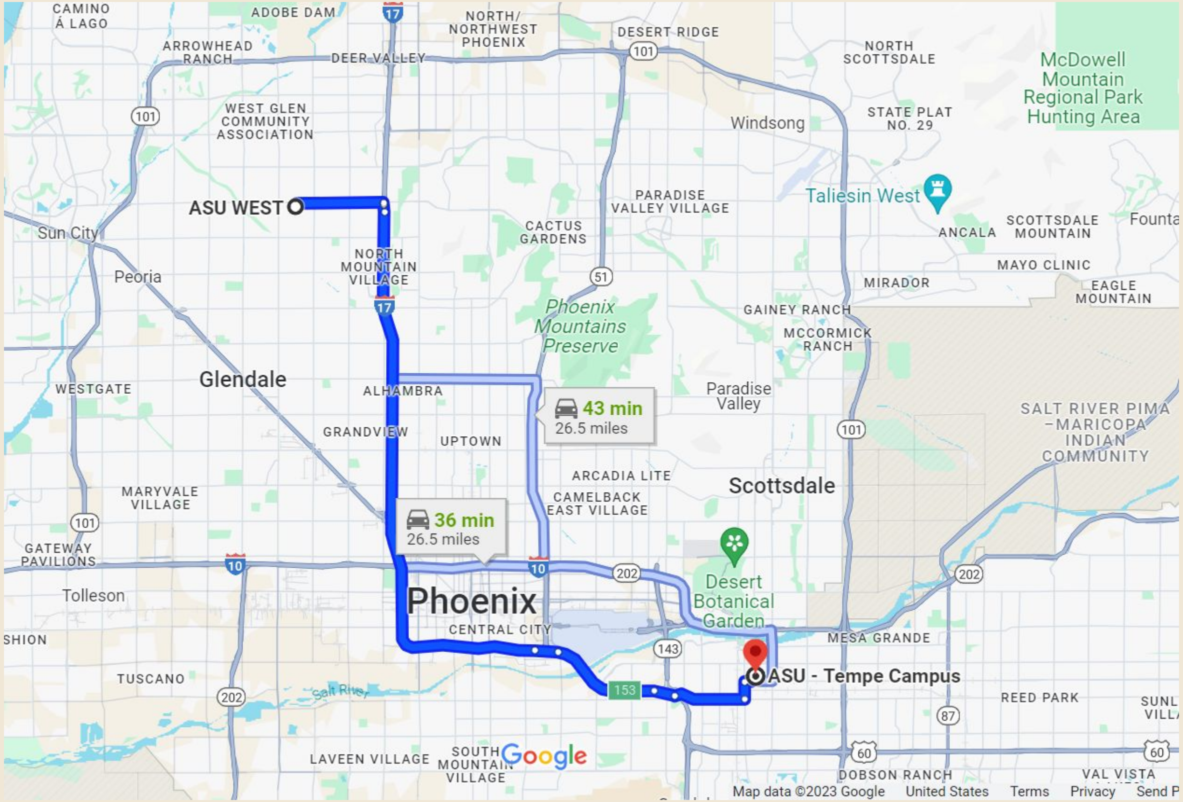
The Hidden Beauty of the A* Algorithm

Adam Kurth, **Agustin Garcia Flores***, Christopher Torres, David Rodriguez, Pavan Rohan Bathula

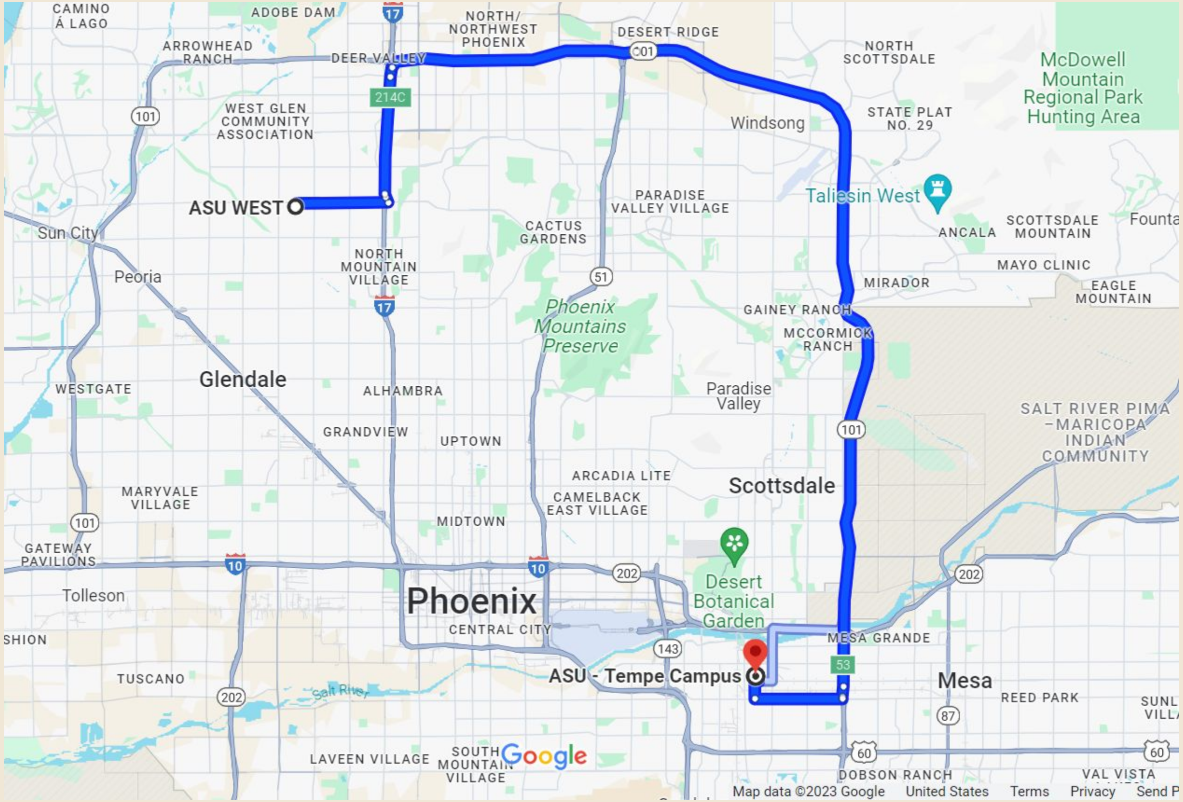
Dr. Malena Español

MAT423: Numerical Analysis I

Motivation


ASU WEST
ASU - Tempe Campus
36 min
26.5 miles
43 min
26.5 miles
Phoenix
Glendale
Scottsdale
Peoria
Sun City
Tolleson
Paradise Valley
Phoenix Mountains Preserve
Desert Botanical Garden
Taliesin West
McDowell Mountain Regional Park Hunting Area
SALT RIVER PIMA -MARICOPA INDIAN COMMUNITY
Map data ©2023 Google
United States
Terms
Privacy

