# Approximating solutions to the vehicle routing problem using wisdom of artificial crowds with genetic algorithms (November 2019)

Jacob T. Cassady

*Abstract*—**This paper presents a novel approach to solving the vehicle routing problem, a generalization of the Traveling Salesman Problem, using Wisdom of Artificial Crowds with genetic algorithms as well as a novel approach to injecting cognitive diversity into an artificial crowd of genetic algorithms. The algorithm presented in this paper was implemented in Python and tested on several datasets producing approximations superior to any single genetic algorithm in the crowd.**

***Index Terms*—Shortest path problem, Genetic algorithms, NP-hard, Routing**

## I. Introduction

Generating and approximating solutions to the Vehicle Routing Problem (VRP) has been the topic of hundreds of scientific papers [1]. VRP is a generalization of the well-known Traveling Salesman Problem (TSP). It was first introduced by George Dantzig and John Ramser in 1954 [2] and studied longer in its special TSP case with record of mathematicians Thomas Kirkman and W. R. Hamilton's work as far back as the 1800s. A thorough survey of the first 50 years of VRP approximation methods was done by Laporte in 2009 [3].

The context of the problem is that there is one depot with one or more vehicles that must deliver goods to one or more customers with preference to minimizing the cost of travel. This contrasts with TSP which only has one vehicle per depot. The problem is considered a non-deterministic polynomial-time hard problem due to its factorial complexity. Advancements in VRP have been of interest to many research domains including scheduling, controls, and more [4] and proven to be valuable for vital industries such as agriculture [5], earth sciences [6], and transportation [7].

## II. Prior Work

### A. Genetic Algorithms

Genetic Algorithms (GA) were first introduced by John Holland in 1962 based on principles from biology, theoretical genetics, automata theory, and artificial adaptive systems [8]. They have been proven to be valuable tools for approximating solutions to problems subject to high time complexity when solved by formal methods with research applications in publication since 1967 [9].

Booker, et al. describes the "central loop" of genetic algorithms as the following steps:

1) *Determine fitness of population. Select pairs from the population according to fitness and selection criteria.*
2) *Apply genetic operators to the pairs, creating offspring.*
3) *Replace weakest classifiers with offspring.*

Studies to determine preferred methods for each of the steps listed above have been done on a variety of problems including TSP [10-12]. These methods have been shown to be effective at approximating solutions for VRP as well with hybrid metaheuristic GAs outperforming even the tabu search (TS) method [13].

### B. Wisdom of Artificial Crowds

"Wisdom of Crowds" (WOC) was first coined by James Surowiecki in 2004 where he argues, "Under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them. Groups do not need to be dominated by exceptionally intelligent people in order to be smart. Even if most of the people within a group are not especially well-informed or rational, it can still reach a collectively wise decision" [14]. He continues to describe five criteria required for a crowd to be wise:

1) *Cognitive Diversity – individuals should not share information*
2) *Independence – individual's opinions are not influence by other individual's opinions*
3) *Decentralization – individuals can specialize*
4) *Aggregation – there exists an aggregation method for combining individual's opinions.*
5) *Trust – individuals trust the group to be fair.*

This concept of WOC has been applied to a variety of problems including TSP with promising results. Researchers at University of California, Irvine and the University of Adelaide observed the average performance of their aggregation method out performing even the best individual when applying WOC to the results of individuals' attempts at producing optimal paths for TSPs [15].

"Wisdom of Artificial Crowds" (WoAC) is a metaheuristic algorithm inspired by the nature-based behavior utilized in WOC [16]. WoAC is implemented as a post processing algorithm that takes a collection of individual solutions as an input to produce an aggregate solution that is often superior than any individual solution in the population. This concept has been shown to successfully approximate optimal solutions to TSP when using a crowd of genetic algorithms [17]. Cognitive diversity was achieved with different initializations of the genetic algorithm's population.

