Efficient Feature Selection for Intrusion Detection Systems with Priority Queue-based GRASP

Vagner E. Quincozes
Computer Science Department
Federal Fluminense University
Niterói, Brazil
https://orcid.org/0000-0002-6688-5572

Silvio E. Quincozes
Campus Alegrete
Federal University of Pampa
Alegrete, Brazil
https://orcid.org/0000-0001-6793-4033

Célio Albuquerque
Computer Science Department
Federal Fluminense University
Niterói, Brazil
https://orcid.org/0000-0002-7959-6569

Diego Passos
DEETC
Instituto Superior de Engenharia de Lisboa (ISEL)
Lisboa, Portugal
https://orcid.org/0000-0002-9707-1176

Daniel Mossé
Computer Science Department
University of Pittsburgh
Pittsburgh, United States
https://orcid.org/0000-0002-9508-9815

2024 IEEE 13th International Conference on Cloud Networking (CloudNet) | 979-8-3503-7656-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/CLOUDNET62863.2024.10815746

***Abstract*—The Greedy Randomized Adaptive Search Procedure for Feature Selection (GRASP-FS) is a recently-proposed metaheuristic that optimizes the feature selection process for Intrusion Detection Systems (IDS) by combining exploration and refinement techniques for more assertive intrusion detection. However, GRASP-FS may be time and resource-consuming for large datasets. In this work, we propose GRASPQ-FS, an extended version of GRASP-FS using Priority Queues to reduce resource consumption and processing time. As an additional contribution, we provide a comprehensive analysis of the most suitable parameters for our GRASPQ-FS. Our results reveal that GRASPQ-FS can speed up feature selection up to 90% over GRASP-FS, without compromising F1-Score. Also, we observed that a priority queue with 50 solutions saved 50% in execution time while increasing the F1-Score by 4.5%.**

***Index Terms*—Intrusion Detection System (IDS), Feature Selection, GRASP, Cyber-Physical System (CPS)**

I. Introduction

In critical systems, cyber attacks have become a growing threat, jeopardizing the security of vital information across various sectors [1]. These attacks range from privacy breaches to disruptions of critical services, such as the deployment of the BiBiWiper malware by Iranian state-linked adversaries targeting Israeli companies during the 2023 Israel-Hamas conflict [2]. This highlights the need for robust Intrusion Detection Systems (IDS). IDSs can be deployed in various environments, including cloud infrastructures, to monitor and protect against a wide range of cyber threats [3]. They process a range of network features and are essential for identifying intrusion attempts or unauthorized use of networks and systems, acting as a crucial line of defense against such threats [4], [5].

However, the efficiency of IDSs is inherently linked to their ability to process and analyze large volumes of data, making feature selection (FS) a fundamental step. FS aims to identify the most relevant data attributes, discarding features that are irrelevant or redundant, to improve both computational efficiency and effectiveness in detecting attacks. In this context, the Greedy Randomized Adaptive Search Procedure (GRASP) emerges as a promising tool, providing a balance between exploration and refinement in the search for optimal solutions [5].

GRASP consists of two phases: construction and local search. In the construction phase, a set of initial solutions is generated, which are later refined in the local search phase. This iterative process is key to the continuous performance improvement of the system. As the local search progresses, it explores and refines the initial solutions. This approach allows GRASP to balance the generation of initial solutions and the optimization of these solutions, making it efficient for feature selection in intrusion detection systems, where accuracy and computational efficiency are of utmost importance. GRASP for Feature Selection (GRASP-FS [5]) has been proposed, adapted, and employed in the context of feature selection for IDS. However, GRASP-FS can still consume excessive computational resources in scenarios where the processed dataset is very large. In scenarios where response time is an important factor, such as in intrusion detection in critical infrastructure, efficiency is of paramount importance.

In this work, we observe that the construction phase of GRASPQ-FS is the dominant factor. Thus, we propose GRASP *with Priority Queue for Feature Selection* (GRASPQ-FS) to significantly improve the construction phase with the use of a very simple mechanism: a priority queue. GRASPQ-FS optimizes the use of computational resources and reduces processing time, while maintaining efficiency in FS. Our GRASPQ-FS sorts solutions during the construction phase based on an established priority criteria. Thus, only the most promising solutions advance to the local search phase, drastically reducing the required number of iterations. Furthermore, we provide (a) an open-source implementation[1], (b) an evaluation of GRASPQ-FS over several parameters, and (c) an

[1]Available in: https://github.com/vagnerereno/GRASPQ-FS

analysis that shows the most suitable parameters for the new algorithm.

The remainder of the paper is organized as follows. Section II presents the theoretical foundation. Section III discusses related works. Section IV details the GRASPQ-FS proposal. Section V presents the conducted experiments, and Section VI discusses their results. Finally, Section VII concludes the paper and discusses future work.

II. Background

In this section, a conceptual and theoretical basis is presented. In Subsection II-A, Intrusion Detection Systems will be explored. In Subsection II-B, the GRASP metaheuristic will be discussed.

A. Intrusion Detection System

IDSs play a crucial role in the security of networks and computer systems. They monitor data traffic and system activities, seeking to identify behavior patterns that may indicate an attack or intrusion attempt. By analyzing a wide range of data, from traffic patterns to virus signatures and hacker tactics, IDSs are capable of alerting system administrators to potentially malicious activities. The efficacy of an IDS depends not only on its real-time monitoring capacity but also on its ability to distinguish between normal and abnormal behaviors, thereby minimizing false alarms and ensuring a quick and accurate response [4].

