International Journal Scientific and Profesional (IJ-ChiProf)
Vol 4 Issue 2 March-May 2025, pp: 509-515
ISSN: 2829-2618 (Online)
DOI: https://doi.org/10.56988/chiprof.v4i2.85

Implementation of Greedy Algorithm in Pattern Matching for Text Recognition System

Risky Amelia[1], Tata Sutabri[2],
[1,2]Magister of Informatics Engineering, Universitas Bina Darma, Indonesia
Email: riskiamelia918@gmail.com[1], tata.sutabri@gmail.com[2]

Article Info
Article history:
Received April 21, 2025
Revised April 21, 2025
Accepted April 23, 2025

Keywords:
Greedy algorithm,
Text Recognition,
Pattern Matching

ABSTRACT
The Greedy pattern-matching algorithm is a phrase pattern-matching method that works by selecting the optimal solution at each step without backtracking. This approach is applied in text recognition systems for keyword search, natural language processing, and automatic text filters. This research analyzes the performance of the algorithm through computational experiments and literature review by evaluating the efficiency of execution time, number of character comparisons, and matching success rate. The results show that the algorithm offers high speed in pattern matching, especially on large datasets, as it is able to optimally shift the search index. However, its accuracy decreases when handling complex patterns or phrases that have many similarities. By combining this algorithm with heuristics or data preprocessing techniques, its drawbacks can be minimized, thus remaining an effective solution in text recognition systems that require fast and real-time processing.

Corresponding Author:
Tata Sutabri,
Magister of Informatics Engineering,
Universitas Bina Darma, Indonesia
Email: tata.sutabri@gmail.com

1. INTRODUCTION

Phrase pattern matching in text recognition systems is a fundamental element in many applications, including information retrieval, natural language processing, and character recognition. Efficiency in pattern matching is critical to improving the accuracy and speed of text processing, especially in environments with large volumes of data. One prominent approach to solving this problem is the greedy algorithm, which offers a faster computational method compared to other more complex approaches. By choosing the locally best solution at each step, greedy algorithms are able to reduce the computational burden and speed up the phrase pattern-matching process in text recognition systems.

A greedy algorithm operates by making the best decision at each step based on the information available at that time, with the expectation that this set of decisions will lead to an overall optimal solution. In the context of pattern matching, these algorithms can be used to identify the frequency of pattern occurrence in text and evaluate local refinement-based matches. Research by [1] shows that the combination of greedy methods with other techniques, such as neighborhood-based search, can provide better results in text-processing tasks.

One application of greedy algorithms in pattern matching is in signal compression techniques, such as Orthogonal Matching Pursuit (OMP). This technique allows the reconstruction of sparse signals from noisy measurements with high efficiency [2]. [3] also highlighted that greedy algorithms such as OMP and CoSaMP are very effective in recovering signals that have sparsity properties, demonstrating the superiority of these methods in dealing with large-scale problems.

In text recognition systems, greedy algorithms can be used to improve pattern-matching efficiency by reducing time complexity. [4] showed that the application of greedy algorithms in text recognition systems can help filter and recognize patterns better, especially in noisy text data. This is particularly relevant in applications such as face recognition, where fast and accurate matching is needed to handle variation and occlusion. [5] developed a greedy-based convolution algorithm capable of identifying sources in multidimensional data. The algorithm demonstrated effectiveness in image and signal matching, making it an attractive option for application in phrase pattern matching

in text. The main advantages of greedy algorithms lie in their efficiency and simplicity in implementation. [6] notes that greedy algorithms are often used in streaming algorithms due to their ability to handle rapidly changing data. However, this method also has limitations, especially in guaranteeing a global optimal solution. A combination approach can be used. [7] showed that greedy algorithms trained with good parameters can provide more optimal results than untrained methods.