## III. Proposed Approach

### A. *Genetic Algorithm*

The genetic algorithm implemented follows the central loop described by Booker, et al. closely [9]. The algorithm is initialized with a given population size, crossover method, crossover probability, mutation method, mutation probability, and epoch threshold. The population size denotes the number of chromosomes in the GAs population. The crossover probability denotes the percentage of the population to be replaced by crossover at the start of a generation. A crossover probability of 1 would mean offspring will replace all parents while a crossover probability of 0 would mean that no parents would crossover to produce offspring. The mutation probability denotes what percent of the population after crossover will undergo mutation. A mutation probability of 1 would mean all chromosomes in the population are mutated while a mutation probability of 0 would mean none of the chromosomes in the population are mutated. Epoch threshold denotes how many generations the GA must produce without seeing an improvement before finishing.

The crossover methods implemented were inspired by Otman and Abouchabaka's work on comparing adaptive crossover operators for GAs aimed at approximating solutions to TSP [10]. The mutation methods implemented were inspired by similar work from Abdoun, et al. where they analyzed the performance of mutation operators for similar purposes [11].

*Roulette Wheel Selection*

The selection method implemented was influenced by the work of Zhong, et al. [12]. The roulette wheel selection methods aim to reduce the likelihood all chromosomes will converge to a single local optimum solution. To accomplish this, it calculates the probability of selection for each chromosome by dividing its fitness value by the sum of all fitness values of the chromosomes in the population. This differs from a straightforward elitist selection method that selects a percentage of the best performing chromosomes.

*Uniform Crossover*

The Uniform Crossover produces a child by alternating randomly between the alleles of the two parents.

*Ordered Crossover (OX)*

The Ordered Crossover method implemented comes from a book by Goldberg, in which he argues using OX is useful for problems that are ordered based [18]. The implementation used in this paper was slightly altered to produce only one child. OX is performed by randomly partitioning two parent chromosomes into three contiguous sections.

**Table 1. The partition of a parent**

| | | |
|---|---|---|
| S1 | S2 | S3 |

The child inherits sections S1 and S3 from parent 1 while S2 is determined from the left-over alleles in parent 2 while retaining order.

*Partially Mapped Crossover (PMX)*

The Partially Mapped Crossover method implemented was first described in a 1985 book from Goldberg and Lingle [19] in which the parents are partitioned randomly into three sections as shown in Table 1. Sequences S1 and S3 from parent 1 are copied into the child. S2 is determined from parent 2's alleles by starting at S2 and skipping over any allele that is already present in the child.

*TWORS Mutation*

The TWORS Mutation method randomly swaps two alleles in a chromosome.

*Reverse Sequence Mutation (RSM)*

The Reverse Sequence Mutation method randomly partitions the chromosome into three sections as shown in Table 1. The Sequence S2 is reversed while S1 and S3 remain constant.

*Fitness Function*

The fitness function is a greedy implementation in which the customers are evenly divided between the depot's vehicles. Table 2 below shows an example of how a series of 11 vertices with one depot and 4 vehicles would divide its customers while evaluating its fitness. In this example, the depot is at vertex 9, while all other vertices 1-8 and 10-11 are customers. The fitness function is then calculated by iterating over the vehicles, adding the distance between the depot and the first customer, summing the distance between each adjacent customer in the vehicles list, and then adding the distance from the last customer back to the depot.

**Table 2. Vehicle Customer Allocation**

| Vehicle 1 | | | Vehicle 2 | | Vehicle 3 | | Vehicle 4 | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 2 | 8 | 6 | 4 | 1 | 10 | 11 |

### B. *Wisdom of Artificial Crowds*

The Wisdom of Artificial Crowds algorithm implemented was inspired by the implementation presented by Yampolskiy, et al. [17] although a novel approach was used for injecting cognitive diversity. In contrast to injecting cognitive diversity through a series of initializations of the same genetic algorithm, multiple GAs with differing crossover methods and mutation methods were used. Each GA has the same population size, selection method, and epoch threshold.