Feature selection is a fundamental aspect of optimizing IDSs. Features are essentially the attributes of the data that the system analyzes, such as traffic patterns, packet types, and connection frequencies. Feature selection involves identifying the most relevant attributes for effective intrusion detection, discarding those that are irrelevant or redundant. A well-defined feature set not only increases the accuracy in identifying threats but also reduces the volume of data to be processed, resulting in higher computational efficiency. Traditionally, feature selection can be performed through statistical techniques (i.e., filter-based selection), machine learning algorithms (i.e., wrapper selection), or can be implemented as part of the decision-making process in classifier algorithms (i.e., embedded selection) [5].

Feature selection can also employ metaheuristics, like GRASP, which offer a more adaptive and efficient approach.

B. GRASP

GRASP, introduced by [6], is well-known metaheuristic that has been used in combinatorial optimization problems in different areas [7], [8]. Essentially, GRASP is characterized by its iterative and adaptive approach, divided into two main phases: the construction phase and the local search phase. In the construction phase, GRASP generates initial solutions through an iterative process that mixes deterministic and random elements. The deterministic component, often a greedy algorithm, is used to build a solution step by step, selecting, at each stage, an element that seems to be the most promising. Concurrently, randomness is introduced to ensure diversity in the initial solutions, preventing the process from getting stuck in local optima. This balance between greedy and random decision-making allows GRASP to efficiently explore the search space, generating a varied set of starting solutions. After generating the initial solutions, the local search phase begins. Each initial solution is refined through a local search process, which explores the neighbors of the current solution in an attempt to find a better one. This process continues until a stopping criterion is reached, such as a time limit or the inability to find further improvements. The local search is crucial for enhancing the solutions generated in the construction phase, increasing the chances of finding a global optimum. Figure 1 illustrates how GRASP-FS works.

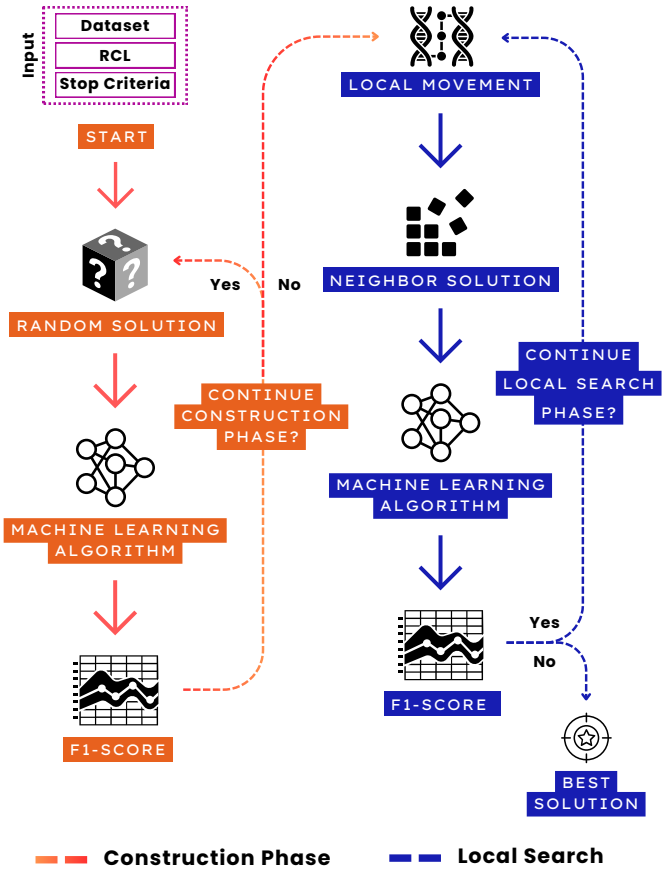


Fig. 1. GRASP-FS.

GRASP has proven effective in a wide range of problems, especially those characterized by large and complex search spaces [8]. Its flexibility and the ability to balance exploration (searching new areas of the solution space) and intensification (refining promising solutions) make it a powerful tool for optimization problems [8].

III. Related Work

There are various feature selection (FS) techniques in the literature [9]. However, in this section, our focus will be on approaches that employ the GRASP metaheuristic, chosen for its efficiency demonstrated in previous studies [10].

Yusta [7] demonstrated that GRASP could be more effective than other algorithms like Sequential Forward Floating Selection (SFFS), Tabu Search, Genetic, and Memetic algorithms for selecting the best subset of features to classify instances from various databases. Esseghir and Amir [11] also applied GRASP to FS on some of these datasets and others, exploring different classifiers.

Bermejo *et al.* [12] used GRASP to handle large and high-dimensional datasets, reducing the number of wrapping evaluations through the Incremental Wrapper Subset Selection (IWSS) algorithm in the construction phase. Moshki *et al.* [13] combined GRASP with an extended version of Simulated Annealing in local search, with new parameters to balance between accuracy and execution time, albeit with limitations.