In practical applications, the Boyer-Moore algorithm is one example of a pattern-matching algorithm that relies on greedy strategies to improve the efficiency of text search [8]. Implementations that integrate these approaches can speed up the matching process by utilizing the information available during each search step. In more complex problems, such as multinational pattern matching, the use of greedy algorithms can provide an edge in information retrieval from large-scale text. [9] showed that although greedy algorithms do not always guarantee an optimal solution in every scenario, their ability to reduce search time and improve matching accuracy still makes them attractive for application in text recognition systems.

Compared to brute force methods, greedy algorithms offer advantages in processing speed and resource usage efficiency. [10] compared the efficiency of the Boyer-Moore algorithm with the brute force approach and found that the greedy strategy resulted in faster execution time with lower computational load. In the development of the phrase pattern matching module, the use of efficient data structures is essential to support the greedy algorithm. Structures such as suffix trees and prefix-based matching functions allow quick access to information during the matching process. [11] notes that the application of greedy algorithms in this context can save time and computational resources significantly, making it an appropriate choice for the development of efficient text recognition systems.

2. RESEARCH METHOD

2.1 Research Strategies

This research uses computational experimentation methods to analyze the performance of the Greedy pattern-matching algorithm in the phrase pattern-matching system in text. This approach aims to evaluate the efficiency of the algorithm in terms of execution time, number of character comparisons, and matching success rate in various text scenarios. In addition to supporting the findings, this research also uses the literature review method to analyze the implementation of the Greedy algorithm in phrase pattern matching in text recognition systems. With data synchronization, we have to be careful about whether our data is synchronized with social media or internet media [12]. Kajian literatur dilakukan dengan menelaah berbagai sumber seperti jurnal ilmiah, buku, serta penelitian terdahulu yang membahas teknik pencocokan pola dan optimasi algoritma Greedy.

2.2 Data Collection Methods

According to [13], one of the important factors in the construction or development of information systems is understanding the existing system and its problems. Some commonly used techniques include interview techniques, questionnaire techniques, direct observation techniques, and sampling techniques. According to [14], data are facts that will be made useful information. The dataset used in this research contains a long text and various phrase patterns that you want to match in the text.

2.3 Greedy Algorithm

The greedy algorithm is one method often used in the development of optimization algorithms. Its basic concept comes from a local approach, where decisions are made based on the best available option at each step with the hope that the local solution obtained will produce an optimal global solution. This approach makes greedy algorithms very useful in various optimization problems, especially those that require quick decisions without taking into account long-term consequences [15].

3 RESULTS AND DISCUSSION

3.1 Searchable Phrase Patterns

To facilitate pattern matching, there are several phrase patterns to look for:

a. *"text recognition"*
b. *"pattern matching algorithm"*
c. *"This technology enables users."*
d. *"text format"*

The Source Text used in this case is: *"Artificial intelligence-based text recognition systems are growing rapidly. This technology allows users to search for specific phrases in documents quickly and accurately. By using efficient pattern matching algorithms, these systems can recognize word patterns in various text formats."*

International Journal Scientific and Profesional (IJ-ChiProf)
Vol 4 Issue 2 March-May 2025, pp: 509-515
ISSN: 2829-2618 (Online)
DOI: https://doi.org/10.56988/chiprof.v4i2.85

3.2 Matching Result

Based on text analysis of product descriptions shows that the use of more casual and personalized language is increasing from year to year [16]. The following are the results of matching with the Greedy Pattern Matching Algorithm.

Table 1. Matching Results with Greedy Pattern Matching Algorithm

Phrase Pattern	Index of Occurrence in the Text
"text recognition"	7
"pattern matching algorithm"	166
"This technology allows users."	73
"text format"	225

The Greedy Pattern Matching algorithm has the advantage of speed and efficiency as it immediately shifts the index as far as possible without backtracking. With this approach, searching for phrases in text is done in a relatively constant time for each match, making it suitable for large documents. In addition, the matching accuracy is quite high as the searched pattern is found in the right index without any errors in the matching process. This makes it effective for text recognition systems that require fast and accurate search of keywords or phrases.

