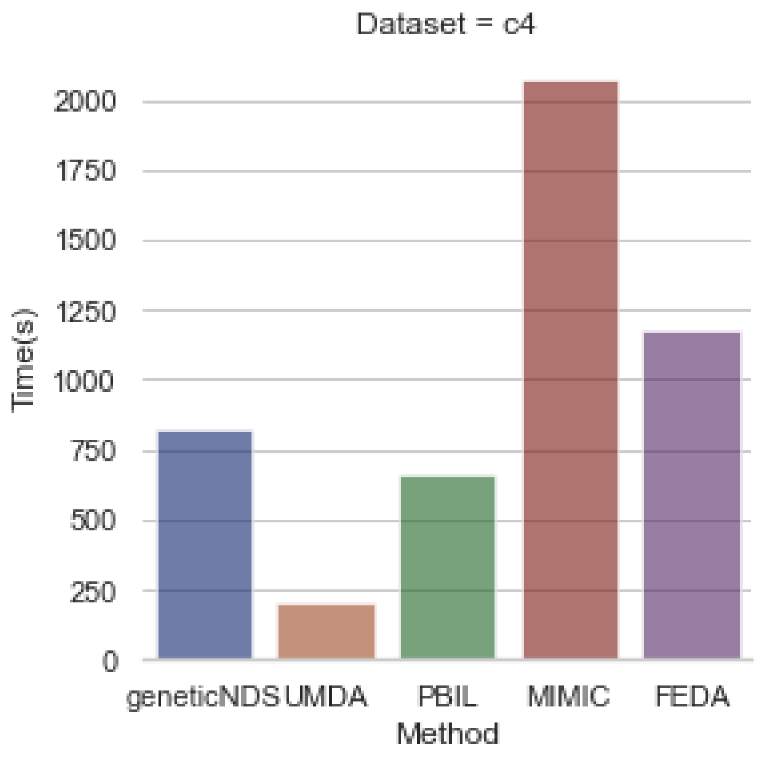

Dataset = a3
Time(s)
3000
2500
2000
1500
1000
500
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

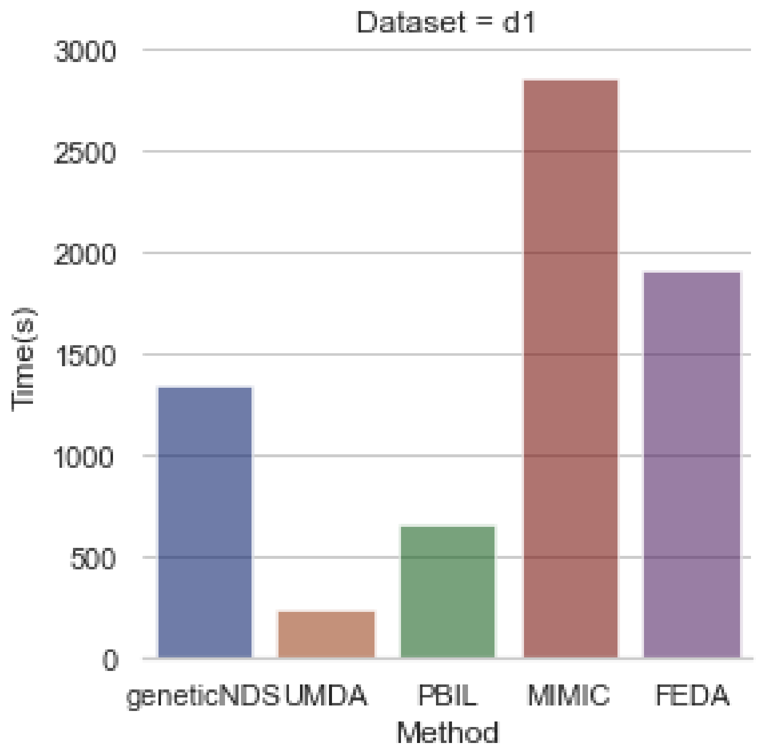

Dataset = a4
Time(s)
3000
2500
2000
1500
1000
500
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method


Dataset = c1
Time(s)
1750
1500
1250
1000
750
500
250
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

