Atefe Rajabi 40230563

Libreries

Moduls

Functionality

1. DateProcessor
2. Recombination
3. Chromosome
4. NSGAII
5. Fast-NSGAII
6. PlotProcessor

Results

1. DS02
2. DS04
3. DS05
4. DS07
5. DS08
6. DS10

Overall Result

1. Wise initialization
2. Effect of columns number expansion
3. Effect of samples number expansion
4. Effect of fast-non-dominated sorting parallelization
5. Effect of objective evaluation parallelization
6. ….

Libraries Used:

- scikit-learn: Utilized for machine learning tasks, including the implementation of K-nearest neighbors and Random Forest classifiers.

- pymoo: A Python library for multi-objective optimization algorithms. It provides tools for handling optimization problems with multiple objectives.

- concurrent.futures: Used for parallel execution of tasks to enhance performance.

- matplotlib: Employed for data visualization, particularly for creating scatter plots and line plots.

- pandas: Used for data manipulation and handling datasets.

- numpy: A fundamental library for numerical operations in Python, used extensively for array manipulations.

- json: Used for handling JSON configuration files.

- docx: A library for creating and modifying Microsoft Word (.docx) files.

Modular Organization:

1. NSGAConfig: This module handles the loading of configuration parameters from a JSON file. It provides methods to retrieve specific parameters for a given dataset.

2. Recombination: Defines different crossover methods for recombining genetic information between parent chromosomes.

3. Chromosome: Represents an individual solution in the population. It includes methods for dominance checking and crowding distance calculation.

4. DataProcessor: Handles loading and processing of input datasets. It provides methods to obtain the number of features and labels from a dataset.

5. NSGAII: The main class implementing the NSGA-II algorithm. It includes methods for initialization, crossover, mutation, dominance comparison, and environmental selection. The algorithm's main loop, as well as fitness functions and termination criteria, are defined here.

6. Main Execution Loop: The script includes a loop to run the NSGA-II algorithm on multiple datasets. The results, such as Pareto fronts, IGD values, and hypervolume values, are visualized and summarized.

Overall Functionality:

1. Initialization: The algorithm initializes a population of solutions randomly or based on specific criteria. Four methods have been implemented: random initialization, wise initialization based on feature importance, wise initialization based on the maximum number of features in the true Pareto front for each dataset, inspired by the plots in the provided paper, and randomly diverse initialization.

2. Crossover and Mutation: Parent solutions are selected, and crossover is applied with a certain probability. The process of selecting crossover is adaptive, as mentioned in the provided paper. Mutation is also applied with a predefined probability.

3. Fitness Calculation: Fitness functions are applied to evaluate the performance of solutions based on classification error and solution size. Additionally, this process is performed in parallel; after the generation of offspring, all of them are evaluated simultaneously since they are independent.

4. Environmental Selection: The NSGA-II environmental selection mechanism is applied to identify non-dominated solutions and maintain diversity in the population. Additionally, the first part of non-dominated sorting is performed in parallel, as checking chromosome domination against each other can be executed simultaneously.

5. Convergence Criteria: The algorithm monitors changes in hypervolume and IGD values. If no improvement is observed for a certain number of generations, the algorithm terminates.

The `NSGAII` class implements the NSGA-II algorithm, and it includes several methods to handle different aspects of the algorithm. Here are the main functions implemented in the `NSGAII` class:

1. __init__:

- Initializes the NSGA-II algorithm with parameters such as population size, maximum number of generations, crossover and mutation probabilities, and other configuration details.

2. initialize_population:

Random_initialize_population
Wise_random_initialize_population
Wise_initialize_population
Wise_initialize_population2

- The population is initialized, as mentioned before, using four different methods: random initialization, wise initialization – which involves three distinct approaches. One is based on important features selected using the random forest algorithm, and the other is based on the Pareto front of other algorithms, considered the true Pareto front, and the last one is diverse generating population based on features presence.

Wise_random_initialize_population:

Controlled Number of Active Features: Instead of randomly generating the entire chromosome, the number of active features (num_active_features) is controlled. This ensures that each chromosome starts with a different number of features, promoting diversity.

Ensuring Representation of All Features: After the initial population is created, the function checks to ensure that every feature is represented at least once across all chromosomes. If any feature is missing, it is added to a randomly selected chromosome. This step ensures that no feature is completely left out in the initial population, giving each feature a chance to prove its worth in the evolutionary process.

3. create_offspring:

- Generates offspring through crossover and mutation operations. It selects parent individuals based on their ranks and crowding distances, applies crossover and mutation, and adds the resulting offspring to the next generation.

4. evaluate_population:

Fitness_function_1
Fitness_function_2

- Since the problem is multi-objective, there are two fitness functions: the first one calculates the classification error using KNN, while the other measures the size of the feature space.

5. fast_non_dominated_sort:

- Performs fast non-dominated sorting of the population based on dominance relationships, categorizing solutions into different fronts according to their dominance levels. Initially, we calculate the domination relationships for each chromosome, forming a list that indicates which other chromosomes is dominated by it. Subsequently, we create the first front (front_0) comprising chromosomes that are not dominated by any other. To identify subsequent fronts, we use the domination information from the previous front, ensuring that each new front consists of solutions dominated only by the chromosomes in the preceding front.

6. crowding_distance_assignment:

- The calculation involves sorting the solutions in the front based on each objective individually. The boundary solutions (first and last) are assigned a crowding distance of infinity to ensure they are always selected for the next generation. For the interior solutions, the crowding distance is incremented by the difference in the objective values of adjacent solutions along each objective axis. The sum of crowding distances across all objectives is used to measure the overall density of solutions. Solutions with larger crowding distances are preferred because they are in less crowded regions, indicating a more diverse and spread-out set of solutions.

7. environmental_selection:

- Individuals for the next generation are selected based on non-dominated sorting and crowding distance. This method combines the current population and offspring, performs non-dominated sorting, and selects individuals for the next generation. Additionally, for the last front that is not completely fit in the population, crowding distance is calculated, and a larger distance is preferable.

8. get_hypervolume:

- Calculates the hypervolume indicator for a set of points in the objective space. Hypervolume is a metric used to evaluate the quality of Pareto fronts.

9. get_IGD:

- Calculates the inverted generational distance (IGD) metric for a set of points in the objective space. IGD measures the convergence of a Pareto front to the true Pareto front.

10. nsga2:

- The NSGA-II algorithm is executed, involving the initialization of the population, evaluation of the initial population, and iterative creation of offspring. Individuals for the next generation are selected until the maximum number of generations is reached. Additionally, a termination condition is based on a lack of improvement situation, configured differently for each dataset.

11. init_OSP:

- This method initializes the operator success probabilities (OSP) for each crossover operator used in NSGA-II. It sets equal probabilities for all crossover operators initially, ensuring a fair starting point for exploration.

12. update_OSP:

- The method adjusts the operator success probabilities based on the historical performance of crossover operators during the evolutionary process. It considers the number of times each operator was rewarded and penalized, updating the probabilities to favor operators that have demonstrated better performance in generating promising offspring.

13. credit_assignment:

- This method evaluates the dominance relationships between parent and offspring solutions and assigns credits (rewards and penalties) accordingly. It considers non-dominated solutions and penalizes or rewards them based on their ability to dominate or be dominated by other solutions, contributing to the credit assignment mechanism in NSGA-II.

14. dominance_comparison:

- The dominance_comparison method identifies non-dominated solutions within a set of chromosomes. It categorizes solutions into non-dominated and dominated sets based on their dominance relationships. This information is crucial for evaluating the Pareto front and guiding the environmental selection process.

15. avoid_zero_offspring:

- This method ensures that offspring generated through mutation or crossover are not entirely composed of zero values. It applies uniform mutation until a non-zero offspring is obtained, helping maintain diversity in the population and preventing degenerate solutions with all zero values.

16. roulette wheel selection

- This selection method simulates a roulette wheel, where each crossover operator has a section of the wheel proportional to its probability. The random value determines which section of the wheel is selected, and the corresponding crossover operator is chosen. This mechanism ensures that crossover operators with higher probabilities are more likely to be selected, mimicking a probabilistic selection process.