However, this algorithm has its limitations, especially in handling word variations. If there is a change in word form, such as the addition or removal of a word in the pattern, the algorithm cannot find it because it works with exact matching. In addition, this method is less optimal for texts with many repetitions of similar patterns, as it will still process the entire text without considering further optimization as done by other pattern-matching algorithms, such as Knuth-Morris-Pratt (KMP) or Boyer-Moore.

3.3 Simulation of Calculation

The following is a simulation of the calculation using the basic parameters:

a. T = "Artificial intelligence-based text recognition systems are growing rapidly." (Length n = 76 characters)
b. P = "text recognition" (Length m = 16 characters)
c. i = initial index in the text
d. j = index in the pattern

There is an initial comparison as follows:

a. Starting from i = 0, compare the first character of the pattern with the 0th character of the text.
b. If there is no match, shift i to the right (i = i + 1) and repeat.

The matches found:

At i = 7, the first character of the pattern matches the text

Table 2. Table of Matches

Index (i+j)	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**
Characters in Text	p	e	n	g	e	n	a	l	a	n	(space)	t	e	k	s
Characters in Pattern	p	e	n	g	e	n	a	l	a	n	(space)	t	e	k	s
Match?	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Since all patterns match, index seven is added to the list of matching results.

a. In the best case, use O(n/m), as the patterns match immediately.
b. In the Average Case: O(n), the patterns must be matched sequentially.
c. In the Worst Case: O(nm), if every character almost matches but fails at the last character.

This simulation shows that the Greedy Pattern Matching algorithm works efficiently for patterns that can be found early in the text. However, for texts that have many variations of words or patterns that are similar but not the same, this algorithm can be less optimal than other methods, such as KMP or Boyer-Moore.

Vol 4 Issue 2 2025, pp: 509-515
ISSN: 2829-2618 (Online)

3.4 Discussion

a. Pattern Matching Method in Text Processing

Greedy algorithms are widely used in various pattern-matching applications, especially in text recognition systems. This method attracts attention due to its simplicity in choosing the best solution at each step without having to see the long-term impact. Despite its simplicity, this approach has proven to be quite efficient in saving time and resources [17]; [18]. In pattern matching, greedy algorithms work by selecting the best step in each iteration based on locally optimal criteria. This makes it particularly useful in text recognition systems, where search speed is a major factor. For example, Purnama et al.'s research shows that greedy algorithms can be used in shortest-path selection to optimize resource usage [17]. [18] It also proved its effectiveness in finding the shortest travel route in Samarinda, which shows how this method can be applied in daily life.

On the other hand, in the field of text processing, pattern matching is often combined with different algorithms to improve efficiency. For example, Boyer-Moore and Brute Force algorithms have long been used for faster string matching [19]. Firmansyah et al. emphasized that pattern matching is not just about matching word by word but should also take into account the overall context [19]. In other words, the development of greedy algorithms can be combined with different techniques to get more accurate results. The greedy algorithm can also be integrated with various preprocessing methods. Filcha and Hayaty, in their research, discuss how the Rabin-Karp algorithm is used to detect plagiarism more quickly and accurately [20]. In the process, techniques such as text cleaning, tokenization, and stopword reduction greatly help improve the efficiency of greedy-based algorithms. In data analysis, greedy algorithms play an important role, especially in finding association patterns in large datasets. For example, the FP-Growth algorithm that is often used in data mining helps companies find sales patterns that can be used for marketing strategies [21]; [22]. Nofianti et al. also showed how this algorithm can be used in data analysis for strategic decision-making [23]

However, there are challenges in applying greedy algorithms, especially in ensuring that the results are optimal overall, not just at the local level. This is important in complex problems with many variables and rapidly changing conditions, such as pattern recognition in text [24]. Therefore, greedy algorithms need to be combined with other methods to optimize the results. For example, the KMP algorithm discussed by Cakrawijaya and Kriswantara can be used for more complex text searches [25]. Although the greedy algorithm has many advantages, future research needs to focus on how to overcome its limitations. In very large or unstructured texts, integration with other methods could be a solution to improve the efficiency of pattern matching [26]; [27]. That way, the application of greedy algorithms in text recognition systems will be more effective and can provide greater benefits for researchers and practitioners in this field.

b. Pros and Cons of the Greedy Approach

The greedy approach is a widely used method in developing algorithms for various problems in computer science and mathematics. It operates by making locally optimal decisions in each step, with the expectation that these decisions will result in a globally optimal solution. However, while this approach has some advantages, it also has disadvantages that can affect its effectiveness in solving complex problems. In this talk, I will discuss the advantages and disadvantages of the greedy approach using a number of relevant references.