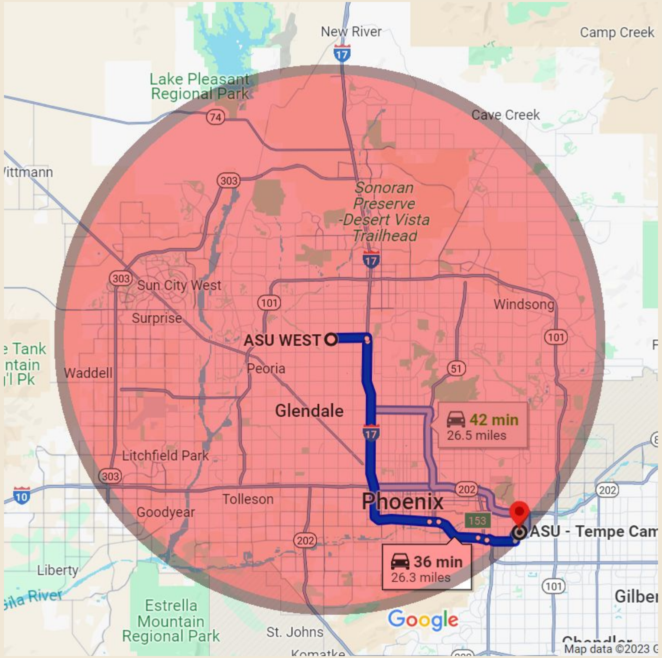
Motivation



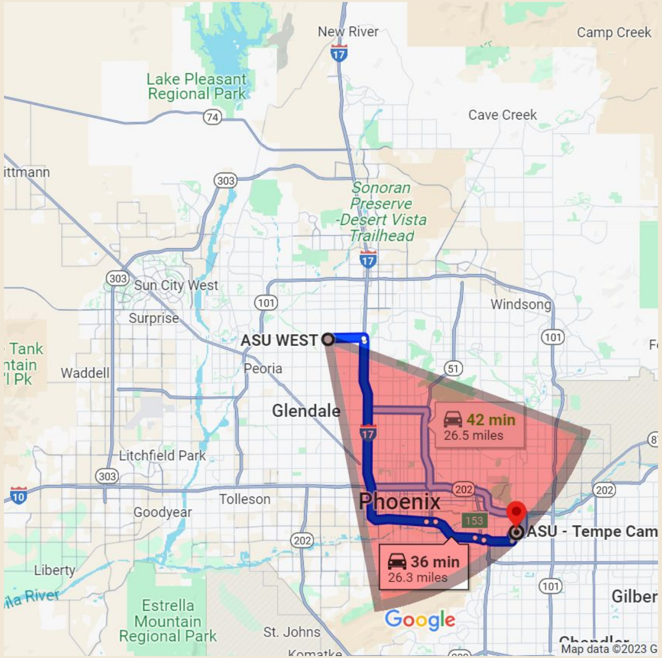
ASU WEST
ASU - Tempe Campus
Phoenix
Glendale
Scottsdale
Mesa
Peoria
Sun City
Tolleson
Paradise Valley
Phoenix Mountains Preserve
Desert Botanical Garden
Taliesin West
McDowell Mountain Regional Park Hunting Area
SALT RIVER PIMA -MARICOPA INDIAN COMMUNITY
DEER VALLEY
DESERT RIDGE
NORTH/ NORTHWEST PHOENIX
ARROWHEAD RANCH
ADOBE DAM
CAMINO Á LAGO
WEST GLEN COMMUNITY ASSOCIATION
NORTH MOUNTAIN VILLAGE
CACTUS GARDENS
PARADISE VALLEY VILLAGE
Windsong
NORTH SCOTTSDALE
STATE PLAT NO. 29
ANCALA
SCOTTSDALE MOUNTAIN
MAYO CLINIC
MIRADOR
EAGLE MOUNTAIN
GAINEY RANCH
MCCORMICK RANCH
WESTGATE
ALHAMBRA
GRANDVIEW
UPTOWN
ARCADIA LITE
CAMELBACK EAST VILLAGE
MARYVALE VILLAGE
MIDTOWN
GATEWAY PAVILIONS
CENTRAL CITY
MESA GRANDE
REED PARK
TUSCANO
LAVEN VILLAGE
SOUTH MOUNTAIN VILLAGE
DOBSON RANCH
VAL VISTA
Salt River
Google
Map data ©2023 Google
United States
Terms
Privacy

