Genetic Engineering Algorithm (GEA): An Efficient Metaheuristic Algorithm for Solving Combinatorial Optimization Problems

Majid Sohrabi[1,2] [0000−0003−3695−604X], Amir M. Fathollahi-Fard[3] [0000−0002−5939−9795], and Vasilii A. Gromov[1] [0000−0001−5891−6597]

[1] School of Data Analysis and Artificial Intelligence, Faculty of Computer Science, HSE University, 11 Pokrovsky Boulevard, 109028, Moscow, Russian Federation
[2] Laboratory for Models and Methods of Computational Pragmatics, HSE University, Moscow, Russia Federation
(msohrabi@hse.ru)
(stroller@rambler.ru)
[3] School of Management Sciences, Université du Québec à Montréal, B.P. 8888, Succ. Centre-ville, Montréal, QC, H3C 3P8, Canada
(fathollahifard.amirmohammad@courrier.uqam.ca)

Abstract. Genetic Algorithms (GAs) are known for their efficiency in solving combinatorial optimization problems, thanks to their ability to explore diverse solution spaces, handle various representations, exploit parallelism, preserve good solutions, adapt to changing dynamics, handle combinatorial diversity, and provide heuristic search. However, limitations such as premature convergence, lack of problem-specific knowledge, and randomness of crossover and mutation operators make GAs generally inefficient in finding an optimal solution. To address these limitations, this paper proposes a new metaheuristic algorithm called the Genetic Engineering Algorithm (GEA) that draws inspiration from genetic engineering concepts. GEA redesigns the traditional GA while incorporating new search methods to isolate, purify, insert, and express new genes based on existing ones, leading to the emergence of desired traits and the production of specific chromosomes based on the selected genes. Comparative evaluations against state-of-the-art algorithms on benchmark instances demonstrate the superior performance of GEA, showcasing its potential as an innovative and efficient solution for combinatorial optimization problems.

Keywords: Genetic Algorithm · Metaheuristic Algorithms · Genetic Engineering · Combinatorial Optimization.

1 Introduction

Combinatorial optimization problems belonging to the class of NP-hard ones pose significant challenges in various domains, requiring efficient algorithms to

find optimal or near-optimal solutions. Genetic Algorithms (GAs) [1] have emerged as a popular choice due to their ability to explore diverse solution spaces, adapt to changing dynamics, and provide heuristic search. However, the limitations of GAs, including computational complexity, premature convergence, lack of problem-specific knowledge, and the need for parameter tuning, motivate the search for innovative approaches to enhance their efficiency [2]. These drawbacks encourage our attempts to redesign GA using the genetic engineering concept to be highly efficient for solving combinatorial optimization problems.

According to the literature on metaheuristic algorithms, GAs stand as the earliest population-based algorithms prioritizing the discovery of satisfactory solutions within a reasonable computational timeframe, rather than exclusively pursuing optimality [3]. While the field boasts an array of novel metaheuristic algorithms [4]– [18], it is worth noting that the literature on GAs is exceptionally rich, featuring numerous research contributions that introduce various GA variants equipped with advanced genetic programming and engineering techniques [19]– [27]. For instance, Gero and Kazakov [28] conducted a study focusing on the identification of useful genetic material while minimizing the presence of harmful genetic components, leading to the proposition of a novel GA. Kameya and Prayoonsri [29] introduced a GA-based approach grounded in pattern recognition to identify essential patterns within favorable chromosomes and protect them from undesirable crossovers. Ding et al. [30] delved into the integration of a back-propagation (BP) neural network with GA. Liang et al. [31] proposed a suite of adaptive elitist-population strategies that found application within the GA framework.

Additionally, Dasgupta et al. [32] integrated a load-balancing strategy for cloud computing with GAs. Elsayed et al. [33] enhanced GAs with a novel multi-parent crossover operator. Peng and Li [34] put forth an improved DV-Hop algorithm based on GAs, while Askarzadeh [35] explored memory-based GAs. Reddy et al. [36] developed a hybrid GA infused with fuzzy logic, and Fathollahi-Fard et al. [37] proposed a hybrid of GA with other innovative metaheuristics. Furthermore, Fathollahi-Fard et al. [38] devised a revised non-dominated sorting genetic algorithm by introducing novel search operators. Last but not least, Kolaee et al. [39] introduced a local search-based non-dominated sorting genetic algorithm tailored to solving routing problems within the tourism industry. However, none of the studies reviewed thus far have proposed the introduction of new search operators grounded in a diverse array of methods aimed at isolating, purifying, inserting, and expressing new genes within existing GA chromosomes, as we have undertaken in this study.

Recently, a wide range of population-based algorithms has been proposed to address challenging optimization problems. Examples include Cuckoo Search (CS) [4], Whale Optimization Algorithm (WOA) [5], Sine Cosine Algorithm (SCA) [6], Harris Hawks Optimization (HHO) [7], Squirrel Search Algorithm (SSA) [8], Red Deer Algorithm (RDA) [9], Sparrow Search Algorithm (SSA) [10], Capuchin Search Algorithm [11], Aquila Optimizer (AO) [12], Chameleon Swarm Algorithm (CSA) [13], Aptenodytes Forsteri Optimization (AFO) [14],

Dung Beetle Optimizer (DBO) [15], Beluga Whale Optimization (BWO) [16], and others. However, it is essential to note that the "No Free Lunch" theorem [17] suggests that no metaheuristic algorithm can outperform others for all optimization problems. Hence, there is a constant demand for the development of new metaheuristic algorithms that exhibit improved performance across different problem domains [18].

