**Student No: 216016510**

**Optimisation research paper**

**Due date: 15 October 2024**

**Solving the Traveling Salesman Problem Using Genetic Algorithms**

**Abstract.** The Traveling Salesman Problem (TSP) is one of the most well-known optimisation problems. The main objective of TSP is to find the shortest possible route that visits a given number of cities exactly once and returns to the starting city. This paper proposes a solution to the travelling salesman problem using genetic algorithm (GA). The algorithm aims to optimize the route by simulating natural selection, where the best solutions evolve over generations. This paper explains the algorithm's fitness function, selection, crossover, and mutation mechanisms. It also compares the genetic algorithm with traditional approaches and discusses future improvements. Our findings indicate that genetic algorithms provide an efficient and practical solution to TSP, particularly for larger instances where other optimisation solutions are computationally expensive.

Keywords: Genetic algorithms (GAs), Travelling salesman (TSP), crossover, mutation

1. **Introduction**

The Traveling Salesman Problem (TSP) is an optimization problem where the goal is to find the shortest possible route that visits each city exactly once and returns to the starting point. The problem has been widely studied due to its applicability in various fields, including logistics and transportation. However, as the number of cities increases, the complexity of finding the optimal solution grows exponentially. For n cities, the number of possible routes is (n-1)!, making it not feasible to solve larger instances using algorithms that can be computationally expensive.

Traditional algorithms, such as dynamic programming, are effective for small problem sizes but become less practical for larger datasets due to their exponential time complexity. Consequently, heuristic and metaheuristic approaches have been developed to tackle TSP. Among these, genetic algorithms (GAs) have shown significant promise due to their ability to explore large solution spaces and converge on near-optimal solutions efficiently.

GAs are inspired by the principles from natural evolution, which include selection, crossover, and mutation. These algorithms iteratively evolve a population of potential solutions toward better solutions. In the context of TSP, the individuals in a population represent possible tours, and the fitness function measures the quality of each tour based on its total length. By selecting and combining the best-performing individuals, GAs aim to find the shortest tour. However, the performance of genetic algorithms is dependent on the choice of parameters such as mutation and crossover rates, population size, and fitness function weighting. These parameters determine the balance between exploration and exploitation in the search process, which in turn affects the convergence rate and solution quality.

This paper investigates the application of GAs to the TSP, with a focus on how different parameter settings impact the algorithm’s performance. We also compare the GA with other heuristic approaches and discuss the advantages and limitations of using GAs for solving TSP.

1. **Literature Study**

The Traveling Salesman Problem (TSP) has been extensively studied due to its NP-hard nature and its various practical applications, from logistics and manufacturing to DNA sequencing. It remains a fundamental problem for testing optimization algorithms. Historically, early research on TSP was dominated by methods that could guarantee optimal solutions but were limited by computational complexity. Dynamic programming, introduced by Bellman [1], and the branch-and-bound method proposed by Little et al. [2], were among the first approaches capable of solving the travelling salesman problem. However, their time complexity rendered them not feasible for solving large instances of TSP, making them suitable only for smaller datasets.

As computational constraints became more apparent, research shifted towards heuristic and metaheuristic algorithms, which aim to provide near-optimal solutions in a reasonable time frame. The nearest-neighbour algorithm, a simple and greedy heuristic, constructs a path by always choosing the closest unvisited city. Despite its efficiency, it often produces suboptimal solutions because it tends to get trapped in local minima [3]. To address these limitations, researchers developed more sophisticated heuristics and metaheuristics.

Simulated annealing (SA) and tabu search are two popular metaheuristics that were applied to the TSP to overcome local minima traps. Simulated annealing, inspired by the annealing process in metallurgy, introduces a probabilistic approach that allows for occasional uphill moves to escape local optima. Tabu search, on the other hand, incorporates memory structures that prevent revisiting previously explored poor solutions. While these methods offer improved performance over simple heuristics, they still face challenges when applied to large-scale instances of TSP.

Genetic Algorithms (GAs) emerged as a powerful metaheuristic for solving the TSP, particularly due to their adaptability and suitability for parallel processing. First introduced by Holland in the 1970s [4], Genetic algorithms simulate the process of natural evolution by evolving a population of candidate solutions (tours) over successive generations. Key genetic operators include selection, crossover, and mutation, which allow for the exploration and exploitation of the search space. The fitness function in the context of TSP typically evaluates the quality of a solution based on the total tour distance.

