Scalability Results

Assumptions

- To evaluate scalability, I tested problem instances with 5, 10, 15, 20, and 25 customers.
- The original instance (15 customers) was both scaled down and scaled up.
- For scaling down, 5 or 10 customers were randomly removed from the original instance.
- For scaling up, additional customers were generated by applying random Gaussian jitter (with increasing standard deviation) to the original customers' x and y coordinates.

Instance	$\lvert V_c \rvert$	$\lvert V_s \rvert$	avg. execution time (s)	avg. number of routes found	avg. optimal cost of route	avg. B&B tree nodes explored
r209C5_td	5	5	8.30	2	184.29	1
r209C10_td	10	5	8.89	2	270.88	1
r209C15_td	15	5	69.94	3	379.81	5
r209C20_td	20	5	229.42	5	579.96	9
r209C25_td	25	5	1596.67	5	672.57	59

Results Explanation

- For the 5-customer tests, I randomly removed 10 customers from the original instance and generated five distinct sub-instances, each containing 5 customers.
- Then, I ran the algorithm on these sub-instances and computed the average execution time, average number of explored nodes, average number of routes found, and the average solution cost.
- The same procedure was applied for the 10-customer tests, this time removing 5 customers from the original instance for each sub-instance.

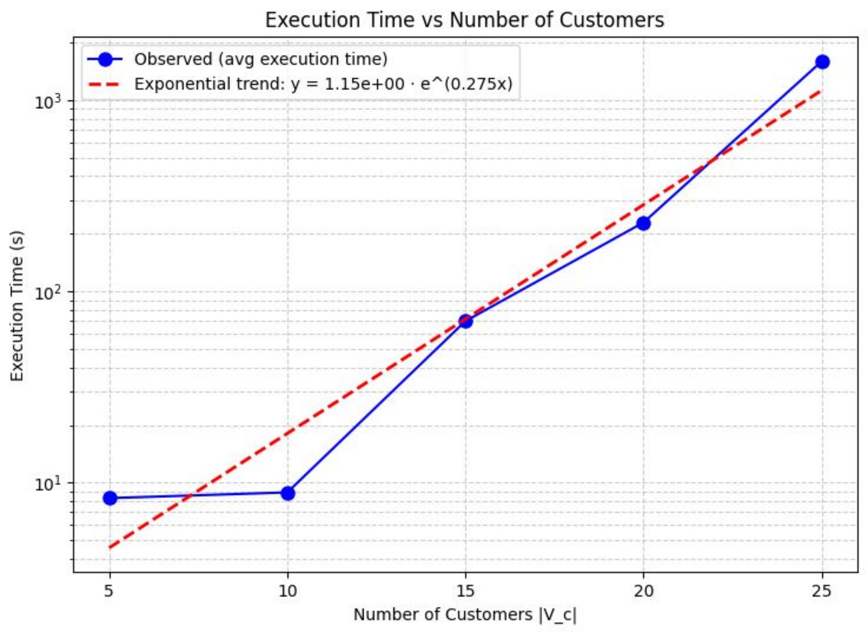
For the 20 and 25-customer tests, I started from the original 15-customer instance and generated additional customer nodes using Gaussian jitter. For the 20-customer case, I created 5 new customer nodes and for the 25-customer case, I created 10 new customer nodes.

