Applying a genetic algorithm and wisdom of artificial crowds hybrid algorithm to the CVRP

Nick York
CES Department
Speed School of Engineering
University of Louisville, USA
neyork01@louisville.edu

***Abstract*— NP-hard problems grow exponentially as size increases. Heuristics allow finding approximate solutions in polynomial time. Traveling salesman problems (TSPs) are NP-hard problems. Researchers have applied the wisdom of artificial crowds (WoAC) to TSPs, but other problems remain. Vehicle routing problems (VRPs) are also NP-Hard. This paper explores a genetic algorithm (GA) and WoAC hybrid algorithm to solve the capacitated vehicle routing problem (CVRP). I wrote a program to implement the hybrid algorithm and test it by solving CVRPs of different sizes and complexity. Generally, the GA came within 20% of the optimal solution, with smaller problems performing better. Applying WoAC to the GA produced worse results. Future research should experiment by increasing GA population size to increase the available solution space.**

***Index Terms*—capacitated vehicle routing problem; CVRP; traveling salesman problem; TSP; genetic algorithm; GA; the wisdom of artificial crowds; WoAC; hybrid algorithm**

I. Introduction

THE vehicle routing problem (VRP) is a combinatorial optimization problem. Dantzig and Ramser formulated the VRP by generalizing the TSP [1]. TSPs require finding the minimum cost route for a salesman to visit a list of cities without visiting a city more than once, except for the starting city. The salesman must return to their starting city at the end. Cities are represented by Cartesian coordinates and numbered starting at 1. The cost to travel between two cities is the Euclidean distance between them, calculated as

$$cost = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2} \quad (1)$$

The VRP was described as dispatching gasoline trucks to make deliveries to gas stations. Each truck starts from and returns to a central terminal. The gas station locations (P) and demands (q) are known beforehand. Trucks are required to make q_i deliveries at every P_i location, excluding the terminal.

Trucks have capacity, C. The constraint on C can be defined in one of two ways. In the first instance

$$C \geq \sum_i q_i \quad (2)$$

the problem resembles the TSP, as a single truck can make all deliveries. However, in the second instance

$$C \ll \sum_i q_i \quad (3)$$

a single truck cannot make all deliveries.

Each truck visits one or more gas stations until cumulative demand meets or exceeds C. Each gas station is visited exactly once. The objective is to visit all the gas stations while minimizing the total distance (or time) traveled. The authors consider other formulations, including multiple demands for N different products, multiple truck capacities, and under or over-delivery of a fixed percentage based on demand.

The capacitated VRP (CVRP) is a variant whose primary constraint is capacity. This paper explores minimizing total distance for the CVRP using a hybrid algorithm.

II. Prior Work

Dantzig and Ramser's paper inspired research into new variants of and solutions to the VRP [2], [3]. Variants include the CVRP, the VRP with Pickup and Delivery (VRP-PD), and Dynamic Vehicle Routing Problems (DVRPs). The CVRP is NP-Hard. Exact algorithms can solve for up to 100 deliveries depending on the variant and other factors. However exact algorithms are insufficient for solving large-scale real-world applications. Decades of research on CVRPs has focused on approximate algorithms – heuristics, metaheuristics, hybrid algorithms – that approach the optimal solution for large problems.

Algorithms themselves are not the only area of research. Numerous encoding schemes have been applied to the CVRP, including vectors and matrices with binary, integer, and real numbers [4]. Relaxing problem constraints can yield highly optimized solutions [5]-[6].

Christofides and Eilon tested solving the CVRP using a branch-and-bound approach, a "savings" approach, and the 3-opt algorithm [7]. The authors found that branch-and-bound (an exact algorithm) was substantially less effective than solving for an equivalent TSP. The largest problem the authors solved using branch-and-bound included 13 customers. Computation time and storage requirements made the branch-and-bound approach intractable for larger problems. On the other hand, 3-

opt outperformed the "savings" approach in all but one of ten problems for minimum distance, with the largest problem solved including 100 customers.

Another approach is to use metaheuristics to solve the CVRP. Nazif and Lee tested solving the CVRP using their genetic algorithm and several other commonly used metaheuristics [8]. The authors found that their genetic algorithm was competitive both in quality of solutions and time to find a solution. Gunawan, Susyanto, and Bahar proposed a mathematical model for solving the VRP-PD with a genetic algorithm [9].