Shortest Path Algorithms

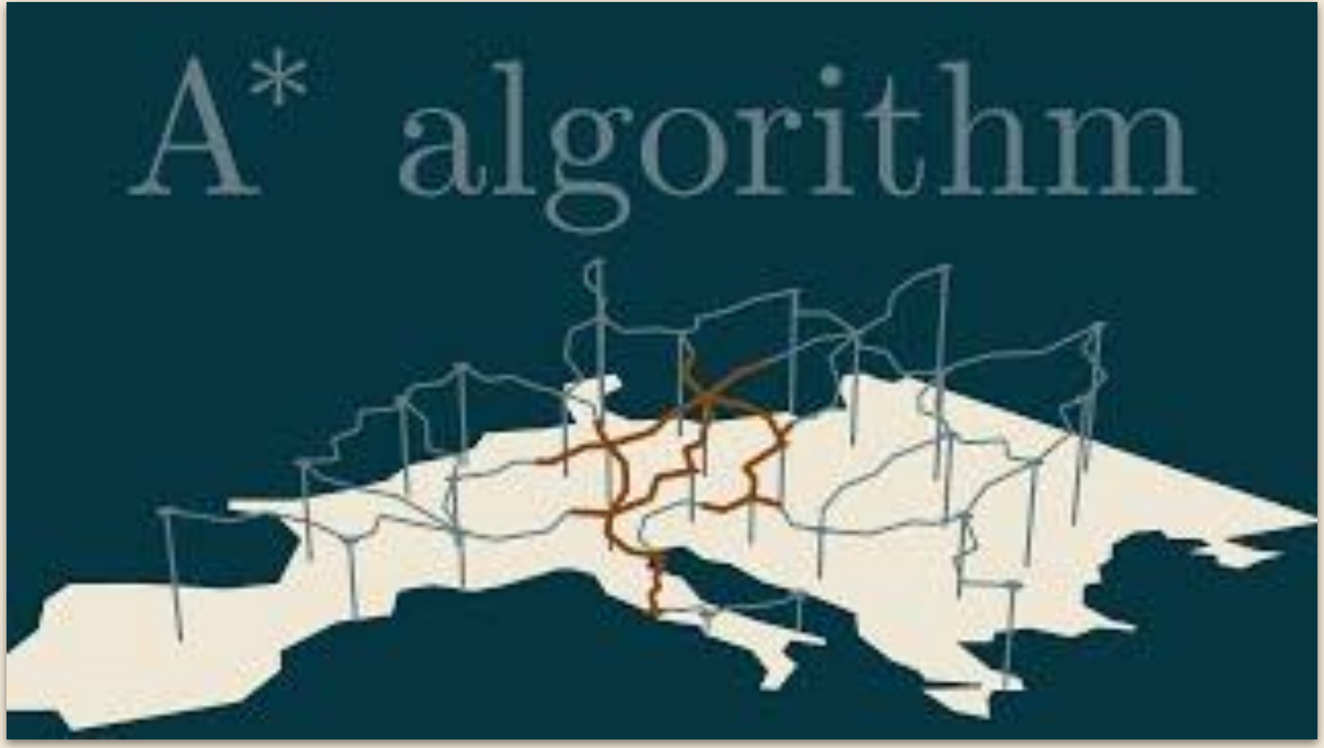
- Dijkstra's Algorithm



- A* Algorithm



Visual Representation

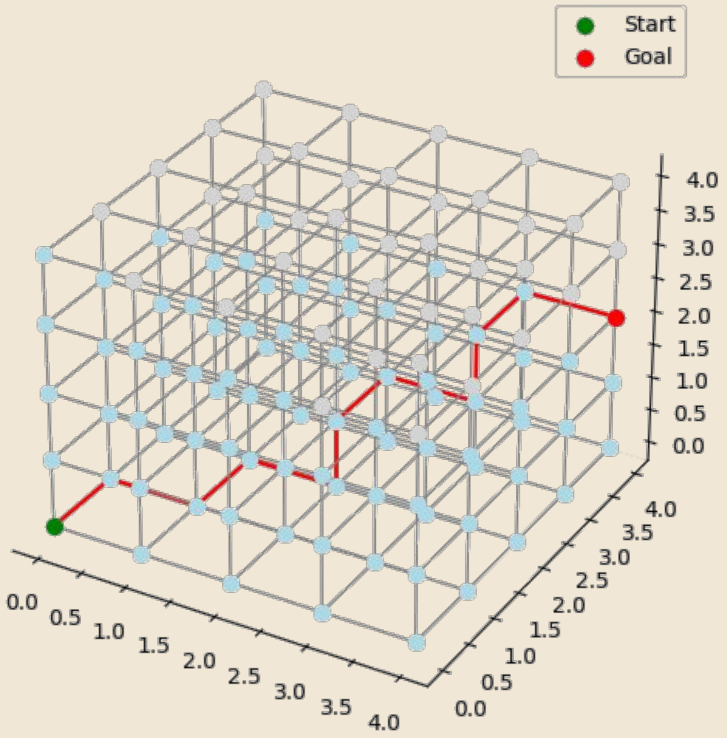


https://youtu.be/A60q6dcoCjw?si=O6S90JPafFu0uy53

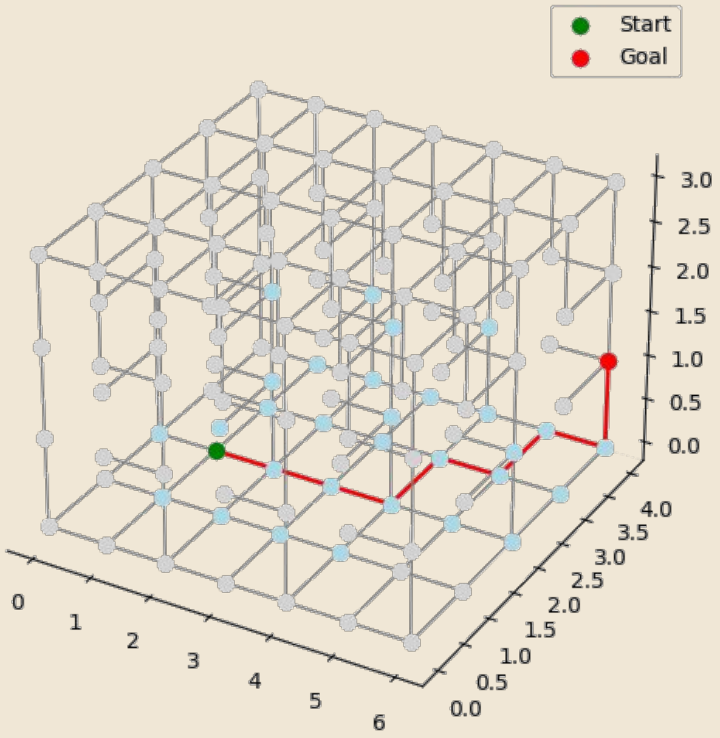
Visual Representation

Past Efforts

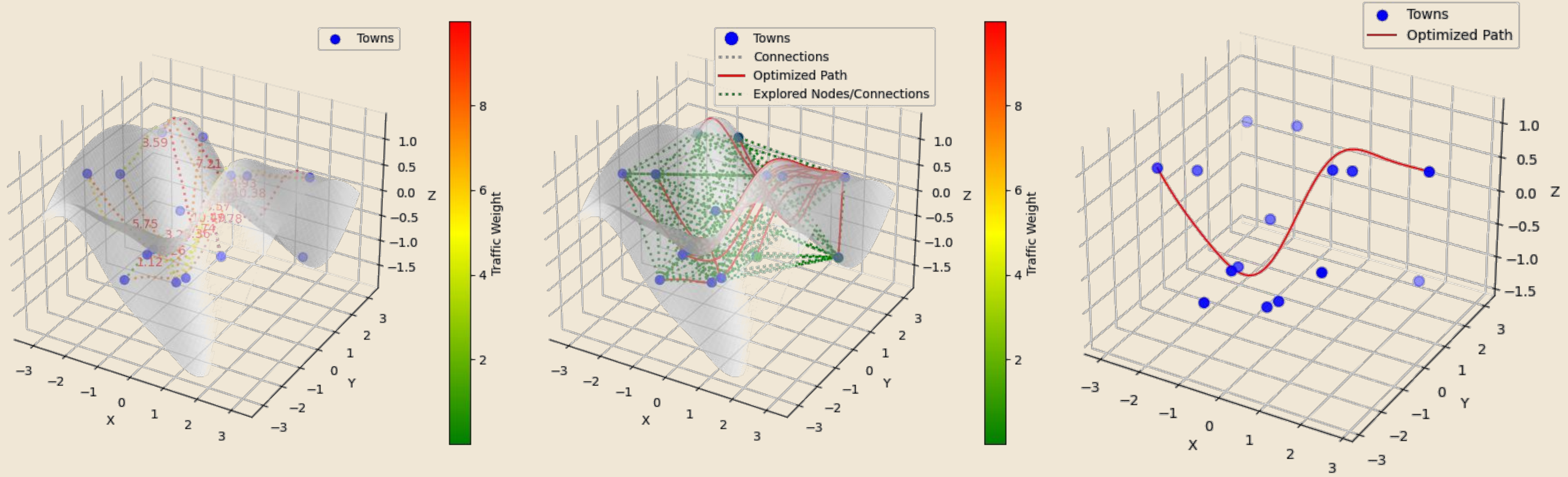
Shortest path from (0, 0, 0) to (4, 4, 2)



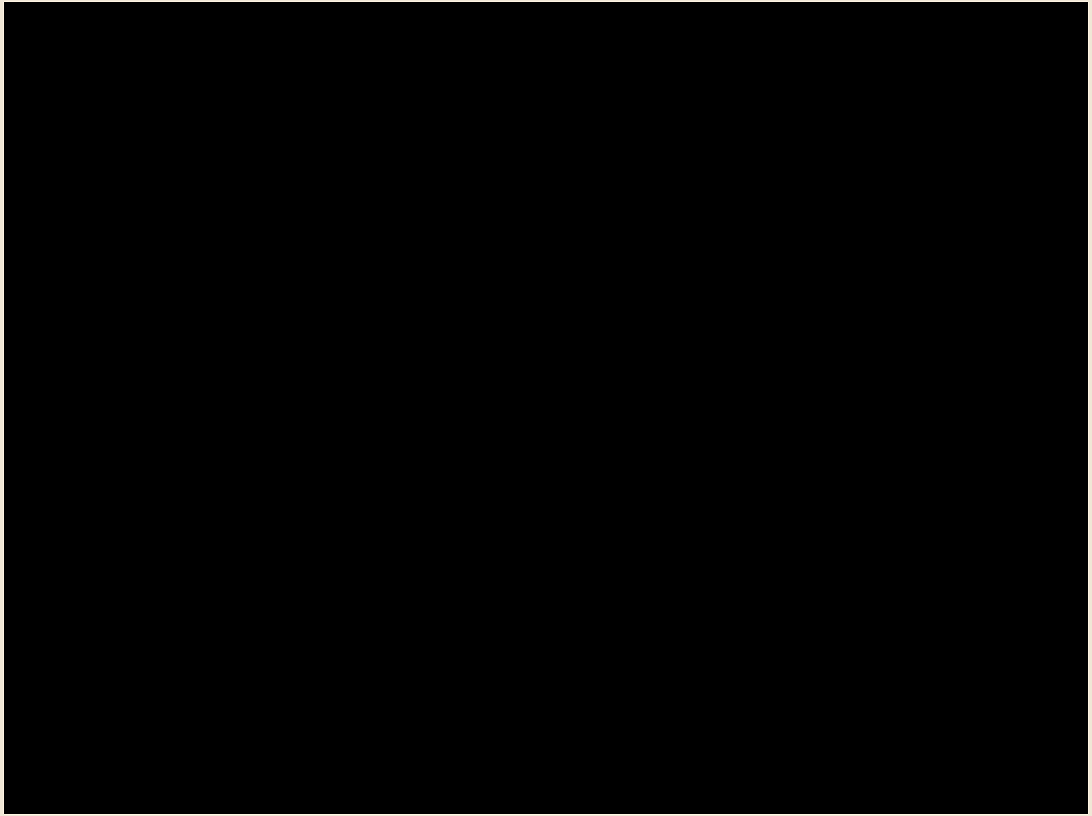
Shortest path from (1, 2, 0) to (6, 4, 1)



Connecting Towns on a Mountain



DEMO



Algorithm Analysis (Part 1)

- General
  - The algorithm’s purpose is to find the shortest path from a starting to end node.
  - Algorithm is an extension of Dijkstra's algorithm in that it does not use priority queues, but instead heaps (i.e. Binary Trees).
- Heuristics
  - A heuristic function is useful for the A* algorithm as it can increase efficiency in certain scenarios.
    - Make no mistake, these heuristic functions should be considered as educated guesses.
  - Types of approximation heuristics:
    - ***Manhattan Distance*** - Constrained to four directions
    - ***Diagonal Distance*** - Constrained to eight directions
    - ***Euclidean Distance*** - Not constrained
  - It should be noted that other functions exist and may cause some unwanted results where the algorithm will find the ending node, but not the optimal path to it.