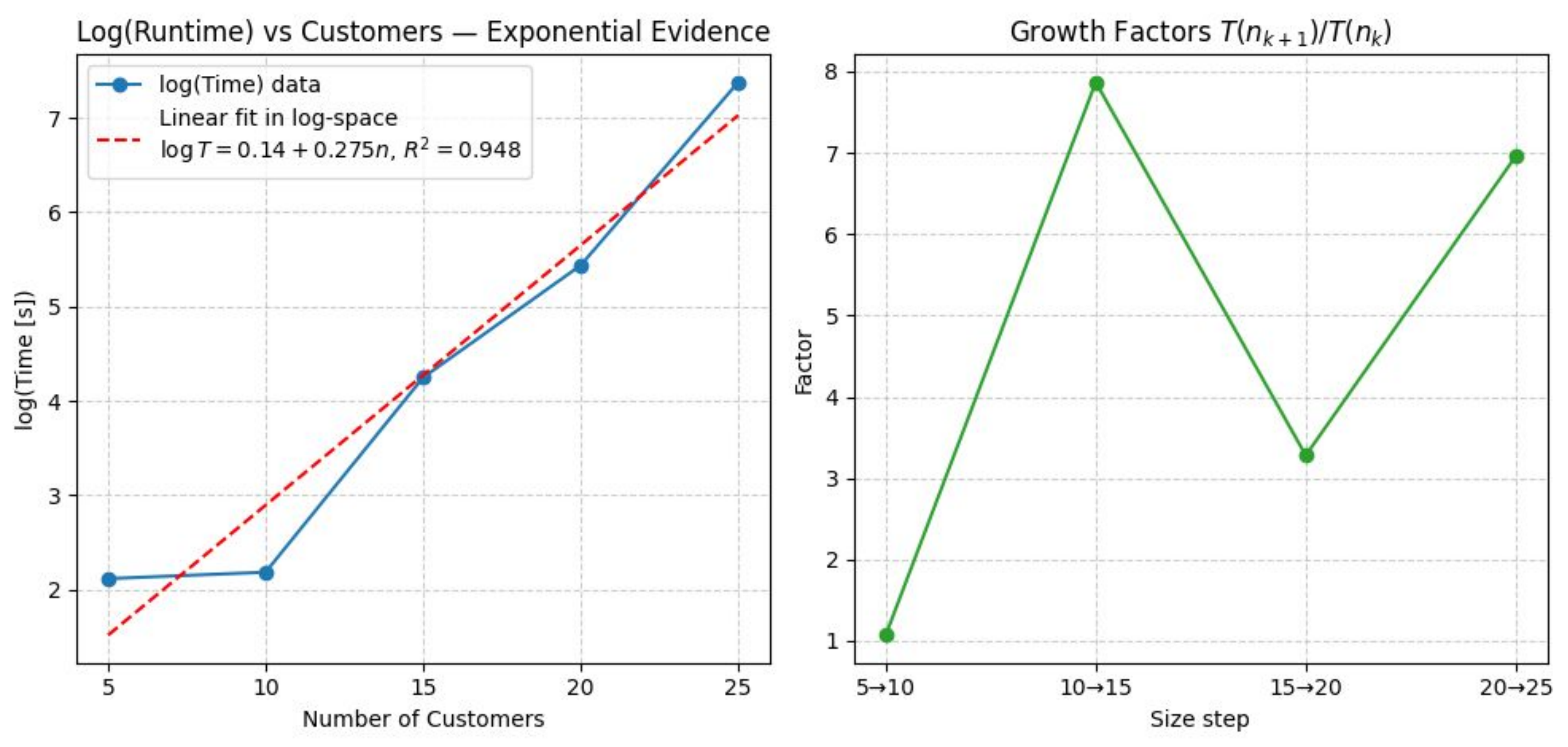
The jitter was applied with increasing variance to simulate progressively more dispersed customer locations. For each case, I produced five sub-instances using standard deviation values of $\sigma = [2.0, 4.0, 6.0, 8.0, 10.0]$.

In every iteration, I randomly selected one of the existing customer nodes as the mean position and generated the new nodes around it using the specified σ.

Finally, for each set of sub-instances, I ran the algorithm and calculated the average execution time, average number of explored nodes, average number of routes found, and the average solution cost.

