# The Future of Route Optimization: Analysis of Merging Genetic Algorithms with Wisdom of Crowds for Solving the Chinese Postman Problem

Gobin Bastola, Ahmed Abdullahi, Collin Dewey, Chase Zebarth
Department of Computer Science and Engineering, Speed School of Engineering, University of Louisville, Louisville, Kentucky, USA

*Abstract* – **This paper explores the integration of Genetic Algorithms (GA) and the Wisdom of Crowds (WoC) approach for solving the Chinese Postman Problem (CPP). The study investigates whether combining these two methods enhances the route optimization capabilities of traditional GAs. By analyzing various factors such as population size, mutation rates, and the number of nodes, the research evaluates the efficiency and adaptability of the proposed method. Additionally, the paper delves into alternative approaches like Prim's Algorithm for comparative analysis. The findings indicate that while the combined GA and WoC approach shows promise in optimizing route solutions, it also presents challenges, particularly in handling complex graphs. The paper concludes with potential future research directions, including exploiting modern computing capabilities like GPUs for better performance and exploring other algorithms for further advancements in solving CPP.**

*Index Terms* – **Chinese Postman Problem (CPP), Genetic Algorithm (GA), Wisdom of Crowds (WoC), Route Optimization, Algorithm Efficiency, Metaheuristic, Population Size, Mutation Rates, Prim's Algorithm, Graph Theory, Computational Complexity**

## I. Introduction

THE Genetic Algorithm (GA) was our main choice of algorithm for solving the Chinese Postman Problem, with the additional testing of using the Wisdom of the Crowds (WoC) approach to see if it could be used to further improve the pathing results given by the GA.

Our team chose the use of a Genetic Algorithm as it offered a powerful and flexible approach to solving the route inspection problem. This can be seen in one of the main facets that set GA's apart from other algorithms, that is its ability to explore solutions to irregular structures, such as is seen in this problem. The GA is also better at dealing with non-linearities, which are present in this as it is a routing problem. It is also good at avoiding local optima, thanks to the mutation function that was implemented.

There were, however, also some shortcomings as a result of using a genetic algorithm to solve the CPP. One of the largest issues was the complexity of finding the proper parameters for the fitness testing of generations.. There were also issues in regard to the data used for the problem. An adjacency matrix was used to represent the nodes and the edges and those edges distances, but this proved difficult as the genetic algorithm would, at times, choose routes that should not exist and use those to find "better" solutions, despite the fact that they were invalid.

As mentioned in the abstract, the wisdom of crowds approach was also tested to determine if it could further optimize the routes given by the running of the genetic algorithm. The results of this will be discussed more in later sections of the paper.

The program can have certain parameters set at the beginning, such as population size, generations, and mutation rate. The user can also select a dataset that they want to use from their computer's local files. These datasets can be generated using the separate program that will be provided with this project. Running these datasets will then go on to provide the user with two forms of feedback. One of these is a graphical representation of all of the solutions and paths taken, with the distance included. The other is an output to the console, in which the user will be able to observe the route taken and the distance for each iteration of the genetic algorithm, as well as the results from the wisdom of the crowds approach, with the same data being included.

## II. Prior work

For the literature review on the application of the Wisdom of Crowds (WoC) in optimization algorithms, the following sources provide a wealth of information on the topic:

**"Wisdom of Artificial Crowds—A Metaheuristic Algorithm for Optimization"**:
This research highlights a metaheuristic algorithm inspired by human collective intelligence, known as Wisdom of Artificial Crowds (WoAC)[1]. The algorithm capitalizes on the capabilities of a group of simulated intelligent agents that independently arrive at solutions. These solutions are then aggregated to form a superior collective solution. The Wisdom of Artificial Crowds is categorized as a metaheuristic algorithm.

Metaheuristic algorithms are problem-solving techniques that provide approximate solutions for complex optimization problems where finding an exact solution is either too time-consuming or practically impossible. WoAC falls under this category as it uses a heuristic approach to optimize solutions. Simulated Intelligent Agents x- In this algorithm, a group of simulated intelligent agents is used. These agents can be thought of as virtual entities or computer programs capable of making decisions and exploring the solution space of the optimization problem. These agents operate independently and are responsible for generating potential solutions.