Diez *et al.* [14] proposed the GRASP Forest (G-Forest) for constructing decision tree ensembles, applying GRASP concepts in FS and choosing split points at each tree node. G-Forest creates a Restricted Candidate List (RCL) with features that have an Information Gain (IG) above a specific threshold, selecting them randomly, followed by a cyclic procedure to find the desired number of trees. The authors do not implement the complete metaheuristic, only the construction phase of GRASP. In a subsequent work [15], they extend G-Forest to the Annealed Randomness Forest (GAR-Forest) proposal. The key idea is introducing a parameter that controls the randomness during the construction phase, varying from a totally random to a totally greedy procedure.

While the previous works are not specific to the domain of intrusion detection, Kanakarajan and Muniasamy [16] applied GAR-Forest to detect attacks in traditional networks, achieving an accuracy of 77.26% and an F1-Score of 72% on the multiclass dataset KDD. Similarly, in [5] the GRASP metaheuristic is adapted for FS for intrusion detection in critical infrastructure networks. This adaptation, called GRASP-FS, is employed to select feature subsets to construct IDSs specialized in specific attacks, with results superior to those obtained by traditional filter-based FS methods.

Regarding scalability, there are relevant proposals that expand GRASP-FS through innovative architectures. For instance, [17] proposed a scalable microservices-based architecture to improve the local search phase. These initiatives represent significant advancements in the quest for GRASP-FS scalability. However, they diverge from the objective of this paper, which focuses on improving the computational performance of GRASP-FS with priority queues.

The related literature shows that, despite the various applications and adaptations of GRASP in several contexts, including FS for IDS, there is a significant gap concerning computational efficiency, especially in scenarios with large volumes of data. Additionally, there are challenges of scalability and efficiency, particularly in situations where response time is critical, such as in the detection of intrusions in essential infrastructures.

IV. GRASPQ-FS

In this section, we present our GRASPQ-FS, an efficient extension of the GRASP-FS algorithm, which integrates a simple priority queue to optimize FS in IDSs. Specifically, in Subsection IV-A we detail the functioning of the GRASPQ-FS, explaining the interaction between the constructive and local search phases. Then, in Subsection IV-B we present the GRASPQ-FS algorithm.

A. GRASPQ-FS Overview

Before starting the construction phase, GRASPQ-FS generates the *Restricted Candidate List* (RCL, as described above) by selecting the features with the highest IGs (the target size of the RCL is a parameter). In the construction phase, GRASPQ-FS generates a series of random solutions using features from the RCL. Each solution is used to generate a classifier with the desired algorithm (e.g., Naive Bayes), and that classifier is evaluated in terms of F1-Score to determine the solution quality.

The higher-quality solutions are organized into a priority queue for further processing. This queue is limited to the N best solutions, ensuring a focus on the most promising feature combinations. These top N solutions in the queue are forwarded for in-depth processing in the local search phase. During this phase, the GRASPQ-FS selects and removes from the queue the solution with the highest F1-Score. From this top solution, the algorithm uses the RCL to generate and evaluate neighboring solutions, applying the same F1-Score metric. When a neighboring solution with a higher F1-Score is found, it is elevated to the status of the best global solution. This process continues until a local stopping criterion is reached or the solutions in the priority queue are exhausted.

B. GRASPQ-FS Algorithm

Algorithm 1 shows GRASPQ-FS, structured as a construction phase (lines 11-22) and a local search phase (lines 23-32). The former uses a randomized greedy approach to generate initial solutions, while the latter aims to refine these solutions, that is, to explore neighboring solutions of better quality.

The algorithm begins by receiving the set of features from a dataset (`fSet`), the maximum number of iterations (`maxIt`), the size of the RCL, the number of features for each generated subset (`numF`), and the desired size of the priority queue. The expected result is the best subset of features. Variables are initialized between lines 2 to 6, defining the RCL, the best feature set, the initial F1-Score, and the priority queue, all starting at zero, in addition to establishing the iteration counter. The loop between lines 7 and 9 calculates the IG for each feature, which are used to define the RCL on line 10.

The construction phase begins at line 11 and continues through line 22. At each iteration, the algorithm generates initial solutions and evaluates their quality (based on F1-Score). The solutions are then added to the priority queue (if there is still space or if they are better than other solutions in the queue).

Lines 23 to 31 correspond to the local search phase, which tries to improve solutions in the queue. The initial solutions are removed from the priority queue in decreasing order of quality and submitted to the local search function. The best

solution found after the local search is evaluated and, if its F1-Score is higher than the best F1-Score found so far, the best feature set is updated with the current solution. Finally,