Execution Time vs | V_c|





Log(Runtime) vs Customers — Exponential Evidence

Growth Factors $T(n_{k+1})/T(n_k)$

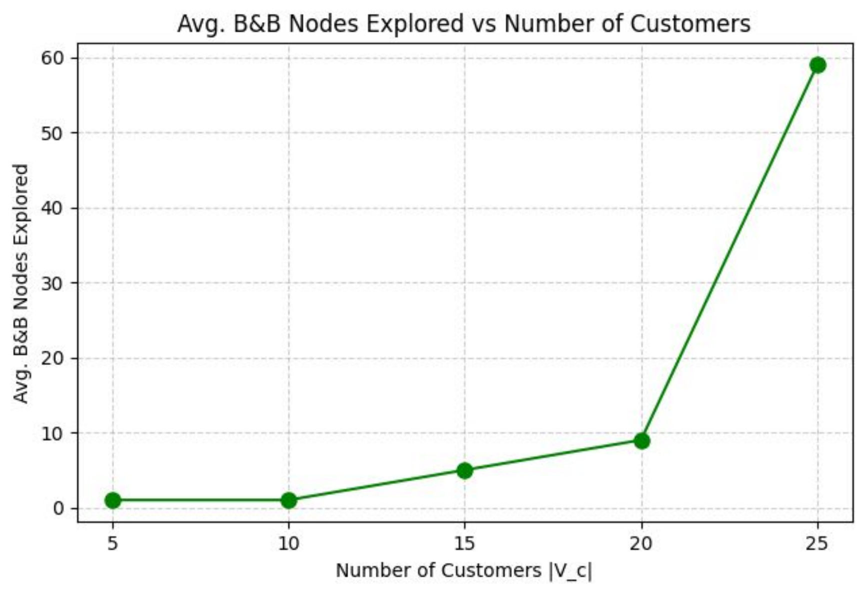
- The plot on the left shows the natural logarithm of the execution time (in seconds) plotted against the number of customers. Each blue dot corresponds to one of your measured instances (5, 10, 15, 20, 25 customers). The red line is the best-fit linear regression performed on the log-transformed values, following the model: T(n) = a + b n,
- where T(n) is the execution time for n customers. The slope b=0.275 and intercept a=0.14 were obtained using least squares regression (np.polyfit) on the (n,logT) pairs.
- The coefficient of determination R^2=0.948 indicates how well the line fits the points, where a value close to 1 suggests a very strong correlation. A linear trend in log-space implies that T(n) grows approximately as exp{0.275 n}, providing visual evidence of exponential-like growth, consistent with NP-hard behavior.

- The plot on the right shows the growth factors between consecutive problem sizes. Each green marker corresponds to the ratio: $T(n_{k+1})/T(n_k)$

- These values quantify how many times slower the solver becomes as the problem size increases.

- The large jumps between steps clearly show that the solver’s runtime increases very quickly as the problem size grows, supporting the exponential trend seen in the left panel.

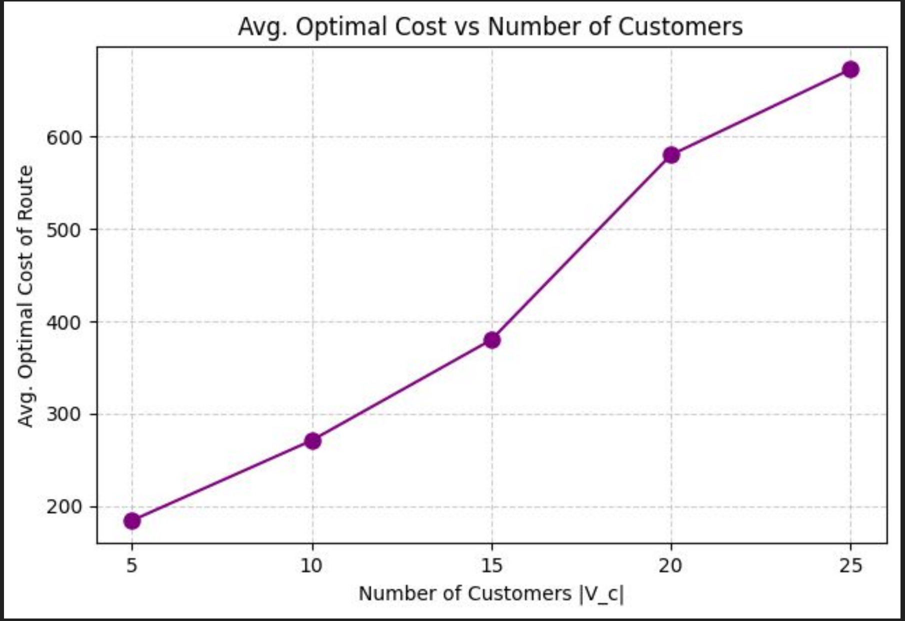
Average B&B Nodes explored vs $|V_c|$



- The plot shows how the average number of B&B nodes explored grows as the number of customers increases.
- For small instances (5–10 customers), very few nodes are explored. As the problem size grows to 15 and 20 customers, the number of explored nodes rises gradually.
- At 25 customers, there is a jump to nearly 60 nodes, illustrating how the search space expands rapidly with problem size.

Average Optimal Cost vs $|V_c|$

The graph illustrates that as customer count grows, the optimal routing cost rises non-linearly. This behavior reflects both the larger coverage required and the combinatorial complexity of finding optimal routes. It complements the runtime and B&B node growth plots, showing that the problem becomes more computationally costly as its size increases.



The End