International Journal of Robotics and Control Systems
Vol. 4, No. 4, 2024, pp. 1934-1946
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Accelerating Convergence in Data Offloading Solutions: A Greedy-Assisted Genetic Algorithm Approach

Mulki Indana Zulfa [a,1,*], Stephen Prasetya Chrismawan [a,2], Adhwa Moyafi Hartoyo [a,3], Wafdan Musa Nursakti [b,4], Waleed Ali Ahmed [c,5]

[a] Jenderal Soedirman University, Mayjen Sungkono Km.5, Purbalingga 53371, Indonesia
[b] Monster Technologies, Lingkaran Cyber Point Timur, Cyberjaya 63000, Malaysia
[b] King Abdulaziz University, Rabigh, 25732, Saudi Arabia
[1] mulki_indanazulfa@unsoed.ac.id; [2] stephen.chrismawan@mhs.unsoed.ac.id; [3] adhwa.hartoyo@mhs.unsoed.ac.id;
[4] wafdan.nursakti@monster.com; [5] waabdullah@kau.edu.sa
* Corresponding Author

ARTICLE INFO

Article history
Received September 25, 2024
Revised October 28, 2024
Accepted November 12, 2024

Keywords
Data Offloading;
Genetic Algorithm;
Greedy Algorithm;
Dynamic Knapsack Problem;
Edge Networks

ABSTRACT

Data offloading, a technique that distributes data across the network, is crucial for alleviating congestion and enhancing system performance. One challenge in this process is optimizing web caching, which can be modeled as a dynamic knapsack problem in edge networks. This study introduces a Greedy-Assisted Genetic Algorithm (GA-Greedy) to tackle this challenge, accelerating convergence and improving solution quality. The greedy heuristic is integrated into the GA at two stages: during initialization to create a superior starting population, and at the end of each iteration to refine solutions generated through genetic operations. The GA-Greedy's effectiveness was evaluated using the IRcache dataset, focusing on hit ratio—an indicator of successful cache accesses that reduces network load and speeds up data retrieval. Results show that GA-Greedy outperforms traditional GA and standalone greedy algorithms, especially with smaller cache sizes. For instance, with a 3K cache size, the half-greedy GA achieved a hit ratio of 0.55, compared to 0.2 for the pure GA and 0.1 for the greedy algorithm. Similarly, the full-greedy GA reached a hit ratio of 0.45. By enhancing convergence and guiding the search, GA-Greedy enables more efficient data distribution in edge networks, reducing latency and improving user experience.

1. Introduction

The evolution of the internet, beginning from its origins as ARPANET [1], [2] and evolving into the interconnected and intelligent world of web 3.0, reflects the massive increase in global user [3]. Once a very rudimentary means for disseminating information across the world, the World Wide Web Protocol [4] which supports everything from web 1.0 [5] to the grander and interactive social media and platforms of web 2.0 [6] is transforming into an intelligent web 3.0 [7] where computers can almost 'understand', subsequently work with information as if they were humans themselves [8]. This rapid growth has put significant strain on traditional cloud computing systems, emphasizing the industry's shift towards edge solutions that offer more efficient management of data-intensive tasks [9]. Data offloading has emerged as a critical strategy to manage network congestion and enhance overall system performance [10]. This method intelligently distributes data across the network,

alleviating the load on central servers. However, the problem of how to select the most effective strategy for offloading the data remains, pushing further explication in the UE toward such betterment of the approaches as optimization [11]. In this manner, genetic algorithms (GA) are powerful in implementation because they are inspired by the principle of evolution in solving complex optimization problems [12], [13]. They are able to borrow from the evolutional biological processes like selection, crossover, and mutations to improve a given population iteratively [14]. Despite it, conventional GA are inefficient because they require too many resources especially brawn one in large-scale scenarios of data offloading [15]. To address this, a more efficient strategy utilizing fitness-based Olympic-type selection has been proposed, improving the speed and accuracy of finding near-optimal solutions.

The integration of a greedy heuristic into GA can further enhance its performance, particularly for the dynamic and complex task of data offloading in caching systems [16], [17]. Demand for high-speed internet access is also being influenced by growing numbers of users of bandwidth demanding applications i.e. video [18] and gaming [19] applications. There is a need for alternate approaches for efficient solutions on the network's edge with the growth of these bandwidth apps demand. Edge computing is the solution that augments cloud computing by keeping computing and data intensive tasks within proximity to the end user and therefore improving system response time [20]. Considering the greater efficiency and versatility of our proposed improved GA, it could assist in the optimal decision making of data offloading on edge networks, so that users are able to steadily and quickly access information and services.

Several studies have demonstrated the effectiveness of combining greedy heuristics with genetic algorithms in diverse applications. For example, Tong et al. [21] developed the IGAA, a greedy-optimized annealing algorithm for optimizing the operation paths of automated seedling transplanting devices, a problem analogous to the Travelling Salesperson Problem. Cui et al. [22] utilized a parallel distribution technique with a greedy algorithm to optimize multiclass hybrid flow shop scheduling problems, enhancing computational efficiency. Paulavicius et al. [16] applied a greedy GA in tourist trip planning, maximizing user satisfaction within given constraints. Shukla et al. [23] integrated GA with a Markov model to generate classical Indian music compositions that adhere to specific musical structures. These examples highlight the versatility of greedy-enhanced GAs in solving complex real-world problems.

Akila et al. [24] tackled premature convergence in multi-objective optimization with a unique strategy that improved classification accuracy. Gangavarapu et al. [25] presented a hybrid feature selection method for high-dimensional medical data using genetic algorithms. Brum et al. [26] explored how Ant Colony Optimization (ACO) could develop a novel iterative greedy algorithm for the Non-Permutation Flow Shop Scheduling Problem, highlighting the potential of such combinations. These examples demonstrate the adaptability of greedy-enhanced methods in optimizing complex systems, particularly in scenarios that demand both global exploration and local refinement.

Based on the research presentation, greedy has proven to be quite effective in helping the performance of genetic algorithms and other meta-heuristic algorithms. The Genetic algorithms are the useful tools for optimizing complex problems by imitating processes like natural selections. Their main strength is being able to navigate large search spaces and find the best solutions, even for hard functions. They work with the assistance of randomness to promote rapid exploration in a wide range of directions, while also making it difficult to be entrapped by local optima and maximizing searches globally [27]. Unfortunately, they tend to be very effective though random effects sometimes lead to premature convergence to sub-optimal solutions. For this, it is essential that greedy algorithms, which are well-known for locating local optima faster than their alternatives, are integrated [28].