Algorithm 1: GRASPQ-FS WITH PRIORITY QUEUE

```
   input : fSet // complete feature set
           maxIt // number of iterations
           RCLsize // number of features in RCL
           numF // number of features to select
           priorityQueue // priority queue size
   output: bestFS // best feature subset
1  begin
2      rcl ← ∅ // restricted candidate list
3      bestFS ← ∅
4      bestFS.F1Score ← 0
5      priorityQueue ← ∅ // priority queue
6      numIterations ← 0
7      foreach feature ∈ fSet do
8          feature ← calculateIG(feature)
9      end
10     rcl ← topIGRanked(fSet, RCLsize)
       // Start of the Construction Phase
11     while numIterations < maxIt do
12         initialSolution ← construction(numF, rcl,
             RCLsize, fSet)
13         initialSolution.F1Score ←
             evaluate(initialSolution)
14         if priorityQueue < priorityQueue.maximumSize()
             then
15             addpriorityQueue(initialSolution,
                 initialSolution.F1Score)
16         end
17         else if initialSolution.F1Score >
             priorityQueue.lowerF1Score() then
18             priorityQueue.removeLowestF1Score()
19             addpriorityQueue(initialSolution,
                 initialSolution.F1Score)
20         end
21         numIterations ← numIterations + 1
22     end
       // Start of the Local Search Phase
23     while priorityQueue ≠ ∅ do
24         initialSolution ← priorityQueue.dequeue()
25         bestLocalFS ← localSearch(initialSolution, rcl)
26         bestLocalFS.F1Score ← evaluate(bestLocalFS)
27         if bestLocalFS.F1Score > bestFS.F1Score then
28             bestFS ← bestLocalFS
29             bestFS.F1Score ← bestLocalFS.F1Score
30         end
31     end
32     return bestFS
33 end
```

at line 32, the algorithm returns the best feature set found, concluding the GRASPQ-FS process.

V. EXPERIMENTAL SETUP

In this section, we present the preparation of the practical experiments conducted to evaluate the efficacy of the proposed algorithm using the ERENO data generator [18].

A. Dataset

For the experiments conducted in this study, we used ERENO [18], a network traffic generation tool. ERENO is specially designed to simulate traffic in digital power substations adhering to the IEC-61850 standard, a globally recognized and adopted standard [19]. ERENO generates both GOOSE and SV messages, essential protocols in this domain.

We used ERENO to generate a dataset[2] consisting of 4,000 samples, which was split into 50% for training and 50% for testing. This dataset includes records of normal samples and seven types of cyber attacks: Random Replay, Inverse Replay, Masquerade Fake Fault, Masquerade Fake Normal, Injection, High StNum, and Poisoned High Rate. These attacks were chosen because they represent the most common and challenging threats in modern industrial control systems [18].

The dataset consists of 51 features related to the network traffic, providing a detailed view of control messages, communication frequency, and other traffic patterns. The features include data extracted from both the SV (e.g., electric current and voltage values for different phases of the electrical network) and GOOSE messages (e.g., status number, sequence, and circuit breaker status), as well as enriched features that are built by combining the former ones, including insights into the electrical signal characteristics and features that illustrate the differences between messages, among others [18].

B. Testing Environment and Parameters

The experiments were conducted on a computer with an Intel® Core™ i5-10300H CPU @ 2.50GHz, which has a maximum CPU frequency of 4500 MHz and a minimum of 800 MHz. It also has an NVIDIA GeForce GTX 1650 Ti for graphical processing and 16 GB of total system memory.

To evaluate FS performance, we used two simple classification algorithms, Naive Bayes (NB) and K-Nearest Neighbors (KNN), since neither of them uses an internal feature selection. To evaluate the sensitivity of GRASPQ-FS to its parameters, we considered the following values:

- 100 iterations for the construction phase.
- 100 iterations for the local search.
- Priority queues of sizes 10, 30, and 50.
- RCLs with 15, 25, and 40 features.
- Initial solutions with 5 and 10 features.

We use GRASP-FS (i.e., without priority queue) as a baseline, where every 100 solutions generated in the construction phase are subjected to 100 additional iterations in the local search phase, resulting in a total of 10,000 iterations. In contrast, with our priority queue of size N, the number of iterations in the local search phase is reduced to $100N$.

VI. RESULTS AND DISCUSSION

Table I presents the results related to the effect of parameters on the performance of GRASPQ-FS (bottom part is the baseline), revealing important insights. First, when analyzing the performance of NB and KNN algorithms under different

[2]The datasets are available in: https://github.com/vagnerereno/GRASPQ-FS

TABLE I
GRASPQ-FS F1-SCORE AND EXECUTION TIME OF THE NAIVE BAYES (NB) AND K-NEAREST NEIGHBORS (KNN) ALGORITHMS

Initial Solution	Priority Queue	RCL	Best Solution F1-Score		Construction Time (Seconds)		Local Search Time (Seconds)	
			NB	KNN	NB	KNN	NB	KNN
5	10	15	82.55%	85.24%	0.75	11.24	6.98	100.3
		25	84.37%	81.37%	0.75	10.68	7.07	106.5
		40	81.18%	81.98%	0.82	9.96	7.88	96.1
10	10	15	83.23%	79.42%	0.89	13.53	9.73	139.2
		25	82.49%	77.21%	0.79	20.47	7.66	186.2
		40	88.08%	83.77%	0.94	12.54	8.84	127.1
5	30	15	82.47%	83.13%	0.76	10.66	23.61	302.5
		25	81.82%	81.09%	0.75	10.15	22.37	293.7
		40	86.21%	84.23%	0.76	10.17	21.44	294.5
10	30	15	83.23%	79.79%	0.81	14.22	22.56	404.8
		25	83.66%	79.22%	0.81	16.10	23.59	403.8
		40	90.40%	88.09%	0.81	12.28	24.07	354.8
5	50	15	83.70%	85.24%	0.68	10.37	37.37	515.4
		25	87.31%	84.56%	0.92	10.68	35.25	497.7
		40	89.40%	84.72%	0.68	10.87	36.73	474.1
10	50	15	83.23%	81.93%	1.17	13.83	48.88	646.8
		25	82.73%	84.40%	0.83	13.52	39.46	663.1
		40	88.52%	85.57%	0.75	13.13	39.26	614.6
5	Without Priority Queue	15	83.70%	85.24%	0.74	10.02	65.65	991.8
		25	82.75%	81.68%	0.90	9.95	70.87	990.0
		40	86.52%	85.45%	0.77	9.66	72.25	956.5
10	Without Priority Queue	15	83.47%	81.93%	0.78	13.15	67.92	1266
		25	90.54%	79.79%	0.80	13.17	72.29	1278
		40	90.43%	88.90%	0.81	12.73	74.44	1266