In this paper, we propose a new metaheuristic algorithm, the Genetic Engineering Algorithm (GEA), inspired by genetic engineering concepts. Genetic engineering encompasses a diverse range of methods used to isolate, purify, insert, and express new genes based on existing ones, resulting in the emergence of desired traits and the production of specific chromosomes based on the selected genes. By drawing parallels from this field, we aim to redefine the optimization process and overcome the limitations inherent in traditional GAs. The techniques used in GEA enable more precise manipulation of the optimization process, leveraging problem-specific insights and reducing randomness in mutation and crossover operations. By introducing the concept of gene manipulation within the population, GEA aims to enhance the exploration and exploitation of the solution space, leading to improved convergence and solution quality. To evaluate the effectiveness of GEA, we conduct extensive experiments on a set of benchmark instances and compare its performance against state-of-the-art metaheuristic algorithms. The results demonstrate the superior performance of GEA in terms of convergence speed, solution quality, and robustness. This signifies the potential of GEA as a novel and efficient approach for solving combinatorial optimization problems.

The rest of the paper is organized as follows: Section 2 presents the main inspiration of our GEA based on the genetic engineering concept. Section 3 studies the design and implementation details of the proposed GEA based on genetic engineering operators. Section 4 presents the experimental setup and discusses the comparative results with other algorithms. Finally, Section 5 concludes the paper, by emphasizing the significance of GEA as an innovative solution for efficient combinatorial optimization problems and outlining potential directions for future research.

2 Inspiration

Genetic engineering (GE) has transitioned from speculative fascination to a groundbreaking reality with wide-ranging applications. This methodology exhibits significant potential in disease treatment, exemplified by cancer immunotherapy [19], and the utilization of CRISPR technology to eliminate the HIV virus from infected cell genomes, offering prospects for a cure [20]. It extends to the realm of human genetics, impacting characteristics in newborns [21], and holds promise in agriculture for producing high-yield crops [22]. For instance, the development of Golden Rice, engineered to combat vitamin A deficiency and prevent blindness worldwide, underscores GE's potential [23].

Dominant chromosomes wield a pivotal influence on genetics, shaping specific trait expressions. Through meticulous manipulation, scientists can surpass or enhance genetic characteristics for desired outcomes. Precision in specifying dominant chromosomes related to plant yield, for instance, has led to high-yield species addressing global challenges such as climate change, pollution, and food shortages [24].

Directed mutations entail precise DNA alterations in organisms to induce advantageous changes. Researchers can introduce specific mutations in genes to promote beneficial traits or suppress harmful ones. This approach finds applications in medicine, agriculture, and environmental conservation. By directing mutations in disease-causing genes, scientists are developing innovative treatments for genetic disorders like cystic fibrosis and muscular dystrophy. Achieving this involves two critical steps: identifying crucial genes for enhancing traits and carefully manipulating non-informative genes to achieve desired outcomes [23].

Desired genes possess specific qualities or capabilities that scientists aim to introduce into an organism. These genes enhance crop resistance in agriculture, contributing to sustainable farming and increased food production. In medicine, they hold the key to curing genetic disorders [25]. Targeting desirable genes through mutation takes genetic engineering to a new level of precision. Techniques like CRISPR-Cas9 [26] enable precise gene editing, allowing correction of mutations responsible for diseases like sickle cell anemia or Huntington's disease, offering hope to millions [24].

Gene injection, another innovative approach, delivers therapeutic genes directly into the body to treat or prevent diseases. This method holds promise for conditions like cancer and cardiovascular disorders. By injecting genes producing therapeutic proteins, scientists can enhance natural defense mechanisms, stimulate tissue regeneration, or target cancer cells. Gene injection therapy represents a powerful tool in personalized medicine [25].

In conclusion, genetic engineering is a groundbreaking reality with diverse applications in medicine, agriculture, and environmental conservation. Through dominant chromosomes, directed mutations, and desired gene identification, scientists shape genetics for high-yield crops and disease treatment. Techniques like precise gene editing and gene injection therapy enable unprecedented precision and customization. As genetic engineering advances, it promises to revolutionize various domains, ushering in an era of personalized medicine and sustainability. Building on these GE techniques, this paper proposes a novel metaheuristic algorithm, GEA.