GA is a metaheuristic based on the theory of evolution. The theory of evolution states the fittest individuals survive to maturity to reproduce and pass their genes onto their offspring. Data is encoded as genes, collections of genes as chromosomes, and collections of chromosomes as populations.

A population changes as new chromosomes replace old ones over successive generations. Chromosomes are selected for a chance to reproduce each generation based on their fitness. Fitness is a problem-specific measure of utility. During reproduction, a crossover operator exchanges parts of two chromosomes to produce two child chromosomes. Crossover repeats until the child population is the same size as the parent population.

Afterward, a mutation operator changes one or more genes for some number of child chromosomes based on a predefined mutation rate. The child population replaces the parent population at this point. The new population is evaluated to see whether it meets some measure known as a stopping criterion. If the stopping criterion is not met, then the algorithm repeats until it is.

Besides GAs, swarm intelligence metaheuristics have been researched to solve the CVRP [10]. Despite a large body of research spanning decades, I could not find published research applying the wisdom of artificial crowds (WoAC) to the CVRP. (WoAC uses artificial crowds instead of human crowds to generate and aggregate solutions.) WoAC has been applied to the TSP; Yampolskiy and El-Barkouky found WoAC can find more optimal solutions of the TSP by aggregating solutions by a genetic algorithm [11].

Though crowds and swarms are what we might call "collective intelligence" they remain distinct [12]. Specifically, WoAC utilizes the collective wisdom of an artificial crowd to arrive at a solution that is more optimal than the experts' solutions alone. A percentage of the most optimal solutions ("elites") are aggregated and combined to build a new solution.

According to Surowiecki [13] wisdom of crowds relies on four properties to find an optimal solution. First, a crowd must be diverse conceptually (how they arrive at a solution) and cognitively (how "smart" they are). Second, members of a crowd must arrive at solutions independent of other members so that new solutions are considered, regardless of how a solution is found or how optimal it may be. The third and fourth are decentralization and aggregation, which mean no one individual directs the actions of the crowd and that solutions are aggregated in a way that preserves their independence (i.e., solutions are not negatively weighted relative to the most optimal solution).

Lastly, hybrid algorithms combine two or more algorithms, including exact and approximate algorithms, to solve problems [14]. Hybrid algorithms may focus on reducing the time to find a solution or finding a more optimal solution. Zhang, Cai, Ye, Si, and Nguyen use tabu search with the artificial bee colony algorithm to solve the VRP more optimally than individual heuristics alone [15].

III. Proposed Approach

A. Overview

I wrote a Python program that allows the user to specify which file or files to use based on their filename or file path. Each file contains a CVRP instance (see Data). The program reads and parses the vehicle capacity, coordinates, demands, and depot in the CVRP file(s). Once a file is parsed, there are two stages to the hybrid algorithm: running the GA and running WoAC.

B. Genetic Algorithm

Customers and the depot are encoded as genes. Collections of genes are encoded as chromosomes, except for the depot. Collections of chromosomes are encoded as populations. Genes have three attributes:

label – a unique identifier
coords – Cartesian coordinates
demand – demand (0 for depot)

Chromosomes have several attributes:

genes – list of customer genes
depot – depot encoded as gene
capacity – vehicle capacity
routes – list of vehicle routes
cost – the cumulative cost of vehicle routes
fitness – utility of the chromosome

A function is called to build each route, starting from the first gene in the genes attribute and adding genes until their demand meets or exceeds capacity (or there are no genes left to add) at which point the route is added to the routes attribute. Then, the cost attribute is calculated (1), including leaving and returning to the depot. The fitness attribute is calculated as

$$fitness = 1/cost \tag{4}$$

Population has several functional attributes (the remainder are used for tracking statistics for each generation):

population_size – number of chromosomes to use
mutation_rate – probability a chromosome is mutated
elites – number of chromosomes to automatically add to the

selection pool
optimal – chromosome with the minimum cost
chromosomes – list of chromosomes in the population
depot – depot encoded as gene
capacity – vehicle capacity

The first attributes are fixed values: population_size is 100, mutation_rate is 1%, and elites is 20 (one-fifth of population_size). The optimal attribute is updated at the end of each generation if a lower cost chromosome is found. The depot and capacity attributes are used to initialize the initial population chromosomes and child chromosomes during crossover. The chromosomes attribute is initialized by randomly sampling genes without replacement to construct 100 chromosomes.