configurations, there was not a clear trend indicating that larger or smaller initial solutions were consistently superior in terms of F1-Score. In some cases, an initial solution size of 5 showed better results, while in others, a size of 10 was more effective.

Additionally, the RCL size did not have a significant impact on processing time. Configurations with RCL sizes of 15, 25, and 40 showed similar processing times. However, it was observed that in most cases, a larger RCL size (i.e., 40) leads to a higher F1-Score.

It is important to note that the runtime for the construction phase remains constant across different configurations. This is due to the fixed number of iterations set for this phase, which does not depend on other parameters. In contrast, the local search phase shows a significant variation in runtime, directly influenced by the size of the priority queue: smaller priority queues lead to a quicker local search. This trend confirms that reducing the number of solutions to be evaluated can improve the algorithm's efficiency without compromising the quality of the final solutions.

Regarding the effect of the priority queue on the algorithm's performance, the results indicate that the presence of a priority queue with 10 solutions has a minimal impact on the F1-Score. Specifically, for KNN, the configuration with 30 solutions in the priority queue achieved an F1-Score of 88.09%, only 0.81% lower than the highest recorded F1-Score of 88.90% (in the baseline, without a priority queue), as underlined in Table I. Moreover, this configuration led to a time reduction of 911 seconds, or approximately 72%, in the local search phase compared to the configuration without a priority queue.

A similar trend is observed with Naive Bayes. Implementing a priority queue of 10 solutions decreased the processing time from 72.29 seconds to 8.84 seconds, representing a substantial time reduction of over 90%, while the F1-Score suffered a minor decrease from 90.54% to 88.08% (underlined in Table I). This configuration reflects a significant improvement in operational efficiency, with a relatively small loss in F1-Score. Additionally, setting the priority queue to a size of 30 achieved an F1-Score of 90.40% with a processing time of 24.07 seconds in the local search. These findings reinforce the feasibility of applying a limited priority queue as a strategic approach to increase processing efficiency without negatively affecting accuracy significantly.

When comparing the two algorithms, NB presents shorter execution times in the local search phase compared to KNN, due to its lower computational complexity. This difference becomes more evident as we increase the size of the priority queue, showcasing how the choice of algorithm can influence the efficiency of the feature selection process.

Finally, the results demonstrate that setting up an optimized priority queue can be crucial for improving the efficiency of the local search in GRASPQ-FS without harming performance. Overall, for both KNN and Naive Bayes, a priority queue with 10 solutions emerges as an effective choice.