Results:

Dataset: DS02:

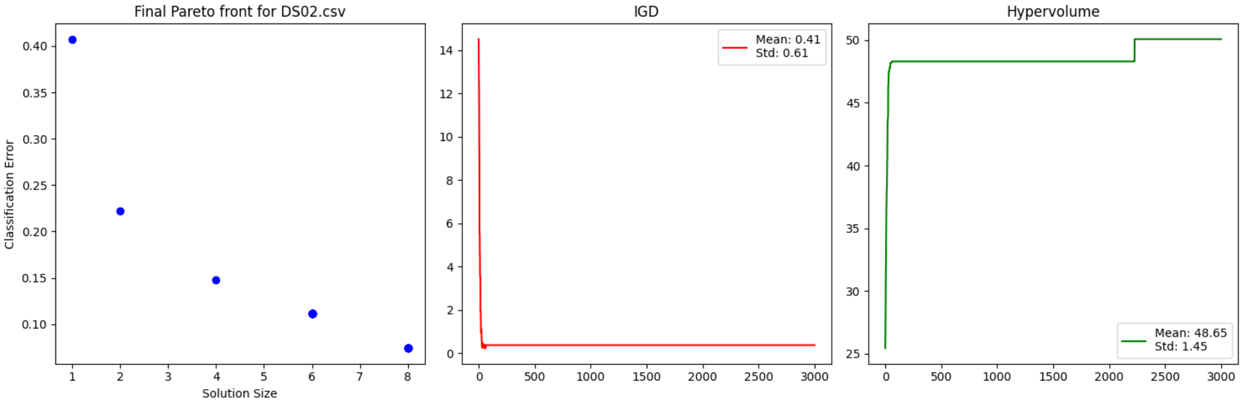
Num-features: 56

Num-samples: 27

Initialization: random

MAXFE: 300000

Evaluations: 100000, Hypervolume: 48.29629629629629
Evaluations: 200000, Hypervolume: 48.29629629629629
Evaluations: 300000, Hypervolume: 50.07407407407407

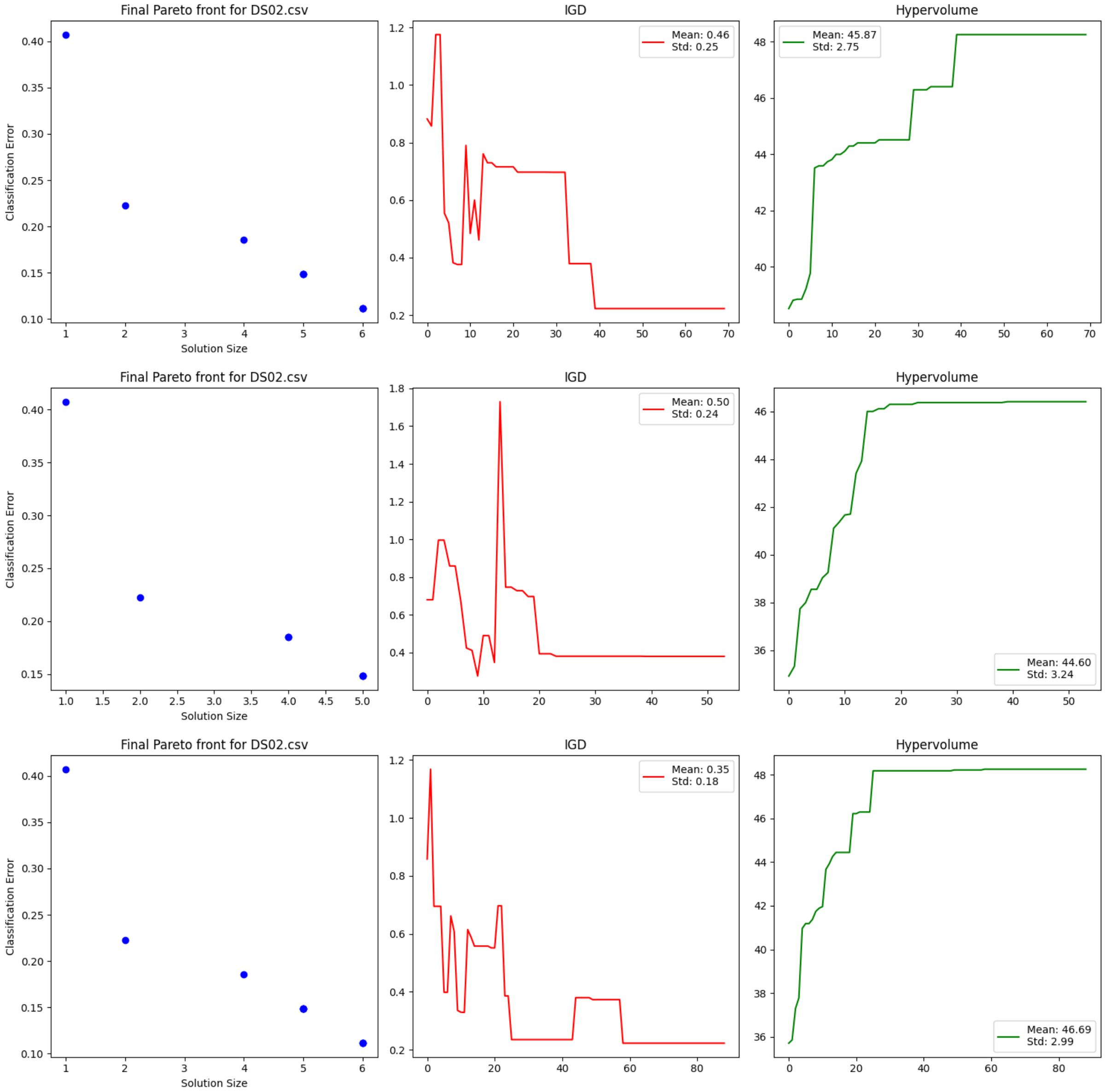


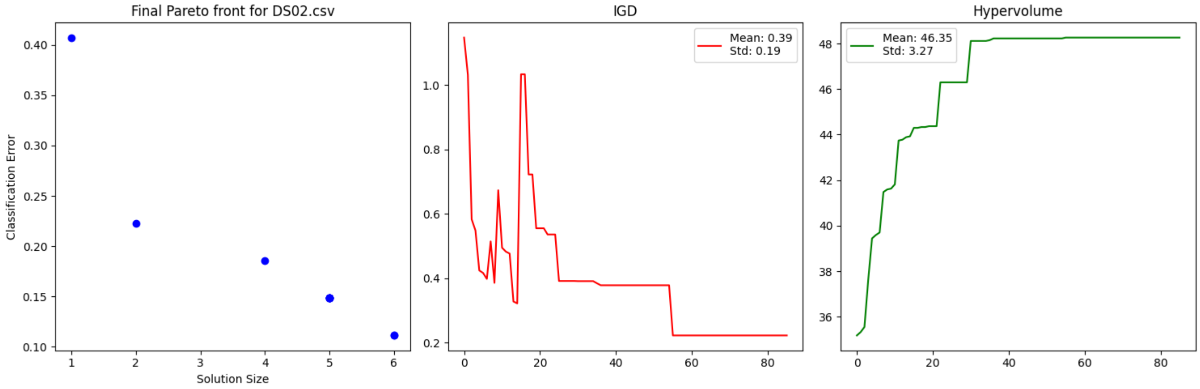
Initialization: random

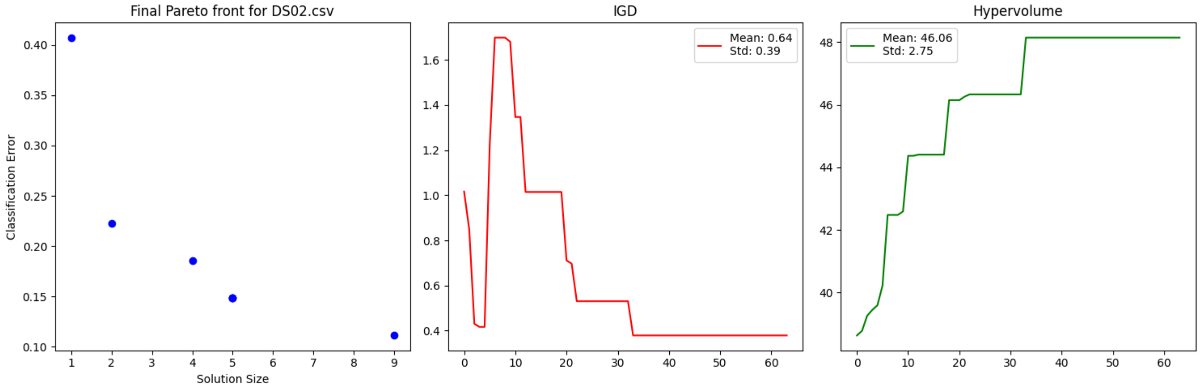
MAXFE: 10000

Termination condition: hv_threshold: 0.1, igd_threshold: 0.005, no_improvement_limit: 30

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS02.csv	70	45.8693	2.74661	0.456694	0.254774	2
DS02.csv	54	44.5988	3.23781	0.503211	0.244547	1
DS02.csv	89	46.6866	2.99346	0.348842	0.179997	2
DS02.csv	86	46.3488	3.266	0.388342	0.194258	2
DS02.csv	64	46.0642	2.74646	0.639479	0.385097	1
		45.91354	2.998068	0.4673135	0.2517346	


