Project 6 Research Presentation

Nick York
University of Louisville
Fall 2021 - CSE 545-50 Artificial Intelligence
Taught by Dr. Yampolskiy

Introduction

- Capacitated Vehicle Routing Problem (CVRP)
  - Variant of VRP
  - Combinatorial optimization problem
    - NP-Hard
  - Applicable to supply chain management
- Generalization of traveling salesman problems (TSP) [1]
  - Traveling salesman (singular) ◻ delivery trucks (plural)
  - Locations (P) and their demands (q) are known
  - Trucks have limited capacity (C)
  - Trucks start from and return to a “depot”
- Parameters on constraint determine problem-type
  - $C \geq \sum_i q_i$ ◻ TSP
  - $C \ll \sum_i q_i$ ◻ CVRP
- Objective: minimize total distance

Prior Work (1 of 2)

- Over half a century of research [2-3]
  - New variants
    - VRP with Time Windows (VRPTW)
    - Dynamic VRP (DVRP)
  - Exact algorithms (branch-and-bound algorithms, dynamic programming)
    - Less effective for solving VRP than TSP
      - TSP – up to thousands of vertices
      - VRP – up to 100 vertices
    - Intractable for very large real-world datasets
  - Approximate Algorithms
    - Heuristics (savings algorithms; cluster first, route second)
    - Metaheuristics (genetic algorithms, ant colony optimization, tabu search)
    - Hybrid algorithms

Approach (1 of 4)

- Wrote and modified Python program
- Used 2-stage hybrid algorithm
  - Stage 1: genetic algorithm (GA)
  - Stage 2: wisdom of artificial crowds (WoAC)
    - Based on Yampolskiy, Ashby, and Hassan's implementation of WoAC for solving TSP [6]

Stage 1:

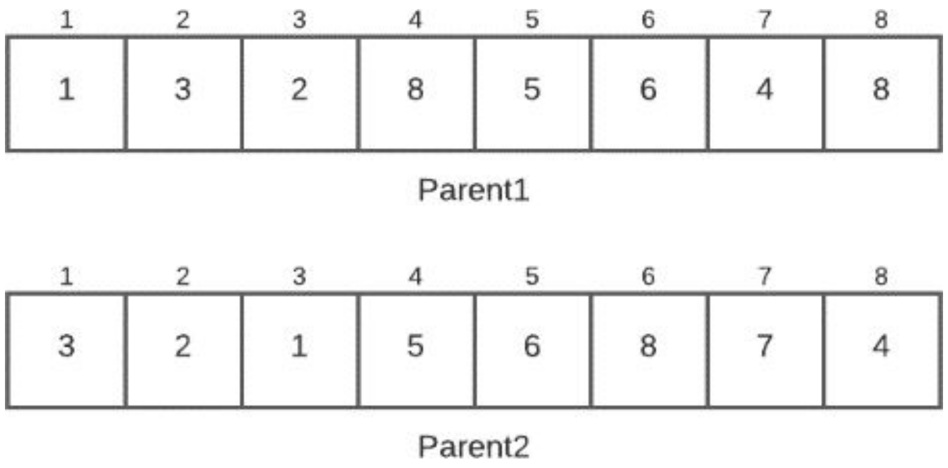
- Problem data is parsed (node coordinates, demands, capacity, depot)
- Data is encoded as Objects
  - Genes
    - Demand
  - Chromosomes
    - Routes built based on genes (their demands) and capacity
    - Cost – total distance to travel each route (based on Euclidean distance)
  - Populations
    - Size - 100
    - Elites - 20

Approach (2 of 4)

Stage 1 (cont.)