Larranaga et al. [5] provided a detailed review of various genetic operators applied to TSP, such as partially mapped crossover (PMX), order crossover (OX), and cycle crossover (CX), each influencing the search process differently. PMX, for instance, attempts to preserve relative ordering between parents, while OX ensures that the offspring inherits a sequence of contiguous cities from one parent. These crossover operators play a crucial role in maintaining diversity within the population and avoiding premature convergence.

Additionally, researchers like Potvin [6] and Osman et al. [7] have explored hybrid approaches that combine GAs with local search techniques to improve solution quality and convergence speed. Potvin's work focused on integrating GAs with a 2-opt or 3-opt local search to refine solutions obtained by GAs. These hybrid algorithms have shown remarkable improvements in solving large instances of TSP, as local search provides fine-tuning to already promising solutions generated by the genetic process. Osman's comparison of GAs, SA, and Ant Colony Optimization (ACO) revealed that GAs, when appropriately tuned, could outperform other metaheuristics in various benchmark instances of TSP, especially for larger datasets.

While genetic algorithms are effective in avoiding local minima and exploring the solution space, they are highly sensitive to parameter settings. Michalewicz [8] highlighted the impact of population size, crossover rate, and mutation rate on the performance of GAs. A large population may provide better diversity, but it increases computational time, while a small population may lead to premature convergence. Similarly, the mutation rate must be carefully balanced to ensure sufficient exploration without destabilizing the search process. To mitigate the challenges posed by parameter tuning, adaptive genetic algorithms have been developed, which adjust parameters dynamically based on the current state of the search. Grefenstette [9] proposed such an adaptive approach, showing that it could significantly improve GA performance by adjusting mutation and crossover rates according to the diversity of the population.

In addition to adaptive genetic algorithms, various hybrid techniques have been explored to enhance the performance of GAs. For instance, Memetic Algorithms (MAs), which combine the evolutionary aspects of GAs with local optimization techniques, have gained traction in solving complex optimization problems like the TSP. Merz and Freisleben [10] demonstrated the effectiveness of Memetic Algorithms on TSP by applying local search heuristics, such as 2-opt or 3-opt, to refine solutions during the evolutionary process. Their experiments showed that memetic algorithms significantly improved solution quality and convergence speed over traditional genetic algorithms, particularly on large instances of TSP.

Parallel Genetic Algorithms (PGAs) are another approach that has been explored to improve computational efficiency when solving the TSP. PGAs distribute the computational workload across multiple processors, allowing for simultaneous exploration of different regions of the search space. According to Alba and Troya [11], PGAs have the potential to accelerate convergence while maintaining solution diversity, which is critical for avoiding premature convergence in GAs. Their research demonstrated that parallel implementations could solve larger TSP instances in significantly less time compared to sequential GAs, without sacrificing solution quality.

Ant Colony Optimization is another metaheuristic that has been applied to the travelling salesman problem successfully. Dorigo and Gambardella [12] proposed the first ACO algorithm for TSP, inspired by the behaviour of ants. ACO builds solutions through a process of collective learning, where artificial ants traverse the solution space and deposit pheromones on edges, guiding subsequent ants towards promising areas of the search space. Osman and Kelly [7] compared ACO with GAs and SA on benchmark TSP instances, finding that ACO was particularly effective for small to medium-sized problems, while GAs demonstrated better scalability for larger instances.

Finally, in recent years, hybrid evolutionary techniques that combine elements of different metaheuristics have been proposed. For example, Genetic Algorithm-based Simulated Annealing (GASA) merges the evolutionary framework of GAs with the probabilistic hill-climbing characteristics of Simulated Annealing to escape local optima. Such hybrid algorithms have shown improved performance in terms of the quality of the solution and the convergence speed, especially when dealing with large, complex instances of TSP [13].

In summary, the literature on TSP illustrates the evolution of solution strategies, from exact methods to heuristic and metaheuristic approaches, with GAs standing out as one of the most versatile and effective methods. The adaptability of GAs and their ability to be combined with other techniques (local search, parallelization, and hybridization) offer great potential for solving large-scale TSP instances efficiently.

1. **Problem statement**

The Traveling Salesman Problem is regarded as an NP-hard problem that presents a significant challenge when attempting to find the optimal solution for large instances. The travelling salesman problem seeks for the shortest possible route that visits each city exactly once and returns to the starting point. As the number of cities increases, the number of possible routes grows, making traditional methods like dynamic programming not feasible due to their computational complexity. The problem is particularly relevant in real-world applications such as logistics, delivery routing, and network optimization, where finding efficient routes can result in significant cost savings and improved operational efficiency.