The GA applies selection, crossover, mutation, and evaluation until the stopping criterion is met. Selection automatically adds the fittest chromosomes from chromosomes to the selection pool, based on elites. The remaining chromosomes are selected based on probability; the fittest chromosomes have the highest probability of selection.

Next is crossover. While there are fewer child chromosomes than population_size, two parent chromosomes are randomly selected from the selection pool to mate. Fig. 1 shows two parents with eight genes each to illustrate crossover.

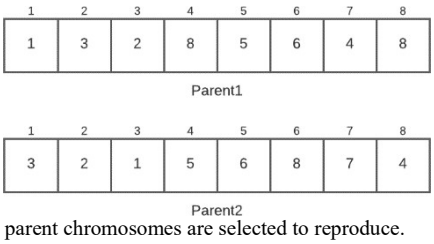

Fig. 1. Two parent chromosomes are selected to reproduce.

The crossover operator randomly selects a gene sequence from the first parent and combines it with genes from the second parent not in the sequence to start creating two child chromosomes. The lightly shaded squares in Fig. 2 represent the gene sequence added to the first child chromosome. The darkly shaded squares are genes before the sequence to be added to the second child.

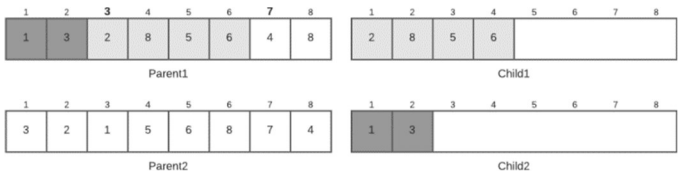

Fig. 2. Child chromosomes after the first genes are added. The bold 3 and 7 signify the start and end of the gene sequence.

The genes in the second parent not in the first child are added next. The remaining genes in the second parent are added to the second child, followed by those after the sequence in parent one. Fig. 3 shows the result of crossover.

After crossover is mutation – the mutation operator takes each child and generates a random number that if less than mutation_rate initiates mutation.

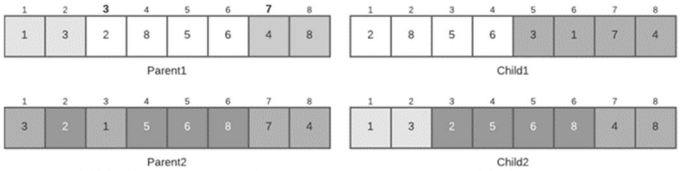

Fig. 3. Child chromosomes after remaining genes are added.

I use Reverse Sequence Mutation as the mutation operator [16]. The mutation operator randomly selects a gene sequence in the child, reverses its order, and inserts it back into the chromosome (Fig. 4).

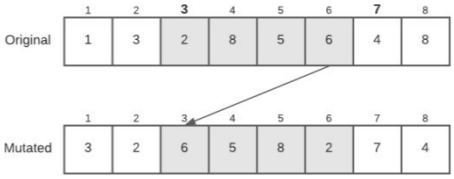

Fig. 4. Result of Reverse Sequence Mutation. The bold 3 and 7 signify the start and end of the gene sequence to reverse.

The children are now assigned to the chromosomes attribute and evaluated. The GA uses a predefined number of generations to run as the stopping criterion. Because of this, I used a while-loop is used instead of a traditional evaluation function. The evaluation function is repurposed to select the optimal chromosome for that generation and track statistics for each generation. If a chromosome from chromosomes has a lower cost, then optimal replaced with it. The final value optimal is returned by the GA as the solution.

C. Wisdom of Artificial Crowds

I modified Yampolskiy, Ashby, and Hassan's implementation of WoAC used to solve TSP for CVRP [17]. Their approach to aggregation counts how often an edge appears in a percentage of the most optimal solutions (elites) to create an agreement matrix. Then the authors use a novel metaheuristic to find a new optimal solution using the agreement matrix, which they describe at a high level.