The contribution of this research is enhanced GA with greedy for optimizing data offloading in caching system at edge network. The greedy algorithm acts as an initial guide in the optimization process using GA. In the initialization stage, greedy is used to select initial individuals that have the potential for good solutions based on certain criteria, thus forming a better quality initial GA

population. At the end of the iteration, when GA has produced a number of potential solutions, the greedy algorithm plays a role again. In this instance, greedy is employed to assess the various solutions and choose the one, which more than others, has the highest or most efficient fitness level. That is why, greedy ensures that the combination of solutions generated at the end of GA is better than any local optimum and is, in fact, a true global optimum.

2. Method

2.1. Dataset

This study has access to the IRcache dataset, which archives caching records from proxy caches in a distributed architecture. The first creator of the IRcache dataset is Alex Rousskov [29]. It has sine went under the administration of National Laboratory of Applied Network Research [30]. The IRcache dataset is representative of a global proxy network provided by a cluster of proxy servers located in five cities in the United States: Urbana-Champaign (UC), Boulder (BO2), Silicon Valley (SV), San Diego (SD), and New York (NY). There are a lot of datasets aimed at understanding how to model the cache replacement policy in data in cache memory and the IRcache dataset is one of such datasets. The properties of traffic of the internet in such a case are also capture in the dataset. Within the last seven years this IRcache dataset is used by Paul et al. when presented simCache architecture framework [31], Ibrahim et al [32] while doing research in cache replacement and Li et al [33] while doing research on content caching optimization. The properties, which are presented in the IRcache dataset are outlined in Table 1.

In relation to requested online objects, the dataset records a variety of metadata, such as object size (size), response time, and request frequency. By nature the logs provide an insight into the behavior of real-world caching systems. A sample of the raw IRcache dataset is presented in Table 1. However, for this research, which uses the KP model, some key columns have been used.

- iddata: It indicates the column with a unique identifier assigned to each object request. Iddata, or "id," has been employed in this study to track particular things in order to identify them uniquely and subject them to optimization in the future.
- elapsed: The time in milliseconds required for the object request was completed. Response time holds the key deciding factor of the profit of an object in KP model to be selected, meaning that in a cache, objects with a faster response time will be more profitable [34], [35].
- size: This data presents the size of the object in bytes that was requested. Within the context of Knapsack Problem, size is how its called the weight of an object which determines how much space in a cache would be taken by this object [36], [37].
- Uniform Resource Locators (URL): The URL records requests for specific web objects. From this column, we calculated the cumulative number of accesses for each object based on its URL. This cumulative access count is then used as a factor to determine the value of the object [38]. Frequently requested objects are deemed more valuable to cache since they can significantly reduce network load [39], [40].

2.1.1. Data Transformation Process

The IRcache dataset used in this study consists of entries that record object size (size), access time (elapsed), and the URL of the object accessed. To adapt this dataset to the 0/1 knapsack problem, we perform several transformations. For each item, we first compute a cumulative access metric that indicates how frequently the object URL is accessed within the dataset. In order to shed light on the complexity of the knapsack problem under consideration, we also determine the total number of unique items following the transformation and normalization procedures. This metric will serve as the value in the context of the knapsack problem. Next, since the values of size and elapsed have a wide range, we normalize these values using (1). This normalization ensures that all features have comparable scales, which is essential for the knapsack algorithm. The normalized values are directly

integrated into the computation of the objective function, where they impact the choice of items by proportionally modifying the weight and profit of each item to conform to the knapsack limitations. Following normalization, we use (2) to create the objective function for the knapsack problem. Under the knapsack capacity constraints, this objective function seeks to maximize the cumulative access value of the chosen objects while maintaining the total normalized size and access time. The weighting factors for access value, size, and elapsed time are denoted by b1, b2, and b3, respectively. Lastly, the hit ratio serves as the primary performance indicator for the cache system. Finally, to evaluate the cache system's performance, we use the hit ratio as the main metric. The hit ratio is calculated from the objective function using (3) [41], [42]. This metric reflects the proportion of object requests that are successfully satisfied from the cache, which is a direct indicator of the system's effectiveness. With this transformation and metric, we can effectively apply the 0/1 knapsack model to the IRcache dataset and evaluate various caching strategies based on the resulting hit ratio. A sample of the pivoted dataset for KP testing is shown in Table 2.

$$Norm_{var(x)} = \frac{1}{n}\sum_{i=1}^{n}\frac{var_{(x)} - var_{(min)}}{var_{(max)} - var_{(x)}} \quad (1)$$

$$Func = b_1 * (1 - v_{acc}) + b_2 * (1 - v_{elp}) + b_3 * (1 - v_{size}) \quad (2)$$

$$hit\ ratio = \frac{\sum_{i=1}^{N} r_i}{N} \quad (3)$$

Research with KP01 modeling is generally tested with choice of data sets of 10, 20, 50 to 100 [43]-[45]. The number of rows in the IRcache dataset is in the thousands. However, in this research, we tried a larger number of KP data, up to more than 300 rows of data. Therefore, it was also possible to make use of the dataset for testing the problem of Knapsack after pivoting the dataset.Provided the dataset assisted in optimizing the object weight and object value. By utilizing this refineddataset, we were able to apply the Knapsack Problem to examine various cache replacement policies.

Table 1. IRcache dataset sample

Field	Value
id	: 784
time_s	: 1282363557
elapsed	: 159
ipclient	: 171.27.148.136
code	: TCP_HIT/200
size	: 3106
method	: GET
url	: http://www.tokobagus.com/upload/users/561/561065/purple-1279870160_list.jpg
type	: image/jpeg

2.2. Proposed Method

2.2.1. Genetic Algorithm

Due to their capacity to explore search spaces, genetic algorithms have been used in several optimization challenges, including data dumping. Nonetheless, the sluggish convergence of classical genetic algorithms in relation to the specified issue space continues to provide a barrier, particularly in increasingly complex problem domains [46]. We propose a greedy heuristic to guide the genetic algorithm towards regions of the solution space that provide more benefits. The genetic algorithm (GA) is considered the foundation of the mission, using the idea of individuals within a population, where each person represents a potential candidate solution for the data offloading issue. These methodologies use a chromosomal model of solutions, populating the chromosomes with binary or

numeric representations that denote the intermediates and cloud resources to which tasks are delegated.