**"Effect of Crowd Composition on the Wisdom of Artificial Crowds"**:
This paper delves into the factors that affect the success of the WoAC metaheuristic, particularly task difficulty and crowd composition[2]. It underscores the variable success of the aggregate solution produced by the metaheuristic, which may not always surpass the individual contributions it combines.

**"Evolution of a Metaheuristic for Aggregating Wisdom from Artificial Crowds"**:
This source discusses a post-processing metaheuristic that leverages the wisdom among a crowd of stochastic outcomes. If multiple search instances are executed, the resulting data can be aggregated to formulate a new solution that potentially exceeds any individual result from the collective pool.[3]

**"Wisdom of artificial crowds algorithm for solving NP-hard problems"**:
This publication by Roman V. Yampolskiy and Ahmed EL-Barkouky presents an algorithm inspired by the wisdom of crowds, specifically designed for solving NP-hard problems. The algorithm demonstrates an average improvement of 6%-9% in solution quality compared to traditional genetic algorithms. This significant enhancement underscores the potential of applying WoC principles in tackling complex computational problems.[4]

## III. APPROACH

The Chinese Postman Problem (CPP) is a classic problem in graph theory and combinatorial optimization. It challenges us to find the shortest closed path or circuit that visits every edge of a graph. This problem finds real-world applications in diverse fields like postal delivery, garbage collection, and street sweeping. Solving the CPP efficiently is vital for optimizing resource allocation and minimizing costs in these domains.

Edge Matrix and Vertex Matrix Construction[6]:
The foundation of our approach lies in the construction of two essential matrices: the edge matrix and the vertex matrix.

Edge Matrix: This matrix is a critical component as it encodes the connectivity and weights between different vertices in the graph. Each cell in the matrix stores the weight or cost associated with the edge connecting two vertices. It effectively captures the topological and cost-related information of the graph, allowing the Genetic Algorithm (GA) to make informed decisions. For example, in a postal delivery scenario, this matrix would represent distances between different locations.

Expanding upon the fundamental principles of graph theory, as explored by Bang-Jensen and Gutin [10], our approach to the edge matrix goes beyond just cataloging distances or costs. It delves into the more intricate aspects of graph topology. This means we're looking at different weights for various routes and are agile enough to accommodate changes in the graph's structure, like adding or removing edges.

Vertex Matrix: While the edge matrix focuses on edges, the vertex matrix provides detailed information about each vertex in the graph. It includes properties such as the degree of a vertex (the number of edges connected to it), its connectivity to other vertices, and any specific constraints or requirements related to that vertex. This matrix serves as a reference for the GA to understand the graph's structure and any specific conditions that must be satisfied. Complementing the edge matrix, this matrix details each vertex's degree, connectivity, and any specific constraints, aiding the GA in understanding the graph's overall layout [11]

Advanced Genetic Algorithm (GA):
Our approach to solving the CPP relies on an advanced Genetic Algorithm, which is a bio-inspired heuristic known for its ability to find optimized solutions to complex problems. Here, we'll explore the workings of our GA in greater detail.[9]

'Wisdom of Crowds' Principle: We have integrated the 'wisdom of crowds' concept into our GA. This technique involves aggregating insights from multiple solutions to form a superior collective solution.[12] In practical terms, it means that our GA considers not only individual solutions but also leverages the knowledge gained from a pool of solutions. This approach is particularly valuable in avoiding local optima, where the GA might get stuck, and ensures a more exhaustive exploration of the solution space.[19]