Algorithm Analysis (Part 2)

- Time Complexity
  - With the addition of different heuristics available for the A* algorithm; we must factor in that calculation when regarding the complexity of the algorithm.
  - Since the heuristic function has a major impact on the performance of a A* search, then it safe to say that a good heuristic allows A* to be away many of the b^d nodes that an uninformed search would expand.
  - Lets express this in terms of an effective branching b, which can then calculate the number of nodes.
    - $N + 1 = 1 + b + (b)^2 + ... + (b)^d$
    - So, good heuristics are determined by a low effective branching factor with the optimal being b=1.
  - Therefore, the time complexity is polynomial time when the map space is a tree, there is a single final node, and at the heuristic function meets the following condition:
    - $|h(x) - h*(x)| = O(\log h^*(x))$
    - Where h* is the optimal heuristic, and the exact cost to get from x to the final node.

Regression and Least Squares

- Degrees of freedom: 1499
- R squared: 97.09% proportion of the variance in the dependent variable explained by the independent.
- For polynomial models, we usually assume iid (independent, identical, distributed) assumptions. But this is **not necessary** for our purposes.

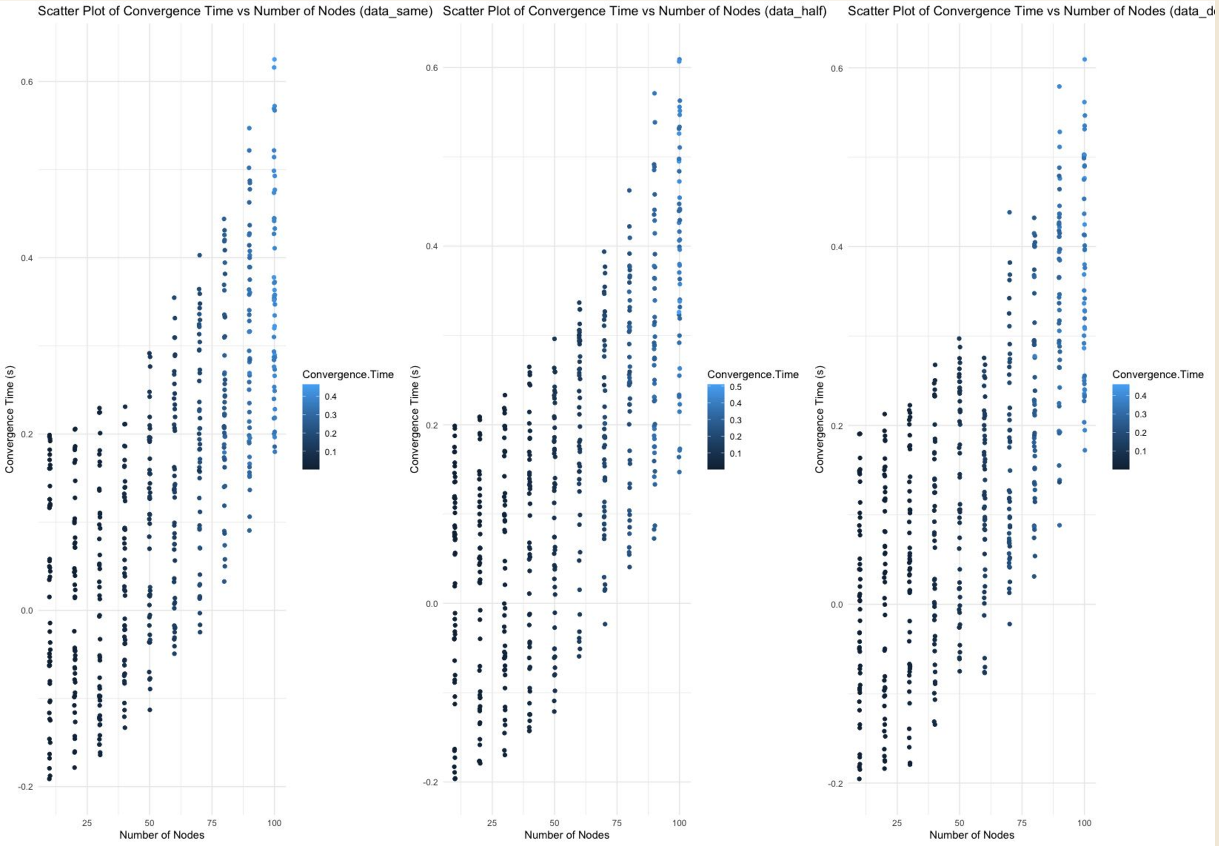
For sake of time, I will explain only the highlights of my analysis.