3 Proposed GEA

While the GA [1] is a well-established evolutionary algorithm that commonly employs classical mutation and crossover operators, this study proposes a novel approach by incorporating GE techniques. The overall flowchart for the proposed GEA is shown in Figure 1. One can skip any operator from Crossover to Gene Injection to customize the algorithm for different purposes and check the

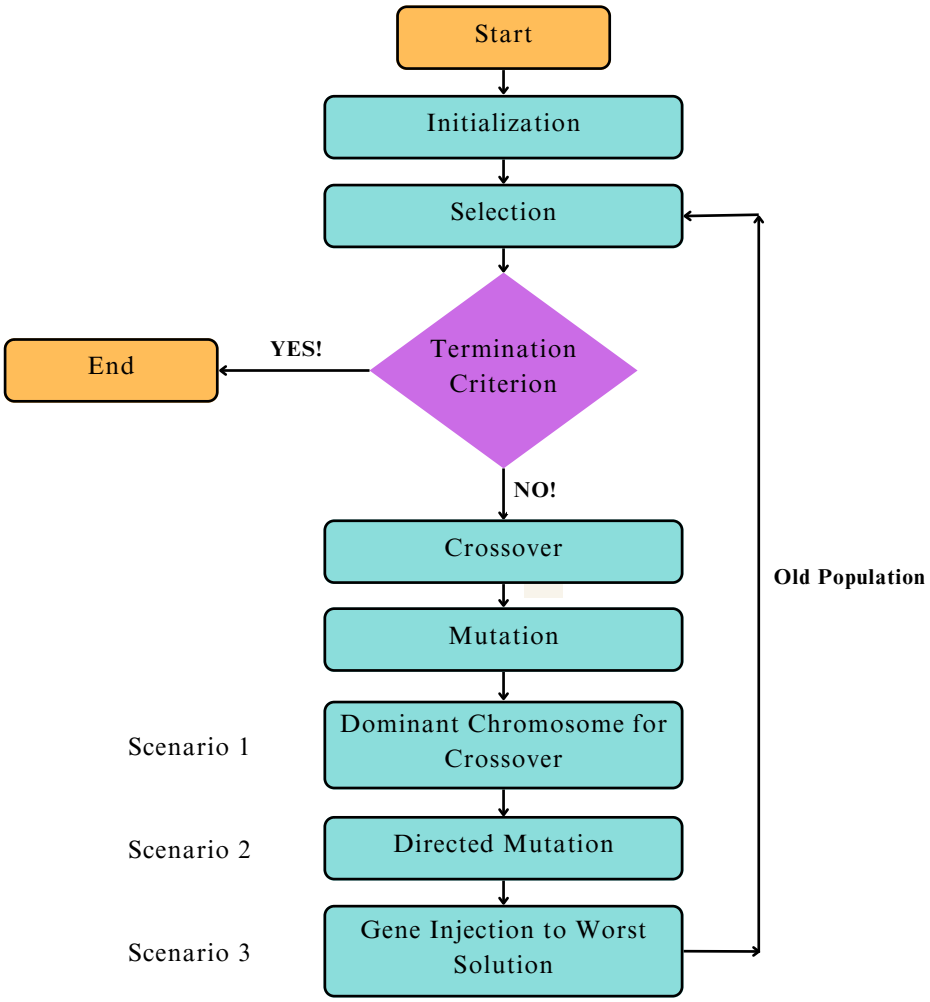

Fig. 1: Flowchart for the proposed GEA.

performance of the method with partial operators. The GEA similar to other meta-heuristics starts with an initial population that is the counterpart of this method. This algorithm encompasses a diverse range of methods used to isolate, purify, insert, and express specific genes within a host organism, ultimately leading to the emergence of desired traits and the production of specific chromosomes based on the selected genes. The flowchart of the proposed GEA is shown in Figure 1.

After generating the initial population, all individuals will be evaluated based on the specific fitness function. It is worth noting that each problem is defined by a specific and unique fitness function to present a solution in combinatorial optimization. We can classify different integer programming problems such as vehicle routing optimization, flow-shop scheduling problems, knapsack problems, and facility location planning as combinatorial optimization problems [18]. The chromosome definition or the solution presentation in each type of combinatorial optimization problem is different. For example, the chromosome in routing optimization is defined as the sequence of visits [2]. In flow-shop scheduling problems, the chromosome is considered as the sequence of a set of jobs on machines [18]. In addition, the solution definition in facility location planning and knapsack problems is defined by 0-1 or binary variables [3]. In our GEA, the examples for explaining the search operators are based on a binary chromosome where each gene may be zero or one. In this new metaheuristic algorithm, in addition to mutation and crossover operators, we have three genetic engineering operators as three scenarios explained as follows:

Senario 1. **Finding Dominant Chromosome (most repeated genes):** The first scenario of GEA focuses on identifying the dominant chromosome by considering a percentage, denoted as $p\%$, of the best individuals in the population. The value of p is initially defined by the user and can be optimized based on the specific problem at hand. A chromosome is deemed dominant if it possesses the highest number of repeated genes among the best $p\%$ of individuals. The process of identifying the dominant genes and constructing the dominant chromosome is outlined by Equations (1) and (2). To provide clarity on this operation, Figure 2 illustrates an example of finding the dominant chromosome from the elite population. Additionally, the pseudocode for this operation is presented in Algorithm 1.

$$RM_i = [\Sigma_{j=1}^{M} gene_j] \tag{1}$$

$$DC = max(RM) \tag{2}$$

Where M, RM, and DC represent the number of individuals in $p\%$ of the population, repetition matrix, and dominant chromosome respectively.