Table 2. IRcache dataset sample

iddata	size (weigh)	elapsed	access count
404052406	3106	23	30
404018280	9906	13	21
40403542	3649	7	8
404021968	20410	4	11

The genetic algorithm drives global exploration and optimization. Each individual is a possible data offloading solution. These solutions are represented as chromosomes, using binary or numeric representations to indicate which tasks are offloaded to edge servers or cloud resources. A well-defined fitness function that evaluates the quality of data offloading strategies is used to determine each individual's (gen) fitness [47]. This fitness function frequently minimizes cost, delay, and energy consumption while taking service level agreement and bandwidth constraints into account. High-fitness chromosomes are selected, guaranteeing that better solutions will contribute genetic material to the following generation. The tournament [48], roulette wheel [49], and rank-based selection [50] are common. Tournament selection was used in this study because it can moderate selective pressure and preserve solution variety, both of which are important for meeting the cache optimization problem's convergence requirements. A crucial genetic operator called crossover enables genetic information to be shared between chosen parents. Crossover creates offspring with features from both parents by recombining chromosomal segments, which may lead to new and better solutions [51]. Fig. 1 shows how the crossover mechanism is performed from parent-1 (P-1) and parent-2 (P-2) which produces offspring (P'). Mutation, another important operator, randomly modifies the genes, introducing diversity into the population [52]. Exploring new areas of the solution space maintains population variety and avoids premature convergence to local optima. GA uses crossover, mutation, and selection to grow the population over several generations. More suited chromosomes and nearly ideal data dumping mechanisms result from this evolutionary process' filtering of solutions.

2.2.2. Greedy Algorithm

A greedy strategy is critical in accelerating the convergence of optimization procedure, since it offers a way of making local decisions at each step [54]. This algorithm proceeds on the basis of a given rule or a decided criterion which values short term benefits only. When applied in data offloading, the greedy algorithm could select tasks with the most amount of data or most computationally intensive and offload them to the edge server or the cloud where the less current load or latency exists. The details about the greedy algorithms adaptations may differ based on the problem definitions and goals. Nevertheless, it is therefore proposed as its most important characteristic the ability to provide good initial solutions or improve given solutions using local optimization techniques [55]. This occurs because such systems are short-sighted and care only about the local gains without regard for the repercussions of their actions or the optimality of the solution as a whole. In the case of the hybrid approach, the greedy strategy can be incorporated into the genetic algorithm in a numberof different ways [56]. For one, it can be employed to obtain a better quality initial population for theGA so that the population is closer to the optimal solution making the number of generations requiredfor convergence fewer. In fact, the greedy method comes in each generation of the GA to post-processthe solutions obtained after crossover and mutation thereby improving the quality and speed of the search.

2.2.3. Greedy-Assisted Genetic Algorithm

This paper presents a new optimized technique, the Greedy-Assisted Genetic Algorithm (GA-Greedy) for solving the dynamic web caching knapsack problem. By using a greedy approach inside the framework of genetic algorithms, the GA-Greedy further improved the initial population's quality and optimized the evolution process [57]. The greedyheuristic has been implemented first in GA-Greedy since it has the objective of creating a populationof solutions that has an edge. So as to make the

generation of pure genetic algorithms population easier the random method is resorted to, an approximate solution may be attained after many generations [58]. To speed up the process, a greedy heuristic is used to build the initial population of chromosomes. This ensures that the search is focused on the best area of the solution space. Also, the GA-Greedy algorithm restructures the population at the end of every loop and replaces some of the solutions with better ones using the greedy heuristic. The goal is to improvethe offspring of the children produced by the selection, crossover, and mutation genetic operations. The greedy heuristic gives extra attention to the offspring as soon as they are produced, and if on theother hand a child either goes outside the given capacity of the knapsack or does not have maximum fitness, the child supports policies concerning local improvement [55]. This phase allows for precision and a very reasonable high standard of the population quality to be maintained throughout the optimization phase.

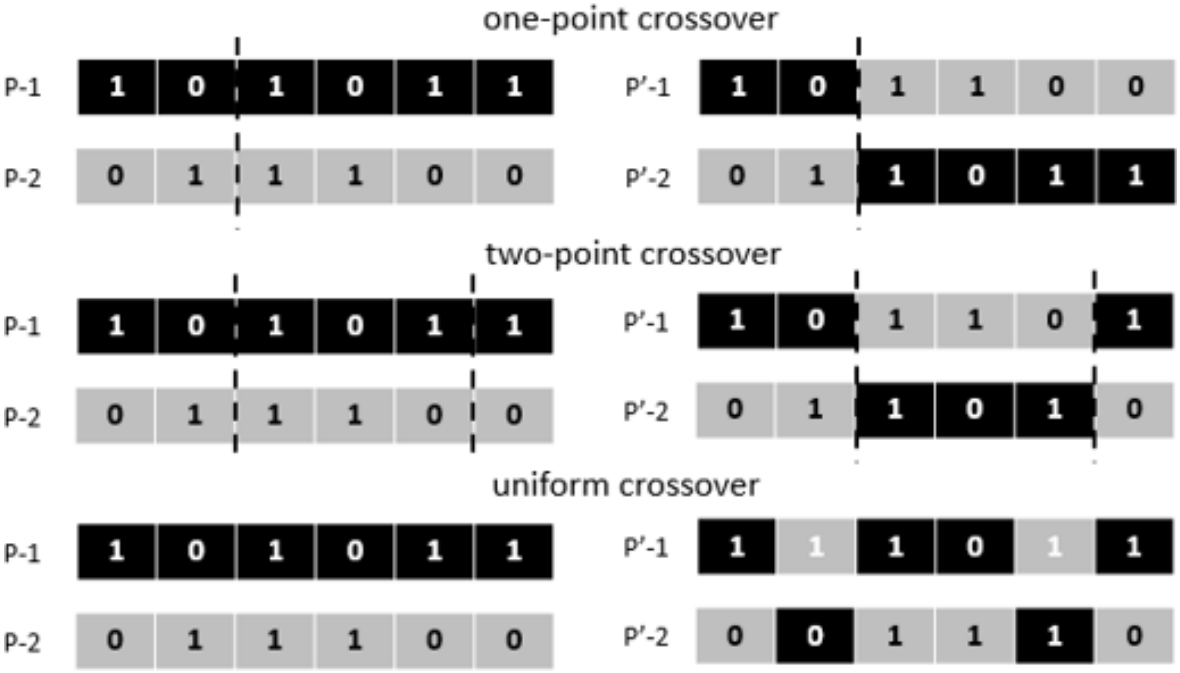

Fig. 1. Some crossover mechanisms in genetic algorithm [53]