Gathering the chromosomes to be used in the crowd is done by using predetermined weights for each GA. For example, a GA with a weight of 0.05 and a population of 100 chromosomes would provide 5 chromosomes to the crowd before solution aggregation. These weights were selected after developing an understanding of a GA with these methods' performance when approximating TSP. The weights used for different crossover and mutation combinations of GA during experimentation are summarized in Table 3 below.

**Table 3. Weights of GAs with Crossover and Mutation Method Combinations**

| Crossover Method | Mutation Method | Weight |
|---|---|---|
| Uniform | TWORS | 0.05 |
| Uniform | RSM | 0.05 |
| OX | TWORS | 0.6 |
| OX | RSM | 0.3 |
| PMX | TWORS | 0.05 |
| PMX | RSM | 0.05 |

After the crowd's population has been determined, the frequency of each edge is calculated. Edges that occur more frequent than 0.3 times the highest frequency of an individual edge are used in the aggregate. If two edges share the same starting or ending vertex, the edge with the highest frequency is used in the final solution. If both edges are equally as frequent and have similar starting or ending vertices, then the shorter edge is used in the final solution. Unvisited vertices are then chosen using a greedy heuristic, picking the vertex closest to one of its vehicle's route's edge. The vertex is then placed in the customer list at a position where the addition to distance traveled is reduced.

## IV. Experimental Results

### *A. Data*

The test data was generated using software called Concorde [20]. Concorde is an ANSI C program that produces optimal solutions to TSP graphs. It also has the capability to generate new instances of TSP of any size with either predefined coordinates or random distribution. Please see the example below for reference to a 6 city test file generated by Concorde.

```
NAME: concorde6
TYPE: TSP
COMMENT: Generated by CCutil_writetsplib
COMMENT: Write called for by Concorde GUI
DIMENSION: 6
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 30.147059 79.746835
2 45.996732 78.270042
3 61.192810 78.270042
4 60.049020 48.734177
5 43.545752 49.156118
6 30.228758 51.476793
```

### *B. Results*

*GA Results: 1 vehicle, 1 depot, 21 customers*

Figures 1-6 show improvements in minimum cost or maximum fitness in runs of each genetic algorithm. The algorithms share a population size of 20, a vehicle count of 1, a crossover probability of 50%, a mutation probability of 5% and an epoch threshold of 10. As you can see, all GAs improve over time with the combination of OX and RSM producing the most optimal approximations.