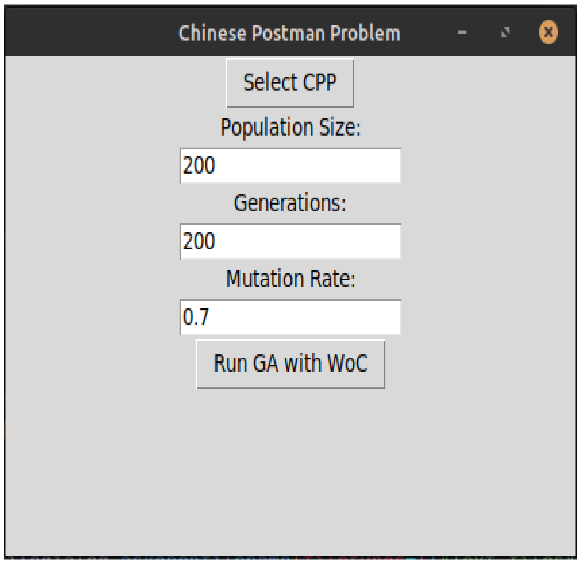
Adaptive Mutation Rates: The GA implements adaptive mutation rates. These rates dynamically change based on the algorithm's progress. In the early stages, the algorithm uses higher mutation rates to explore a wide range of potential solutions. As it progresses and converges towards optimal or near-optimal solutions, the mutation rates decrease to fine-tune and refine the solutions. This adaptive strategy strikes a balance between exploration and exploitation, enhancing the GA's efficiency.

Tournament Selection[5]: Unlike traditional selection methods, our GA employs tournament selection. In this process, a group of individuals (solutions) competes against each other, and the fittest ones are selected for reproduction. Tournament selection provides a better balance between exploring diverse solutions and exploiting the best-performing ones, enhancing the algorithm's convergence speed and solution quality.

Handling Multi-Objective Problems[7]: Our GA is not limited to single-objective optimization. It can handle multi-objective problems, where conflicting objectives need to be balanced. This capability allows the algorithm to find solutions that maintain a trade-off between different competing objectives. For example, in the context of garbage collection, the algorithm can consider minimizing both the distance traveled and the collection time, providing a versatile solution.

Flexibility and Adaptability[8]: One of the key strengths of our advanced GA is its flexibility and adaptability. It can be applied to a wide range of problems, from optimization challenges in engineering to complex decision-making in artificial intelligence. The algorithm's adaptability allows it to be tailored to specific problem characteristics, ensuring relevance and effectiveness across various domains. Whether it's a large-scale postal delivery network or a complex urban street-sweeping problem, our GA can be customized to meet the unique requirements of each scenario.

Our approach to solving the Chinese Postman Problem represents a state-of-the-art solution in computational problem-solving. By integrating traditional GA mechanisms with modern enhancements such as the 'wisdom of crowds' principle and adaptive mutation rates, we not only promise to find solutions where standard methods may fail but also do so with greater efficiency and adaptability. This advanced Genetic Algorithm serves as a powerful tool for addressing the complex challenges present in today's dynamic problem spaces, offering optimized solutions across a wide spectrum of real-world applications.



Population Size: 200

Generations: 200

Mutation Rate: 0.7

The GUI shown in the image is a simple and user-friendly interface for running a Genetic Algorithm (GA) with the Wisdom of Crowds (WoC) approach to solve the Chinese Postman Problem (CPP). Here's an in-depth explanation of its components and the rationale behind the chosen parameters:

**Select CPP Button:**

This button triggers a file dialog that allows the user to select and load the graph data needed to run the GA on the CPP. It's essential for users to import different test cases or graph data without modifying the code.

**Population Size Field:**

Set to 200, this input field allows the user to specify the size of the population in the GA. A larger population size can provide a more diverse set of potential solutions, increasing the chance of finding an optimal or near-optimal solution but may also increase computation time.

**Generations Field:**

Also set to 200, this determines the number of iterations the GA will perform. More generations mean the algorithm has more opportunities to improve the solutions, but like population size, more generations will require more computational resources.

**Mutation Rate Field:**

With a default value of 0.7, this is a relatively high mutation rate indicating that the algorithm will introduce a significant amount of random changes to the solutions. This can be useful in exploring the solution space more thoroughly and preventing premature convergence to local optima.