Solving TSP efficiently is highly relevant in industries such as transportation, where companies aim to minimize travel costs, delivery times, and fuel consumption. For instance, in a logistics network, optimizing delivery routes can result in significant cost reductions, improved customer satisfaction, and better resource management. Similarly, in areas like circuit board design and telecommunications, optimizing paths and routes can lead to substantial performance gains. However, the challenge of finding the best route is intensified by the scale of the problem in these applications, which can involve hundreds or thousands of nodes.

In this research, we aim to develop a robust and efficient solution for the TSP using a Genetic Algorithm (GA) approach. GAs are metaheuristic search algorithms that mimic the process of natural selection to solve optimization problems. The goal of this study is to create a GA-based framework that can generate near-optimal solutions for large TSP instances within a practical time frame. This involves designing an algorithm capable of evolving a population of potential routes, evaluating their fitness based on total travel distance, and applying genetic operations such as crossover and mutation to explore the solution space. The ultimate aim is to minimize the overall distance while ensuring that the solution is both feasible and efficient.

A significant challenge in solving the TSP is the tendency of heuristic algorithms to get stuck in local minima. Traditional approaches like the nearest neighbour heuristic often provide quick solutions but fail to find the global optimum due to their greediness. Genetic algorithms, with their population-based approach, offer an advantage here by maintaining diversity within the potential solutions. The diversity allows the algorithm to explore a range of possibilities and reduces the likelihood of premature convergence to suboptimal routes. By strategically combining and evolving potential solutions, GAs can traverse the solution space more effectively and escape local optima, increasing the probability of finding near-optimal solutions.

Moreover, the solution is implemented in a Windows Forms Application to allow for visualization of the algorithm's progress. This visual interface will provide real-time feedback on how the population evolves over iterations, showcasing improvements in fitness and changes in route structure. Visualizing the search process helps in understanding how the algorithm balances exploration and exploitation while navigating the trade-off between solution quality and computation time. The ultimate objective is to design an algorithm that can minimize the total travel distance for large instances of TSP while ensuring computational efficiency and adaptability to different problem sizes.

1. **Solution Approach**

To solve the Traveling Salesman Problem (TSP) using a genetic algorithm (GA), we implemented the algorithm in a C# Windows Forms Application. The solution involves several key steps: generating an initial population of potential routes, applying a fitness function to evaluate these routes, and performing genetic operations (selection, crossover, and mutation) to evolve the population over multiple generations.

Figure 1. Process Flowchart

* 1. **Initialization**:

The initial population is generated randomly, with each individual in the population representing a potential route (or chromosome) for visiting all cities. The cities are represented as coordinates on a 2D plane, and each route is a permutation of these cities. For instance, a population size of 50 means there are 50 different potential routes in the initial generation. Random initialization ensures diversity, which is critical in the early stages of the algorithm to avoid premature convergence.

* 1. **Fitness Function**:

The fitness function is the core metric used to evaluate the quality of the solution. In the TSP, the fitness of a route is inversely proportional to its total distance. The total distance is calculated by summing the Euclidean distances between consecutive cities in the route, including the return to the starting city. The shorter the route, the higher its fitness value. This encourages the selection of shorter routes for the next generation.

* 1. **Selection (Roulette Wheel Selection):**

For selecting individuals for reproduction, we used Roulette Wheel Selection. In this method, each individual is assigned a probability of being selected based on its fitness relative to the rest of the population. Routes with shorter distances (higher fitness) are more likely to be chosen, but there is still a chance for less-fit individuals to be selected. This probabilistic selection encourages exploration while maintaining focus on the best-performing routes. In each iteration, parents are selected for crossover based on their assigned probability.

* 1. **Crossover**:

Crossover combines two parent routes to produce offspring that inherit features from both parents. During crossover, a random subset of cities is selected from one parent and filled into the offspring, while the remaining cities are copied from the other parent in the order they appear.

* 1. **Mutation**:

To maintain diversity and avoid local optima, mutation is applied to some offspring. We used swap mutation, where two cities in a route are randomly selected and swapped. This introduces variability into the population, helping to explore new parts of the solution space. The mutation rate, which defines the probability of mutation occurring, is an important parameter. A low mutation rate prevents the algorithm from becoming stuck in local optima, while a high mutation rate ensures diversity but may disrupt good solutions.