Even though the greedy heuristic improves the results obtained from the GA-Greedy algorithm, the underlying genetic operators remains the same as shown in Fig. 2. In the selection phase, parent solutions are determined by this viability criterion, while crossover focuses precisely on transferring month attributes from parent solutions to produce children. These procedures which come in a developed form using the greedy heuristic, are important for managing exploration inside the solution and are effective in its application [54]. Therefore, the anticipation that illogical, yet practical solutions are frantically implemented would be detrimental in enhancing the genetic algorithm framework incorporating the greedy heuristic. It is presumed genetically engineered algorithms would help in making use of an initial close solution as a guide to systematically search for a better solution in a shorter period. In addition, there will also be an increase in the confidence in the quality of the solutions which will be available as they will be equally feasible and also better in terms of fitness values. This work uses the transformed IRcache dataset to apply the GA-Greedy approach to solve the knapsack problem in the context of online caching optimization. In order to provide a good starting point, the Greedy heuristic starts by initializing the population, either half or all. Greedy is also used to replace some of the genes that perform the worst, modifying them to fit within the maximum size restriction. The GA naturally manages the next steps, preserving diversity and lowering the possibility of local optima by permitting adequate variance in the progeny without overly forceful Greedy application.

3. Results and Discussion

3.1. Result

The following graphs show the fitness values and the comparison of the average fitness values of individuals in the first generation. These graphs are presented to illustrate the implications of Greedy-assisted GA on fitness values, which impact the hit ratio. The fitness values for 100 first-generation individuals using three distinct GA algorithms and a maximum cache size of 200K are shown in Fig. 3. In subfigure (a), the full-greedy assisted GA demonstrates a tight clustering of fitness values between 113 and 114, reflecting a more optimal population at initialization. This outcome is due to the entire population being influenced by the Greedy algorithm, which generates a more refined initial fitness distribution. In contrast, subfigure (b) illustrates that the half-greedy assisted GA produces a broader fitness range, from 50 to 117, because half of the population is initialized via Greedy, while the other half is randomly generated by the GA. Subfigure (c) highlights the pure GA's low fitness values, ranging between 9 and 20, as its initial population is entirely random, resulting in a less controlled and lower-quality starting point.

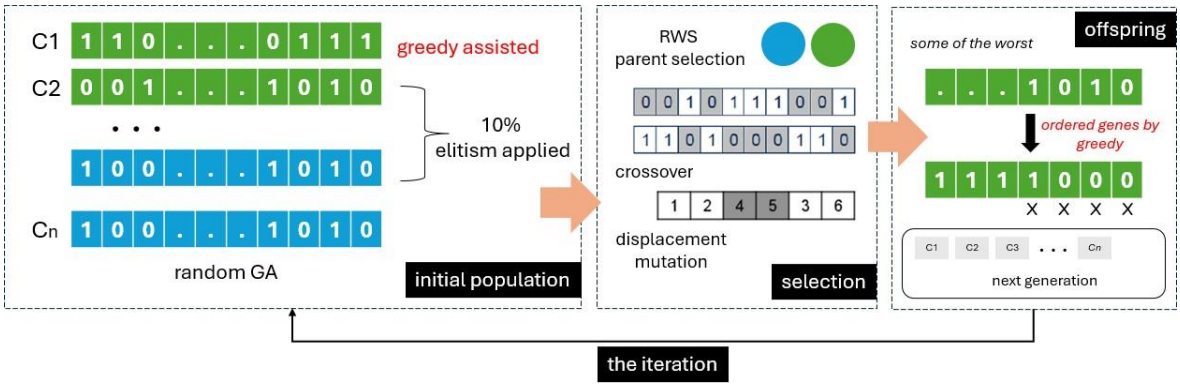

Fig. 2. The proposed greedy assisted GA

The hit percentage for several cities across a range of cache sizes is examined in Fig. 4. Both the pure GA and Greedy algorithms struggle to produce the best results across all graphs (a–d), showing low hit ratios from 3K to 90K cache sizes. The initial population created in the first generation, on the other hand, has a significant impact on the half-greedy and full-greedy assisted GAs' persistent superior performance. For instance, subfigure (a), representing city SV, shows that the half- greedy assisted GA consistently leads across all cache sizes, suggesting that the city's data offloading patterns benefit from higher variability in the initial population. In subfigure (b) for city NY, both greedy-assisted GAs excel across the cache sizes, with the full-greedy assisted GA outperforming between 3K and 15K, and the half-greedy taking the lead from 20K to 300K.

Fig. 4 also highlights the limitations of exact optimization algorithms like Branch and Bound and Dynamic Programming, which underperform due to their inability to maintain sufficient variation in the data offloading process. In subfigures (c) and (d), representing cities UC and BO2, both the full-greedy and half-greedy assisted GAs maintain superior hit ratios across cache sizes, while the pure GA consistently delivers the lowest hit ratios. These findings emphasize the importance of population initialization in determining cache performance, with greedy-assisted GAs providing a significant advantage in generating a more robust starting population, which subsequently impacts the overall hit ratio performance.

3.2. Discussion

This study's actual findings demonstrate the effectiveness of the GA-Greedy method in solving the dynamic web caching knapsack issue. The relevance of incorporating a greedy heuristic into the GA framework is highlighted by GA-Greedy's consistent dominance over greedy algorithms and pure GA, particularly when cache sizes are smaller. This integration speeds convergence and improves

solution quality by strategically directing the search process and improving the produced progeny. The enhanced hit rates attained by the GA-Greedy result in a more efficient use of edge network resources. The GA-Greedy optimizes data offloading choices, alleviating the strain on backhaul lines and core network infrastructure, resulting in reduced latency and enhanced overall network performance. This improved efficiency is especially vital in edge computing settings where resources are often limited and user demands are elevated. Further confirming GA-Greedy's potential in practical applications, a comparison with earlier research shows that it performs better in terms of hit rate and convergence, especially in edge contexts with constraints.