Final Pareto front for DS02.csv
Classification Error
Solution Size
0.10
0.15
0.20
0.25
0.30
0.35
0.40
1
2
3
4
5
6
IGD
Mean: 0.46
Std: 0.25
0.2
0.4
0.6
0.8
1.0
1.2
0
10
20
30
40
50
60
70
Hypervolume
Mean: 45.87
Std: 2.75
40
42
44
46
48
Final Pareto front for DS02.csv
Classification Error
Solution Size
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
IGD
Mean: 0.50
Std: 0.24
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Hypervolume
Mean: 44.60
Std: 3.24
36
38
40
42
44
46
Final Pareto front for DS02.csv
Classification Error
Solution Size
IGD
Mean: 0.35
Std: 0.18
0
20
40
60
80
Hypervolume
Mean: 46.69
Std: 2.99


Final Pareto front for DS02.csv
Classification Error
0.40
0.35
0.30
0.25
0.20
0.15
0.10
1
2
3
4
5
6
Solution Size
IGD
Mean: 0.39
Std: 0.19
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
Hypervolume
Mean: 46.35
Std: 3.27
48
46
44
42
40
38
36
0
20
40
60
80


Final Pareto front for DS02.csv
Classification Error
0.40
0.35
0.30
0.25
0.20
0.15
0.10
1
2
3
4
5
6
7
8
9
Solution Size
IGD
Mean: 0.64
Std: 0.39
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50
60
Hypervolume
Mean: 46.06
Std: 2.75
48
46
44
42
40
0
10
20
30
40
50
60

Dataset: DS04

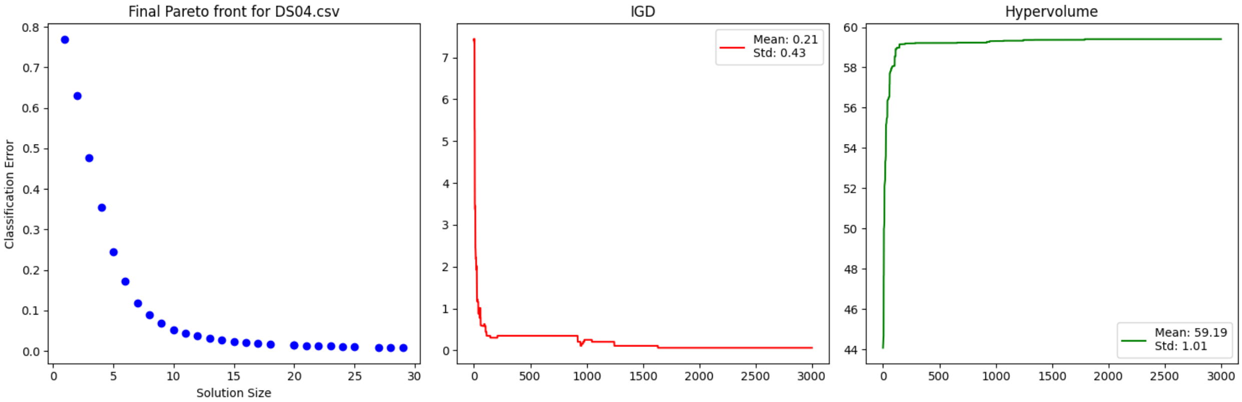
Num-features: 64

Num-samples: 1000

Initialization: random

MAXFE: 300000

number of features: 64
Evaluations: 100000, Hypervolume: 59.31214747681813
Evaluations: 200000, Hypervolume: 59.40911570252888
Evaluations: 300000, Hypervolume: 59.40911570252888

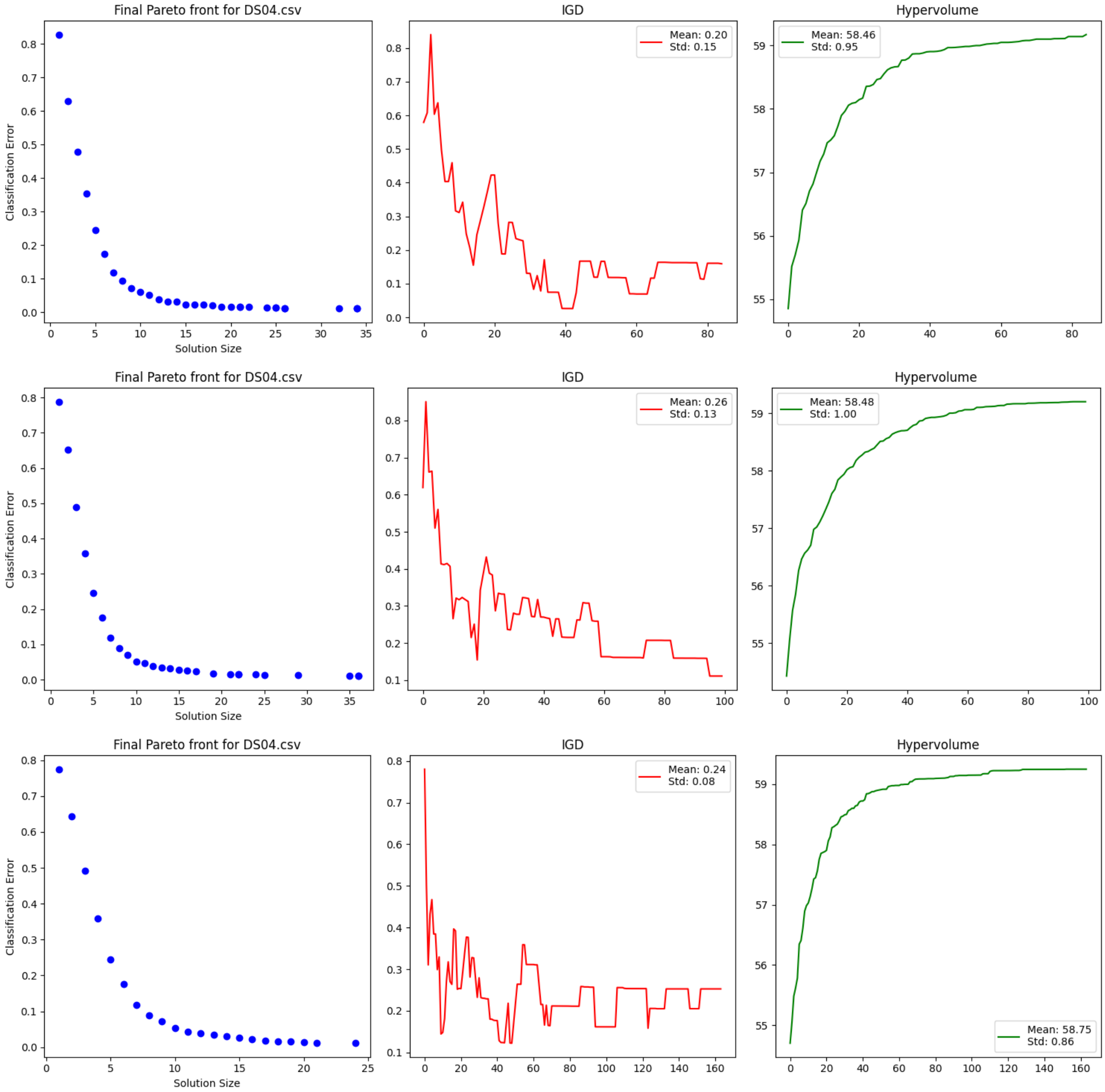


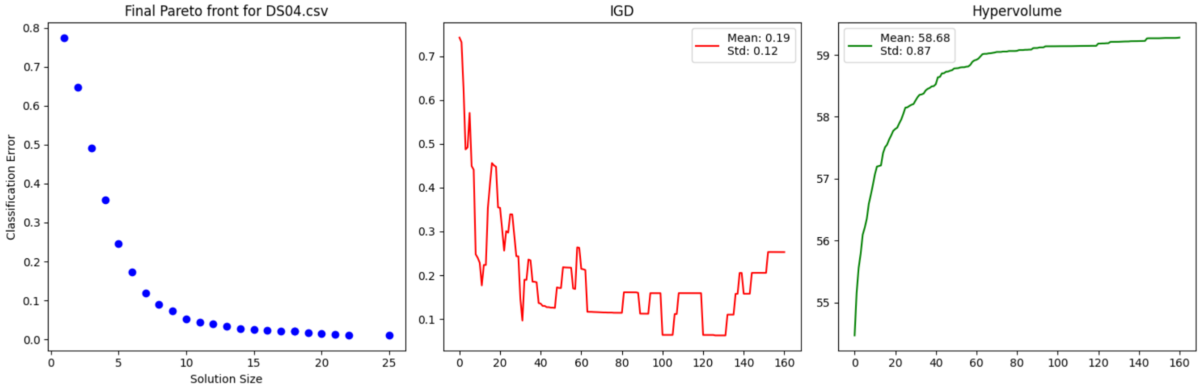
Initialization: random

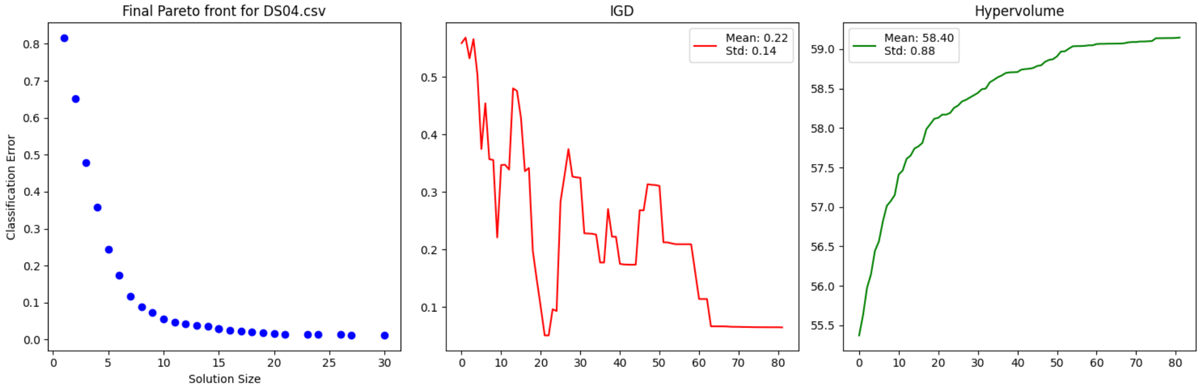
MAXFE: 20000

Termination condition: hv_threshold: 1, igd_threshold: 0.05, no_improvement_limit: 30