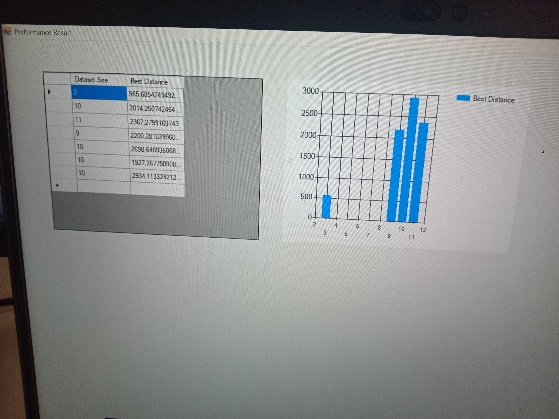
* 1. **Generational Evolution**:

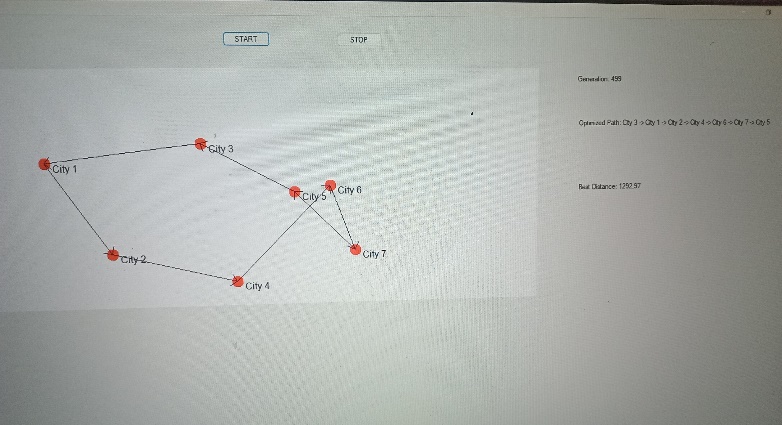
After selection, crossover, and mutation, the new generation is formed, replacing the old population. The best individuals (routes with the shortest distance) are carried over to the next generation (elitism), ensuring that the best solutions are preserved. The algorithm continues this process for a set number of generations, after which it outputs the best route found.

* 1. **Stopping Criteria**:

The algorithm is set to stop after a fixed number of generations or if no improvement is observed for a predetermined number of generations. This ensures computational efficiency and prevents the algorithm from running indefinitely.

In the below provided images, we can observe the outcome of this approach. In Image 1, a table and graph summarize the results for datasets of varying sizes. The algorithm successfully finds shorter paths as dataset size changes, with performance trends visible in the bar graph. Image 2 displays the application after finding an optimal solution for a 7-city TSP. The city connections are visualized with arrows, clearly depicting the sequence of travel between cities. The display also shows the current generation number, optimized path, and the best distance found, providing real-time feedback on the algorithm’s progress. This graphical representation allows users to track how the solution improves with each generation and verify the final path.





1. **Performance Evaluation and Impact of Crossover and Mutation Values**
   1. **Crossover Rate:**

The crossover rate determines how frequently crossover is applied to produce offspring. Higher crossover rates (e.g., 0.8) ensure that more offspring are produced through crossover, encouraging exploration of the solution space. Lower crossover rates may limit the algorithm's ability to combine good routes effectively. In our experiments, a crossover rate of 0.8 yielded faster convergence and higher-quality solutions.

* 1. **Mutation Rate:**

Mutation plays a crucial role in maintaining diversity. A mutation rate of 0.05 (5%) provided a good balance between exploration and exploitation, introducing enough randomness to avoid local optima without excessively disrupting good solutions. Higher mutation rates (e.g., 0.1) often resulted in too much randomness, while lower rates (e.g., 0.01) caused the algorithm to stagnate.

* 1. **Population Size:**

A larger population provides more diversity but increases computational cost. In our implementation, a population size of 100 was found to be effective, balancing exploration and computational efficiency. Smaller populations (e.g., 20) led to quicker convergence but often resulted in suboptimal solutions, while larger populations (e.g., 150) improved solution quality at the expense of longer runtimes.

* 1. **Generations:**

Running the algorithm for 500–1000 generations allowed the GA to explore the solution space thoroughly and fine-tune the best routes. Beyond a certain number of generations, improvement in the solution quality became minimal.