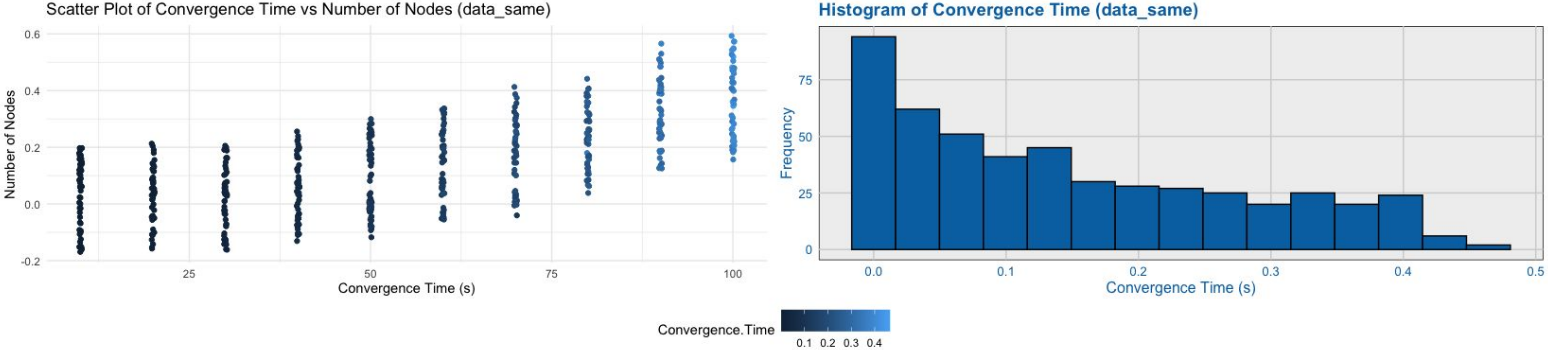
$$E\{Y\}_{\text{added}} = \beta_0 + \beta_1 \text{Total.Cost} + \beta_2 \text{Traffic.Cost} + \beta_3 \text{Number.of.Nodes} + \beta_{31} (\text{Number.of.Nodes})^2$$

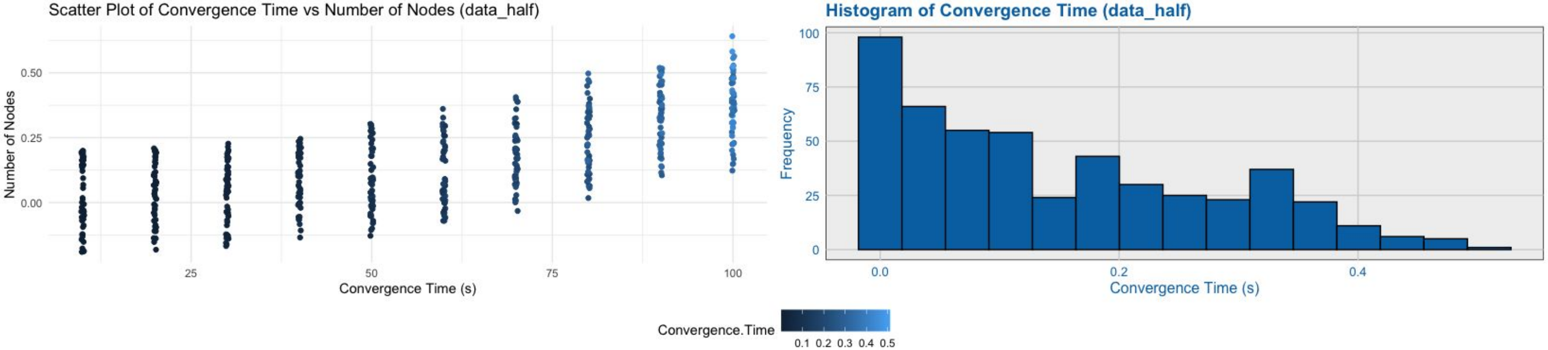
Preview of Convergence Data

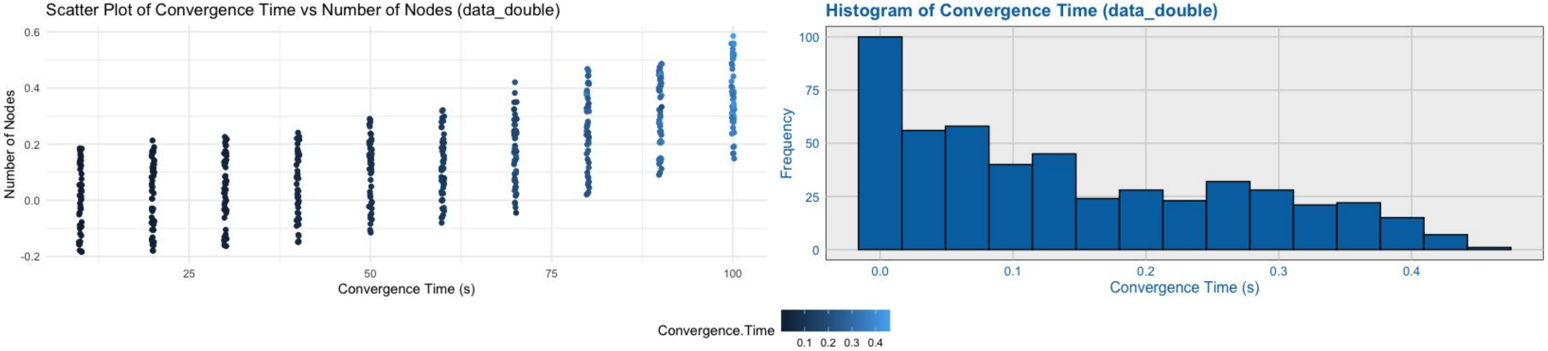
	Convergence.Time	Raw.Cost	Total.Cost	Traffic.Cost	Path	Number.of.Nodes	Number.of.Connections	Elevation.Function	
1	0.0037376881	0.000000000	0.64919569	0.649195694	[1, 5]	10	10	4	elevation_func_1
2	0.0032958984	0.000000000	1.12923265	1.129232654	[0, 7]	10	20	7	elevation_func_1
3	0.0037000179	0.000000000	2.26625532	2.266255324	[0, 2]	10	30	2	elevation_func_1
4	0.0032739639	1.362314629	2.08252390	0.720209275	[3, 9, 4]	10	40	1	elevation_func_1
5	0.0030162334	0.000000000	2.60325916	2.603259161	[2, 5]	10	50	3	elevation_func_1
6	0.0044972897	4.774745049	6.24636958	1.471624534	[3, 4, 0, 8]	10	60	5	elevation_func_1
7	0.0030350685	0.000000000	1.35744622	1.357446223	[3, 6]	10	70	3	elevation_func_1
8	0.0037810802	0.000000000	2.45857158	2.458571579	[3, 6]	10	80	3	elevation_func_1
9	0.0040030479	0.000000000	6.29182205	6.291822053	[8, 9]	10	90	1	elevation_func_1
10	0.0045521259	0.648309413	2.31446890	1.666159484	[3, 7, 6]	10	100	3	elevation_func_1
11	0.0172288418	1.934307183	3.36484211	1.430534923	[1, 18, 5]	20	10	4	elevation_func_1
12	0.0159900188	0.346615310	1.90340073	1.556785420	[4, 15, 16]	20	20	12	elevation_func_1
13	0.0161569118	0.888441196	1.85801347	0.969572279	[8, 10, 0, 14]	20	30	6	elevation_func_1
14	0.0159511566	1.375474874	2.89345378	1.517978910	[1, 19, 6]	20	40	5	elevation_func_1
15	0.0166480541	6.165972464	6.59038140	0.424408934	[16, 12, 6, 18]	20	50	2	elevation_func_1
16	0.0159838200	0.564971895	0.79858123	0.233609337	[1, 8, 14]	20	60	13	elevation_func_1
17	0.0167651176	0.581589022	1.81406037	1.232471347	[3, 16, 14]	20	70	11	elevation_func_1