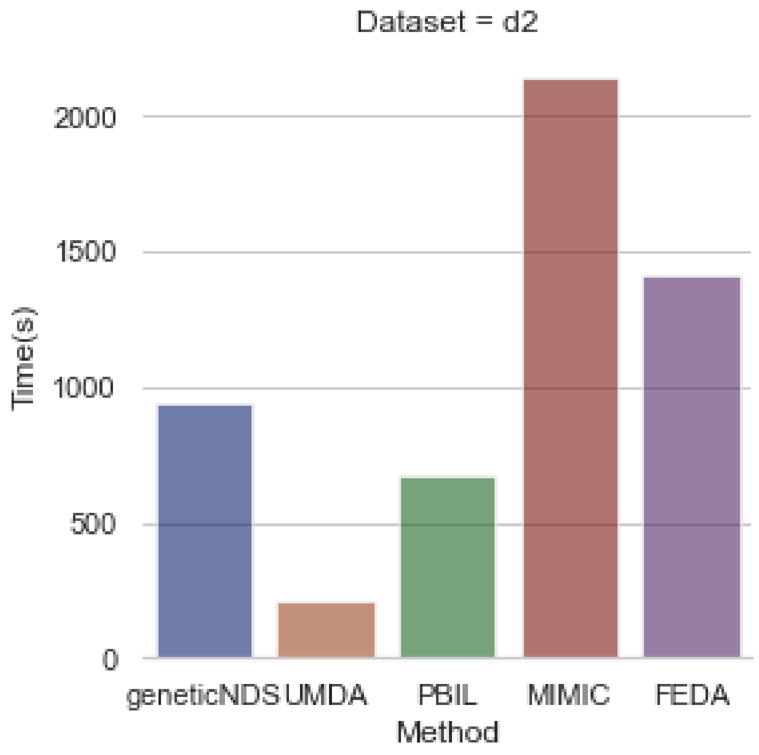

Dataset = c2
2000
1750
1500
1250
1000
750
500
250
0
Time(s)
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method


Dataset = c3
2000
1750
1500
1250
1000
750
500
250
0
Time(s)
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

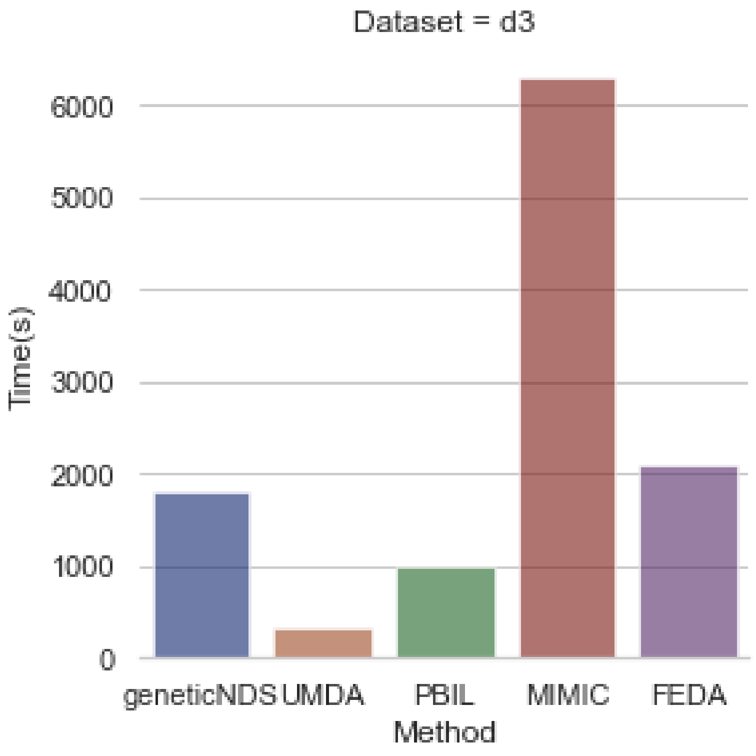

Dataset = c4
Time(s)
2000
1750
1500
1250
1000
750
500
250
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method