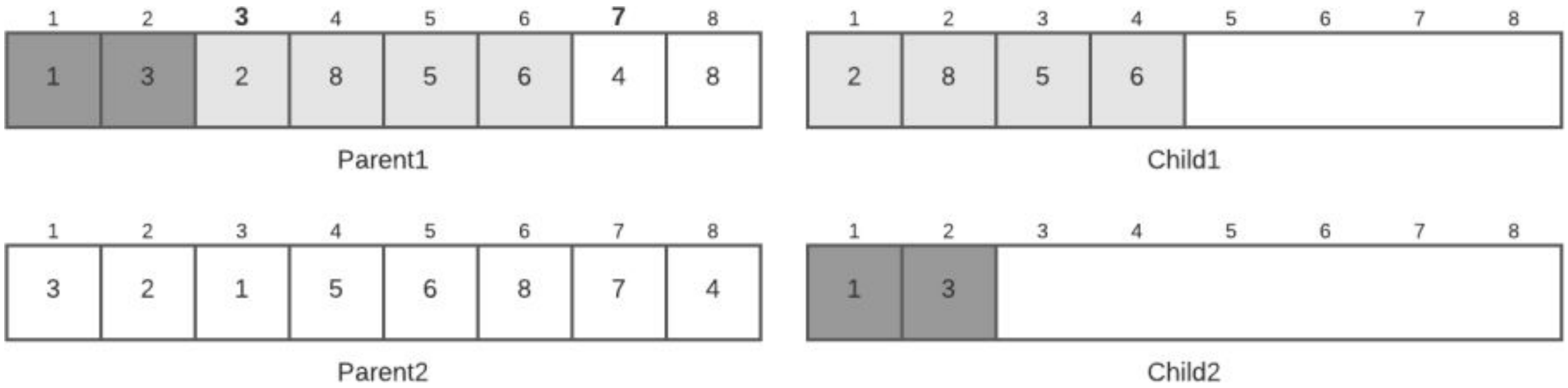
- Genetic algorithm runs
  - Selection
    - 20 elites are automatically added (ensures fittest chromosomes are selected)
    - Fitness: 1/cost
  - Crossover
    - Randomly selected parents from selection pool to mate
    - Children are created until they match population size
  - Mutation
    - Used Reverse Sequence Mutation method [7]
    - Mutation rate: 1%
  - Evaluation
    - Stopping criterion is generations to run

Crossover (1 of 3)



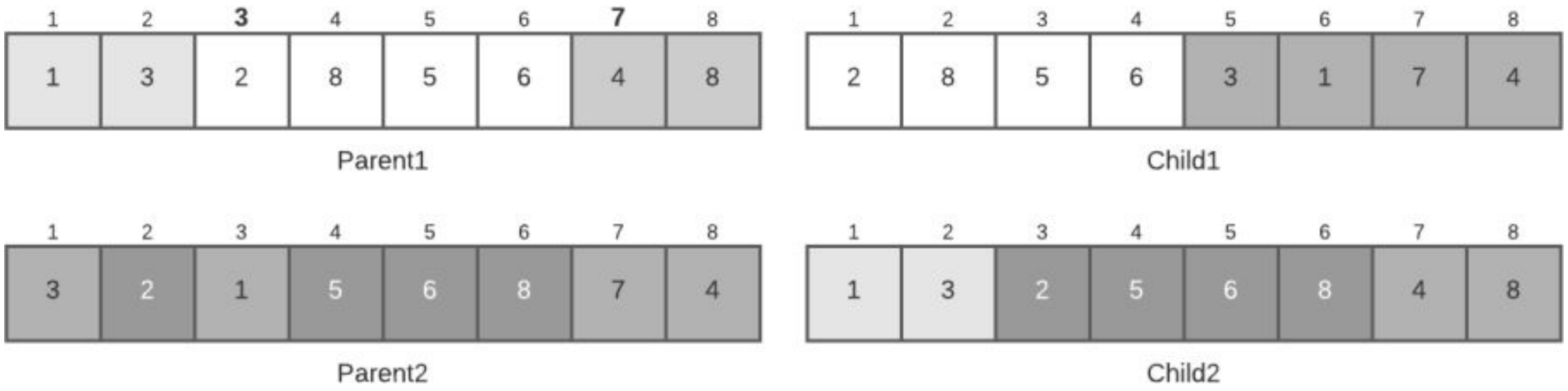
Two parent chromosomes are selected randomly from the selection pool.