Their heuristic builds a new solution one connection at a time. The heuristic starts by adding the edge with the maximum agreement (i.e., node 5 to node 4 makes the solution [5, 4]). Then, the heuristic searches for an edge with the maximum agreement that includes the solution's starting node (i.e., 5) but that is not already in the solution (i.e., 4). The same step is repeated for the solution's ending node.

Whichever edge has a higher agreement is added to the solution (i.e., adding the edge between node 3 and node 5 makes the solution [3, 5, 4]). If there's a tie, then the edge between the ending node and the closest node to it (1) that is not yet in the solution is added. This process repeats until there are no more edges to add, and the solution is complete.

Unlike TSP, CVRP has multiple routes per solution to aggregate and in the new solution to build. For aggregation, the

program iterates over the elites and the routes attribute in them to count edges in each route. A gene's index in the list of genes (i.e., the depot gene is at index 0) is used to identify the starting and ending gene (i.e., an edge between gene 4 and gene 5 is added to row 3, column 4 of the agreement matrix).

Unlike Yampolskiy, Ashby, and Hassan's heuristic, each route selects an edge starting from the depot and with the highest agreement. Once the first route is finished, the second route selects an edge starting from the depot and with the highest agreement that is not yet in the solution.

Routes are built the same way as they are to initialize the routes attribute in a chromosome. The only difference is how the next gene to add is selected. The heuristic only adds edges to the ending node.

A helper function finds an edge starting from the end node with the highest agreement not yet in the solution and adds it if one exists. Otherwise, the heuristic adds the depot, and the route is complete. The process of building routes repeats until there are no edges left to add and the solution is complete. A chromosome is initialized based on the solution and returned from WoAC.

IV. Experimental Results

The GA was run for 10 iterations; the WoAC is run for 4 iterations, each using a different number of elites in increments of five, starting at 5 and going 20. The optimal solution, runtime, and summary statistics for each generation (mean, min, and max cost) are returned for each iteration of GA. The new solution and runtime are returned for each iteration of WoAC. The best solution cost, average solution cost, the standard deviation of cost, number of vehicles (routes) in the best solution, average runtime, and total runtime are found for GA and WoAC.

The number of Elites Used (how many of the most optimal chromosomes from each GA are aggregated) and % Difference (1 minus the cost ratio of best chromosome from WoAC over the best chromosome of the GA) are found for WoAC. If the % Difference is 0%, then the chromosomes' costs are identical. A negative percentage indicates WoAC performed worse than the GA, and vice versa for a positive percentage. The one new statistic included is the number of vehicles (same as the number of routes) in the optimal chromosomes for GA and WoAC. The remaining statistics are compiled using Excel and are self-explanatory.

A. Data

I used a subset of CVRP files containing instances from the set E benchmarks. The set E benchmarks are problem instances used by Christofides and Eilon, including their original instances and those of Dantzig and Ramser; Gaskell; and Gillet and Miller [4]. I downloaded a ZIP file of the benchmarks from the Capacitated Vehicle Routing Problem Library (Table I) [18].

CVRP instance files have a .vrp file extension. Each filename specifies the benchmark set it belongs to, n number of nodes (n-1 customers), and K minimum number of vehicles (i.e., E-n13-k4.vrp). Additionally, the known optimal solution is available for each instance on their repository. The subset I choose represents less and more complex problems, with problem difficulty based on the number of customers and Q (i.e., many customers and small Q requires more vehicles, which requires optimizing more routes).

Table I
Set E Benchmarks Used

Instance	Number of Customers (n-1)	Minimum Number of Vehicles (K)	Capacity (Q)	Tightness	Optimal
E-n22-k4	21	4	6000	94%	375
E-n51-k5	50	5	160	97%	521
E-n76-k7	75	7	220	89%	682
E-n101-k8	100	8	200	91%	815

Uchoa et al. explain that Tightness is a ratio of the sum of vehicle capacity (KQ) and total demand [19]. The closer the ratio is to one, the more difficult it is to solve for the optimal solution.

The files use the TSPLIB format found in TSP files used in previous projects [20]. Keywords and their meaning are listed in Table II.