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS04.csv	85	58.465	0.945286	0.204455	0.153442	29
DS04.csv	100	58.4846	0.995055	0.262293	0.12707	34
DS04.csv	164	58.7548	0.857002	0.244203	0.0768767	56
DS04.csv	161	58.683	0.87241	0.193206	0.118905	52
DS04.csv	82	58.3991	0.884551	0.223908	0.140849	28
		58.5573	0.9108608	0.225612	0.12342	


Final Pareto front for DS04.csv
Classification Error
Solution Size
IGD
Mean: 0.20
Std: 0.15
Hypervolume
Mean: 58.46
Std: 0.95
Final Pareto front for DS04.csv
Classification Error
Solution Size
IGD
Mean: 0.26
Std: 0.13
Hypervolume
Mean: 58.48
Std: 1.00
Final Pareto front for DS04.csv
Classification Error
Solution Size
IGD
Mean: 0.24
Std: 0.08
Hypervolume
Mean: 58.75
Std: 0.86


Final Pareto front for DS04.csv
Classification Error
Solution Size
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
IGD
Mean: 0.19
Std: 0.12
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140
160
Hypervolume
Mean: 58.68
Std: 0.87
55
56
57
58
59
0
20
40
60
80
100
120
140
160


Final Pareto front for DS04.csv
Classification Error
Solution Size
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
5
10
15
20
25
30
IGD
Mean: 0.22
Std: 0.14
0.1
0.2
0.3
0.4
0.5
0
10
20
30
40
50
60
70
80
Hypervolume
Mean: 58.40
Std: 0.88
55.5
56.0
56.5
57.0
57.5
58.0
58.5
59.0
0
10
20
30
40
50
60
70
80

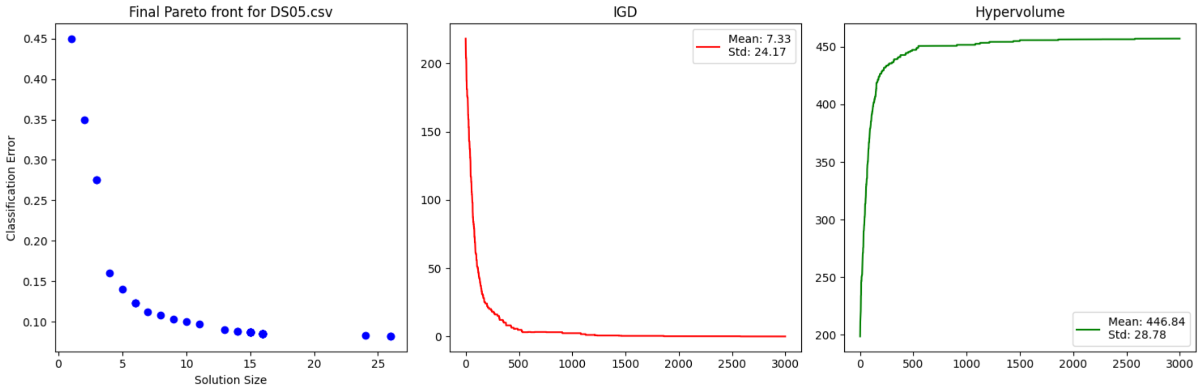
Dataset: DS05

Num-features: 500

Num-samples: 600

Initialization: random

MAXFE: 300000



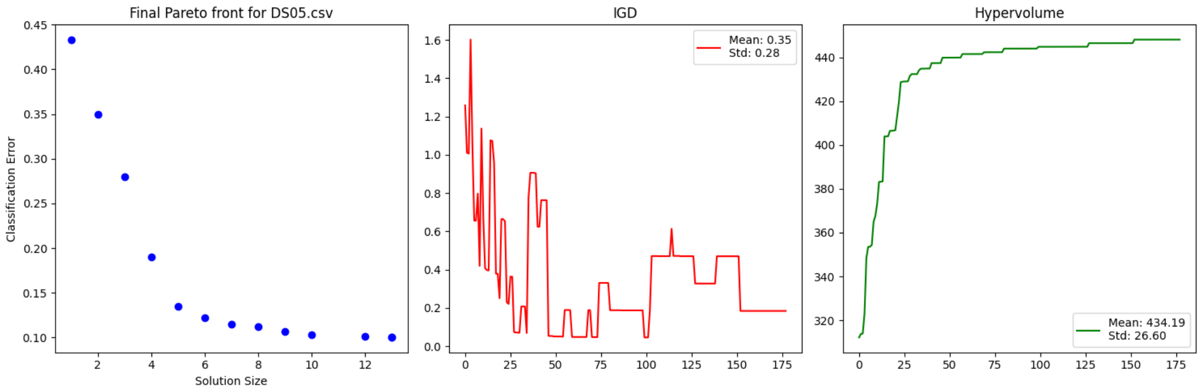
Initialization: wise_2 (based on their Pareto front max feature size)

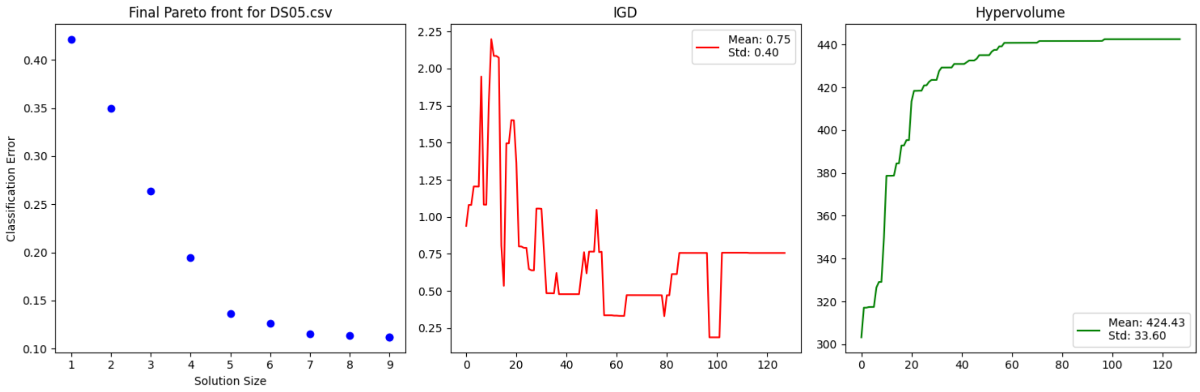
MAXFE: 30000

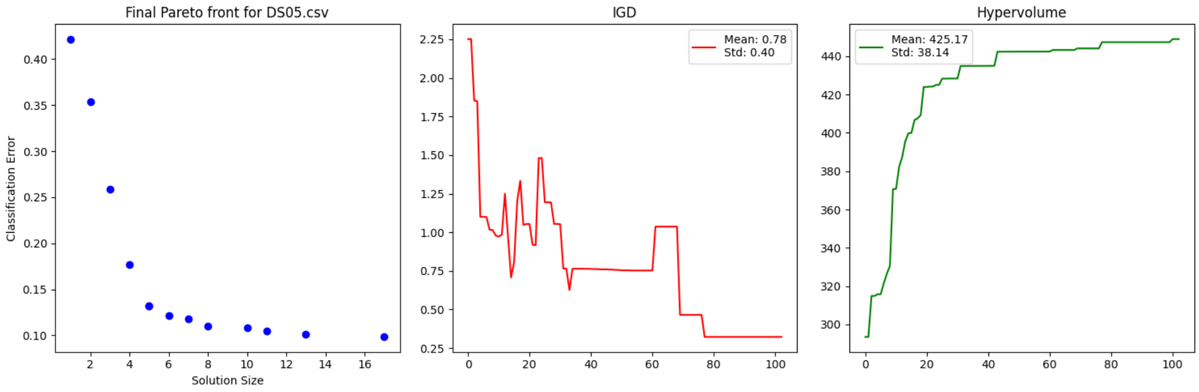
Termination condition: hv_threshold: 1, igd_threshold: 0.1, no_improvement_limit: 30