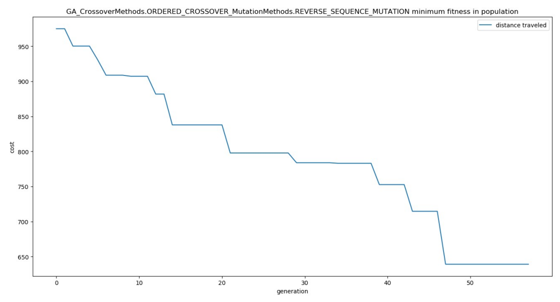

**Figure 1 : OX / RSM Minimum Cost in Population Per Generation (1 vehicle)**

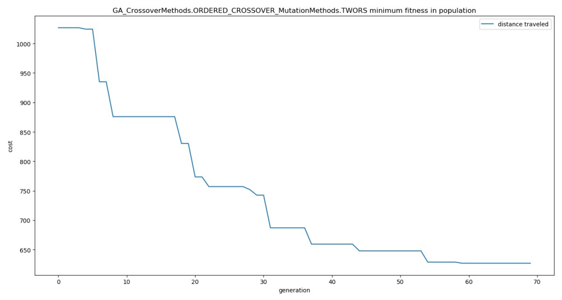

**Figure 2 : OX / TWORS Minimum Cost in Population Per Generation (1 vehicle)**

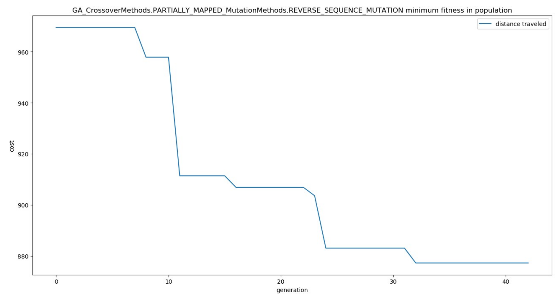

**Figure 3 : PMX / RSM Minimum Cost in Population Per Generation (1 vehicle)**

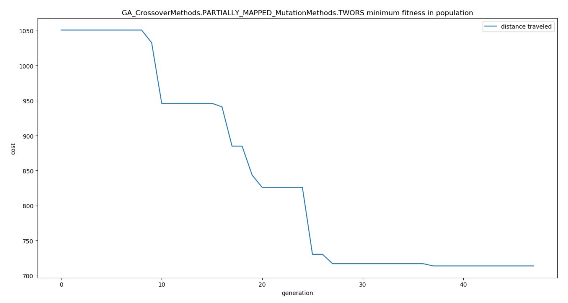

**Figure 4 : PMX / TWORS Minimum Cost in Population Per Generation (1 vehicle)**

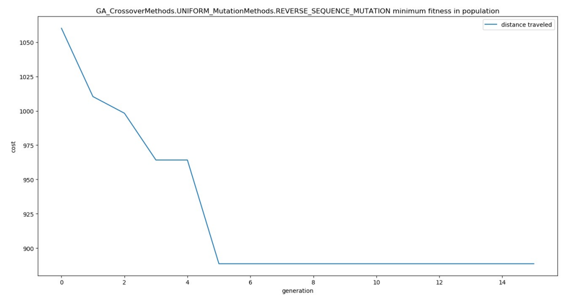

**Figure 5 : UNIFORM / RSM Minimum Cost in Population Per Generation (1 vehicle)**

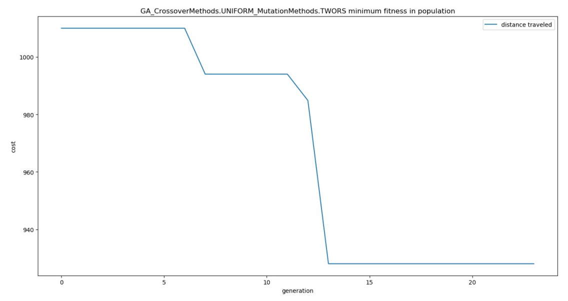

**Figure 6 : UNIFORM / TWORS Minimum Cost in Population Per Generation (1 vehicle)**

*GA Results: 3 vehicles, 1 depot, 21 customers*

Figures 7-12 show improvements in minimum cost or maximum fitness in runs of each genetic algorithm. The algorithms share a population size of 20, a vehicle count of 3, a crossover probability of 50%, a mutation probability of 5% and an epoch threshold of 10. As you can see, all GAs improve over time with the combination of OX and RSM producing the most optimal approximations.