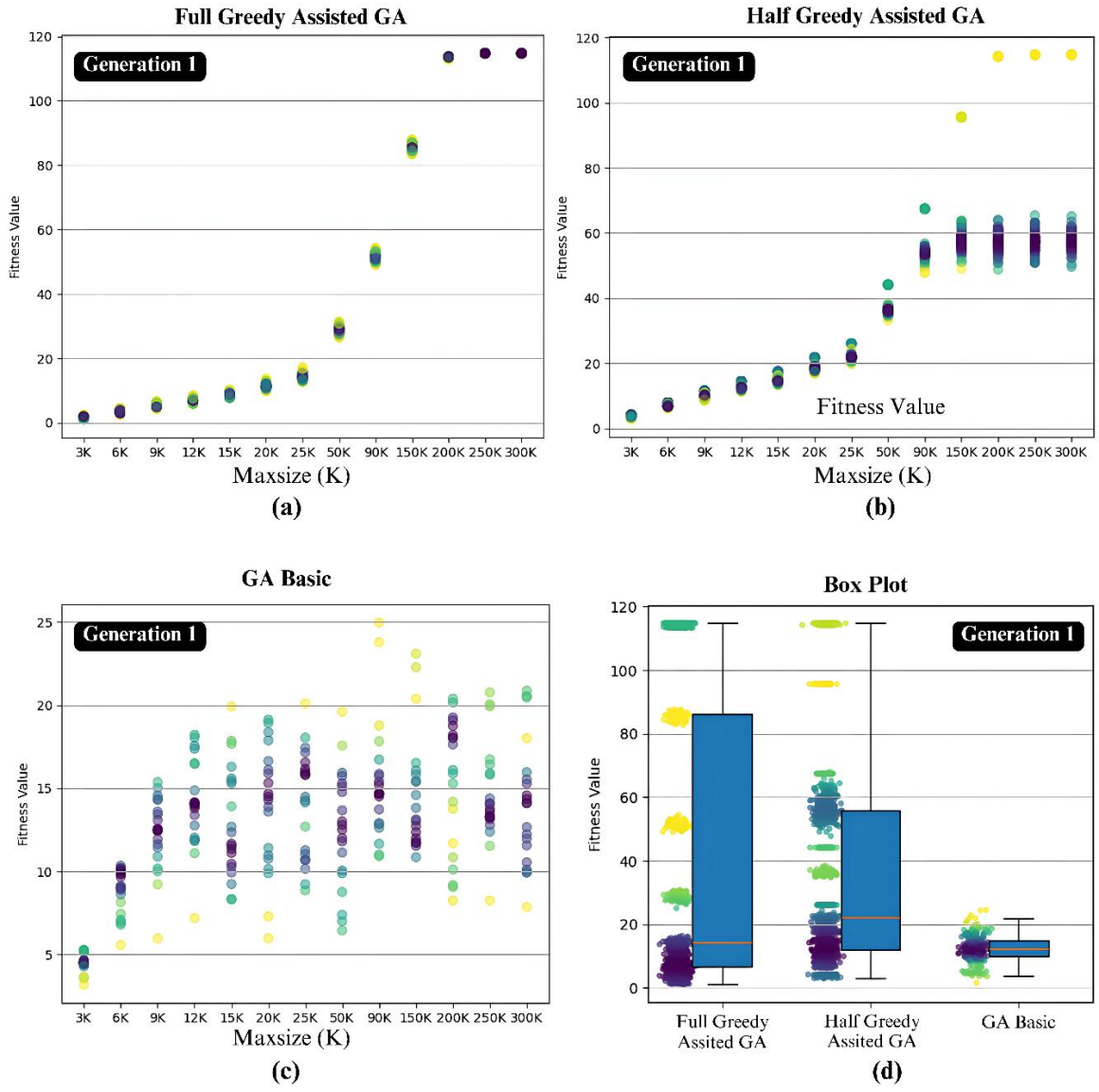

Fig. 3. Greedy-assisted implication on first generation

The effectiveness of GA-Greedy in improving data offloading extends beyond web caching. The fundamental ideas of this methodology may be applied to several sectors necessitating effective data distribution techniques, including content delivery networks, Internet of Things (IoT) systems, and vehicle networks. In these situations, GA-Greedy's capacity to adjust to changing circumstances and optimize resource distribution may markedly improve system performance and user experience. Although the GA-Greedy exhibits encouraging outcomes, it is essential to recognize its limitations. The algorithm's efficacy may be affected by the dataset's particular attributes and the selection of parameters. Subsequent study may investigate the formulation of adaptive mechanisms to optimize

the GA-Greedy's behavior according to the issue setting. Moreover, exploring the incorporation of other optimization methods or machine learning models may augment the algorithm's efficacy.