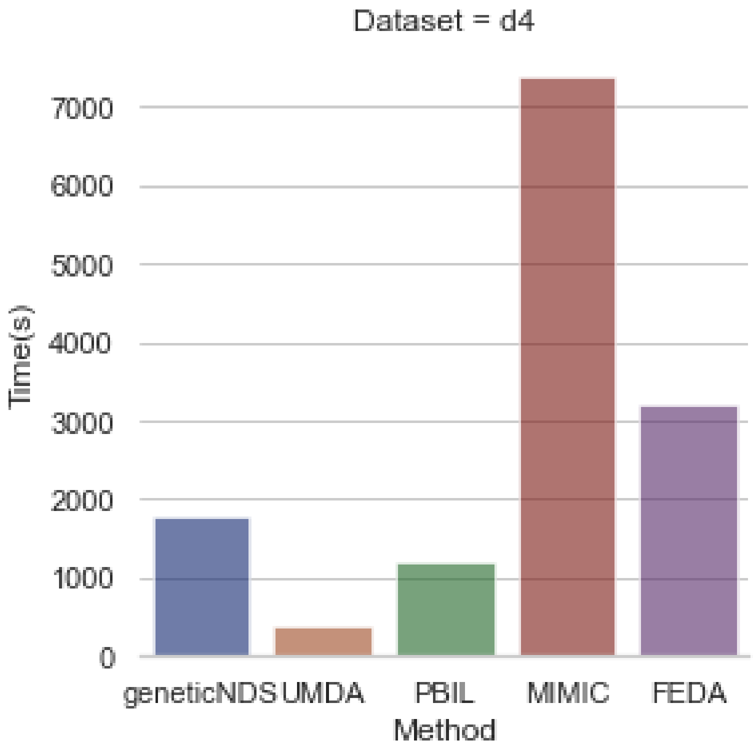
Dataset = d1
Time(s)
3000
2500
2000
1500
1000
500
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method



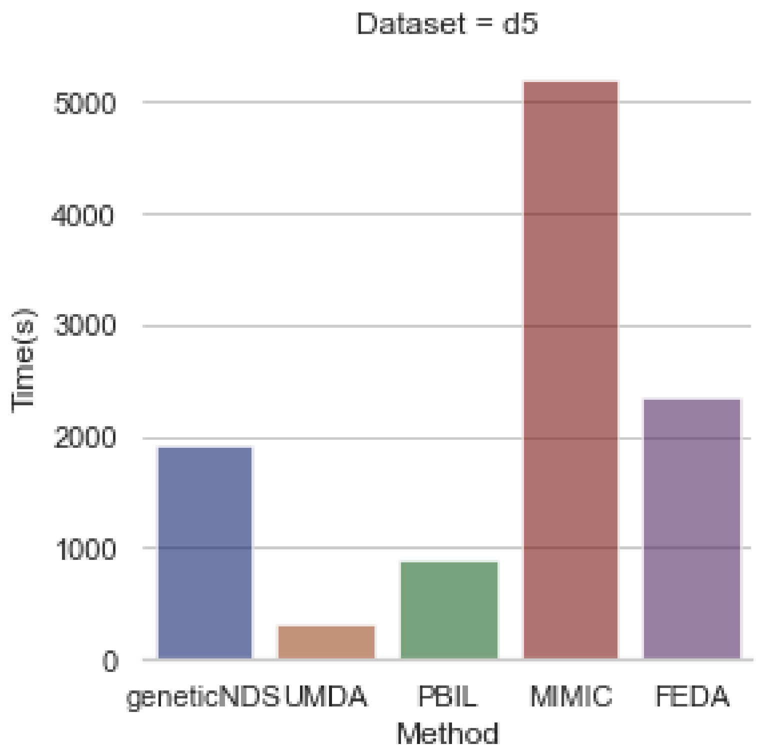
Dataset = d2
Time(s)
2000
1500
1000
500
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