Table II
TSPLIB Keyword Meanings

Keyword	Meaning
NAME	Identifies the data file.
COMMENT	Additional comment(s) on the data.
TYPE	Specifies the nature of the data.
DIMENSION	For a CVRP, it is the total number of nodes and depots.
EDGE_WEIGHT_TYPE	Specifies how the edge "weights" (or "lengths") are given if they are given explicitly.
CAPACITY	Specifies the truck capacity in a CVRP.
NODE_COORD_SECTION	List of node coordinates. An integer gives the number of a node. Two real numbers give the associated coordinates.
DEMAND_SECTION	List of demands of all nodes in a CVRP. The first integer specifies node number, the second its demand. The depot node(s) must also occur in this section. Any depot node's demand is 0.
DEPOT_SECTION	List of possible depot nodes in a CVRP. This list is terminated by -1.
EOF	Terminates the input data. This entry is optional.

The TYPE is CVRP for all files. DIMENSION is the same as n, the number of nodes. EDGE_WEIGHT_TYPE is given as EUC_2D, which specifies edge weights are given by the 2D Euclidean distance between nodes (1). CAPACITY is simply

an integer representing how much of a product a vehicle can hold. Nodes (both customers and depots) are given an integer to identify it and two real number coordinate values under NODE_COORD_SECTION. Each node is assigned a demand based on its identifying integer under DEMAND_SECTION. The DEPOT_SECTION lists depots; the files chosen list a single depot.

B. Results

The results and statistics for all four instances are written to .csv files under a folder named Results. Charts and plots can be displayed as a GUI or (by default) saved as a .png file under Results. I used Lucidchart to plot charts side-by-side. Fig. 5 demonstrates how the optimal solution is constructed over generations. Fig. 6 demonstrates how the optimal solution's cost changes over generations. Fig. 7 shows the optimal solution for the GA and WoAC for each instance.

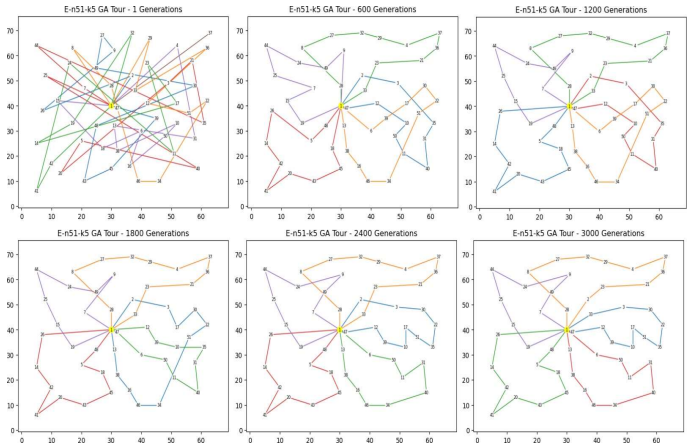

Fig. 5. Demonstration of how optimal routes change over generations using the GA using benchmark E-n51-k5.

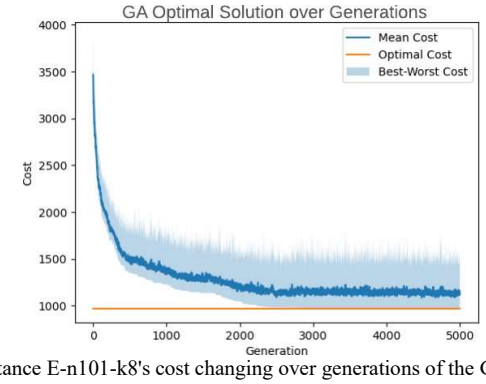

Fig. 6. Instance E-n101-k8's cost changing over generations of the GA

I used Excel to create tables from the .csv files to compare the optimality, runtime, and several other factors for GA and WoAC (Table III).

I also compared the known optimal solution for each instance with the experimental optimal for GA and WoAC (Table IV) and the effect of starting population on optimality and runtime for the GA (Table V).