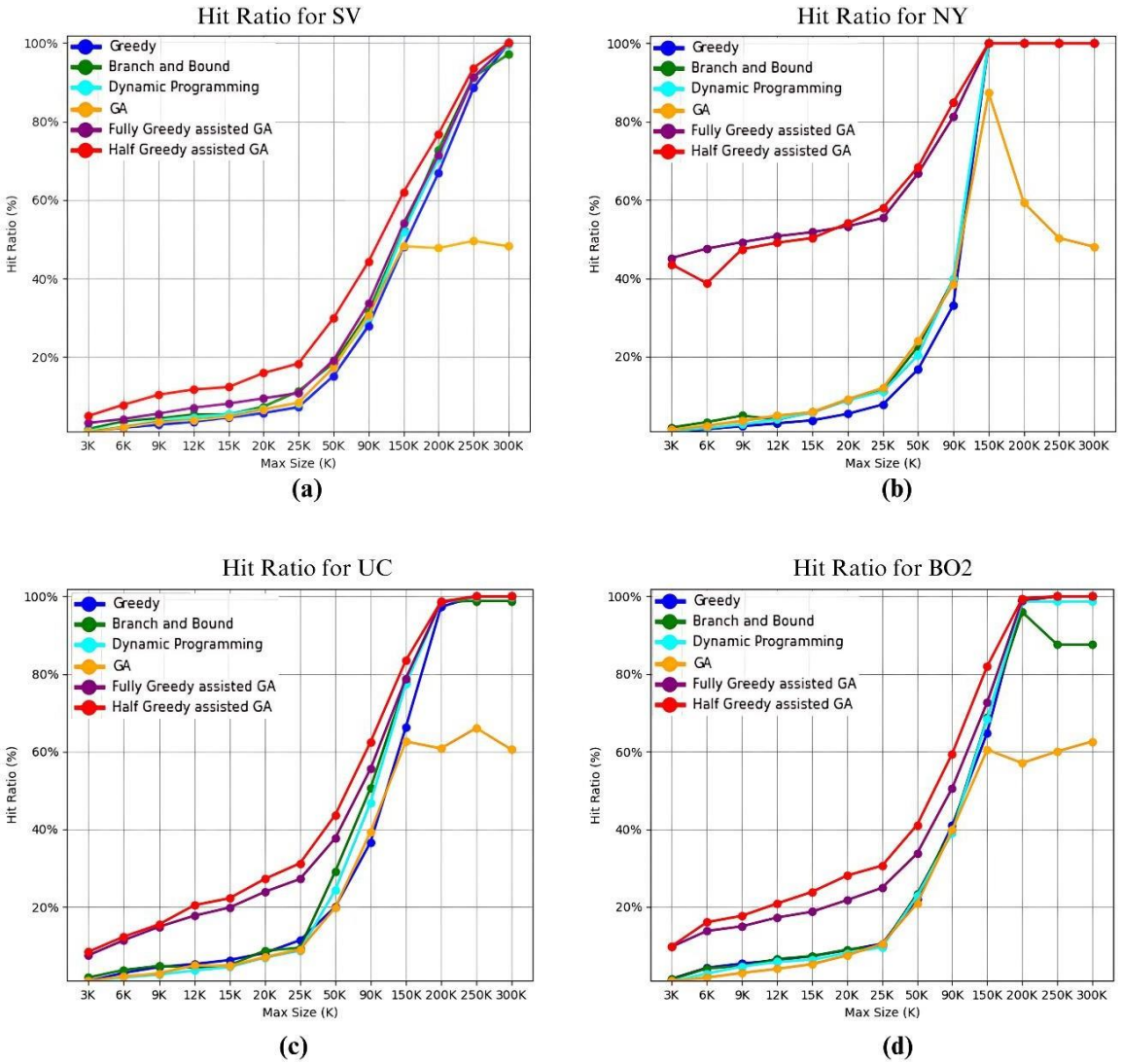

Fig. 4. Hit ratio comparation

4. Conclusion

In order to solve the problem of the dynamic web caching knapsack, this study introduces GA-Greedy, a novel hybrid approach that integrates a greedy heuristic within a Genetic Algorithm (GA) framework to address the dynamic web caching knapsack problem. GA-Greedy consistently outperforms both pure GA and traditional greedy algorithms, achieving a hit ratio of approximately 55% with a 3K cache size, compared to around 20% for the pure GA and 10% for the standalone greedy algorithm, based on empirical tests with the IRcache dataset. This demonstrates GA-Greedy's ability to optimize data offloading in edge networks by reducing latency and backhaul strain, leading to faster data retrieval and improved user experience. However, the performance of GA-Greedy can vary with different datasets and parameter configurations, suggesting the need for adaptive mechanisms to adjust parameters dynamically. Future research should focus on integrating AI and machine learning techniques, such as reinforcement learning, to enhance GA-Greedy's adaptability, and exploring its application in IoT, content delivery networks, and vehicular systems. Overall, GA-Greedy proves to be a versatile and robust solution for optimizing data offloading in dynamic environments, making it a valuable tool for future edge computing applications.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research was funded by a DRTPM grant from the Ministry of Education, Culture, Research and Technology based on decree number 0667/E5/AL.04/2024 and contract 20.44/UN23.35.5/PT.01.00/VI/2024.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] D. Clark, "The design philosophy of the DARPA internet protocols," *Symposium proceedings on Communications architectures and protocols*, pp. 106-114, 1988, https://doi.org/10.1145/52324.52336.

[2] M. El Khadiri, W. C. Yeh, and H. Cancela, "An efficient factoring algorithm for the quickest path multi-state flow network reliability problem," *Computers & Industrial Engineering*, vol. 179, p. 109221, 2023, https://doi.org/10.1016/j.cie.2023.109221.

[3] M. Candela, V. Luconi, and A. Vecchio, "A worldwide study on the geographic locality of Internet routes," *Computer Networks*, vol. 201, p. 108555, 2021, https://doi.org/10.1016/j.comnet.2021.108555.

[4] T. Berners-Lee, R. Cailliau, J. F. Groff, and B. Pollermann, "World-wide web: The information universe," *Internet Research*, vol. 2, no. 1, pp. 52-58, 1992, https://doi.org/10.1108/eb047254.

[5] Z. JinCheng and D. L. K. Chuen, "Chapter 28 - Understanding the Evolution of the Internet: Web 1.0 to Web3.0, Web3, and Web 3+∗," *Handbook of Digital Currency (Second Edition)*, pp. 553-581, 2024, https://doi.org/10.1016/B978-0-323-98973-2.00052-6.

[6] A. I. Díaz-Cano and A. Esplugues-Cebrián, "Web 2.0 as a new support for breastfeeding: Perception of mothers and professionals through a qualitative approach," *Enfermería Clínica (English Edition)*, vol. 34, no. 1, pp. 34-48, 2024, https://doi.org/10.1016/j.enfcle.2023.11.001.

[7] J. Zhu, F. Li, and J. Chen, "A Survey of Blockchain, Artificial Intelligence, and Edge Computing for Web 3.0," *Computer Science Review*, vol. 54, p. 100667, 2023, https://doi.org/10.1016/j.cosrev.2024.100667.

[8] E. Demuro and L. Gurney, "Artificial intelligence and the ethnographic encounter: Transhuman language ontologies, or what it means 'to write like a human, think like a machine," *Language & Communication*, vol. 96, pp. 1-12, 2024, https://doi.org/10.1016/j.langcom.2024.02.002.

[9] D. Wang, X. An, X. Zhou, and X. Lü, "Data cache optimization model based on cyclic genetic ant colony algorithm in edge computing environment," *International Journal of Distributed Sensor Networks*, vol. 15, no. 8, 2019, https://doi.org/10.1177/1550147719867864.

[10] M. I. Zulfa, R. Hartanto, and A. E. Permanasari, "Caching strategy for Web application – a systematic literature review," *International Journal of Web Information Systems*, vol. 16, no. 5, pp. 545-569, 2020, https://doi.org/10.1108/IJWIS-06-2020-0032.

[11] S. Zhang, W. Sun and J. Liu, "Spatially Cooperative Caching and Optimization for Heterogeneous Network," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11260-11270, 2019, https://doi.org/10.1109/TVT.2019.2941115.

[12] M. I. Zulfa, R. Hartanto and A. E. Permanasari, "Performance Comparison of Swarm Intelligence Algorithms for Web Caching Strategy," *2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pp. 45-51, 2021, https://doi.org/10.1109/COMNETSAT53002.2021.9530778.