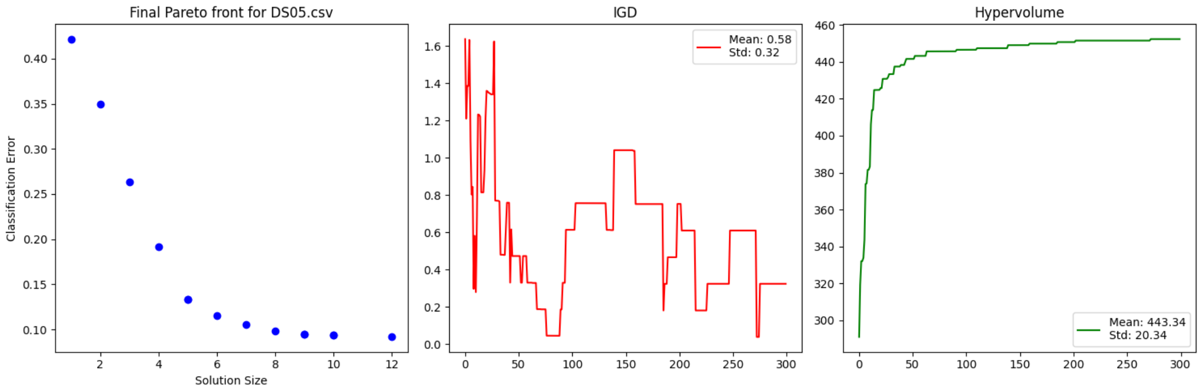
(However, these parameters may differ run to run. The best one is reported.)

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS05.csv	178	434.188	26.5986	0.352397	0.276438	37
DS05.csv	128	424.434	33.6046	0.747464	0.397666	28
DS05.csv	103	425.171	38.136	0.7811	0.402544	21
DS05.csv	300	443.34	20.337	0.582867	0.31953	62
		431.78325	29.6690	0.61595	0.3490445	


Final Pareto front for DS05.csv
Classification Error
0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10
2 4 6 8 10 12
Solution Size
IGD
Mean: 0.35
Std: 0.28
1.6 1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0
0 25 50 75 100 125 150 175
Hypervolume
440 420 400 380 360 340 320
Mean: 434.19
Std: 26.60
0 25 50 75 100 125 150 175


Final Pareto front for DS05.csv
Classification Error
0.40 0.35 0.30 0.25 0.20 0.15 0.10
1 2 3 4 5 6 7 8 9
Solution Size
IGD
Mean: 0.75
Std: 0.40
2.25 2.00 1.75 1.50 1.25 1.00 0.75 0.50 0.25
0 20 40 60 80 100 120
Hypervolume
440 420 400 380 360 340 320 300
Mean: 424.43
Std: 33.60
0 20 40 60 80 100 120


Final Pareto front for DS05.csv
Classification Error
0.40 0.35 0.30 0.25 0.20 0.15 0.10
2 4 6 8 10 12 14 16
Solution Size
IGD
Mean: 0.78
Std: 0.40
2.25 2.00 1.75 1.50 1.25 1.00 0.75 0.50 0.25
0 20 40 60 80 100
Hypervolume
Mean: 425.17
Std: 38.14
440 420 400 380 360 340 320 300
0 20 40 60 80 100


Final Pareto front for DS05.csv
Classification Error
0.40 0.35 0.30 0.25 0.20 0.15 0.10
2 4 6 8 10 12
Solution Size
IGD
Mean: 0.58
Std: 0.32
1.6 1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0
0 50 100 150 200 250 300
Hypervolume
460 440 420 400 380 360 340 320 300
Mean: 443.34
Std: 20.34
0 50 100 150 200 250 300

Dataset: DS07

Num-features: 561

Num-samples: 900

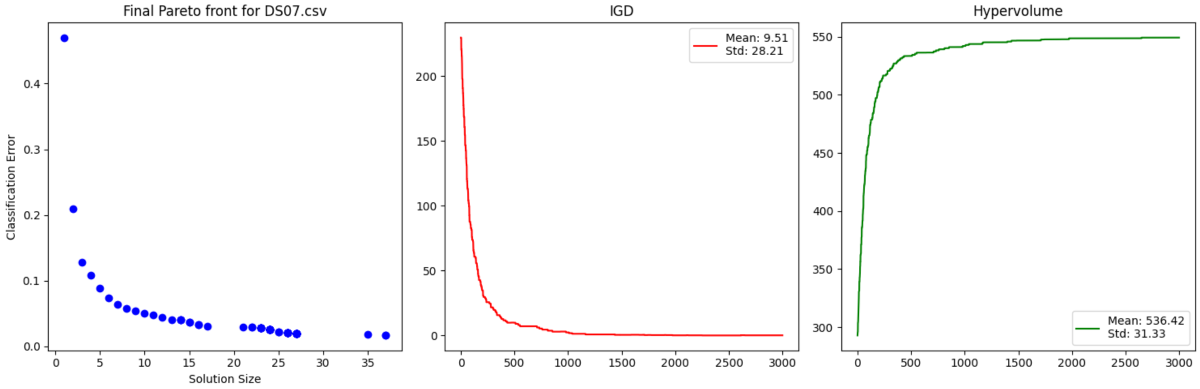
Initialization: random

MAXFE: 300000

Evaluations: 100000, HV: 542.0255555555556, IGD: 2.4288419141085584

Evaluations: 200000, HV: 548.6122222222222, IGD: 0.18200412428615262

Evaluations: 300000, HV: 549.2444444444444, IGD: 0.17772037702437637



Initialization: wise_2 (based on their Pareto front max feature size)

MAXFE: 30000

Termination condition: hv_threshold: 0.5, igd_threshold: 0.1, no_improvement_limit: 40

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS07.csv	123	525.347	10.6764	0.387856	0.275901	36
DS07.csv	207	527.906	8.21121	0.63647	0.374126	64
DS07.csv	97	521.835	7.04768	0.787352	0.312989	30
DS07.csv	116	525.097	7.9582	0.58512	0.223455	37
Avg		525.04625	8.58	0.5991995	0.3014	