Senario 2. **Directed Mutation:** The second scenario in GEA focuses on improving the effectiveness of the mutation operator, which is essential for preventing the algorithm from getting trapped in local optima. In traditional GAs, random mutation is often used, which can be a drawback. To address this limitation, the mutation operator in GEA is targeted and modified to enhance the

Fig. 2: Finding dominant genes from $p\%$ population.

efficiency of the genetic algorithm. In this scenario, specific methods are proposed to direct the mutation process, rather than relying on random selection. One such method involves the detection of desired genes, which enables the algorithm to focus on genes that are known to contribute to desired traits or outcomes. By targeting the mutation process and removing the randomness associated with traditional mutation, the performance of the genetic algorithm can be significantly improved. To provide an illustration of the directed mutation operator, Figure 3 presents an example that demonstrates how the process works. This targeted mutation approach allows the algorithm to prioritize specific genes and introduce beneficial changes in a controlled manner. By doing so, GEA can overcome the limitations of random mutation and enhance its ability to explore the solution space effectively.

$$f(x) = \begin{cases} 1 & \text{if } M_{ij} \ desired \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

– **Desired Gene:** The first step for applying this operator is to find the most repeated genes out of $p\%$ of the best chromosomes which are considered desired genes. The goal here is to consider fixed genes because these genes are considered the most informative elements for generating the elite part of the population. So, their existence in the solution will help the population to stay in the elite part, and by the slight change, they may move towards the global optimum in the near future. This step will generate a pattern matrix with $n * m$. where n is the number of populations in the elite part of the whole population (best $p\%$), and m is the number of genes inside a chromosome which is known as the number of the variables in the problem. This pattern matrix consists of binary elements, in which 1 represents the specific chromosome that is desired and should be fixed, and 0 represents

uninformative genes for which mutation is allowed to be applied. Equation (3) represents how the pattern matrix will be generated. If the number of repetitions reaches a specific threshold then the gene will be desired. The threshold will be specified by the user from the beginning and the parameter can be optimized based on different problems and purposes.

– **Mutation Targeting Desirable Genes:** After generating the pattern for $p\%$ chromosomes with the highest fitness values, a candidate by the roulette wheel will be selected, the mutation applies only on uninformative genes which represent zero in their corresponding pattern matrix. By applying targeting mutation we hope to search the solution space in the proper manner to find the global optimum. The engineered mutation will be repeated to the number of overall mutations only on uninformative genes to invest in the elite part of the population for faster convergence.

Senario 3. **Gene Injection:** The third scenario in GEA focuses on the importance of considering the entire population, including the individuals with the worst fitness values. While the first two scenarios primarily focus on the elite part of the population, it is essential to recognize that even the worst solutions can contribute to the overall improvement of the algorithm. In optimization algorithms, the worst individuals should not be overlooked, as they also possess the potential for beneficial changes. In this scenario, we aim to invest in the worst individuals and employ an engineering approach to enhance their performance. By making slight changes to the worst individuals, they have the opportunity to move towards the global optimum and eventually become part of the elite solutions in subsequent iterations of the algorithm.

To accomplish this, a patterns matrix is constructed for the elite part of the population. Then, individuals from the non-elite part of the population (representing $1 - p\%$) are selected. Based on the pattern matrix, genes from the chromosome with the highest repetition are injected into the selected chromosome. This gene injection operator facilitates the transfer of beneficial genetic information from the dominant chromosome to other individuals in the population, enabling them to improve and contribute to the overall optimization process. Figure 4 provides an example that illustrates how this proposed gene injection operator works. It demonstrates the process of transferring genetic information to enhance the selected chromosome. The dominant chromosome, as coded in Algorithm 1, is necessary for the implementation of this gene injection operator.

By incorporating this third scenario into the GEA, we can harness the potential of even the worst solutions in driving the algorithm toward the global optimum. This approach allows for a more comprehensive exploration of the solution space and facilitates the improvement of the entire population over successive iterations.

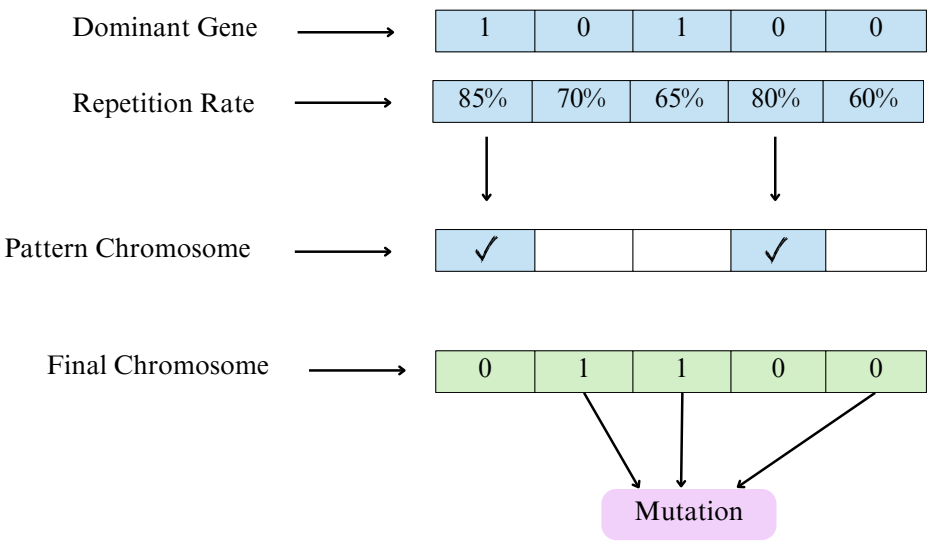

Fig. 3: Directed mutation by fixing informative genes.