Crossover (2 of 3)



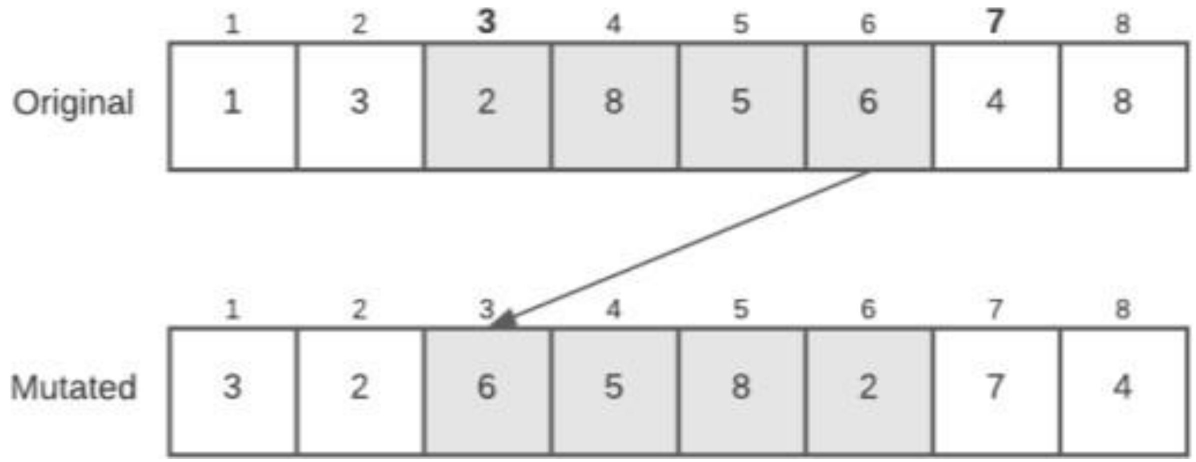
The crossover operator randomly selects a gene sequence from the first parent and combines it with genes from the second parent *not in the sequence* to start creating two child chromosomes. The lightly shaded squares represent the gene sequence added to the first child chromosome. The darkly shaded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

Crossover (3 of 3)



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.

Mutation



The mutation operator randomly selects a gene sequence in the child, reverses its order, and inserts it back into the chromosome

Approach (3 of 4)

Stage 2

- Percentage of most optimal solutions from GA are selected
- Solutions are aggregated
  - Each solution has two or more routes
  - For each route, edges are counted and added to an agreement matrix
- Heuristic builds new solution based on agreement matrix
  - All routes start with an edge starting from depot with highest agreement
  - Edges are added to the end of a route
    - i.e., gene 3 is added to route [5, 4] as [5, 4, 3]
  - If adding a gene exceeds capacity, then depot is added, and a new route started
  - Solution is complete when no genes are left to add

Approach (4 of 4)

- GA is run for 10 iterations
- WoAC is run for 4 iterations, each using a different number of elites
  - 5, 10, 15, and 20 elites (up to one-fifth population size)
- Optimal solution, runtime, and summary statistics for each generation (mean, min, and max cost) returned for GA.
- New solution and runtime returned for WoAC
- Statistics are compiled based on results:
  - The best solution cost, average solution cost, standard deviation of cost, number of vehicles (routes) in the best solution, average runtime, and total runtime are found for GA and WoAC.
  - Number of Elites Used
  - % Difference (only WoAC)
    - 0% - identical solution
    - -% - performed worse than GA
    - +% - performed better than GA

Data (1 of 2)

- Used 4 problem instances from set E benchmarks [8]
- Downloaded .vrp files from Capacitated Vehicle Routing Problem Library [9]
  - Files use TSPLIB95 format (same as .tsp files) [10]

```
NAME : E-n22-k4
COMMENT : ...
TYPE : CVRP
DIMENSION : 22
EDGE_WEIGHT_TYPE : EUC_2D
CAPACITY : 6000
NODE_COORD_SECTION
1 145 215
2 151 264
...
DEMAND_SECTION
1 0
2 1100
...
DEPOT_SECTION
 1
 -1
EOF
```