1. **Future Work**

While genetic algorithms (GAs) provide a promising approach for solving the Traveling Salesman Problem (TSP), several avenues for future research and enhancement remain. One of the primary challenges with GAs is tuning the algorithm’s parameters—such as population size, mutation rate, and crossover probability—to strike the right balance between exploration and exploitation. Poor parameter settings can lead to premature convergence or excessive computational time. Future research could focus on developing adaptive GAs where parameters are dynamically adjusted based on real-time feedback from the population’s performance. Research by Grefenstette [9] introduced adaptive parameter control, showing that such techniques can significantly improve the convergence of GAs. By dynamically adjusting parameters based on performance metrics, adaptive GAs can maintain diversity early on and focus on fine-tuning solutions in later generations.

In addition to parameter tuning, hybrid approaches could be explored to further improve the efficiency of GAs. One potential avenue is to combine GAs with local search methods such as simulated annealing, tabu search, or greedy algorithms. These methods, applied as post-processing steps, can refine the solutions generated by the GA, accelerating convergence to better solutions. For instance, Osman and Kelly [7] demonstrated that hybridizing GAs with local search techniques improved performance across several optimization problems, including TSP. Such hybrid algorithms can reduce the number of generations needed to reach a near-optimal solution, cutting down computational overhead. Future work could involve experimenting with various combinations of GA and local search methods to optimize for different types of TSP instances, including those with time constraints.

Parallelization is another promising direction for enhancing GAs. Genetic algorithms are inherently parallelizable because individuals in a population can be evaluated independently. Alba and Troya 11] explored parallel genetic algorithms (PGAs) and demonstrated that parallelization could substantially reduce computation time while maintaining solution quality. Distributed GA models, where subpopulations evolve in parallel and occasionally exchange genetic material, can efficiently handle large-scale TSP instances involving hundreds or thousands of cities. Future research could explore advanced distributed computing techniques to further enhance parallelization, particularly in real-time applications such as logistics and telecommunication routing, where fast and efficient decision-making is critical.

Real-world applications of the TSP often involve complex constraints that extend beyond the classical model, such as time and capacity constraints, and multi-objective optimization (e.g., minimizing both distance and fuel consumption). Multi-objective optimization has been addressed in works such as Deb [14], where evolutionary algorithms are designed to handle multiple objectives simultaneously. Extending GAs to tackle real-world constraints would significantly enhance their applicability. For example, vehicle routing problems (VRP), involve capacity restrictions for delivery vehicles, and GAs tailored to handle these constraints can be invaluable for industries that rely on efficient distribution. Future work could focus on integrating specific constraints into the genetic algorithm to solve more complex, real-world variants of TSP.

Additionally, machine learning offers potential avenues for optimizing the GA process itself. Researchers such as Pereira et al. [15] have explored the integration of reinforcement learning into GAs, where a machine learning model predicts the most suitable genetic operations (e.g., crossover or mutation) based on the population’s current state. This approach allows the algorithm to adapt over time, becoming more efficient as it learns from past iterations. Future research could build on these findings to develop intelligent GA frameworks capable of adjusting strategies dynamically, potentially accelerating the search process and improving solution quality.

Lastly, visualization tools can be significantly improved to help researchers better understand the behaviour of GAs over time. While visualization is often an overlooked aspect of GA implementation, it plays a crucial role in debugging and analysing the algorithm’s performance. Real-time visualization, as explored by Wu and Zhang [16], can provide insights into population diversity, convergence rates, and the effectiveness of genetic operations, thereby helping researchers fine-tune their models more efficiently. Future work could focus on creating sophisticated, interactive visualization tools that allow for real-time monitoring and intervention during the evolutionary process.

1. **Conclusion**

The Traveling Salesman Problem (TSP) is a classic optimization problem with a wide range of real-world applications, particularly in logistics, delivery routing, and network design. The challenge of finding the shortest possible route that visits each city once and returns to the starting point grows exponentially with the number of cities, making traditional algorithms such as dynamic programming not practical for large instances. This research implemented a genetic algorithm (GA) to tackle the TSP, demonstrating its effectiveness in finding high-quality, near-optimal solutions within a reasonable time frame.

The genetic algorithm’s ability to explore large solution spaces through the processes of selection, crossover, and mutation proved to be a significant advantage. By mimicking natural selection, GAs can efficiently avoid the pitfalls of local minima that typically trap greedy algorithms like the nearest neighbour approach. The population-based nature of GAs allows for diversity in the search space, helping the algorithm explore multiple potential solutions simultaneously. Although GAs do not guarantee the optimal solution, they consistently produce solutions that are close to optimal, offering a practical compromise between solution quality and computational efficiency.