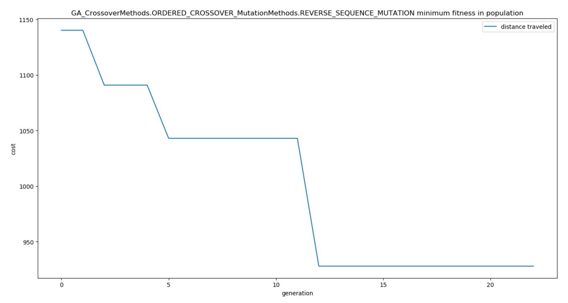

**Figure 7 : OX / RSM Minimum Cost in Population Per Generation (3 vehicles)**

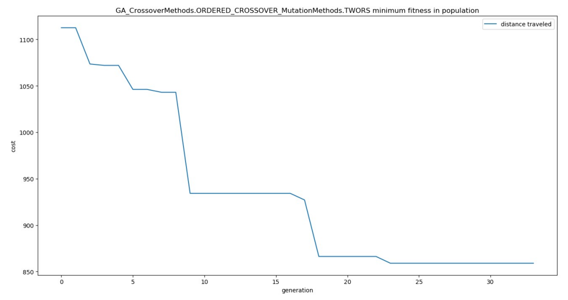

**Figure 8 : OX / TWORS Minimum Cost in Population Per Generation (3 vehicles)**

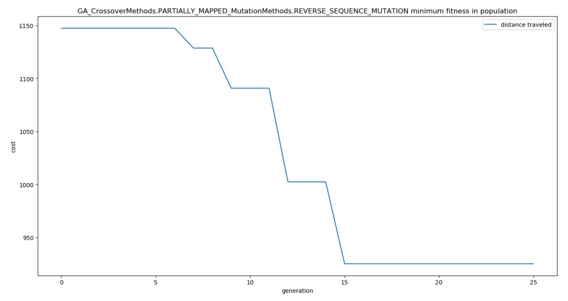

**Figure 9 : PMX / RSM Minimum Cost in Population Per Generation (3 vehicles)**

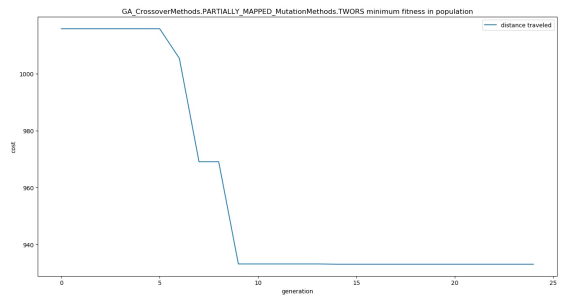

**Figure 10 : PMX / TWORS Minimum Cost in Population Per Generation (3 vehicles)**

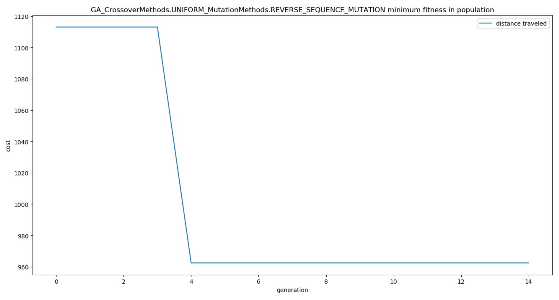

**Figure 11 : UNIFORM / RSM Minimum Cost in Population Per Generation (3 vehicles)**

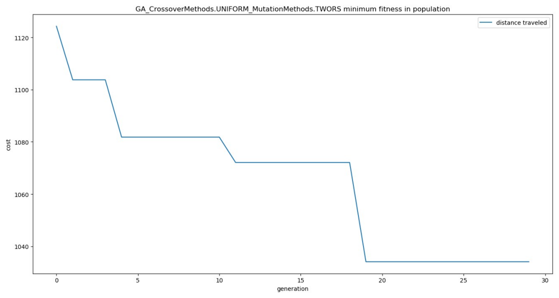

**Figure 12 : UNIFORM / TWORS Minimum Cost in Population Per Generation (3 vehicles)**

*WoAC Results: 1 vehicle, 1 depot, 20 customers*

Table 4 and Figure 13 show comparisons between the different algorithms described in this paper. As you can see, the WoAC algorithm produces a better result at the cost of a post processing overhead. It is important to note the GAs in the crowd of the WoAC algorithm are ran in parallel. This is included in WoAC's runtime.