The main advantages of the greedy approach are its time efficiency and ease of implementation. For example, greedy algorithms often have lower computational complexity compared to more comprehensive optimization algorithms. In research on path planning or network design, many greedy algorithms are implemented because they are fast and easy to apply to real problems, although they may sacrifice the optimality of the results. For example, Li et al. showed that greedy algorithms for routing in vehicular networks can be executed quickly despite the risk to overall performance in complex scenarios[28]. This time efficiency is particularly important in the context of software development and real-time applications that require fast response.

When discussing the advantages of the greedy approach, it is important to consider its application in certain contexts. For example, in research on scheduling in the context of cloud computing, Rakrouki and Alharbe noted that although the greedy approach can quickly solve the scheduling task, it tends not to take into account the total consumption of resources, which can lead to non-optimal placement of virtual machines [29]. This shows that in some applications, greed often produces good solutions quickly but not always the best. The presence of these fast solutions provides a great advantage, especially in scenarios with tight time limits.

Despite these advantages, the main drawback of the greedy approach is its limited ability to find globally optimal solutions. It is easily trapped in local optimum conditions and may make premature decisions that cannot be changed in the next step. This is reinforced by research showing that greedy algorithms tend to give suboptimal results in scenarios where the problem structure does not support optimal local decisions [30]. In addition, this approach does not consider the impact of the decision taken on the next step, making it vulnerable to unfavorable results [31]

As a concrete example, in the study of the Huffman coding problem, although the greedy algorithm is recognized to provide an efficient solution for constructing the complete coding tree, it does not consider the entire coding context thoroughly, which could result in a suboptimal solution when viewed from a long-term point of view [32]. The greedy approach may work for current needs but often neglects the quality of a better long-term solution, exposing significant limitations of this strategy in certain situations.

In the context of algorithm complexity, some studies suggest that the speed of greedy algorithms can come at the cost of trade-offs on accuracy or optimality [33]. While this is not always the case, in many cases, the use of more complex, albeit slower, algorithms may be necessary to achieve better results. This approach is faced with challenges when the algorithm requires more strategic decisions, for example, when it has to balance between exploration and exploitation in a given context.

From an application perspective, as shown in research on path planning for ships avoiding moving obstacles, problems arise when greedy algorithms cannot consider all variables in decision-making, which can lead to poor results in dynamic situations [34]. These limitations remind us that while greedy algorithms can be useful tools, judicious use and proper understanding of their limitations are required to exploit them effectively.

Along with new developments in machine learning and optimization, greedy algorithms continue to gain attention, especially for applications in optimization and mapping, where traditional methods do not always offer adequate solutions. For example, the computation algorithm developed by Tang et al. to predict dynamic resource requirements shows that although greedy algorithms can provide efficiency in certain contexts, it is crucial to combine them with more advanced techniques to enrich the results and quality of the resulting solutions [35].

4. CONCLUSION

Based on the simulation and analysis of the Greedy Pattern Matching algorithm, it can be concluded that this method offers high efficiency in pattern matching as it does not perform backtracking. By immediately shifting the search index as far as possible after finding a match, the algorithm is able to speed up the search process in long texts. This makes it suitable for applications such as artificial intelligence-based text search systems or pattern recognition in large documents. Also, in the literature study, the use of a Greedy algorithm in phrase pattern matching in text recognition systems offers a fast and efficient way to find specific words or phrases in a document. This algorithm works by taking the best decision at each step without considering the previous step, thus speeding up the matching process compared to other, more complicated methods. In practice, this approach is particularly helpful in systems that require high speed, such as keyword searches or automatic text filters. Greedy algorithms also have their downsides, especially when it comes to handling more complex patterns or texts that have many similarities. Since it only focuses on the best solution at each step without considering the long-term effects, the matching results can be less accurate. For example, in text that contains many similar words, it may miss more suitable patterns than other methods such as Knuth-Morris-Pratt or Boyer-Moore, which are more careful in fitting the pattern to the overall text.