Table III
Experimental Results of GA and WoAC

Algorithm	GA				WoAC			
Instance	E-n22-k4	E-n51-k5	E-n76-k7	E-n101-k8	E-n22-k4	E-n51-k5	E-n76-k7	E-n101-k8
Generations	200	3000	3000	5000	N/A	N/A	N/A	N/A
Best	**375.28**	**550.8**	**798.8**	**969.19**	495.76	837.05	1329.69	1713.75
Avg.	381.85	579.04	829.02	1075.04	495.76	837.05	1329.69	1713.75
Std Dev	3.7	18.03	18.78	61.59	0	0	0	0
Vehicles	4	5	7	8	4	5	7	8
Avg. Runtime GA (s)	9.34	37.5	61.33	121.39	0.00027	0.00001	0.00010	0.00014
Total Runtime GA (s)	93.43	374.97	613.31	1213.86	0.00027	0.00005	0.00010	0.00014
Elites Used	N/A	N/A	N/A	N/A	5	5	5	5
% Difference	N/A	N/A	N/A	N/A	-32%	-52%	-66%	-77%
Total Runtime GA+WoAC (s)	N/A	N/A	N/A	N/A	93.70	375.02	613.42	1214.00

Bold values indicate the best value for each generation across either population.

Table IV
Experimental Solution vs. Optimal Solution

Instance	Optimal	Minimum Number of Vehicles (K)	Best GA	% Difference GA	Vehicles GA	Best WoAC	% Difference WoAC	Vehicles WoAC
E-n22-k4	375	4	375.28	0%	4	495.76	-32%	4
E-n51-k5	521	5	550.8	-6%	5	837.05	-61%	5
E-n76-k7	682	7	798.8	-17%	7	1329.69	-95%	7
E-n101-k8	815	8	969.19	-19%	8	1713.75	-110%	8

Bold values indicate the best value for each generation across either population.

Table V
Population Size vs. Generations Run

Population	100				500			
Generations	10	50	100	1000	10	50	100	1000
Best GA	**1186.75**	764.52	**598.9**	565.37	1216.1	**747.22**	604.33	**538.06**
Avg. GA	1305.18	858.48	658.72	617.06	**1260.98**	**783.74**	**623.7**	**587.5**
Avg. Runtime GA (s)	0.12	1.19	5.26	10.42	0.77	8.98	46.51	86.67
Best WoAC	1149.9	898.56	**793.47**	970.07	**966.31**	**852.56**	817.69	**778.9**
Avg. WoAC	1199.82	944.49	855.01	970.07	**1069.33**	**904.29**	**830.41**	**789.37**
Avg. Runtime WoAC (ms)	0.04	0.05	0.05	0.05	0.64	2.09	0.61	0.63
% Difference	3%	-18%	**-32%**	-72%	**21%**	**-14%**	-35%	**-45%**

Using E-n51-k5 benchmark instance.

Bold values indicate the best value for each generation across either population.

V. Conclusion

WoAC creates a valid solution without the need to alter it. However, it generally does much worse than GA. One reason may be that CVRP requires optimizing two or more routes. For the TSP, the agreement matrix is used to build the route from the inside out, adding new edges to the start or end. The first edges tend to have the highest agreement, but the remaining edges with nodes that have yet to be added becomes less optimal, leading the metaheuristic to add less optimal edges over time.

This process repeats in CVRP for each route, where locally optimal edges are selected, and the remainder become less optimal. Another possibility is that changing WoAC to always add edges to the end limited it to less optimal solutions than may exist at the start. Future research should investigate new heuristics for combining solutions from the agreement matrix.