[13] M. I. Zulfa, A. Fadli, A. E. Permanasari, and W. A. Ahmed, "Performance comparison of cache replacement algorithms onvarious internet traffic," *Jurnal INFOTEL*, vol. 15, no. 1, pp. 1-7, 2023, https://doi.org/10.20895/infotel.v15i1.872.

[14] M. O. Okwu and L. K. Tartibu, "Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications," *Studies in Computational Intelligence*, vol. 927, 2021, https://doi.org/10.1007/978-3-030-61111-8.

[15] E. Rajapackiyam *et al*., "An efficient computation offloading in edge environment using genetic algorithm with directed search techniques for IoT applications," *Future Generation Computer Systems*, vol. 158, pp. 378-390, 2024, https://doi.org/10.1016/j.future.2024.04.021.

[16] R. Paulavičius, L. Stripinis, S. Sutavičiūtė, D. Kočegarov, and E. Filatovas, "A novel greedy genetic algorithm-based personalized travel recommendation system," *Expert Systems with Applications*, vol. 230, p. 120580, 2023, https://doi.org/10.1016/j.eswa.2023.120580.

[17] H. Qin, N. Li, T. Wang, G. Yang, and Y. Peng, "CDT: Cross-interface Data Transfer scheme for bandwidth-efficient LoRa communications in energy harvesting multi-hop wireless networks," *Journal of Network and Computer Applications*, vol. 229, p. 103935, 2024, https://doi.org/10.1016/j.jnca.2024.103935.

[18] G. Huang, J. Liu, B. Zhang, and C. Li, "Quality-driven video streaming for ultra-dense OFDMA heterogeneous networks," *Computer Networks*, vol. 218, p. 109398, 2022, https://doi.org/10.1016/j.comnet.2022.109398.

[19] J. Paul, M. Akbari, S. Mondal, S. Das, "Knowledge sharing leads to engagement during Covid-19 for online gamers," *Information & Management*, vol. 61, no. 4, p. 103948, 2024, https://doi.org/10.1016/j.im.2024.103948.

[20] M. I. Bala and M. A. Chishti, "Survey of applications, challenges and opportunities in fog computing," *International Journal of Pervasive Computing and Communications*, vol. 15, no. 2, pp. 80-96, 2019, https://doi.org/10.1108/IJPCC-06-2019-059.

[21] J. Tong *et al*., "Optimizing the path of seedling transplanting with multi-end effectors by using an improved greedy annealing algorithm," *Computers and Electronics in Agriculture*, vol. 201, p. 107276, 2022, https://doi.org/10.1016/j.compag.2022.107276.

[22] H. Cui, X. Li, and L. Gao, "An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem," *Expert Systems with Applications*, vol. 222, p. 119805, 2023, https://doi.org/10.1016/j.eswa.2023.119805.

[23] S. Shukla and H. Banka, "Markov-based genetic algorithm with ϵ-greedy exploration for Indian classical music composition," *Expert Systems with Applications*, vol. 211, p. 118561, 2023, https://doi.org/10.1016/j.eswa.2022.118561.

[24] S. Akila and S. Allin Christe, "A wrapper based binary bat algorithm with greedy crossover for attribute selection," *Expert Systems with Applications*, vol. 187, p. 115828, 2022, https://doi.org/10.1016/j.eswa.2021.115828.

[25] T. Gangavarapu and N. Patil, "A novel filter–wrapper hybrid greedy ensemble approach optimized using the genetic algorithm to reduce the dimensionality of high-dimensional biomedical datasets," *Applied Soft Computing*, vol. 81, p. 105538, 2019, https://doi.org/10.1016/j.asoc.2019.105538.

[26] A. Brum, R. Ruiz, and M. Ritt, "Automatic generation of iterated greedy algorithms for the non-permutation flow shop scheduling problem with total completion time minimization," *Computers & Industrial Engineering*, vol. 163, 2022, https://doi.org/10.1016/j.cie.2021.107843.

[27] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: taxonomy, applications, and open challenges," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13187-13257, 2023, https://doi.org/10.1007/s10462-023-10470-y.

[28] S. R. Kumar and K. D. Singh, "Nature-Inspired Optimization Algorithms: Research Direction and Survey," *Neural and Evolutionary Computing*, 2021, https://doi.org/10.48550/arXiv.2102.04013.

[29] F. Lin, "Solving the knapsack problem with imprecise weight coefficients using genetic algorithms," *European Journal of Operational Research*, vol. 185, no. 1, pp. 133-145, 2008, https://doi.org/10.1016/j.ejor.2006.12.046.

[30] R. Vinayakumar, K. P. Soman and P. Poornachandran, "Secure shell (ssh) traffic analysis with flow based features using shallow and deep networks," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2026-2032, 2017, https://doi.org/10.1109/ICACCI.2017.8126143.

[31] M. I. Zulfa, S. Maryani, Ardiansyah, T. Widiyaningtyas, W. Ali, "Application-Level Caching Approach Based on Enhanced Aging Factor and Pearson Correlation Coefficient," *International Journal on Informatics Visualization*, vol. 8, no. 1, p. 31, 2024, https://doi.org/10.62527/joiv.8.1.2143.

[32] H. Ibrahim, W. Yasin, N. I. Udzir, and B. Process, "Intelligent cooperative web caching policies for media objects based on J48 decision tree and Naïve Bayes supervised machine learning algorithms in structured peer-to-peer systems," *Journal of Information and Communication Technology*, vol. 15, no. 2, pp. 85-116, 2016, https://doi.org/10.32890/jict2016.15.2.5.

[33] X. Li, X. Wang, Z. Sheng, H. Zhou, and V. C. M. Leung, "Resource allocation for cache-enabled cloud-based small cell networks," *Computer Communications*, vol. 127, pp. 20-29, 2018, https://doi.org/10.1016/j.comcom.2018.05.007.

[34] Q. Wang, S. Guo, J. Liu, C. Pan and L. Yang, "Profit Maximization Incentive Mechanism for Resource Providers in Mobile Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 138-149, 2022, https://doi.org/10.1109/TSC.2019.2924002.

[35] K. Cao, L. Li, Y. Cui, T. Wei and S. Hu, "Exploring Placement of Heterogeneous Edge Servers for Response Time Minimization in Mobile Edge-Cloud Computing," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 494-503, 2021, https://doi.org/10.1109/TII.2020.2975897.

[36] B. Sun *et al*., "The Online Knapsack Problem with Departures," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, no. 3, pp. 1-32, 2022, https://doi.org/10.1145/3570618.