Nonetheless, the Greedy algorithm can still be a good choice if used in appropriate situations. With minor adjustments, such as adding pre-filtering techniques or additional heuristics, it can be more accurate without sacrificing speed. Therefore, in automated search or text detection systems, the Greedy approach is still relevant and can be an effective solution, especially when combined with other strategies that can improve the accuracy of matching results.

ACKNOWLEDGEMENTS

The author would like to express sincere gratitude to all parties who have supported the completion of this research. The author also wishes to thank the academic supervisors and lecturers at Universitas Bina Darma for their continuous guidance and constructive feedback throughout the research process. Appreciation is also given to colleagues and fellow students who have shared their insights and motivated during the preparation of this research. Finally, the author's deepest thanks go to her family for their endless support, encouragement, and prayers. This work would not have been possible without the contribution of all the above individuals and institutions.

REFERENCES

[1] I. Akhmetov, R. Mussabayev, and A. Gelbukh, “Reaching for upper bound ROUGE score of extractive summarization methods,” *PeerJ Comput Sci*, vol. 8, p. e1103, Sep. 2022, doi: 10.7717/peerj-cs.1103.

[2] S.-W. PARK, J. PARK, and B. C. JUNG, “On the Sparse Signal Recovery with Parallel Orthogonal Matching Pursuit,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E96.A, no. 12, pp. 2728–2730, 2013, doi: 10.1587/transfun.E96.A.2728.

[3] K. Gupta and A. Majumdar, “Greedy Algorithms for Non-linear Sparse Recovery,” 2016, pp. 99–108. doi: 10.1007/978-81-322-2625-3_9.