**Table 4 : Comparison of Algorithm Performance (1 Depot, 20 Customers, 1 vehicle)**

| ALGORITHM | average result | average runtime | Percent Superior to Avg |
|---|---|---|---|
| UNIFORM_TWORS | 860.4853784 | 0.990 | 50% |
| UNIFORM_RSM | 874.7368728 | 1.024 | 20% |
| PM_TWORS | 871.600 | 1.002 | 30% |
| PM_RSM | 874.501 | 1.100 | 25% |
| ORDERED_TWORS | 862.469 | 1.184 | 40% |
| ORDERED_RSM | 866.795 | 1.002 | 25% |
| WoAC | 736.365 | 6.730 | 95% |
| Average for all | 849.564 | 1.862 | - |

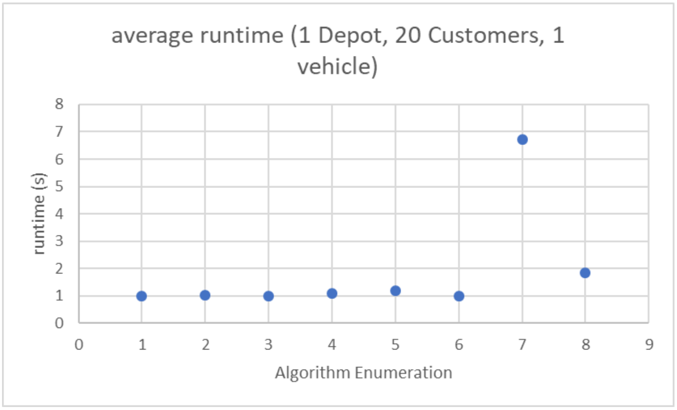

**Figure 13 : Average Runtime per Algorithm**

## V. Conclusions

In conclusion, the GAs described in this paper improve upon approximations of the VRP. Figures 1-12 show the improvements made over subsequent generations to the approximation. The OX crossover is more likely to produce helpful improvements when compared to other crossover methods. The same is true for the RSM mutation method when compared to other mutation methods.

The WoAC post processing algorithm described in this paper is shown to improve upon the approximations produced by individual GAs. This was shown in Figure 13 and Table 4. The WoAC algorithm produces a more optimal approximation at the cost of post processing overhead. 95% of the tests surveyed, the WoAC algorithm outperformed the average for all algorithms which was often lower than the best result for any genetic algorithm.

## VI. Future Work

To understand how the algorithm presented here compares to classical solutions such as the brute force method or the standard TS method for approximating solutions to VRP. These algorithms need to be implemented and ran on the same datasets as VRP to isolate benefits and costs associated with each algorithm.

Furthermore, since the WoAC component of this algorithm can be used as a post processing technique, separate from the GAs described in this paper, more analysis needs to be done on the computational costs of WoAC alone. If it is shown to have low overhead, this method could be used in distributed systems where parallelizing less optimal approximations using GAs could then be processed at a different point improving their collective approximation. This could be shown to be superior over classical methods if the computational cost could be spread out asynchronously on multiple machines.

Lastly, to compare the idea for injecting cognitive diversity to that purposed by Yampolskiy [17], tests need to be ran using both methods with similar sized crowds and genetic algorithm implementations to determine which method is superior. Further analysis could be done to examine the differences in final solutions produced by different combinations of crossover and mutation methods. If it could be shown that different crossover and mutation method pairs produce similarly optimal results with vastly different configurations, the WoAC solution might suffer or prosper as a result of the increased entropy. It would be interesting to better understand the benefit of entropy in these populations.

## VII. Acknowledgements

Thank you to professor Roman Yampolskiy for making the introduction into these topics enjoyable and insightful.

## References

[1] B. L. Golden, S. Raghavan, and E. A. Wasil, *The vehicle routing problem: latest advances and new challenges*. Springer Science & Business Media, 2008.

[2] G. Dantzig, R. Fulkerson, and S. Johnson, "Solutions of a large-scale travelling salesman problem," *Journal of the Operations Research Society of America,* vol. 2, no. 4, pp. 365-462, 1954, doi: https://doi.org/10.1287/opre.2.4.393.