[37] B. Abdollahzadeh, S. Barshandeh, H. Javadi, and N. Epicoco, "An enhanced binary slime mould algorithm for solving the 0–1 knapsack problem," *Engineering with Computers*, vol. 38, pp. 3423-3444, 2022, https://doi.org/10.1007/s00366-021-01470-z.

[38] J. Reynolds, A. Bates, and M. Bailey, "Equivocal URLs: Understanding the Fragmented Space of URL Parser Implementations," *Computer Security–ESORICS 2022*, pp. 166-185, 2022, https://doi.org/10.1007/978-3-031-17143-7_9.

[39] A. Laughter, S. Omari, P. Szczurek, and J. Perry, "Detection of Malicious HTTP Requests Using Header and URL Features," *Proceedings of the Future Technologies Conference (FTC) 2020*, vol. 2, pp. 449-468, 2021, https://doi.org/10.1007/978-3-030-63089-8_29.

[40] N. Fukushi, T. Koide, D. Chiba, H. Nakano, and M. Akiyama, "Understanding Security Risks of Ad-based URL Shortening Services Caused by Users' Behaviors," *Journal of Information Processing*, vol. 30, pp. 865-877, 2022, https://doi.org/10.2197/ipsjjip.30.865.

[41] J. Mertz and I. Nunes, "Understanding Application-Level Caching in Web Applications," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1-34, 2017, https://doi.org/10.1145/3145813.

[42] T. Chen, W. Shang, A. E. Hassan, M. Nasser, and P. Flora, "CacheOptimizer: Helping Developers Configure Caching Frameworks for Hibernate-Based Database-Centric Web Applications," *FSE 2016: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 666-677, 2016, https://doi.org/10.1145/2950290.2950303.

[43] Q. Chen and Y. Shao, "A Hybrid Genetic Algorithm to Solve Zero-One Knapsack Problem," *Applied Informatics and Communication*, pp. 315-320, 2011, https://doi.org/10.1007/978-3-642-23214-5_42.

[44] Y. Feng, J. -H. Yi and G. -G. Wang, "Enhanced Moth Search Algorithm for the Set-Union Knapsack Problems," *IEEE Access*, vol. 7, pp. 173774-173785, 2019, https://doi.org/10.1109/ACCESS.2019.2956839.

[45] Y. Huang, P. Wang, J. Li, X. Chen and T. Li, "A Binary Multi-Scale Quantum Harmonic Oscillator Algorithm for 0–1 Knapsack Problem With Genetic Operator," *IEEE Access*, vol. 7, pp. 137251-137265, 2019, https://doi.org/10.1109/ACCESS.2019.2942340.

[46] S. Parvaze *et al*., "Optimization of Water Distribution Systems Using Genetic Algorithms: A Review," *Archives of Computational Methods in Engineering*, vol. 30, no. 7, pp. 4209-4244, 2023, https://doi.org/10.1007/s11831-023-09944-7.

[47] M. Ahmed *et al*., "Vehicular Communication Network Enabled CAV Data Offloading: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 7869-7897, 2023, https://doi.org/10.1109/TITS.2023.3263643.

[48] S. A. Oke, W. O. Adedeji, M. C. Ikedue, J. Rajan, "Exploiting Tournament Selection-Based Genetic Algorithm in Integrated AHP-Taguchi Analyses-GA Method for Wire Electrical Discharge Machining of AZ91 Magnesium Alloy," *Indonesian Journal of Industrial Engineering & Management*, vol. 4, no. 1, pp. 1-17, 2023, https://doi.org/10.22441/ijiem.v4i1.17387.

[49] A. S. Hameed, H. M. B. Alrikabi, A. A. Abdul–Razaq, Z. H. Ahmed, H. K. Nasser, and M. L. Mutar, "Appling the Roulette Wheel Selection Approach to Address the Issues of Premature Convergence and Stagnation in the Discrete Differential Evolution Algorithm," *Applied Computational Intelligence and Soft Computing*, vol. 2023, no. 1, pp. 1-16, 2023, https://doi.org/10.1155/2023/8892689.

[50] E. Tasci *et al*., "RadWise: A Rank-Based Hybrid Feature Weighting and Selection Method for Proteomic Categorization of Chemoirradiation in Patients with Glioblastoma," *Cancers*, vol. 15, no. 10, p. 2672, 2023, https://doi.org/10.3390/cancers15102672.

[51] R. T. Bye, M. Gribbestad, R. Chandra, O. L. Osen, "A Comparison of GA Crossover and Mutation Methods for the Traveling Salesman Problem," *Innovations in Computational Intelligence and Computer Vision*, pp. 529-542, 2021, https://doi.org/10.1007/978-981-15-6067-5_60.

[52] S. Rani, B. Suri, and R. Goyal, "On the Effectiveness of Using Elitist Genetic Algorithm in Mutation Testing," *Symmetry*, vol. 11, no. 9, p. 1145, 2019, https://doi.org/10.3390/sym11091145.

[53] N. I. Senaratna, "Genetic Algorithms: The Crossover-Mutation Debate," *Neural and Evolutionary Computing*, 2005, https://doi.org/10.48550/arXiv.2102.04013.

[54] F. Yu, Y. Peng, J. Li, G. Zhou, and L. Chen, "An Analysis of Optimization for Car PBS Scheduling Based on Greedy Strategy State Transition Algorithm," *Applied Science*, vol. 13, no. 10, p. 6194, 2023, https://doi.org/10.3390/app13106194.

[55] T. Benecke and S. Mostaghim, "Effects of Optimal Genetic Material in the Initial Population of Evolutionary Algorithms," *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1386-1391, 2023, https://doi.org/10.1109/SSCI52147.2023.10372037.

[56] R. M. Aziz, R. Mahto, K. Goel, A. Das, P. Kumar, and A. Saxena, "Modified Genetic Algorithm with Deep Learning for Fraud Transactions of Ethereum Smart Contract," *Applied Science*, vol. 13, no. 2, p. 697, 2023, https://doi.org/10.3390/app13020697.

[57] L. Pintér, K. Krajczár, F. Őry, J. Szalma, and E. Lempel, "Effect of Intermediate Irrigation on Temperature Rise during Broken NiTi File Removal Using Ultrasonic Device," *Applied Science*, vol. 13, no. 17, p. 9761, 2023, https://doi.org/10.3390/app13179761.

[58] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091-8126, 2021, https://doi.org/10.1007/s11042-020-10139-6.