[4] A. B. Doumi, F. E. Abualadas, M. M. A. Shquier, M. Asassfeh, B. M. Elzaghmouri, and K. M. Alhawity, "New Extracted Features to Recognize Faces Effected by Occlusions and Common Variations," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, pp. 211–239, Oct. 2024, doi: 10.37934/araset.57.1.211239.
[5] G. Bohner and M. Sahani, "Convolutional higher order matching pursuit," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, Sep. 2016, pp. 1–6. doi: 10.1109/MLSP.2016.7738847.
[6] L. Khalil and C. Konrad, *Constructing large matchings via query access to a maximal matching oracle*. 2020.
[7] K. Voulgaris, M. Davies, and M. Yaghoobi, *Deepmp for non-negative sparse decomposition*. 2020.
[8] D. A. Tarigan, A. O. Buaton, B. Briyandana, E. R. Safitri, and R. Rosnelly, "Analysis of String Matching Application on Serial Number Using Boyer Moore Algorithm," *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 6, no. 1, pp. 237–246, Jan. 2024, doi: 10.47709/cnahpc.v6i1.3410.
[9] D. Hendrian, Y. Ueki, K. Narisawa, R. Yoshinaka, and A. Shinohara, "Permuted Pattern Matching Algorithms on Multi-Track Strings," *Algorithms*, vol. 12, no. 4, p. 73, Apr. 2019, doi: 10.3390/a12040073.
[10] C. Irawan and M. R. Pratama, "Perbandingan Algoritma Boyer-Moore dan Brute Force pada Pencarian Kamus Besar Bahasa Indonesia Berbasis Android," *BIOS : Jurnal Teknologi Informasi dan Rekayasa Komputer*, vol. 1, no. 2, pp. 54–60, Feb. 2021, doi: 10.37148/bios.v1i2.13.
[11] A. Kostanyan, "Fuzzy String Matching Using a Prefix Table," *Mathematical Problems of Computer Science*, pp. 116–121, Dec. 2020, doi: 10.51408/1963-0065.
[12] Y. A. Putra and T. Sutabri, "ANALISIS PENYADAPAN PADA APLIKASI WHATSAPP DENGAN MENGGUNAKAN METODE SINKRONISASI DATA," *Blantika: Multidisciplinary Journal*, vol. 1, no. 2, pp. 132–141, Feb. 2023, doi: 10.57096/blantika.v1i2.8.
[13] T. Sutabri, *Analisis Sistem Informasi*. Yogyakarta: Andi, 2012.
[14] T. Sutabri, *Pengantar Teknologi Informasi*. Yogyakarta: Andi, 2014.
[15] E. W. Pratiwi and Mhd. Z. Siambaton, "Aplikasi Penjadwalan Dokter Pada Rumah Sakit Umum Kota Pinang dengan Menggunakan Algoritma Greedy," *Hello World Jurnal Ilmu Komputer*, vol. 1, no. 1, pp. 1–9, Apr. 2022, doi: 10.56211/helloworld.v1i1.4.
[16] R. Amelia and S. Warianti, "ANALISIS STRATEGI MENINGKATKAN VIRALITAS KONTEN PADA SHOPEE LIVE MENGGUNAKAN NATURAL LANGUAGE PROCESSING (NLP)," 2024.
[17] R. D. Saktia Purnama *et al.*, "IMPLEMENTASI PENGGUNAAN ALGORITMA GREEDY BEST FIRST SEARCH UNTUK MENENTUKAN RUTE TERPENDEK DARI CILACAP KE YOGYAKARTA," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, Apr. 2024, doi: 10.23960/jitet.v12i2.4068.
[18] F. Nova Arviantino, W. Gata, L. Kurniawati, Y. A. Setiawan, and D. Priansyah, "Penerapan Algoritma Greedy Dalam Pencarian Jalur Terpendek Pada Masjid–Masjid Di Kota Samarinda," *METIK JURNAL*, vol. 5, no. 1, pp. 8–11, Jun. 2021, doi: 10.47002/metik.v5i1.188.
[19] F. Firmansyah, Fauziah, and N. Hayati, "ANALISIS PERBANDINGAN DAN IMPLEMENTASI STRING MATCHING DAN SQL QUERY PADA SISTEM INFORMASI PERSEDIAAN OBAT BERBASIS WEB APOTEK ERHA FARMA," *Jurnal Ilmiah Teknologi dan Rekayasa*, vol. 27, no. 2, pp. 154–168, Aug. 2022, doi: 10.35760/tr.2022.v27i2.7079.
[20] A. Filcha and M. Hayaty, "Implementasi Algoritma Rabin-Karp untuk Pendeteksi Plagiarisme pada Dokumen Tugas Mahasiswa," *JUITA : Jurnal Informatika*, vol. 7, no. 1, p. 25, May 2019, doi: 10.30595/juita.v7i1.4063.
[21] F. U. Faruq and L. Bachtiar, "PENERAPAN DATA MINING ALGORITMA FP-GROWTH UNTUK MENENTUKAN REKOMENDASI PENJUALAN TANAMAN HIDROPONIK DI MENTAYA PONIK," *ZONAsi: Jurnal Sistem Informasi*, vol. 5, no. 3, pp. 441–451, Sep. 2023, doi: 10.31849/zn.v5i3.15169.
[22] A. N. Ridho, A. P. A. Masa, and P. P. Widagdo, "IMPLEMENTASI MARKET BASKET ANALYSIS PADA DATA PENJUALAN CV. XYZ MENGGUNAKAN ALGORITMA FP-GROWTH," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 3, Aug. 2024, doi: 10.23960/jitet.v12i3.4541.

[23] E. Nofianti, W. A. Triyanto, and N. Latifah, "PENENTUAN STRATEGI PEMASARAN MENGGUNAKAN FREQUENT PATTERN GROWTH (FP-GROWTH) PADA TOKO KOMPUTER," *Indonesian Journal of Technology, Informatics and Science (IJTIS)*, vol. 1, no. 2, pp. 59–62, Jun. 2020, doi: 10.24176/ijtis.v1i2.4941.