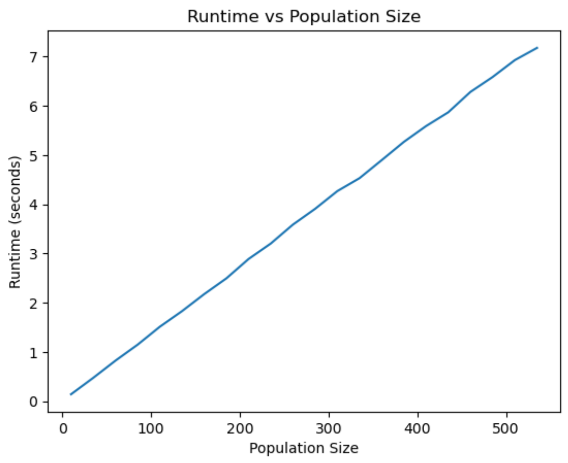
**Run GA with WoC Button:**

This button initiates the GA process. When clicked, it runs the GA with the parameters provided in the other fields. The WoC approach suggests that the algorithm doesn't just rely on a single GA run but aggregates the results from multiple runs to find a better collective solution, potentially leading to more robust and reliable results.
The choices of 200 for both population size and generations, and 0.7 for the mutation rate, are based on prior experimentation that proved these values strike a good balance between solution quality and computational effort for problems of a certain size or complexity. The specific values are defaults that work well in general or starting points for the user to experiment with.

## IV. RESULTS & DISCUSSION

The objective of our research was to assess the efficacy of a Genetic Algorithm (GA) integrated with the Wisdom of Crowds (WoC) approach in solving the Chinese Postman Problem (CPP). We focused on determining whether this innovative combination could enhance the path-finding results beyond what is achievable with a traditional GA. Our experimentation involved generating multiple test cases with varying complexities to simulate real-world scenarios.

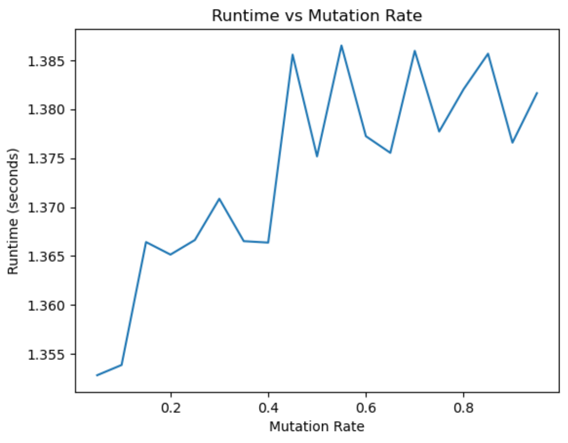
**Runtime vs Population Size:** This graph shows a linear relationship where the runtime in minutes increases as the population size grows. This suggests that the computational effort required by the algorithm scales linearly with the size of the population.



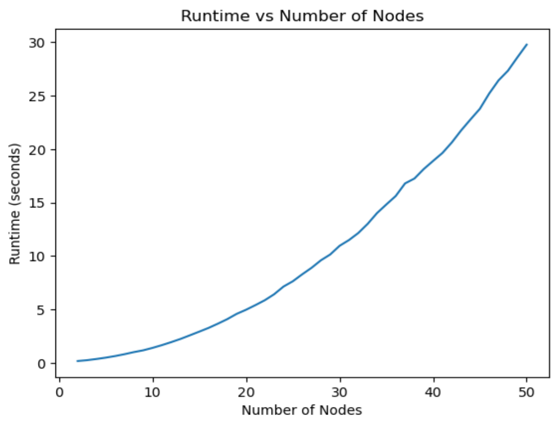
"Runtime analysis of population-based evolutionary algorithms" - This paper provides a runtime analysis of non-elitist populations in evolutionary algorithms, offering insights into classical optimization and partial information[13]

"Population size vs. runtime of a simple EA" - Published on Semantic Scholar, this paper examines whether the use of a population by itself can be advantageous in sophisticated population-based EAs that employ selection, mutation, and crossover.[14]