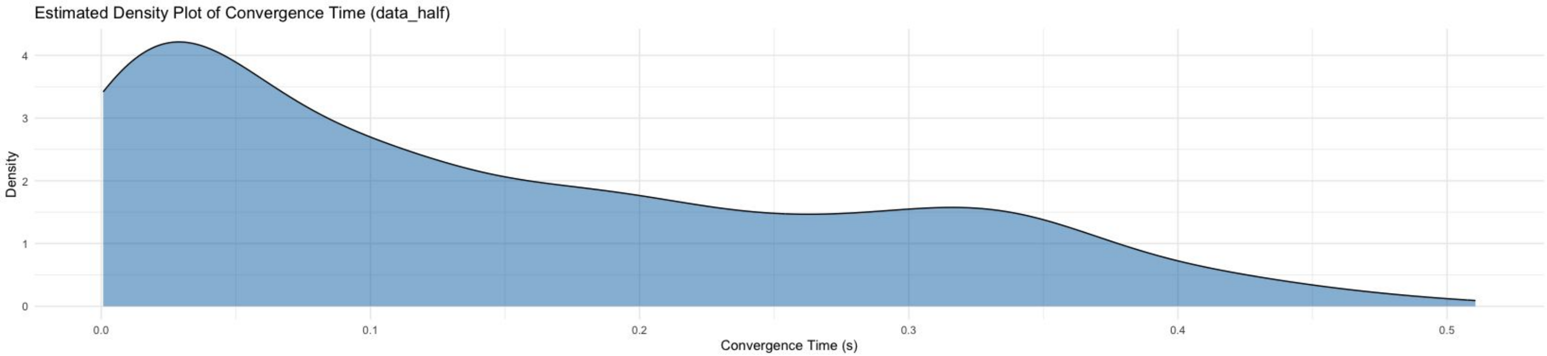
Scatter plots (Num.Nodes vs Convergence.Time)

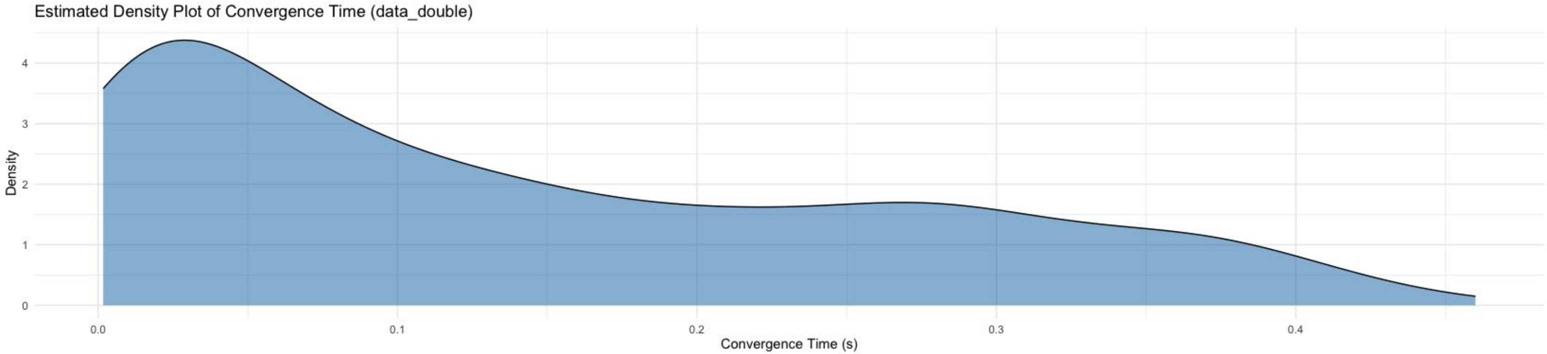


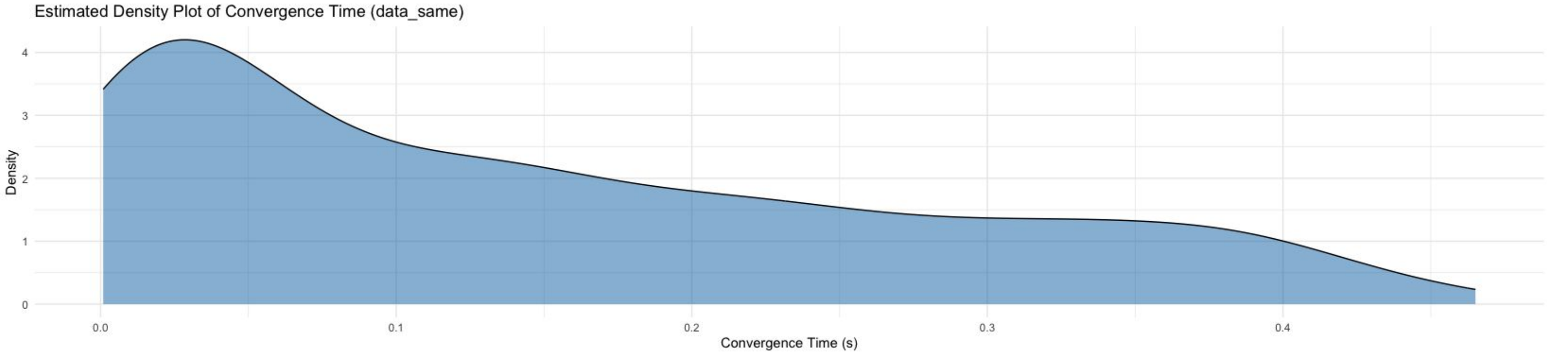

Scatter Plot of Convergence Time vs Number of Nodes (data_same)
Number of Nodes
Convergence Time (s)
Convergence.Time
0.1 0.2 0.3 0.4
Histogram of Convergence Time (data_same)
Frequency
Convergence Time (s)