[24] S. Octarina, G. Sonia, and N. Eliyati, "Implementasi metode Greedy Randomized Adaptive Search Procedure dan model Dotted Board pada Cutting Stock Problem Bentuk Reguler," *Jurnal Penelitian Sains*, vol. 23, no. 1, p. 36, Mar. 2021, doi: 10.56064/jps.v23i1.580.

[25] S. R. Cakrawijaya and B. Kriswantara, "PERBANDINGAN KINERJA ALGORITMA STRING MATCHING BOYER-MOORE & KNUTH-MORRIS-PRATT PADA SEO WEB SERVER," *Komputasi: Jurnal Ilmiah Ilmu Komputer dan Matematika*, vol. 18, no. 2, pp. 97–102, Jul. 2021, doi: 10.33751/komputasi.v18i2.3246.

[26] M. A. Nugroho, E. Wuryanto, and K. Faqih, "Optimizing Uncapacitated Facility Location Problem with Cuckoo Search Algorithm based on Gauss Distribution," *SISTEMASI*, vol. 12, no. 2, p. 361, May 2023, doi: 10.32520/stmsi.v12i2.2467.

[27] G. F. H. Nainggolan, S. Andryana, and A. Gunaryati, "PENCARIAN BERITA PADA WEB PORTAL MENGGUNAKAN ALGORITMA BRUTE FORCE STRING MATCHING," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 6, no. 1, pp. 1–10, May 2021, doi: 10.29100/jipi.v6i1.1824.

[28] Z. Li *et al.*, "Reliable and Scalable Routing Under Hybrid SDVN Architecture: A Graph Learning Based Method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 12, pp. 14022–14036, Dec. 2023, doi: 10.1109/TITS.2023.3300082.

[29] M. A. Rakrouki and N. Alharbe, "QoS-Aware Algorithm Based on Task Flow Scheduling in Cloud Computing Environment," *Sensors*, vol. 22, no. 7, p. 2632, Mar. 2022, doi: 10.3390/s22072632.

[30] Y. Wang, "Review on greedy algorithm," *Theoretical and Natural Science*, vol. 14, no. 1, pp. 233–239, Nov. 2023, doi: 10.54254/2753-8818/14/20241041.

[31] I. F. Lövétei, B. Kővári, and T. Bécsi, "MCTS Based Approach for Solving Real-time Railway Rescheduling Problem," *Periodica Polytechnica Transportation Engineering*, vol. 49, no. 3, pp. 283–291, Sep. 2021, doi: 10.3311/PPtr.18584.

[32] S. Lee, "Greedy Algorithm Implementation in Huffman Coding Theory," *International Journal of Software & Hardware Research in Engineering*, vol. 8, no. 9, Sep. 2020, doi: 10.26821/IJSHRE.8.9.2020.8905.

[33] J. Yu, "Thompson -Greedy Algorithm: An Improvement to the Regret of Thompson Sampling and -Greedy on Multi-Armed Bandit Problems," *Applied and Computational Engineering*, vol. 8, no. 1, pp. 507–516, Aug. 2023, doi: 10.54254/2755-2721/8/20230264.

[34] M. Qi, S. Chen, and H. Bian, "Path planning for ships avoiding movable obstacles based on improved greedy algorithm," in *Second International Conference on Algorithms, Microchips, and Network Applications (AMNA 2023)*, A. Palanisamy Muthuramalingam and K. Subramaniam, Eds., SPIE, May 2023, p. 22. doi: 10.1117/12.2678945.

[35] L. Tang, X. He, P. Zhao, G. Zhao, Y. Zhou, and Q. Chen, "Virtual Network Function Migration Based on Dynamic Resource Requirements Prediction," *IEEE Access*, vol. 7, pp. 112348–112362, 2019, doi: 10.1109/ACCESS.2019.2935014.