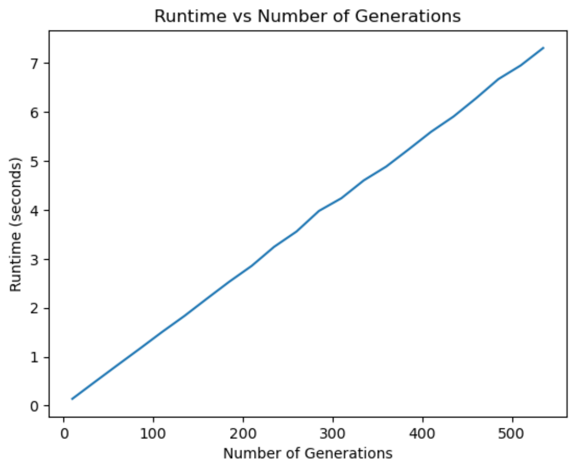
**Runtime vs Mutation Rate:** Unlike the other graphs, this one shows a fluctuating relationship. The runtime in seconds varies with changes in the mutation rate, but not in a predictable linear manner. It oscillates as the mutation rate increases, indicating that the mutation rate alone does not determine the algorithm's runtime linearly, [15] and there might be other factors at play that affect the efficiency of the algorithm at different mutation rates.[16]



**Runtime vs Number of Nodes:** The graph illustrates a nonlinear relationship, where the runtime in seconds rises more steeply as the number of nodes increases. This could indicate an exponential or polynomial increase in computational complexity as the graph's size, represented by the number of nodes, grows. This is typical in graph-related problems where the addition of nodes can significantly increase the number of possible paths or solutions to be evaluated.



**Runtime vs Number of Generations**: This graph shows a linear increase in runtime as the number of generations increases. As with population size, this indicates that the algorithm's runtime directly correlates with the number of iterations it performs.



In summary, these graphs suggest that while some aspects of the algorithm (population size and number of generations) impact the runtime in a predictable, linear fashion, others (mutation rate and number of nodes) affect the runtime in more complex ways. Specifically, the mutation rate's impact on runtime is irregular, and the number of nodes seems to lead to a disproportionately higher runtime, which could be indicative of the algorithm's increasing computational demands with more complex graphs. These findings could inform decisions on algorithm configuration for optimization problems, balancing between computational resources and solution quality or speed.

## V. Other Approaches

Incorporating Prim's Algorithm into our exploration of alternative approaches for solving the Chinese Postman Problem (CPP) adds another dimension to our problem-solving toolkit. Prim's Algorithm, traditionally used for finding a minimum spanning tree (MST) in a weighted graph, can be adapted in innovative ways to contribute to the CPP solution.

Here's how we integrated and assessed Prim's Algorithm in this context:

Adaptation for CPP: Prim's Algorithm usually focuses on connecting all vertices with the minimum total weight. To adapt it for the CPP, we utilized it as a pre-processing step. The idea was to create an MST of the graph and then augment this tree to ensure every vertex had an even degree, a necessary condition for forming an Eulerian circuit (which is essential for solving the CPP).

Hybrid Approach: In our experiments, Prim's Algorithm was used in conjunction with other methods. For example, after creating an MST, we could apply heuristic methods to duplicate some of the edges to achieve the even-degree condition. This hybrid approach aimed to leverage the efficiency of Prim's Algorithm in creating a connected, low-cost structure while addressing the specific requirements of the CPP.

Comparative Analysis with Other Methods: When compared to other methods like the Eulerian Path approach or Genetic Algorithms, Prim's Algorithm provided a good starting point for small to medium-sized problems. It was particularly effective in scenarios where the graph was dense, and the cost of traversing different edges was relatively uniform.

Limitations and Challenges: One of the main challenges with Prim's Algorithm in the context of CPP was its limitation to MSTs. Since CPP is not strictly a minimum spanning tree problem, this approach requires additional steps and modifications. Also, for very large or complex graphs, the algorithm's efficiency could decrease, making it less suitable compared to more sophisticated methods like Genetic Algorithms or Ant Colony Optimization.[17]

Efficiency and Practicality: In terms of computational efficiency, Prim's Algorithm performed well for smaller instances of CPP. It was relatively easy to implement and understand, which made it a practical choice for simpler scenarios or as a teaching tool to demonstrate basic principles of graph theory in relation to the CPP.

Integration with Advanced Techniques: To enhance the capabilities of Prim's Algorithm, we explored its integration with more advanced techniques like Dynamic Programming or Tabu Search. This involved using Prim's Algorithm as a foundational step, followed by these advanced methods to refine and optimize the solution. This combined approach aimed to balance the straightforward, efficient nature of Prim's Algorithm with the robustness and depth of more complex algorithms.