[3] G. Laporte, "Fifty Years of Vehicle Routing " *Transportation Science,* vol. 43, no. 4, pp. 408-416, Nov 2009, doi: https://doi.org/10.1287/trsc.1090.0301.

[4] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A toxonomic review," *Computers & Industrial Engineering,* vol. 57, no. 4, pp. 1472-1483, 2009, doi: https://doi.org/10.1016/j.cie.2009.05.009.

[5] C. Garcia, B. Velazquez-Marti, and J. Estornell, "An application of the vehicle routing problem to biomass transportation," *Biosystems Engineering,* vol. 124, pp. 40-52, 2014, doi: https://doi.org/10.1016/j.biosystemseng.2014.06.009.

[6] C. Lin, K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam, "Survey of Green Vehicle Routing Problem: Past and future trends," *Expert Systems with Applications,* vol. 41, no. 4, 1, pp. 1118-1138, March 2014, doi: https://doi.org/10.1016/j.eswa.2013.07.107.

[7] P. Toth and D. Vigo, "An overview of vehicle routing problems," in *The Vehicle Routing Problem*. Philadelphia, PA, USA: ociety for Industrial and Applied Mathematics, 2002, p. 1.

[8] J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *Journal of the ACM (JACM),* vol. 9, no. 3, pp. 297-314, July 1962, doi: https://doi.org/10.1145/321127.321128.

[9] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier systems and genetic algorithms," *Artificial Intelligence,* vol. 40, no. 1-3, pp. 235-282, September 1989, doi: https://doi.org/10.1016/0004-3702(89)90050-7.

[10] A. Otman and J. Abouchabaka, "A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem," *International Journal of Computer Applications,* vol. 31, no. 11, pp. 49-57, Oct 2011. [Online]. Available: https://arxiv.org/abs/1203.3097.

[11] A. Otman, J. Abouchabaka, and C. Tajani, "Analyzing the performance of mutation operators to solve the traveling salesman problem," *International Journal of Emerging Sciences,* vol. 2, no. 1, pp. 61-77, March 2012.

[12] J. Zhong, X. Hu, M. Gu, and J. Zhang, "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms," presented at the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28-30 Nov., 2005.

[13] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research,* vol. 31, no. 12, pp. 1985-2002, Oct 2004, doi: https://doi.org/10.1016/S0305-0548(03)00158-8.

[14] J. Surowiecki, *The Wisdom of Crowds: Why the Many are Smarter Than the Few and how Collective Wisdom Shapes Business, Economies, Societies, and Nations*. Doubleday, 2004.

[15] S. K. M. Yi, M. Steyvers, and M. D. Lee, "Wisdom of the crowds in traveling salesman problems," 2010. [Online]. Available: http://www.socsci.uci.edu/~mdlee/YiEtAl2010.pdf.

[16] L. H. Ashby and R. V. Yampolskiy, "Genetic algorithm and Wisdom of Artificial Crowds algorithm applied to Light up," presented at the 16th International Conference on Computer Games, Louisville, KY, USA, 27-30 July, 2011.

[17] R. V. Yampolskiy and A. EL-Barkouky, "Wisdom of artifical crowds algorithm for solving NP-hard problems," *International Journal of Bio-Inspired Computation,* vol. 3, no. 6, pp. 358-369, Jan 2011, doi: 10.1504/IJBIC.2011.043624.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1 ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989, p. 372.

[19] D. E. Goldberg and R. J. Linger, "AllelesLociand the Traveling Salesman Problem," in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985: L. Erlbaum Associates Inc. Hillsdale, NJ, USA, pp. 154-159.

[20] W. Cook. "Concorde TSP Solver." http://www.math.uwaterloo.ca/tsp/concorde/index.html (accessed 18 Nov, 2019).