Algorithm 1 Dominant Chromosome

Data: Pop, Prob. Info.
Result: DominantGene, Mask, MaskInverted

while *i less than chromosome length* **do**
  while *j less than No. of Pop.* **do**
    $Genes \leftarrow [Genes, Pop_j(i)]$
  end
  while *there is element in Genes* **do**
    $temp \leftarrow sum(Genes == Genes(1))$
    if *size DominantGene* $== 0$ **then**
      $DominantGene \leftarrow Genes(1)$
      $DominantGeneCounter \leftarrow temp$
    else
      if $temp > DominantGeneCounter$ **then**
        $DominantGene \leftarrow Genes(1)$
        $DominantGeneCounter \leftarrow temp$
      else
        $DominantGene \leftarrow [DominantGene, Genes(1)]$
      end
    end
  end
end
$Mask \leftarrow zeros(1, size(chromosome))$
while *i less than chromosome length* **do**
  if *(DominantGeneCount > threshold) and (threshold not 0)* **then**
    $Mask(i) \leftarrow 1$
  end
end
$MaskInverted \leftarrow notMask$

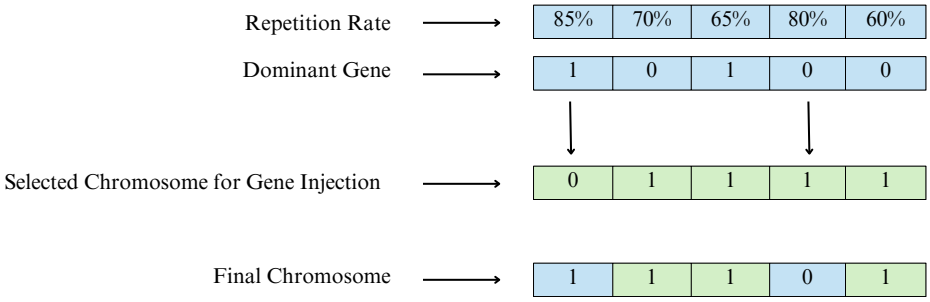

Fig. 4: Injecting informative genes to wort individuals of the population.

4 Experimental Results

Here, we present a comprehensive evaluation of the GEA and demonstrate its efficacy in solving combinatorial optimization problems, particularly in the context of a standard vehicle routing optimization problem. This problem involves determining optimal routes for a fleet of vehicles to visit a set of demand points while minimizing transportation costs. To evaluate the performance of GEA, we compare it not only against the traditional GA but also against three variations of GEA, namely GEA1, GEA2, and GEA3, each utilizing a specific scenario as explained earlier. In GEA, the main loop randomly selects one of these scenarios at each iteration.

For our evaluation, we select six well-established instances from the literature, as referenced by [18] and [9], to benchmark the algorithms. To ensure consistency, we set the maximum number of iterations to 1000 and the population size to 100 for all algorithms. The crossover and mutation percentages are uniformly set to 0.8 and 0.1, respectively, across all algorithms. Moreover, in the case of GEA, the percentages of scenarios considered are 0.5, 0.5, and 0.2 for the first, second, and third scenarios, respectively.

To gauge the performance of the algorithms, we conduct 10 independent runs of each algorithm on every test instance. Subsequently, we report the best, worst, average, and standard deviation of the solutions obtained by each algorithm in Table 1. These results enable us to analyze the robustness of the applied metaheuristic algorithms. Furthermore, Figure 5 illustrates the convergence rate of each algorithm towards its best performance, providing a visual representation of their effectiveness. To complement our analysis, we perform statistical analyses using a 0.95 confidence level, employing normalized standard deviations across all algorithms. The interval plot in Figure 6 showcases the results of these statistical analyses, shedding light on the comparative performance and reliability of the algorithms.

Based on the results presented in Table 1, our findings indicate that GEA, when utilizing all scenarios, outperforms the other algorithms. In most instances,

Table 1: Report of the algorithms results based on criteria of the Best=B, Worst=W, Mean=M, and Standard deviation=Std. (The best values in each criterion and test instance are highlighted in bold.)