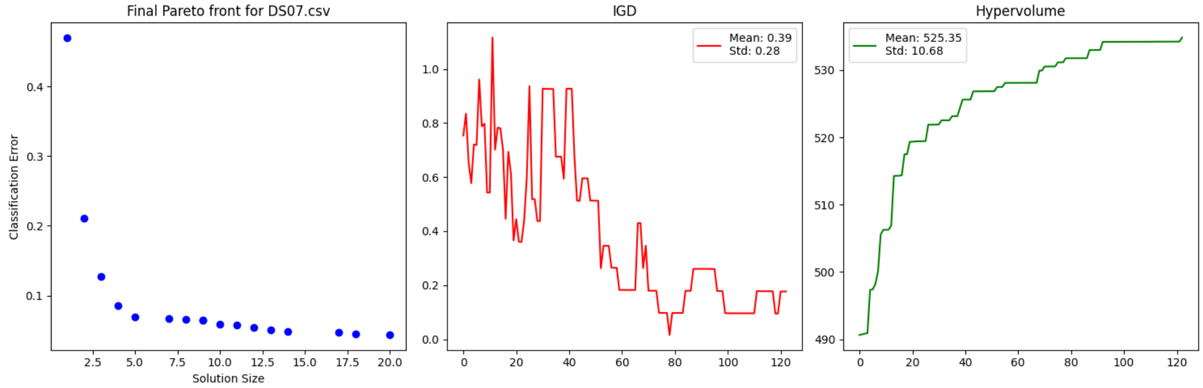
Initialization: wise_random

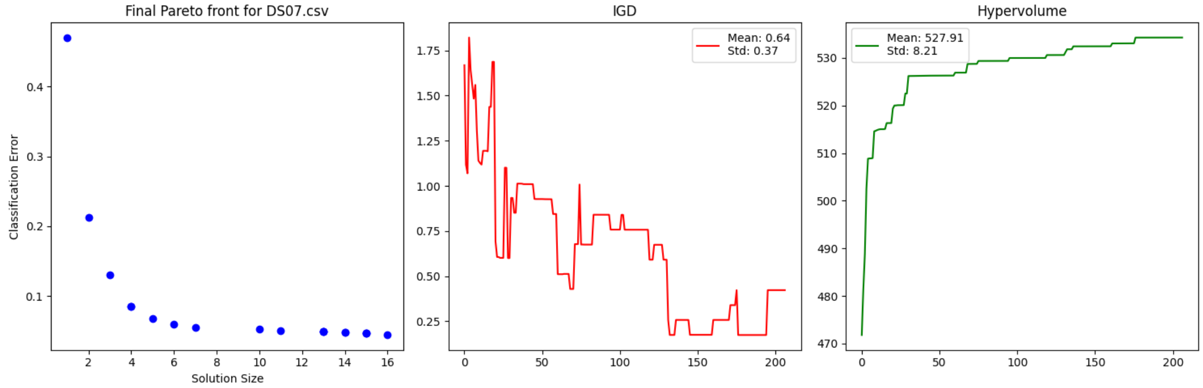
MAXFE: 30000

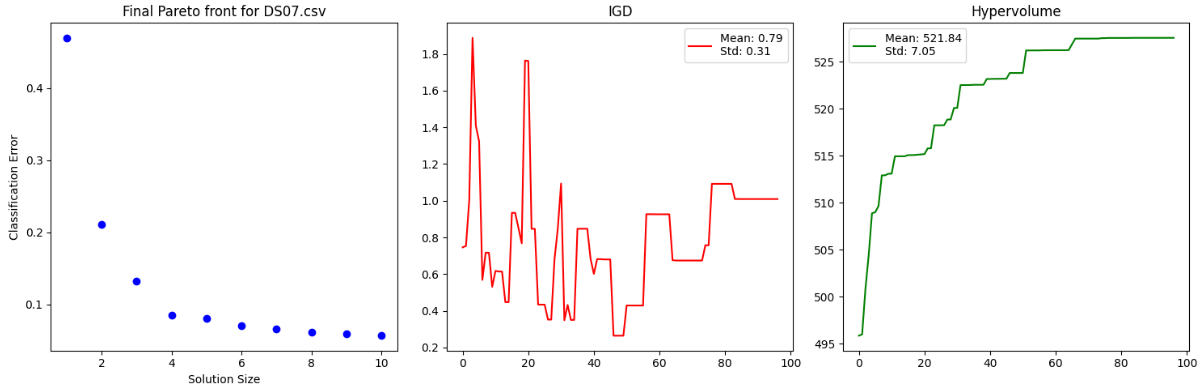
Termination condition: hv_threshold: 0.5, igd_threshold: 0.1, no_improvement_limit: 40

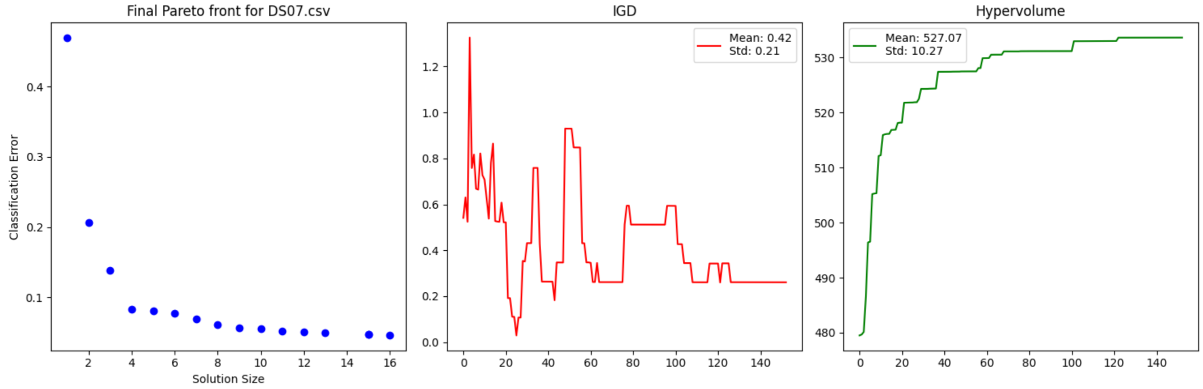
Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS07.csv	349	625.198	5.07735	3.0033	0.890111	69

Wise:

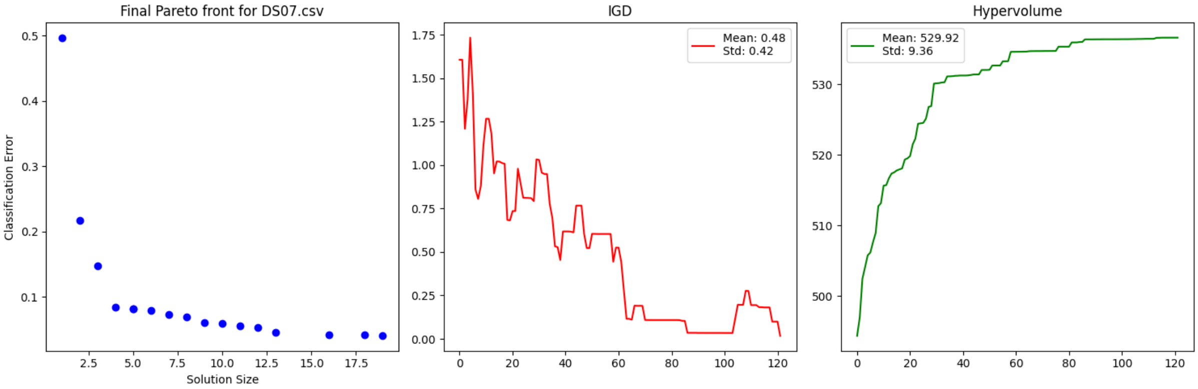

Final Pareto front for DS07.csv
Classification Error
0.1
0.2
0.3
0.4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Solution Size
IGD
Mean: 0.39
Std: 0.28
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
Hypervolume
Mean: 525.35
Std: 10.68
490
500
510
520
530
0
20
40
60
80
100
120


Final Pareto front for DS07.csv
Classification Error
0.1
0.2
0.3
0.4
2
4
6
8
10
12
14
16
Solution Size
IGD
Mean: 0.64
Std: 0.37
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0
50
100
150
200
Hypervolume
Mean: 527.91
Std: 8.21
470
480
490
500
510
520
530
0
50
100
150
200


Final Pareto front for DS07.csv
Classification Error
0.1
0.2
0.3
0.4
2
4
6
8
10
Solution Size
IGD
Mean: 0.79
Std: 0.31
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
20
40
60
80
100
Hypervolume
Mean: 521.84
Std: 7.05
495
500
505
510
515
520
525
0
20
40
60
80
100


Final Pareto front for DS07.csv
Classification Error
0.1
0.2
0.3
0.4
2
4
6
8
10
12
14
16
Solution Size
IGD
Mean: 0.42
Std: 0.21
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
20
40
60
80
100
120
140
Hypervolume
Mean: 527.07
Std: 10.27
480
490
500
510
520
530
0
20
40
60
80
100
120
140

Wise random:


Final Pareto front for DS07.csv
Classification Error
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Solution Size
IGD
Mean: 0.48
Std: 0.42
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
120
Hypervolume
Mean: 529.92
Std: 9.36
530
520
510
500
0
20
40
60
80
100
120