The number of customers appears to factor into the %

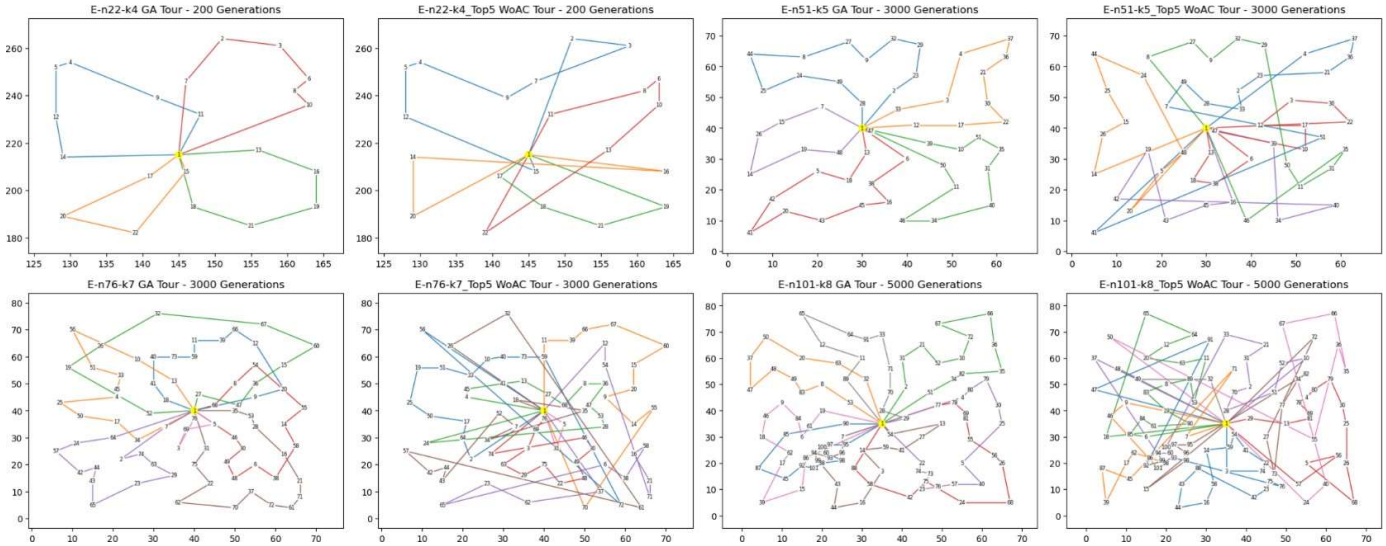

Fig. 7. Most optimal solution found by GA and WoAC for the benchmarks.

Difference between Best GA and Best WoAC. Best WoAC did as well as Best GA for E-n22-k4 (0%) but performed worse on the remaining benchmarks, all of which increases in size (Table III). Likewise, % Difference GA (1 minus the cost ratio of Best GA and Optimal) and % Difference WoAC (1 minus the cost ratio of Best WoAC and Optimal) became progressively worse as the number of customers increased (Table IV).

Comparing the Tightness values in Table I to the % Difference GA in Table IV suggests Tightness had little or no visible effect on the optimality of solutions, or that Tightness values themselves were too alike to reveal differences. While % Difference GA was 0% for E-n22-k4 with Tightness 94%, it was -17% for E-n76-k7 with Tightness 89%. The sample size is too small to draw firm conclusions. Future research should experiment by increasing the number of problem instances tested for more meaningful results.

In terms of speed, the Avg. Runtime GA is slower than Avg. Runtime WoAC (Table III). However, WoAC relies on GA to aggregate solutions. Because of this, combining GA and WoAC only increases runtime by milliseconds. Hypothetically, if fewer generations could be used for GA while getting the same or better solution by applying WoAC afterward, then the hybrid method may perform faster. As it is, I was unable to optimize the program to achieve a reduction in running time relative to GA alone. Without thoroughly testing the hypothesis, I did find that the % Difference for population sizes 100 and 500 were both better when GA ran for fewer generations on E-n51-k5 (Table V). One possibility is that chromosomes converge the more generations GA runs and that there are fewer local improvements to make when combining the solution.

Ideally, GA could be run for more generations to arrive at a better solution. While this does not correct my implementation of WoAC, it would give more optimal solutions from which to aggregate and combine into new solutions.

The number of generations does not guarantee optimality though; Fig. 6 shows how the minimum cost plateaus after so many generations, despite not reaching the optimal (969.19 vs 815). Increasing the solution space by increasing the population size may be necessary to see improvements from increasing the number of generations run.

Two experimental constraints were the computer used to run the metaheuristic and a bottleneck in the GA itself. The computer used is a 4-year-old Asus ZenBook laptop, model UX330UAK, with an Intel Core i5-7200 CPU 2.50-2.71 GHz and 8 GB of RAM. Additionally, larger population sizes increase the GA running by minutes for larger problems but find more optimal solutions than smaller population sizes. To run the metaheuristic in a reasonable amount of time, I used a population size of 100 to generate results and statistics.