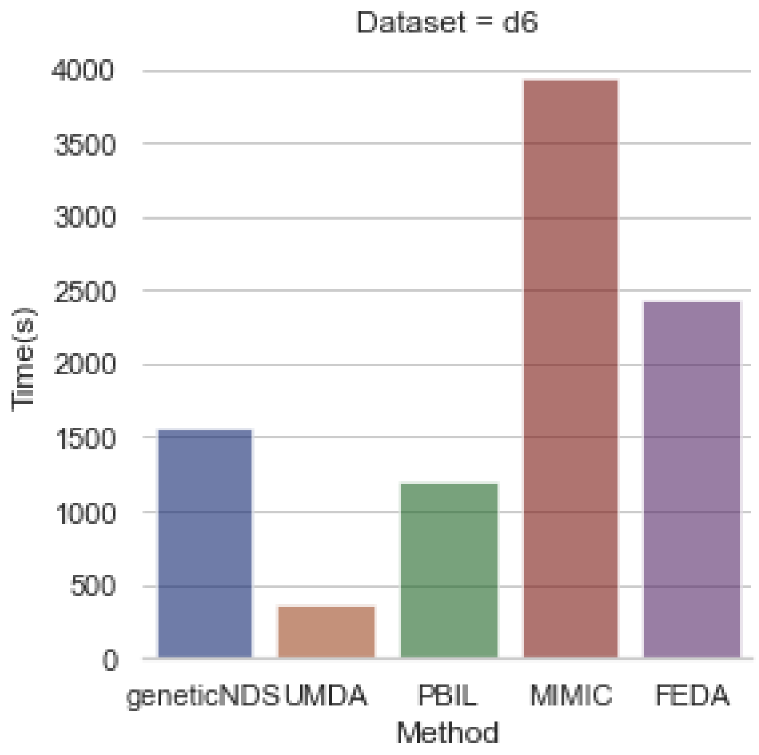


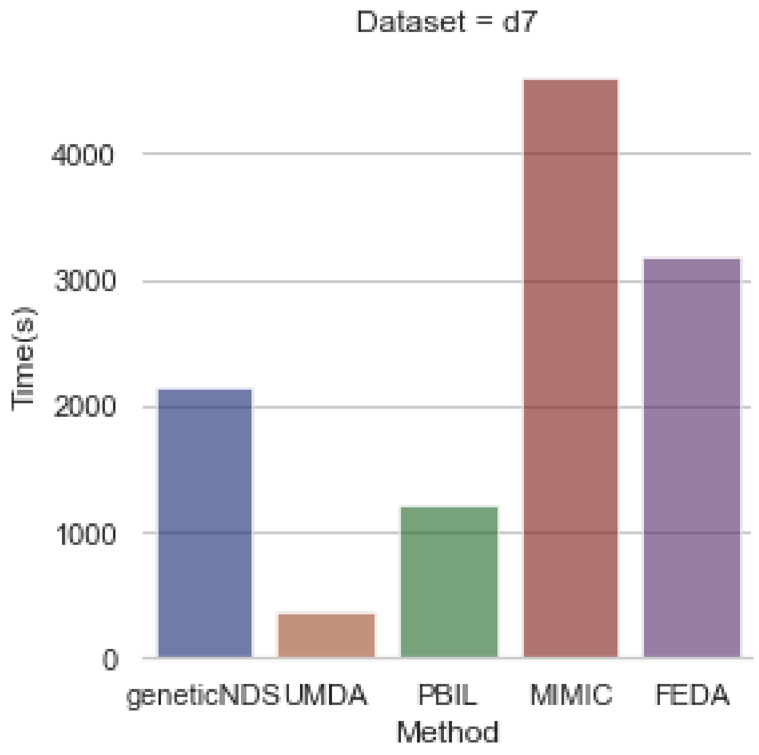
Dataset = d3
Time(s)
0
1000
2000
3000
4000
5000
6000
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