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

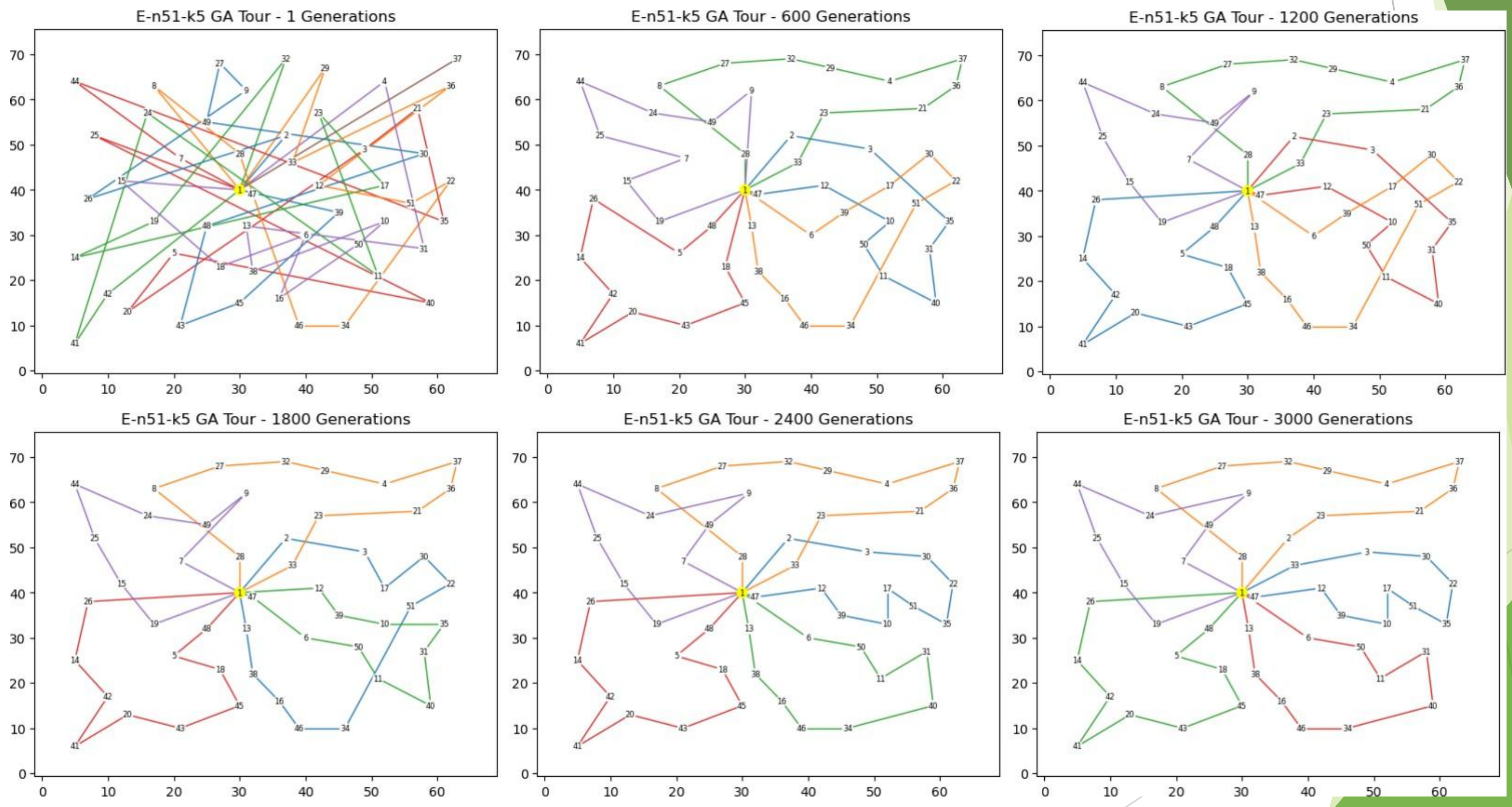
CVRP File (TSPLIB95 Format)

Data is given as keyword/value pairs. The main keywords are:

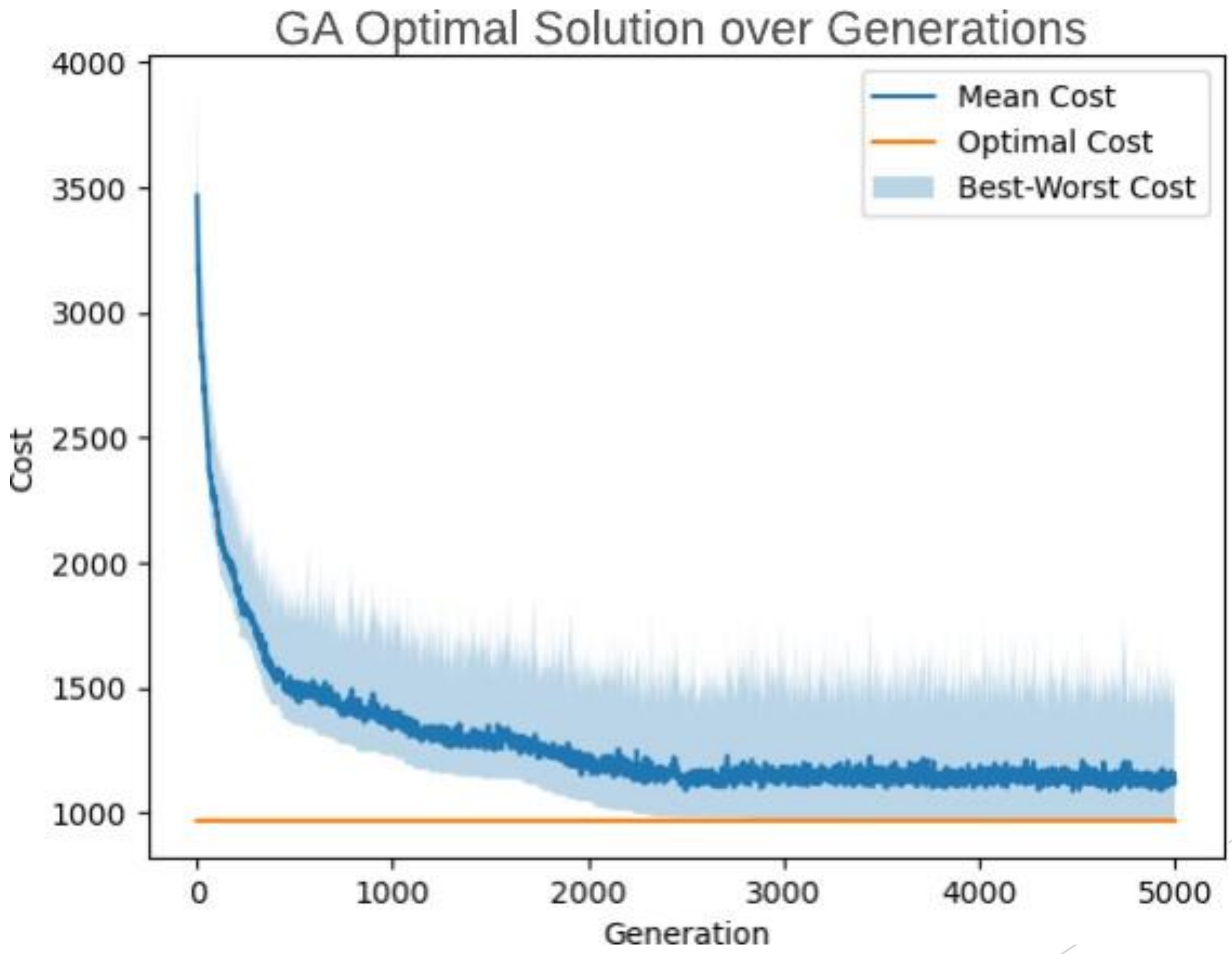
- CAPACITY
  - vehicle capacity
- NODE_COORD_SECTION
  - vertex coordinates
- DEMAND_SECTION
  - corresponding demands
- DEPOT_SECTION
  - depot

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.