A key feature of the implementation was its integration into a C# Windows Forms Application, which allowed for the real-time visualization of the GA’s progress. The graphical display of city routes and the algorithm's iterations provided valuable insights into how the solution evolved over generations. This feature not only enhanced user engagement but also aided in understanding the dynamics of the genetic algorithm, making it easier to track improvements in the fitness function and identify points where the algorithm converged.

Despite the success of this approach, several limitations remain that point to areas for future research. The sensitivity of GAs to parameter settings, such as population size, mutation rate, and crossover probability, requires careful tuning to balance exploration and exploitation effectively. Moreover, incorporating real-world constraints such as time and capacity limits, and multi-objective optimization (e.g., minimizing both distance and cost) would make the GA more applicable to practical scenarios, such as vehicle routing problems and large-scale supply chain optimization. Adaptive and hybrid algorithms, which adjust parameters dynamically or combine GAs with local search techniques, offer promising directions for further refinement.

In conclusion, genetic algorithms represent a powerful, flexible, and scalable solution to the TSP, particularly for large problem instances where traditional methods fail. Their ability to balance exploration and exploitation in a vast solution space makes them a valuable tool in both theoretical and practical optimization problems. However, to fully realize the potential of GAs in real-world applications, future research should focus on improving parameter tuning, integrating domain-specific constraints, and exploring hybrid approaches to further enhance performance and adaptability.

1. **References**:

[1] Bellman, R. (1962). Dynamic Programming Treatment of the Travelling Salesman Problem. Journal of the ACM, 9(1), 61-63. <https://doi.org/10.1145/321105.321111>

[2] Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An Algorithm for the Traveling Salesman Problem. Operations Research, 11(6), 972-989. <https://doi.org/10.1287/opre.11.6.972>

[3] Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research, 21(2), 498-516. <https://doi.org/10.1287/opre.21.2.498>

[4] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. Cambridge, MA: MIT Press. <https://direct.mit.edu/books/monograph/2574/Adaptation-in-Natural-and-Artificial-SystemsAn>

[5] Larrañaga, P., Kuijpers, C., Murga, R. H., Inza, I., & Dizdarevic, S. (2002). Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review, 16(4), 291-317. <https://research.tilburguniversity.edu/en/publications/genetic-algorithms-for-the-travelling-salesman-problem-a-review-o>

[6] Genetic Algorithms. (1996). Genetic algorithms for the traveling salesman problem. Computational Optimization and Applications, 6(2), 173-190. <https://doi.org/10.1007/BF02125403>

[7] Osman, I. H., & Kelly, J. P. (1996). Meta-heuristics. Retrieved from <https://www.researchgate.net/publication/312989901_Meta-heuristics>

.

[8] Michalewicz, Z. (1994). Genetic Algorithms + Data Structures = Evolution Programs (3rd ed.). Springer.

[9] Grefenstette, J. J. (2005). Optimization of control parameters for genetic algorithms. IEEE <https://ieeexplore.ieee.org/document/4075583>

[10] Merz, P., & Freisleben, B. (2002). Memetic algorithms for the traveling salesman problem. Complex Systems, 14(4). Retrieved from <https://www.researchgate.net/publication/2928026_Memetic_Algorithms_for_the_Traveling_Salesman_Problem>

[11] Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. Complexity, 4(4), 31-37. [https://doi.org/10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.0.CO;2-4](https://doi.org/10.1002/(SICI)1099-0526(199903/04)4:4%3c31::AID-CPLX5%3e3.0.CO;2-4)

[12] Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. European Journal of Operational Research, 99(2), 254-265. <https://doi.org/10.1016/S0303-2647(97)01708-5>

[13] Hajek, B. (1988). Cooling schedules for optimal annealing. University of Illinois at Champaign-Urbana. Retrieved from <https://web.mit.edu/6.435/www/Hajek88.pdf>

[14] Deb, K. (2001). Multi-objective optimization using evolutionary algorithms. Wiley. ISBN: 978-0-471-87339-6.

[15] Pereira, L. P., Daza, A. C., & Oliveira, M. A. (2017). Combining reinforcement learning and genetic algorithms for online evolution of robotic controllers. Autonomous Robots, 41(6), 1321-1345.

[16] Wu, X., & Zhang, Y. (2009). Real-time visualization of genetic algorithms. Proceedings of the International Conference on Genetic and Evolutionary Computation, 475-480.