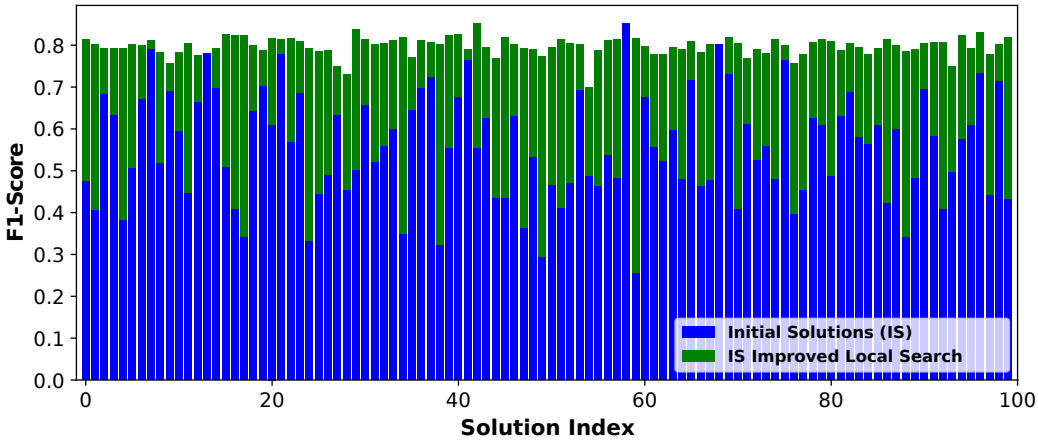


Fig. 2. GRASP-FS, no Priority Queue. All Solutions sent to Local Search

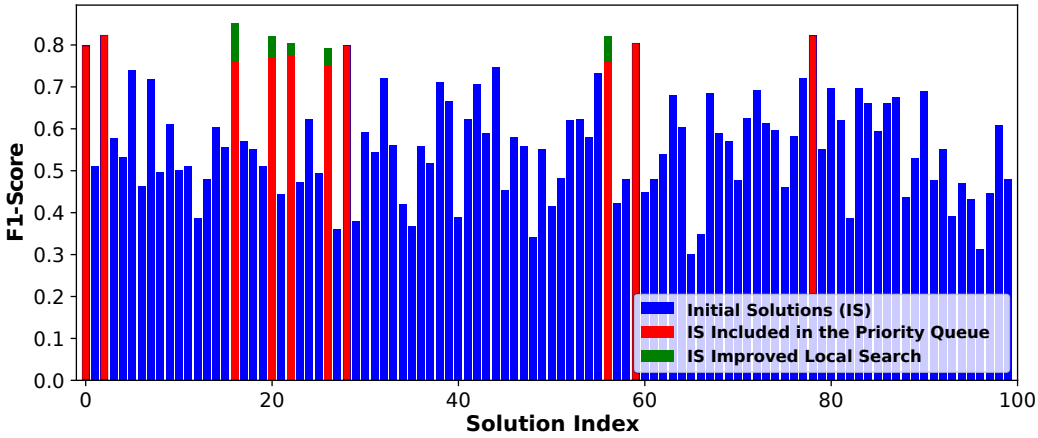


Fig. 3. GRASPQ-FS, only Priority Queue solutions sent to Local Search

A. Analysis of Initial Solutions Submitted to Local Search

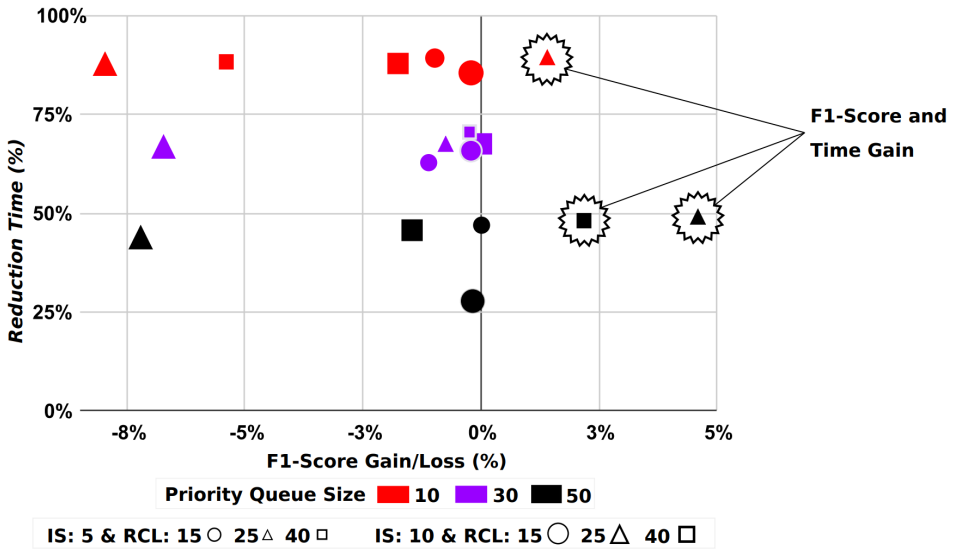
Figures 2 and 3 illustrate the initial solutions processed by the local search phase, contrasting the baseline GRASP-FS (no priority queue) with GRASPQ-FS (with a priority queue of size 10). The experiments were conducted with subsets of 5 initial solutions, using an RCL of 15 features and the KNN algorithm for classification.

We can observe that our priority queue did not compromise the quality of the final solutions, as in both figures the best solution reached an F1-Score of 85.24%. However, the resource savings achieved are evident: there are only 10 red bars in GRASPQ-FS vs 100 blue bars in GRASP-FS, while the green bars (i.e., initial solutions that were improved by the local search) in the two algorithms achieve the same result. This selection introduced by the priority queue allowed for significant optimization of time, which is appropriate in scenarios with limited resources.

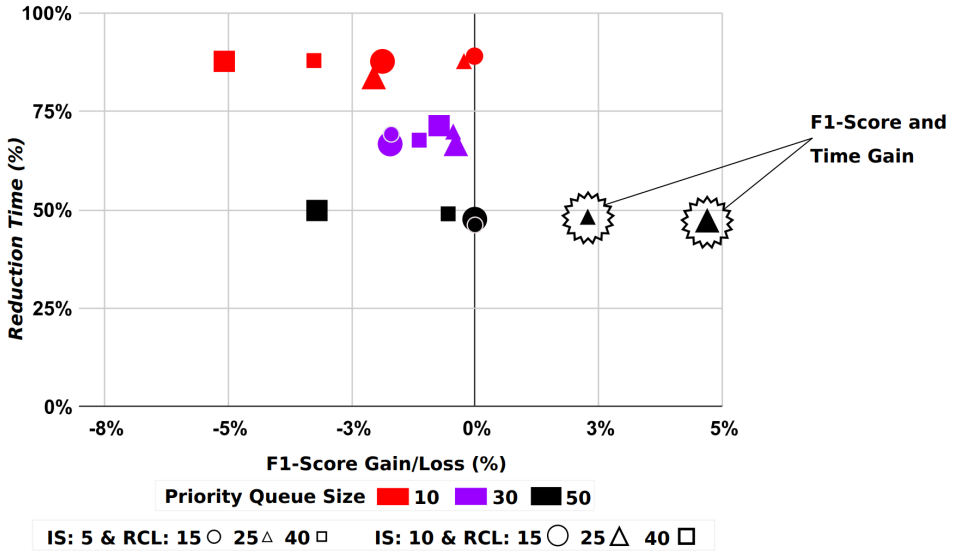
These results reiterate the efficacy and efficiency of the priority queue in the FS process. By focusing on the most efficient solutions from the start, it is possible to achieve a high F1-Score without the need to process an excessive volume of potential solutions.