Test instance		F1	F2	F3	F4	F5	F6
Demand points x Number of vehicle		8x3	10x3	14x4	20x4	25x5	30x5
GA	B	**257.3492**	**268.1687**	**301.6661**	317.6503	326.5457	308.8542
	W	291.6624	269.0742	316.0882	351.2298	363.0131	343.9097
	M	260.7805	268.7120	305.8615	333.5178	338.1742	321.1532
	Std	10.8507	0.4675	5.4002	12.3733	11.3976	11.5798
GEA_1	B	**257.3492**	**268.1687**	**301.6661**	319.3303	319.5602	307.1991
	W	**257.3492**	269.0742	318.8057	342.2278	359.0854	370.7047
	M	**257.3492**	268.2593	304.7409	324.0378	330.8722	328.0479
	Std	**5.99E-14**	0.2863	5.4787	6.8149	10.8851	21.2861
GEA_2	B	**257.3492**	**268.1687**	**301.6661**	**317.1235**	321.5556	**302.5377**
	W	**257.3492**	269.0742	306.3834	353.7992	359.0854	**322.7266**
	M	**257.3492**	268.3498	302.8296	327.4803	333.1713	**311.4745**
	Std	**5.99E-14**	0.3817	1.6555	12.1645	13.4915	**7.3910**
GEA_3	B	**257.3492**	**268.1687**	**301.6661**	317.6503	319.0169	308.8834
	W	**257.3492**	269.0742	306.3834	**331.8416**	**331.3571**	346.2497
	M	**257.3492**	268.4404	302.3684	323.3476	326.245	323.5826
	Std	**5.99E-14**	0.4373	1.58597	5.45505	**4.1059**	13.7657
GEA	B	**257.3492**	**268.1687**	**301.6661**	317.6503	**317.7347**	304.4598
	W	**257.3492**	**268.1687**	**301.6661**	**331.8416**	331.4877	343.6004
	M	**257.3492**	**268.1687**	**301.6661**	**321.4611**	**323.1822**	313.4242
	Std	**5.99E-14**	**5.99E-14**	**0**	**4.7726**	6.0097	11.8294

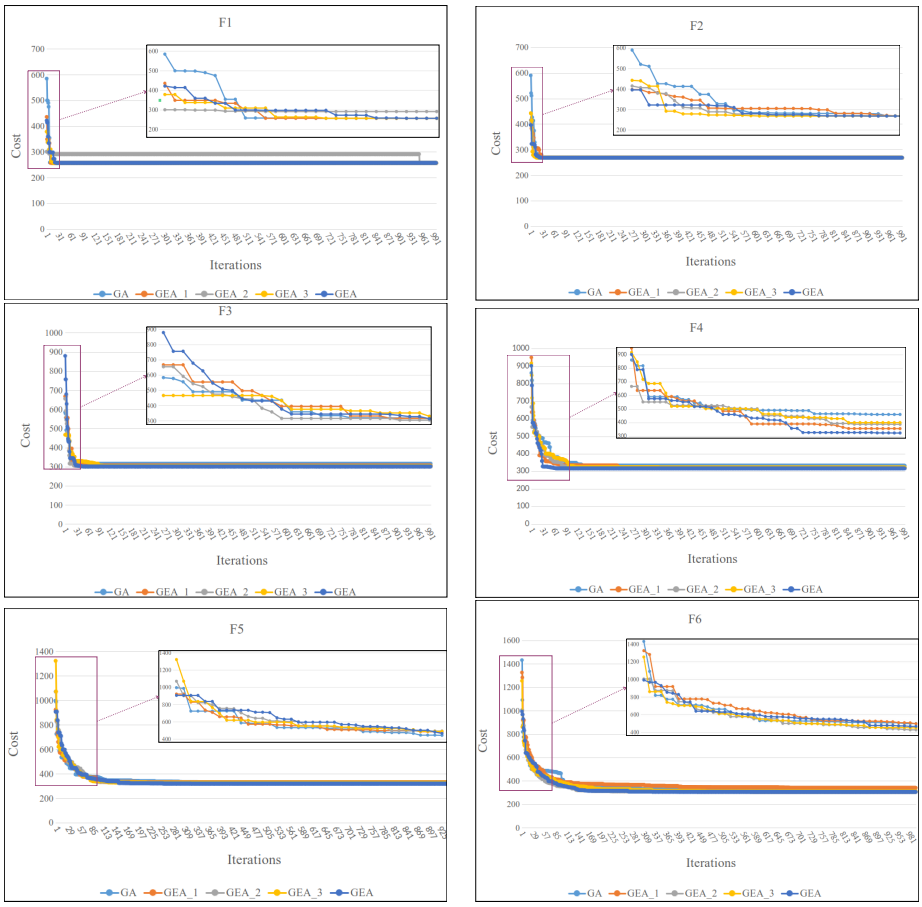

Fig. 5: Convergence rate of the metaheuristic algorithms in all the benchmarked instances.