Table V shows how a population size of 500 has a greater runtime, but more optimal solutions, all else being equal. Population size 500 performed better than population size 100 for Best GA and Best WoAC (see bolded values in Table V). One reason a larger population size may improve optimality is that the solution space (the initial population) is larger and new routes are possible. A newer computer with greater processing power would have run the algorithms in less time, allowing me to test larger population sizes over many iterations in minutes rather than an hour at a time.

References

[1] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," Management Science, vol. 6, no. 1, pp. 80-91, 1959.

[2] S. N. Kumar and R. Panneerselvam, "A Survey on the Vehicle Routing Problem and Its Variants," Intelligent Information Management, vol. 4, no. 3, pp. 66-74, 2012.

[3] G. Laporte, "Fifty Years of Vehicle Routing," Transportation Science, vol. 43, no. 4, pp. 408-416, 2009.

[4] S. J. de Araujo Lima and S. A. de Araújo, "Genetic Algorithm applied to the Capacitated Vehicle Routing Problem: an analysis of the influence of different encoding schemes on the population behavior," American

Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), vol. 73, no. 1, pp. 96-110, 2020.

[5] A. N. Letchford and J.-J. S. González, "The Capacitated Vehicle Routing Problem: Stronger Bounds in Pseudo-Polynomial Time," *European Journal of Operational Research,* vol. 272, no. 1, pp. 24-31, 2019.

[6] R. V. Kulkarni and P. R. Bhave, "Integer programming formulations of vehicle routing problems," *European Journal of Operational Research,* vol. 20, no. 1, pp. 58-67, 1985.

[7] N. Christofides and S. Eilon, "An Algorithm for the Vehicle-Dispatching Problem," Operational Research Society, vol. 20, no. 3, pp. 309-318, 1969.

[8] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," Applied Mathematical Modeling, vol. 36, no. 5, pp. 2110-2117, 2012.

[9] S. Gunawan, N. Susyanto and S. Bahar, "Vehicle Routing Problem with Pick-Up and Deliveries using Genetic Algorithm in Express Delivery Services," in *AIP Conference Proceedings*, Yogyakarta, 2019.

[10] A. K. M. F. Ahmed and J. U. Sun, "Bilayer Local Search Enhanced Particle Swarm Optimization for the Capacitated Vehicle Routing Problem," Algorithms, vol. 11, no. 3, pp. 31-52, 2012.

[11] R. Yampolskiy and A. El-Barkouky, "Wisdom of artificial crowds algorithm for solving NP-hard problems," International Journal of Bio-Inspired Computation, vol. 3, no. 6, pp. 358-369, 2011.

[12] L. Rosenberg, D. Baltaxe and N. Pescetelli, "Crowds vs Swarms, a Comparison of Intelligence," in Swarm/Human Blended Intelligence Workshop (SHBI), Cleveland, 2016.

[13] J. Surowiecki, The Wisdom of Crowds, New York: Doubleday, 2004.

[14] P. Jakob and R. Günther, "Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification.," *Lecture Notes in Computer Science,* vol. 3562, pp. 41-53, 2005.

[15] D. Zhang, S. Cai, F. Ye, Y.-W. Si and T. T. Nguyen, "A hybrid algorithm for a vehicle routing problem with realistic constraints," *Information Sciences,* Vols. 394-395, pp. 167-182, 2017.

[16] O. Abdoun, C. Tajani and J. Abouchabaka, "Analyzing the Performance of Mutation Operators to Solve the Traveling Salesman Problem," International Journal of Emerging Sciences, vol. 2, no. 1, pp. 61-77, 2012.

[17] R. V. Yampolskiy, L. Ashby and L. Hassan, "Wisdom of Artificial Crowds—A Metaheuristic Algorithm," Journal of Intelligent Learning Systems and Applications, vol. 4, no. 2, pp. 98-107, 2012.

[18] I. Xavier, "CVRPLIB - All Instances," 11 11 2021. [Online]. Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/. [Accessed 12 11 2021].

[19] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, A. Subramanian and T. Vidal, "New Benchmark Instances for the Capacitated Vehicle Routing Problem," European Journal of Operational Research, vol. 257, no. 3, pp. 845-858, 2016.

[20] G. Reinelt, "TSPLIB-A Traveling Salesman Problem Library," ORSA Journal on Computing, vol. 3, no. 4, pp. 376-384, 1991.