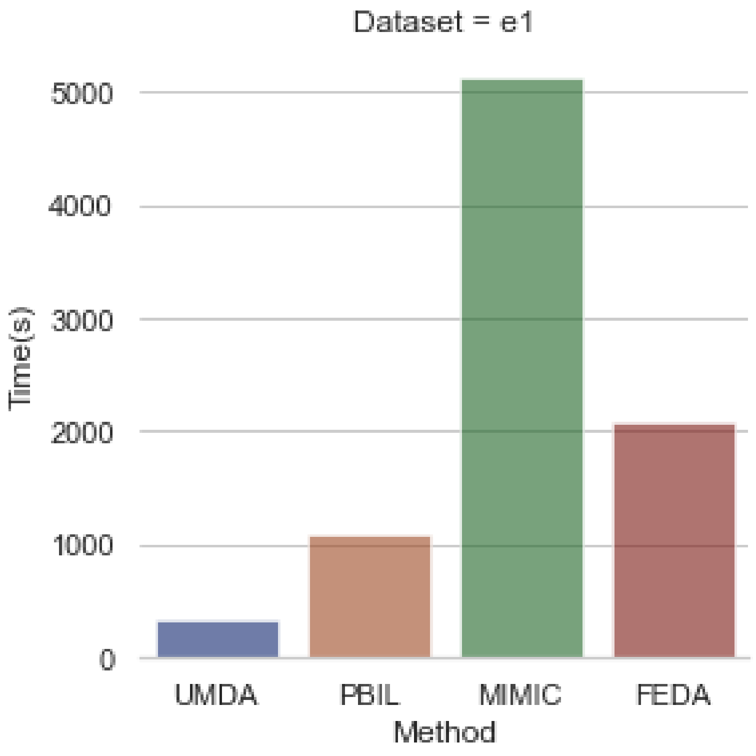


Dataset = d4
Time(s)
0
1000
2000
3000
4000
5000
6000
7000
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

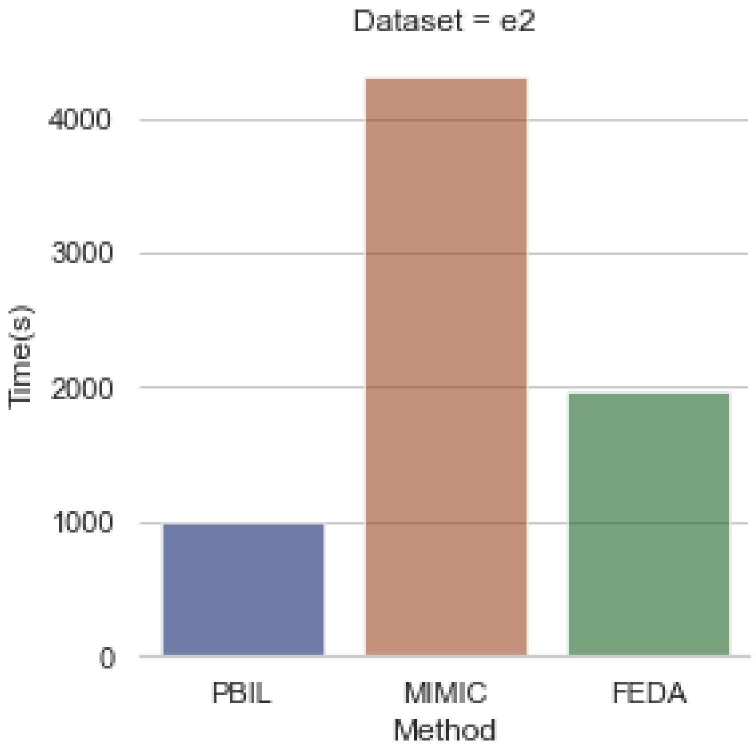

Dataset = d5
Time(s)
0
1000
2000
3000
4000
5000
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method


Dataset = d6
Time(s)
4000
3500
3000
2500
2000
1500
1000
500
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