Results – GA over Generations (E-n51-k5)



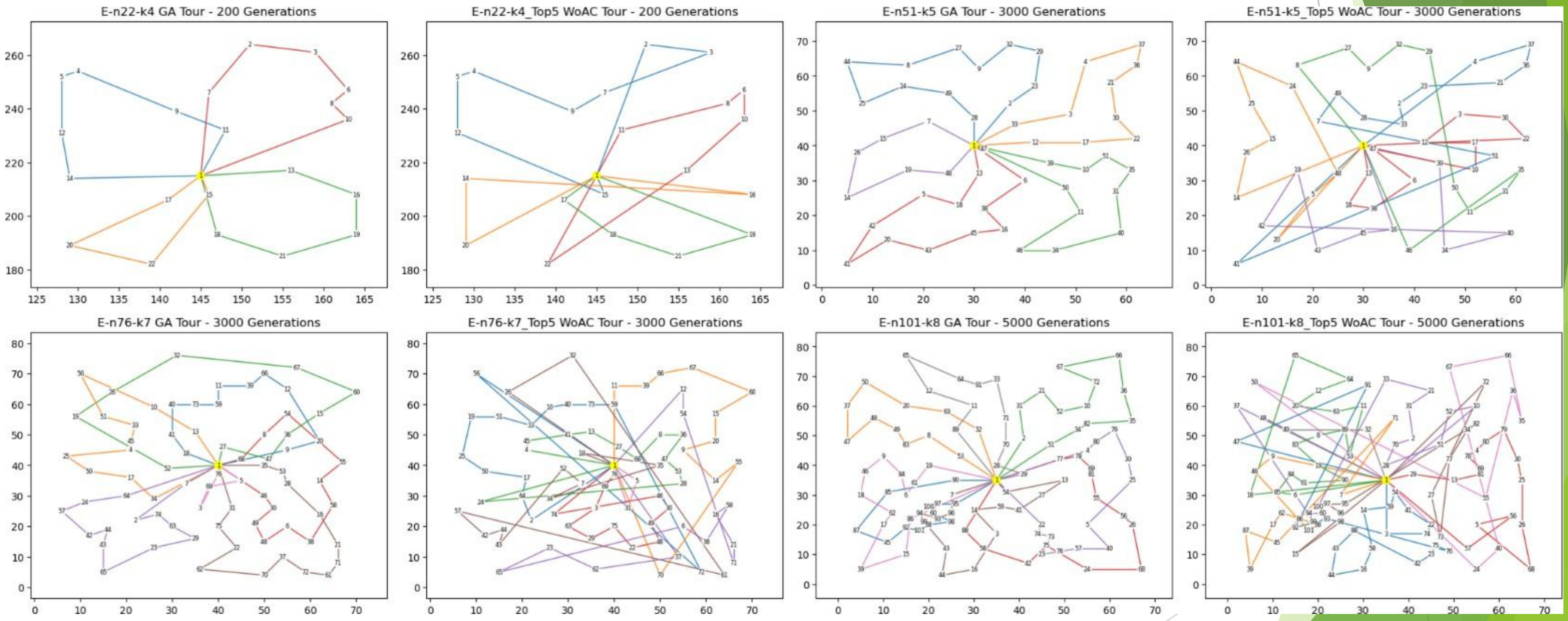
Results – GA over Generations (E-n101-k8)



Results – GA vs. GA+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

Results – GA vs. GA+WoAC



Results – Comparison to 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

Results – Population Size & Generations (E-n51-k5)

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%**

Conclusions

- GA approaches optimal solution
  - Less optimal as problem size increases
- GA+WoAC performs much worse than GA
  - Problem size
  - Heuristic itself
- GA+WoAC avg. runtime relies on GA total runtime
  - GA is much faster when run once than over 10 iterations for GA+WoAC
  - WoAC runs in milliseconds
  - Optimizing generations run could reduce runtime of GA+WoAC
- Larger population size performs better
  - Processing power is insufficient
  - Runtime increases non-linearly as population size increases

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] 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.
- [5] L. Rosenberg, D. Baltaxe and N. Pescetelli, "Crowds vs Swarms, a Comparison of Intelligence," in Swarm/Human Blended Intelligence Workshop (SHBI), Cleveland, 2016.
- [6] 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.
- [7] 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.
- [8] N. Christofides and S. Eilon, "An Algorithm for the Vehicle-Dispatching Problem," Operational Research Society, vol. 20, no. 3, pp. 309-318, 1969.
- [9] 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].
- [10] G. Reinelt, "TSPLIB-A Traveling Salesman Problem Library," ORSA Journal on Computing, vol. 3, no. 4, pp. 376-384, 1991.