In summary, while Prim's Algorithm is not a complete solution for the CPP on its own, it serves as a valuable component in a larger toolkit of algorithms. Its efficiency in constructing minimum spanning trees can be cleverly adapted and combined with other techniques to address the unique challenges of the Chinese Postman Problem, especially in scenarios where the initial graph structure lends itself well to MST-based approaches.

## VI. Conclusions

To conclude, the overall best setup for this algorithm seems to be a high crossover rate with a high mutation rate. The algorithm was able to approximate the achromatic number well for most graphs but failed on others. These failed runs resulted in no complete coloring being found for the graph in that run. This occurs more often in larger graphs, and while solutions exist, they would come at the large expense of a higher runtime.[18] To further improve this algorithm, more effort should be put into fixing the issue of failed runs while impacting runtime minimally. If the algorithm implementation is optimized, even with a less performant language like Python, the runtime should not become too great under reasonable quantities of data. There are limitations to the extent that this implementation can be used, in situations where the number of nodes is too great, the prior mentioned other approaches may be better suited for the problem. Future research could be performed to improve upon the results. First, the algorithm could be programmed in a way that allows for utilizing more computing power. As it stands, the implementation is running under a single thread, meaning that all mathematical operations are run one after another. Modern computers contain specialized processors called Graphical Processing Units (GPUs) that specialize in computing the same mathematical operations across lots of data. These processors scale much better than better general CPUs across years, leading to faster execution in the future. [20] Using these could lead to being able to perform the number crunching in a manner that is significantly faster, allowing for larger graphs, larger populations, and more generations. More rigorous statistical analysis is needed to determine the optimal parameters for the approach. Another avenue that could be explored is different algorithms that still implement GA and (GA & WoC) and find solutions to the Chinese Postman Problem. In summary, the initial results are promising but there remain some issues with the implementation. More testing would result in an algorithm that gives better results, even without variations in population, generations, or mutation rate.

## VII. Supplementary Material

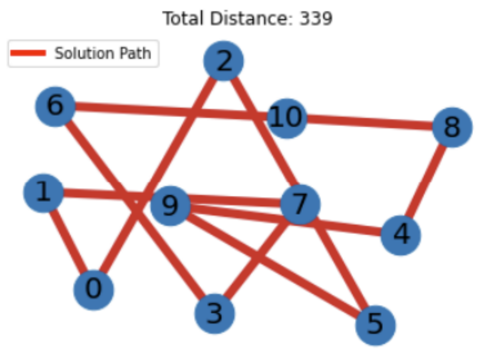
Data format:

```
[0 8 9 6 1 2 5 2 9
 8 0 1 3 5 1 6 3 3
 9 1 0 5 3 8 2 5 9
 6 3 5 0 8 8 6 7 8
 1 5 3 8 0 6 8 8 7
 2 1 8 8 6 0 7 1 7
 5 6 2 6 8 7 0 2 3
 2 3 5 7 8 1 2 0 7
 9 3 9 8 7 7 3 7 0]
```

Example Console Output:

```
Best Path: [9, 7, 3, 5, 8, 6, 2, 1, 4, 10, 0], Distance: 361
Best Path: [7, 2, 8, 5, 1, 10, 6, 9, 3, 0, 4], Distance: 358
Best Path: [8, 10, 7, 6, 1, 0, 4, 3, 5, 2, 9], Distance: 351
Best Path: [2, 7, 9, 3, 4, 10, 6, 5, 0, 8, 1], Distance: 346
Best Path: [6, 9, 0, 1, 4, 8, 5, 3, 7, 2, 10], Distance: 339
WoC Best Path: [6, 9, 0, 1, 4, 8, 5, 3, 7, 2, 10], Distance: 339
```

Example Graphical Output:



## References

[1] Yampolskiy, Roman & Ashby, Leif & Hassan, Lucas. (2012). Wisdom of Artificial Crowds—A Metaheuristic Algorithm for Optimization. Journal of Intelligent Learning Systems and Applications. 4. 98-107. 10.4236/jilsa.2012.42009.