Dataset: DS08

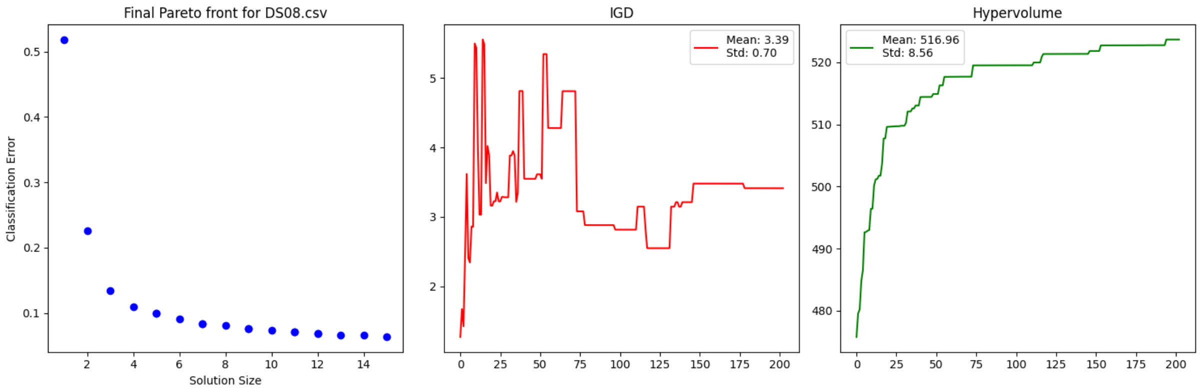
Num-features: 561

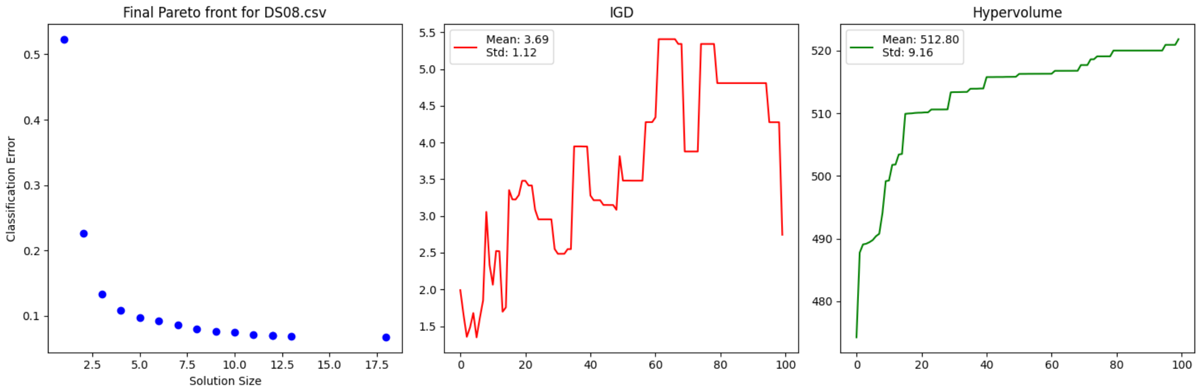
Num-samples: 1200

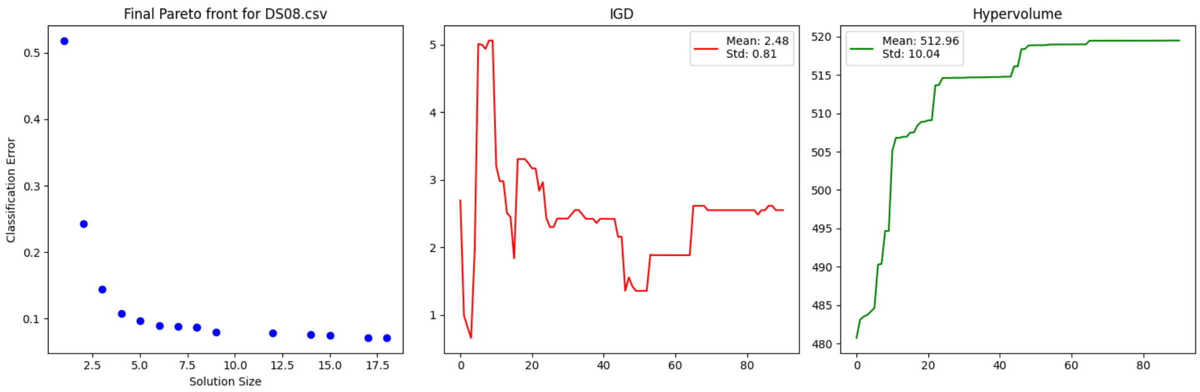
Initialization: wise_2 (based on their Pareto front max feature size)

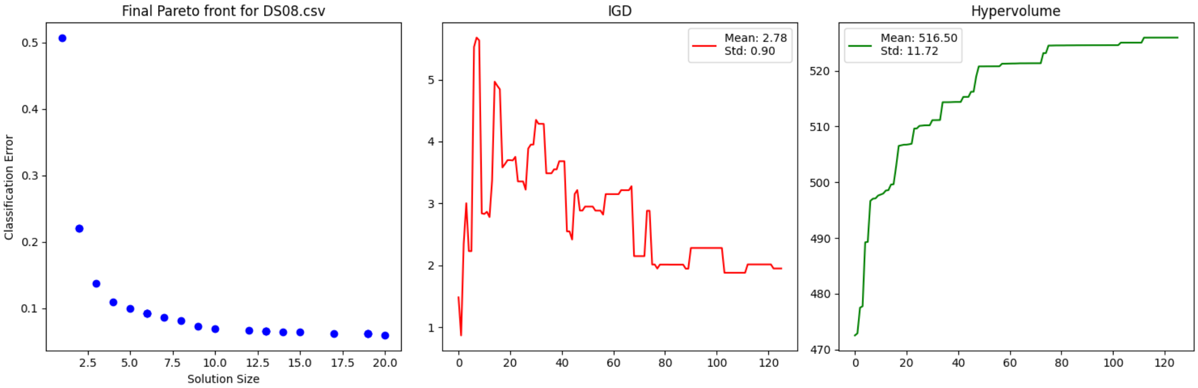
MAXFE: 300000

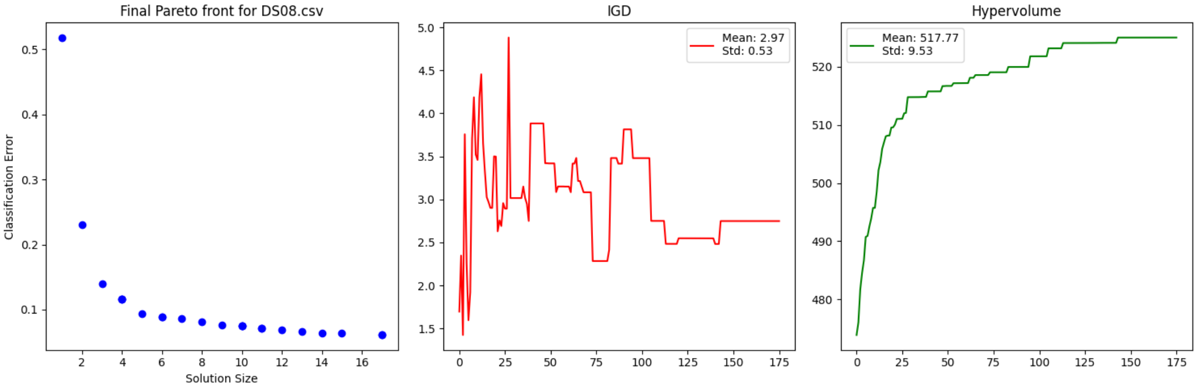
Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS08.csv	203	516.959	8.56181	3.39018	0.703712	85
DS08.csv	100	512.801	9.15733	3.68532	1.12288	32
DS08.csv	91	512.957	10.0353	2.47645	0.806087	29
DS08.csv	126	516.503	11.7208	2.77915	0.897477	46
DS08.csv	176	517.771	9.52984	2.97462	0.529173	75








Final Pareto front for DS08.csv
Classification Error
0.5
0.4
0.3
0.2
0.1
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Solution Size
IGD
Mean: 2.78
Std: 0.90
5
4
3
2
1
0
20
40
60
80
100
120
Hypervolume
Mean: 516.50
Std: 11.72
520
510
500
490
480
470
0
20
40
60
80
100
120


Final Pareto front for DS08.csv
Classification Error
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
12
14
16
Solution Size
IGD
Mean: 2.97
Std: 0.53
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0
25
50
75
100
125
150
175
Hypervolume
Mean: 517.77
Std: 9.53
520
510
500
490
480
0
25
50
75
100
125
150
175

Dataset: DS10

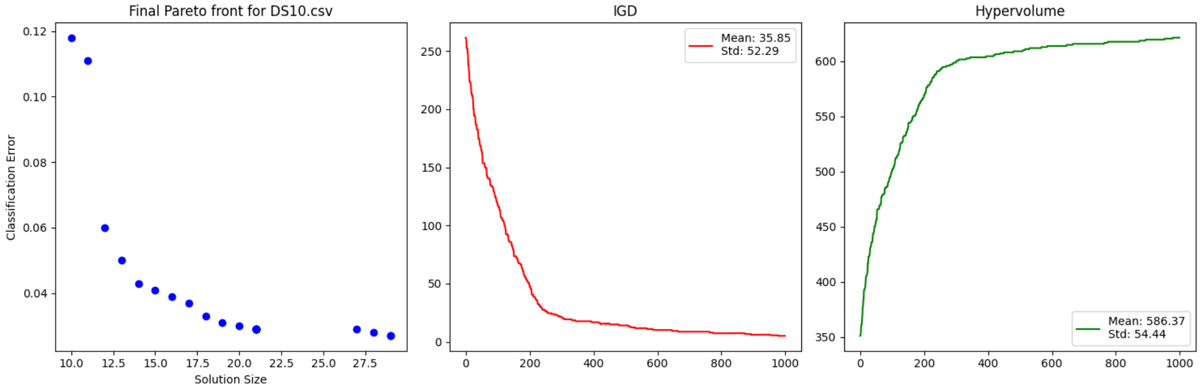
Num-features: 649

Num-samples:

Initialization: random

MAXFE: 100000

Evaluations: 25000, HV: 591.9554824285364, IGD: 26.928896199185708
Evaluations: 50000, HV: 609.0753687819556, IGD: 14.073136268640786
Evaluations: 75000, HV: 615.8413743084402, IGD: 8.64583942329659
Evaluations: 100000, HV: 621.4474144803486, IGD: 5.086199016790782



Initialization: wise

MAXFE: 100000

Termination condition: hv_threshold: 0.5, igd_threshold: 0.05, no_improvement_limit: 50

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS10.csv	273	619.325	6.14583	3.47249	0.918371	99
DS10.csv	197	621.221	7.63149	3.69707	0.643908	41
DS10.csv	194	617.376	8.40519	3.26008	0.88995	58