B. Comparative Performance Analysis between GRASPQ-FS and GRASP-FS

Figures 4(a) and 4(b) present (for NB and KNN, respectively) a comparison between the results of GRASPQ-FS with different parameters and those of the traditional GRASP-FS. Each shape on the charts represents a complete execution of GRASPQ-FS, considering certain sizes of the priority queue, RCL, and initial solution. The value on the vertical axis represents how much shorter the execution time of GRASPQ-FS was compared to an analogous execution (i.e., the same sizes of RCL and initial solution) of GRASP-FS. The horizontal axis shows how much worse or better (in terms of F1 Score) the solution found in that execution of GRASPQ-FS was compared to the "best solution" of the analogous execution of traditional GRASP-FS. These "best solutions" refer to the set of selected features that, in a real scenario, should be used as input for an IDS. We use different shapes and sizes are used to represent combinations of initial solution (IS) sizes and RCL values: small circles, rectangles, and triangles denote an IS size of 5 with RCL sizes of 15, 25, and 40, respectively; similarly, larger shapes correspond to an IS size of 10 with the same RCL sizes. Moreover, we use different colors to highlight



(a) NB: Naive Bayes Algorithm



(b) KNN: K Nearest Neighbors Algorithm

Fig. 4. Comparison of Time Reduction and F1-Score Gain/Loss between GRASPQ-FS and GRASP-FS with Different Priority Queue Configurations, Initial Solutions (IS), and RCL, for NB (top) and KNN (bottom) algorithms

executions with different priority queue sizes (red shows queue = 10, purple shows queue = 30, and black shows queue = 50).

Figure 4(a) shows the results for the NB algorithm. In summary, a priority queue of size of 10 for NB provided an average execution time reduction of 90% compared to traditional GRASP-FS. In particular, when priority queue of 10 is combined with an initial solution with 5 features (IS=5) and an RCL size of 25 (RCL=25), GRASPQ-FS also improved F1-Score by 1.62% (small red triangle). This improvement in efficiency and effectiveness is indicative of the utility of applying a moderately sized priority queue in this specific context.

Also, it can be observed in Figure 4(a) that a larger priority queue, with 50 solutions, results in a reduction of execution time by 25%-50%; specifically, with IS=5 and RCL=25, we see an increase in F1-Score of up to 4.56% (small black triangle). This result suggests that although execution time may increase with larger priority queues compared with smaller ones, the quality of the feature selection, as indicated by the F1-Score, can also improve. This is believed to be due to the method's randomness in generating the initial solutions, which is mitigated with larger priority queues.

Figure 4(b) presents the results for the KNN algorithm. Specifically, the priority queue of size 10 showed a reduction of execution time of about 90%, maintaining the F1-Score stable compared to the baseline GRASP-FS, when using an initial solution size of 5 and an RCL size of 15. Similarly, as observed with NB, increasing the queue size to 50 and RCL

of 25 in KNN resulted in F1-Score improvements of 2.88% and 4.61% and time reductions of 50% and 48%, for IS = 5 and IS = 10, respectively. These results again corroborate that significant optimization of processing time is achievable without compromising the model's accuracy.

Finally, these results show the beneficial effect of varied priority queues in the feature selection process using GRASPQ-FS, providing insights into how fine-tuning parameters can be used to improve the performance and accuracy of machine learning algorithms.

VII. Conclusion

In this work, we introduced the GRASPQ-FS, a simple extension of the state-of-the-art metaheuristic GRASP-FS, incorporating an efficient priority queue into the construction phase. Our proposal has proven to be extremely effective, offering substantial processing gains. Not only did we achieve time savings of up to 90%, but the implementation of a priority queue of size 50 also enhanced the F1-Score by 4.61% while reducing the computation time by approximately 50% compared to the GRASP-FS approach. This advancement is particularly significant for real-time Intrusion Detection Systems (IDSs), where speed and efficiency in processing large volumes of data are essential. The efficiency achieved with our GRASPQ-FS ensures a more agile and effective response against cyber threats, thereby strengthening digital security.

Beyond its significant impact on IDSs, the applicability of GRASPQ-FS extends to other domains that require quick responses, such as financial analysis, real-time medical diagnostics, and other critical scenarios. In all these contexts, the ability to process information quickly, without compromising the quality of solutions, is crucial for efficient and timely decisions.

In future work, we intend to integrate techniques from Explainable Artificial Intelligence (XAI), such as the SHAP methodology [20], to evaluate the selected features, replacing the use of the Information Gain (IG) algorithm in forming the Restricted Candidate List (RCL). Additionally, we plan to explore various datasets available in the literature [21] to further evaluate GRASPQ-FS. For example, we will consider using the Westermo [22] and ICS-Flow [23] datasets.