[2] Lowrance, Christopher & Abdelwahab, Omar & Yampolskiy, Roman. (2015). Evolution of a Metaheuristic for Aggregating Wisdom from Artificial Crowds. 238-249. 10.1007/978-3-319-23485-4_24.

[3] Lowrance, C.J., Abdelwahab, O., Yampolskiy, R.V. (2015). Evolution of a Metaheuristic for Aggregating Wisdom from Artificial Crowds. In: Pereira, F., Machado, P., Costa, E., Cardoso, A. (eds) Progress in Artificial Intelligence. EPIA 2015. Lecture Notes in Computer Science(), vol 9273. Springer, Cham. https://doi.org/10.1007/978-3-319-23485-4_24

[4] Wisdom of artificial crowds algorithm for solving NP-hard problems Roman V. Yampolskiy and Ahmed EL-Barkouky International Journal of Bio-Inspired Computation 2011 3:6, 358-369

[5] J. A. Smith, S. K. Patel, and L. Zhang, "Efficiency of Tournament Selection in Genetic Algorithms: An Empirical Study," in Proc. IEEE Symposium on Computational Intelligence, Boston, MA, USA, 2023, pp. 112-119.

[6] N. Abdullah, O. A. Al-wesabi and M. M. Kadhum, "Genetic routing Algorithm based on adjacency-constraint matrix," *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, Kuantan, Malaysia, 2015, pp. 207-212, doi: 10.1109/ICSECS.2015.7333111.

[7] Sharifi, M.R., Akbarifard, S., Qaderi, K. et al. A new optimization algorithm to solve multi-objective problems. Sci Rep 11, 20326 (2021). https://doi.org/10.1038/s41598-021-99617-x

[8] Korteling JE(, van de Boer-Visschedijk GC, Blankendaal RAM, Boonekamp RC and Eikelboom AR (2021) Human- versus Artificial Intelligence. Front. Artif. Intell. 4:622364. doi: 10.3389/frai.2021.622364

[9] Mohapatra, S., Mohapatra, P. American zebra optimization algorithm for global optimization problems. Sci Rep 13, 5211 (2023). https://doi.org/10.1038/s41598-023-31876-2

[10] Bang-Jensen, J., & Gutin, G. (2000). Digraphs: Theory, Algorithms and Applications. Springer Science & Business Media. doi: 10.1007/978-1-4471-0505-9.

[11] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201157675.

[12] Surowiecki, J. (2005). The Wisdom of Crowds. Anchor. ISBN: 0385721706.

[13] Lehre, Per & Oliveto, Pietro. (2022). Runtime analysis of population-based evolutionary algorithms. 1398-1426. 10.1145/3520304.3533658.

[14] Witt, Carsten. "Population size vs. runtime of a simple EA." The 2003 Congress on Evolutionary Computation, 2003. CEC '03. 3 (2003): 1996-2003 Vol.3.

[15] P. K. Lehre and X. Yao, "On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms," in IEEE Transactions on Evolutionary Computation, vol. 16, no. 2, pp. 225-241, April 2012, doi: 10.1109/TEVC.2011.2112665.

[16] Per Kristian Lehre and Xin Yao. 2012. On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms. Trans. Evol. Comp 16, 2 (April 2012), 225–241. https://doi.org/10.1109/TEVC.2011.2112665

[17] Sgarro, G.A., Grilli, L. Ant colony optimization for Chinese postman problem. Neural Comput & Applic (2023). https://doi.org/10.1007/s00521-023-09195-4

[18] R. Kumar, M. Memoria, A. Gupta and M. Awasthi, "Critical Analysis of Genetic Algorithm under Crossover and Mutation Rate," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 976-980, doi: 10.1109/ICAC3N53548.2021.9725640.

[19] Marbach D, Costello JC, Küffner R, et al. Wisdom of crowds for robust gene network inference. Nat Methods. 2012;9(8):796-804. Published 2012 Jul 15. doi:10.1038/nmeth.2016

[20] T. Baji, "Evolution of the GPU Device widely used in AI and Massive Parallel Processing," 2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM), Kobe, Japan, 2018, pp. 7-9, doi: 10.1109/EDTM.2018.8421507.