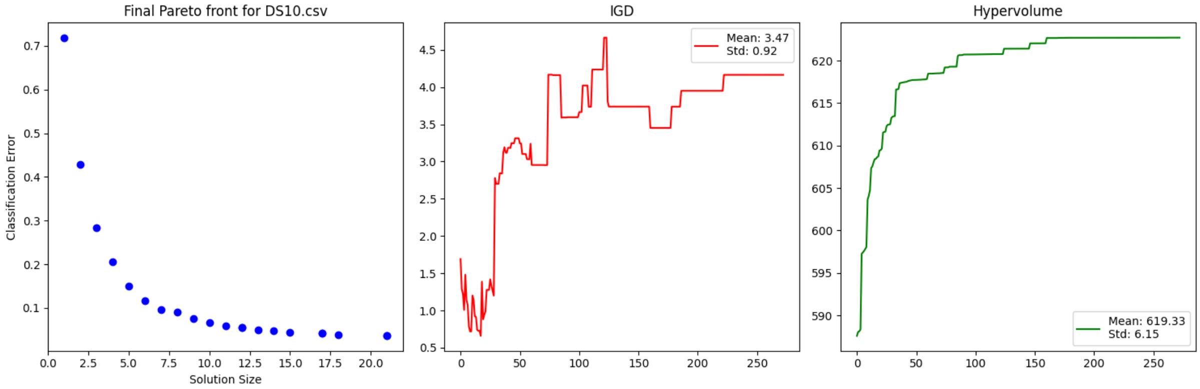
Initialization: wise_random

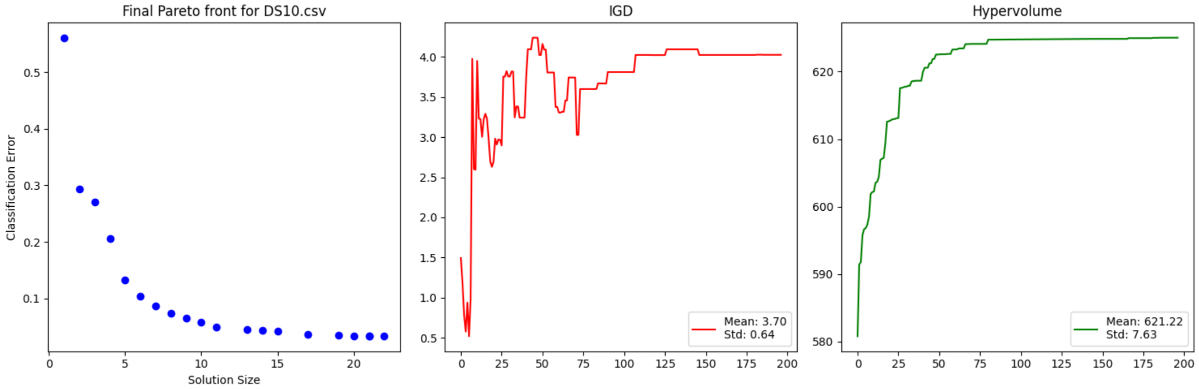
MAXFE: 100000

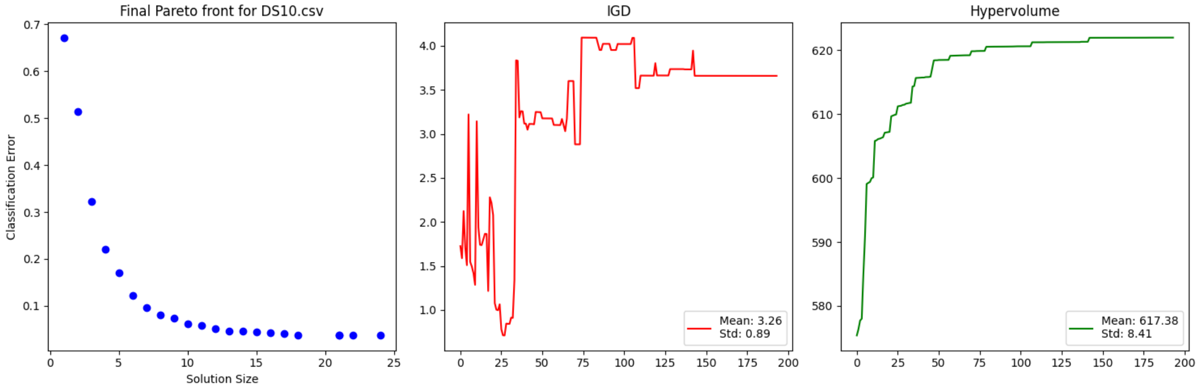
Termination condition: hv_threshold: 0.25, igd_threshold: 0.05, no_improvement_limit: 70

Dataset	Generation#	HV_mean	HV_std	IGD_mean	IGD_std	Duration
DS10.csv	349	625.198	5.07735	3.0033	0.890111	69

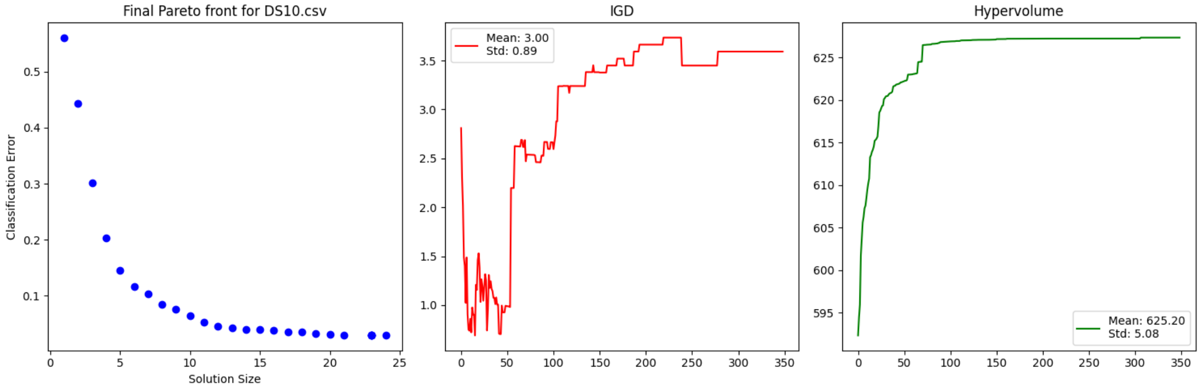
Wise initialization 2 result:







Random wise result:



Overall Result:

1. Wise Initialization: A well-designed initialization strategy can significantly impact the performance of genetic algorithms like NSGAII. Wise initialization might involve starting with a diverse set of solutions to cover a broad area of the search space or initializing with heuristically informed solutions that are likely to be closer to optimal. This approach can lead to faster convergence and improved solution quality, as it reduces the likelihood of the algorithm getting stuck in suboptimal regions of the search space.

2. Effect of Column Number Expansion: Increasing the length of chromosomes (i.e., the number of features) affects the search space's complexity. A larger search space often requires more generations to converge and can increase the risk of premature convergence if not managed properly. It also demands more computational resources and may require adjustments in population size or mutation rates to maintain diversity.

3. Effect of Sample Number Expansion in Datasets: Increasing the number of samples can impact the classification accuracy and the computational cost of the evaluation function (especially in KNN). More samples usually provide a better representation of the underlying distribution, potentially leading to more accurate feature selection. However, it also increases the time required for fitness evaluation.

4. Fast-Non-Dominated Sorting Parallelization: Parallelizing this process can significantly reduce computational time, especially for large population sizes or complex objective functions. Faster sorting allows more generations to be processed in the same time, potentially improving the algorithm's ability to explore and exploit the search space.

5. Objective Evaluation Parallelization: Parallelizing the evaluation of objectives (like classification error and the number of features) can drastically reduce the overall runtime. This is particularly true when using computationally intensive classifiers like KNN. Faster evaluations mean more generations can be processed, enhancing the evolutionary process.

6. Effect of K=3 Fold=3 in Algorithm: This setting in KNN impacts the reliability of the classification error as an objective. Using a 3-fold cross-validation provides a balance between computational cost and the reliability of the error estimate. However, it may not capture the model's performance variance as effectively as a higher number of folds would.

7. Different Crossovers in the Crossover Pool: The inclusion of various crossover operators like one-point, two-point, uniform, shuffle, and reduced surrogate adds diversity to the genetic search. Each operator has different characteristics in terms of how it combines parent chromosomes and explores the search space.

8. Roles of Crossovers in Exploration and Exploitation:

- One-point and Two-point Crossover: Tend to be more exploitative, as they create offspring similar to parents, maintaining building blocks.

- Uniform Crossover: More exploratory, as it mixes genes from parents more uniformly, creating diverse offspring.

- Shuffle Crossover: Can be seen as a balance between exploration and exploitation, depending on how it's implemented.

- Reduced Surrogate Crossover: Likely to be more exploitative, focusing on maintaining and refining existing good solutions.

In summary, each of these aspects has a substantial impact on the results and performance of NSGAII implementation. Balancing these elements is key to optimizing the algorithm's efficiency and effectiveness in feature selection tasks. For instance, wise initialization can kickstart the search process more effectively, while the choice of crossover operators can dictate the balance between exploring new areas of the search space and exploiting known good solutions. The setting of KNN parameters and the parallelization of key processes also play crucial roles in determining the speed and quality of the solutions found. Understanding the interplay of these factors is essential for fine-tuning the algorithm to specific datasets and feature selection challenges.