References

[1] J. A. Kerr, "Assessing Russian Cyber and Information Warfare in Ukraine: Expectations, Realities, and Lessons," Center for Naval Analyses (CNA), Tech. Rep., 2024. [Online]. Available: https://www.cna.org/reports/2023/11/assessing-russian-cyber-and-information-warfare-in-ukraine

[2] CrowdStrike Inc., "CrowdStrike 2024 Global Threat Report," Tech. Rep., 2024, accessed: August 2024. [Online]. Available: https://go.crowdstrike.com/rs/281-OBQ-266/images/GlobalThreatReport2024.pdf

[3] H. Attou, M. Mohy-eddine, A. Guezzaz, S. Benkirane, M. Azrour, A. Alabdultif, and N. Almusallam, "Towards an Intelligent Intrusion Detection System to Detect Malicious Activities in Cloud Computing," *Applied Sciences*, vol. 13, no. 17, p. 9588, 2023.

[4] D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R. M. Mohammad, "Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT," *Journal of Sensor and Actuator Networks*, vol. 12, no. 2, p. 29, 2023.

[5] S. E. Quincozes, D. Passos, C. Albuquerque, L. S. Ochi, and D. Mossé, "GRASP-Based Feature Selection for Intrusion Detection in CPS Perception Layer," in *2020 4th Conference on cloud and internet of things (CIoT)*. IEEE, 2020, pp. 41–48.

[6] T. A. Feo and M. G. Resende, "Greedy Randomized Adaptive Search Procedures," *Journal of global optimization*, vol. 6, pp. 109–133, 1995.

[7] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, 2009.

[8] L. S. Almeida, F. Goerlandt, R. Pelot, and K. Sörensen, "A Greedy Randomized Adaptive Search Procedure (GRASP) for the multi-vehicle prize collecting arc routing for connectivity problem," *Computers & Operations Research*, vol. 143, p. 105804, 2022.

[9] A. Thakkar and R. Lohiya, "A survey on Intrusion Detection System: Feature Selection, Model, Performance Measures, Application Perspective, Challenges, and Future Research Directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2022.

[10] S. E. Quincozes, D. Mossé, D. Passos, C. Albuquerque, L. S. Ochi, and V. F. dos Santos, "On the performance of grasp-based feature selection for cps intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 614–626, 2022.

[11] M. A. Esseghir, "Effective wrapper-filter hybridization through GRASP schemata," in *Feature Selection in Data Mining*, 2010, pp. 45–54.

[12] P. Bermejo, J. A. Gámez, and J. M. Puerta, "A GRASP algorithm for fast hybrid (filter-wrapper) feature subset selection in high-dimensional datasets," *Pattern Recognition Letters*, vol. 32, no. 5, pp. 701–711, 2011.

[13] M. Moshki, P. Kabiri, and A. Mohebalhojeh, "Scalable feature selection in high-dimensional data based on GRASP," *Applied Artificial Intelligence*, vol. 29, no. 3, pp. 283–296, 2015.

[14] J. F. Diez-Pastor, C. García-Osorio, J. J. Rodríguez, and A. Bustillo, "GRASP forest: a new ensemble method for trees," in *International Workshop on Multiple Classifier Systems*. Springer, 2011, pp. 66–75.

[15] J.-F. Díez-Pastor, C. García-Osorio, and J. J. Rodríguez, "Tree ensemble construction using a grasp-based heuristic and annealed randomness," *Information Fusion*, vol. 20, pp. 189–202, 2014.

[16] N. K. Kanakarajan and K. Muniasamy, "Improving the accuracy of intrusion detection using gar-forest with feature selection," in *Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA)*. Springer, 2016, pp. 539–547.

[17] E. F. Silva, N. Naves, S. E. Quincozes, V. E. Quincozes, J. F. Kazienko, and O. Cheikhrouhou, "GDLS-FS: Scaling Feature Selection for Intrusion Detection with GRASP-FS and Distributed Local Search," in *International Conference on Advanced Information Networking and Applications*. Springer, 2023, pp. 199–210.

[18] S. E. Quincozes, C. Albuquerque, D. Passos, and D. Mossé, "ERENO: A Framework for Generating Realistic IEC–61850 Intrusion Detection Datasets for Smart Grids," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 3851–3865, 2024.

[19] R. E. Mackiewicz, "Overview of IEC 61850 and Benefits," in *2006 IEEE Power Engineering Society General Meeting*. IEEE, 2006, pp. 8–pp.

[20] V. E. Quincozes, S. E. Quincozes, J. F. Kazienko, S. Gama, O. Cheikhrouhou, and A. Koubaa, "A survey on IoT application layer protocols, security challenges, and the role of explainable AI in IoT (XAIoT)," *International Journal of Information Security*, vol. 23, no. 3, pp. 1975–2002, 2024.

[21] V. F. Santos, C. Albuquerque, D. Passos, S. E. Quincozes, and D. Mossé, "Assessing Machine Learning Techniques for Intrusion Detection in Cyber-Physical Systems," *Energies*, vol. 16, no. 16, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/16/6058

[22] P. E. Strandberg, D. Söderman, A. Dehlaghi-Ghadim, M. Leon, T. Markovic, S. Punnekkat, M. H. Moghadam, and D. Buffoni, "The Westermo network traffic data set," *Data in Brief*, vol. 50, p. 109512, 2023.

[23] A. Dehlaghi-Ghadim, M. H. Moghadam, A. Balador, and H. Hansson, "Anomaly Detection Dataset for Industrial Control Systems," *IEEE Access*, vol. 11, pp. 107 982–107 996, 2023.