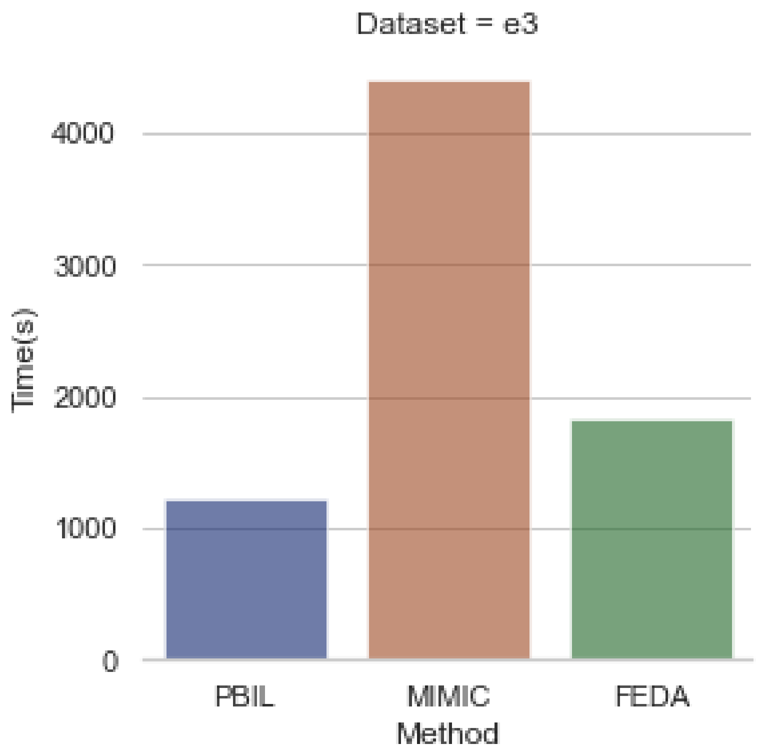

Dataset = d7
Time(s)
4000
3000
2000
1000
0
geneticNDS
UMDA
PBIL
MIMIC
FEDA
Method