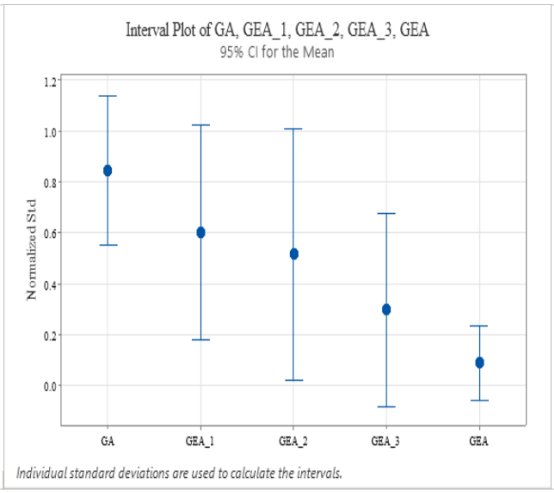

Fig. 6: Interval plot based on 95% confidence level for analyzing the robustness of the metaheuristic algorithms

this algorithm consistently discovers near-optimal solutions superior to those obtained by GA and the other GEA variants. Among the GEA variations, GEA2 stands out as the most successful, confirming the strength of the second scenario in exploring better near-optimal solutions. Figure 5 demonstrates that all algorithms exhibit an acceptable convergence rate across the test instances, with similar solution quality. However, the statistical analyses from Figure 6 conclusively support the highest accuracy of our GEA compared to the other algorithms.

In conclusion, our comprehensive evaluation showcases the effectiveness of the GEA in solving combinatorial optimization problems, specifically the vehicle routing optimization problem. The results presented in Table 1 and Figures 5 and 6 highlight the superior performance and accuracy of GEA, particularly when incorporating all scenarios. These findings provide compelling evidence for the potential of GEA as a robust and reliable metaheuristic algorithm for addressing optimization challenges.

5 Conclusion and Future Work

This study presented a comprehensive evaluation of the GEA and its effectiveness in solving combinatorial optimization problems, focusing on the vehicle routing optimization problem. The results obtained through benchmarking against the traditional GA and different variations of GEA demonstrated the superiority of GEA, particularly when incorporating all scenarios. GEA consistently outperformed other algorithms, yielding better near-optimal solutions in most instances.

The findings of this study contribute to the growing body of research on metaheuristic algorithms and their applications in optimization problems. The

success of GEA in addressing the vehicle routing optimization problem showcases its potential in real-world scenarios where efficient transportation routing is critical. These results provide valuable insights into the efficacy of genetic engineering techniques based on our three scenarios for the development of GEA in solving combinatorial optimization problems and highlight the importance of considering different scenarios in the algorithm design.

Moving forward, several areas for future research can be identified. Firstly, further investigation can be conducted to explore the impact of different parameter settings on the performance of GEA and its variations. Fine-tuning the algorithm's parameters may enhance its effectiveness and lead to even better solutions. Extending the evaluation to other optimization problems and comparing GEA against state-of-the-art algorithms would provide a broader perspective on its performance and competitiveness. Moreover, integrating GEA with other optimization techniques or hybridizing it with machine learning approaches could further enhance its capabilities. Combining the strengths of genetic engineering algorithms with other intelligent algorithms may lead to novel hybrid approaches with improved optimization performance. Lastly, conducting experiments on larger problem instances and assessing the scalability and efficiency of GEA would be beneficial [40]. Scaling up the problem size would provide insights into the algorithm's performance when dealing with more complex and larger-scale optimization challenges [41].

Acknowledgements This paper is an output of a research project implemented as part of the Basic Research Program at the National Research University Higher School of Economics (HSE University). We gratefully acknowledge the HPC facilities at HSE University [27] for providing computational resources. In addition, we would like to acknowledge that the publications from Prof. Mostafa Hajiaghaei-Keshteli inspire us to develop a new metaheuristic algorithm.

References

1. Holland, J., (1975). Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press.
2. Elshaer, R., and Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers and Industrial Engineering, 140, 106242.
3. Katoch, S., Chauhan, S. S., and Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80, 8091-8126.
4. Yang, X. S., and Deb, S. (2010). Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation, 1(4), 330-343.
5. Mirjalili, S., and Lewis, A. (2016). The whale optimization algorithm. Advances in engineering software, 95, 51-67.
6. Mirjalili, S. (2016). SCA: a sine cosine algorithm for solving optimization problems. Knowledge-based systems, 96, 120-133.

7. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., and Chen, H. (2019). Harris hawks optimization: Algorithm and applications. Future generation computer systems, 97, 849-872.
8. Jain, M., Singh, V., and Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm and evolutionary computation, 44, 148-175.
9. Fathollahi-Fard, A. M., Hajiaghaei-Keshteli, M., and Tavakkoli-Moghaddam, R. (2020). Red deer algorithm (RDA): a new nature-inspired meta-heuristic. Soft Computing, 24, 14637-14665.
10. Xue, J., and Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. Systems science and control engineering, 8(1), 22-34.
11. Braik, M., Sheta, A., and Al-Hiary, H. (2021). A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural computing and applications, 33, 2515-2547.
12. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A., and Gandomi, A. H. (2021). Aquila optimizer: a novel meta-heuristic optimization algorithm. Computers and Industrial Engineering, 157, 107250.
13. Braik, M. S. (2021). Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. Expert Systems with Applications, 174, 114685.
14. Yang, Z., Deng, L., Wang, Y., and Liu, J. (2021). Aptenodytes forsteri optimization: Algorithm and applications. Knowledge-Based Systems, 232, 107483.
15. Xue, J., and Shen, B. (2023). Dung beetle optimizer: A new meta-heuristic algorithm for global optimization. The Journal of Supercomputing, 79(7), 7305-7336.
16. Zhong, C., Li, G., and Meng, Z. (2022). Beluga whale optimization: A novel nature-inspired metaheuristic algorithm. Knowledge-Based Systems, 251, 109215.
17. Wolpert, D. H., and Macready, W. G. (1997). No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1), 67-82.
18. Fathollahi-Fard, A. M., Hajiaghaei-Keshteli, M., and Tavakkoli-Moghaddam, R. (2018). The social engineering optimizer (SEO). Engineering applications of artificial intelligence, 72, 267-293.
19. Li, D., Li, X., Zhou, W. L., Huang, Y., Liang, X., Jiang, L., and Wang, W. (2019). Genetically engineered T cells for cancer immunotherapy. Signal Transduction and Targeted Therapy, 4(1), 35.
20. Xiao, Q., Guo, D., and Chen, S. (2019). Application of CRISPR/Cas9-based gene editing in HIV-1/AIDS therapy. Frontiers in cellular and infection microbiology, 9, 69.
21. Raposo, V. L. (2019). The first Chinese edited babies: a leap of faith in science. JBRA assisted reproduction, 23(3), 197.
22. Li, C. (2020). Breeding crops by design for future agriculture. Journal of Zhejiang University. Science. B, 21(6), 423.
23. Dubock, A. (2019). Golden rice: to combat vitamin A deficiency for public health. Vitamin A, 1.
24. Huang, T. K., and Puchta, H. (2021). Novel CRISPR/Cas applications in plants: from prime editing to chromosome engineering. Transgenic research, 30, 529-549.
25. Shahryari, A., Saghaeian Jazi, M., Mohammadi, S., Razavi Nikoo, H., Nazari, Z., Hosseini, E. S., and Lickert, H. (2019). Development and clinical translation of approved gene therapy products for genetic disorders. Frontiers in genetics, 10, 868.

26. Zhuo, C., Zhang, J., Lee, J. H., Jiao, J., Cheng, D., Liu, L., and Li, M. (2021). Spatiotemporal control of CRISPR/Cas9 gene editing. Signal Transduction and Targeted Therapy, 6(1), 238.
27. Kostenetskiy P.S., Chulkevich R.A., and Kozyrev V.I. (2021). HPC Resources of the Higher School of Economics, Journal of Physics: Conference Series, 1740(1), 012050. https://doi.org/10.1088/1742-6596/1740/1/012050
28. Gero, J. S., and Kazakov, V. (2001). A genetic engineering approach to genetic algorithms. Evolutionary computation, 9(1), 71-92.
29. Kameya, Y., and Prayoonsri, C. (2011). Pattern-based preservation of building blocks in genetic algorithms. In 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 2578-2585.
30. Ding, S., Su, C., and Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. Artificial intelligence review, 36, 153-162.
31. Liang, Y., and Leung, K. S. (2011). Genetic algorithm with adaptive elitist-population strategies for multimodal function optimization. Applied Soft Computing, 11(2), 2017-2034.
32. Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., and Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology, 10, 340-347.
33. Elsayed, S. M., Sarker, R. A., and Essam, D. L. (2014). A new genetic algorithm for solving optimization problems. Engineering Applications of Artificial Intelligence, 27, 57-69.
34. Peng, B., and Li, L. (2015). An improved localization algorithm based on genetic algorithm in wireless sensor networks. Cognitive Neurodynamics, 9, 249-256.
35. Askarzadeh, A. (2017). A memory-based genetic algorithm for optimization of power generation in a microgrid. IEEE transactions on sustainable energy, 9(3), 1081-1089.
36. Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Rajput, D. S., Kaluri, R., and Srivastava, G. (2020). Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis. Evolutionary Intelligence, 13, 185-196.
37. Fathollahi-Fard, A. M., Dulebenets, M. A., Hajiaghaei–Keshteli, M., Tavakkoli-Moghaddam, R., Safaeian, M., and Mirzahosseinian, H. (2021). Two hybrid metaheuristic algorithms for a dual-channel closed-loop supply chain network design problem in the tire industry under uncertainty. Advanced engineering informatics, 50, 101418.
38. Fathollahi-Fard, A. M., Tian, G., Ke, H., Fu, Y., and Wong, K. Y. (2023). Efficient Multi-objective Metaheuristic Algorithm for Sustainable Harvest Planning Problem. Computers and Operations Research, 106304.
39. Kolaee, M. H., Al-e, S. M. J. M., and Jabbarzadeh, A. (2023). A local search-based non-dominated sorting genetic algorithm for solving a multi-objective medical tourism trip design problem considering the attractiveness of trips. Engineering Applications of Artificial Intelligence, 124, 106630.
40. Du, D., and Pardalos, P. M. (1998). Handbook of combinatorial optimization (Vol. 4). Springer Science & Business Media.
41. Mart, R., Pardalos, P. M., and Resende, M. G. (2018). Handbook of heuristics. Springer Publishing Company, Incorporated.