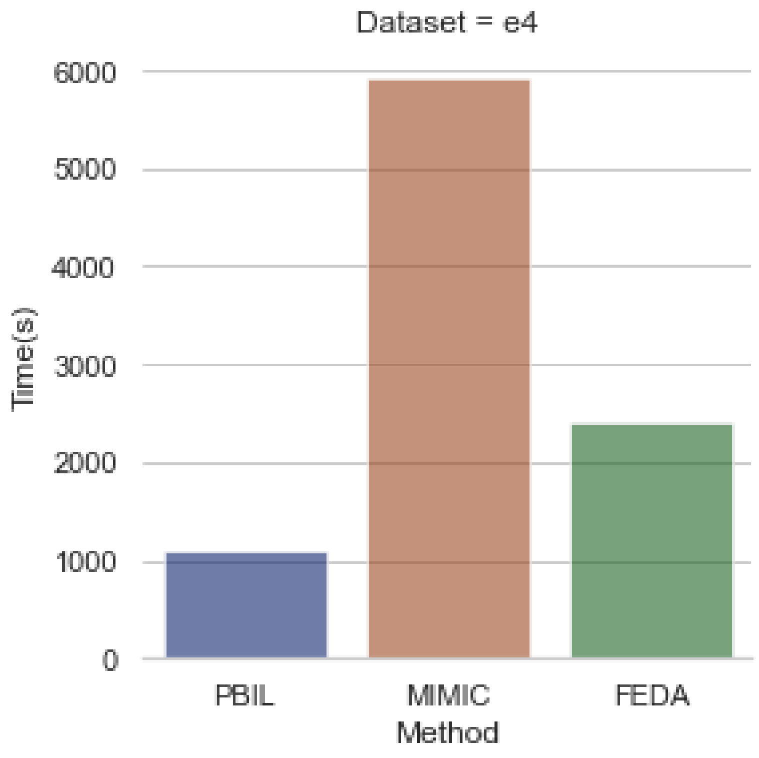


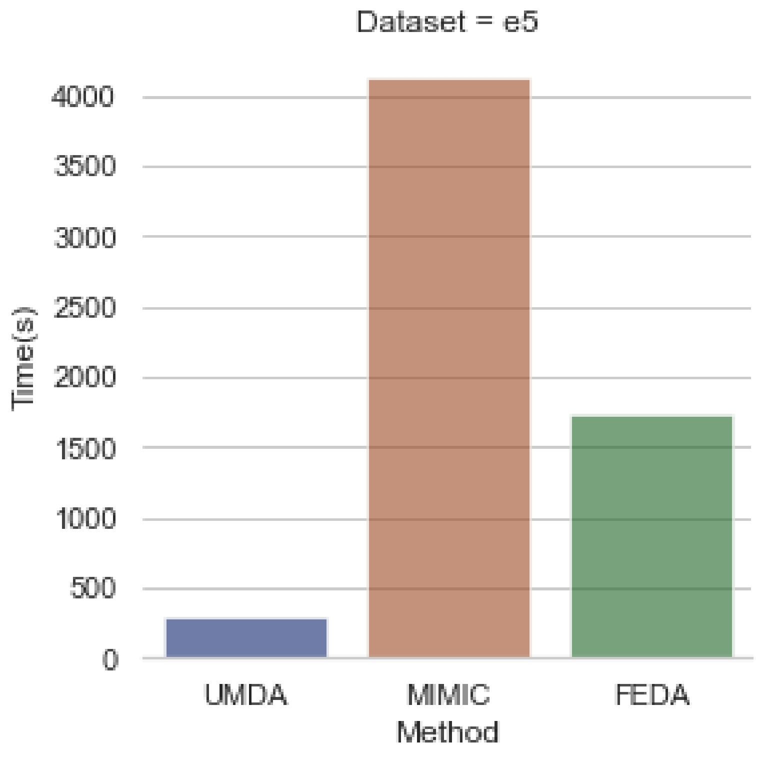
Dataset = e1
Time(s)
0
1000
2000
3000
4000
5000
UMDA
PBIL
MIMIC
FEDA
Method

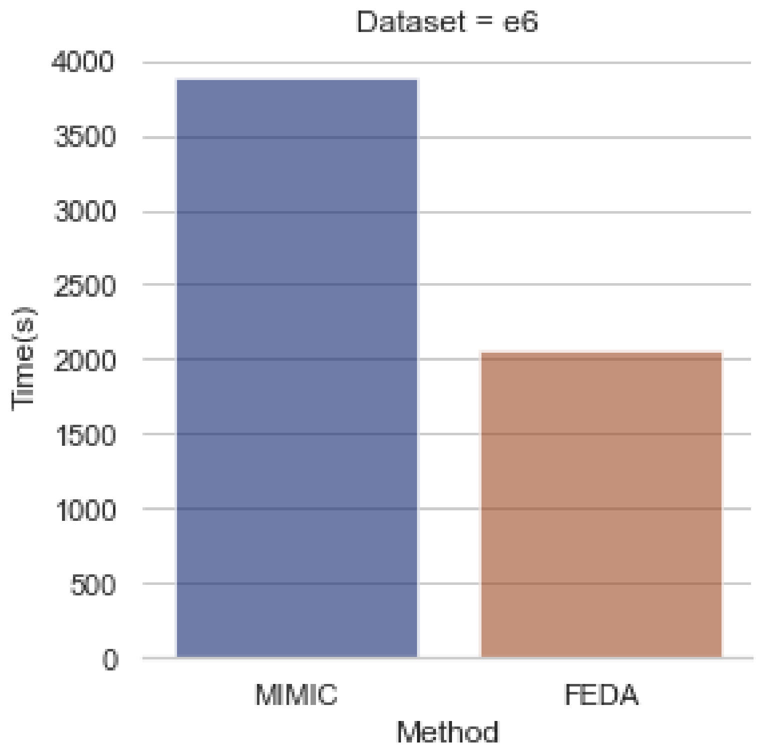


Dataset = e2
Time(s)
0
1000
2000
3000
4000
PBIL
MIMIC
FEDA
Method


Dataset = e3
Time(s)
4000
3000
2000
1000
0
PBIL
MIMIC
FEDA
Method


Dataset = e4
6000
5000
4000
3000
2000
1000
0
Time(s)
PBIL
MIMIC
FEDA
Method


Dataset = e5
4000
3500
3000
2500
2000
1500
1000
500
0
Time(s)
UMDA
MIMIC
FEDA
Method



0.4.1 5. Statistical tests of quality indicators

Antes de meterme en esto, a ver si veis alguna laguna en la experimentación. Por ejemplo me preocupa: - Ausencia de nsga-ii - qué hacer con GRASP - tiempos tan grandes por |NDS| y que ensombrece el tiempo real de learning+sampling - quitar algún dataset dX