Scatter Plot of Convergence Time vs Number of Nodes (data_half)
Number of Nodes
Convergence Time (s)
Convergence.Time
0.1 0.2 0.3 0.4 0.5
Histogram of Convergence Time (data_half)
Frequency
Convergence Time (s)


Scatter Plot of Convergence Time vs Number of Nodes (data_double)
Number of Nodes
Convergence Time (s)
Convergence.Time
0.1 0.2 0.3 0.4
Histogram of Convergence Time (data_double)
Frequency
Convergence Time (s)


Estimated Density Plot of Convergence Time (data_half)
Density
Convergence Time (s)
0.0
0.1
0.2
0.3
0.4
0.5


Estimated Density Plot of Convergence Time (data_double)
Density
Convergence Time (s)
0.0
0.1
0.2
0.3
0.4


Estimated Density Plot of Convergence Time (data_same)
Density
Convergence Time (s)
0.0
0.1
0.2
0.3
0.4

Bayesian Information Criterion (BIC)

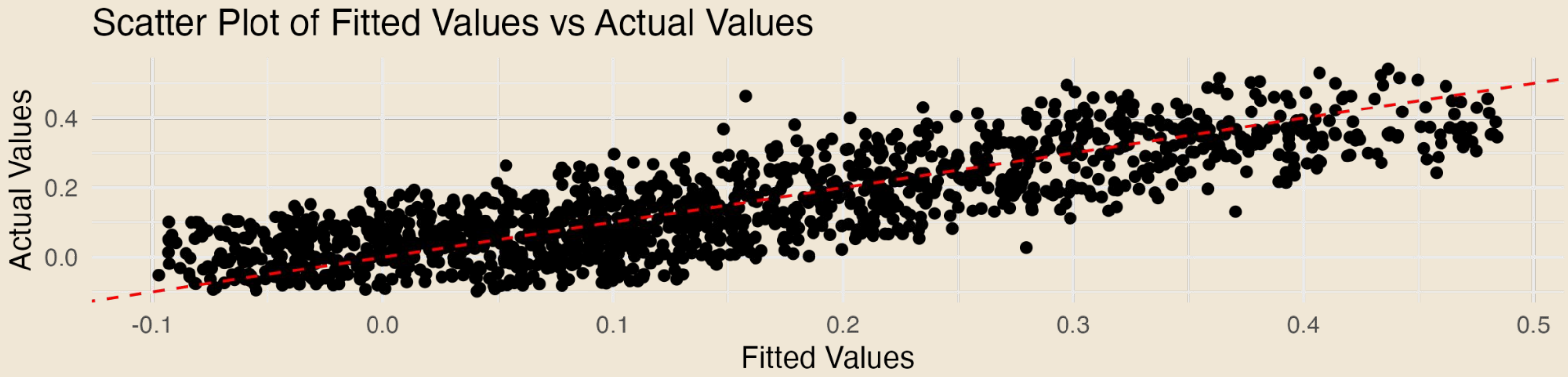
$$\text{BIC} = n\ln\left(\frac{\text{SSE}}{n}\right) + p\ln(n)$$

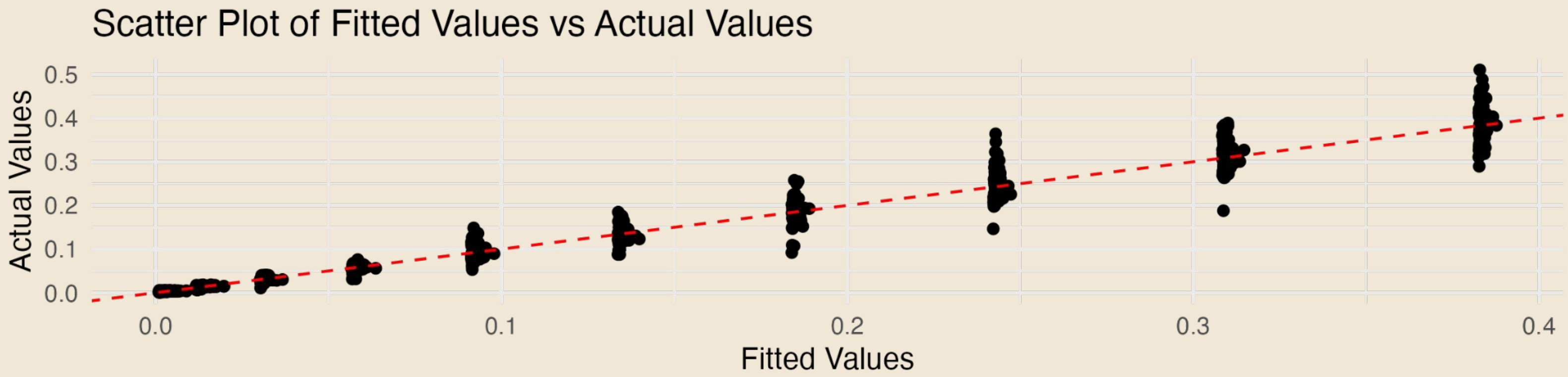
- BIC is an adaptation from AIC where the premise remains the same. Penalizes for complexity while building the model.
- BIC cross examines all possible permutations of predictors, and chooses the one with the most explanatory power, while minimizing the initial value (in red)
- As you can see, this is the model we ended up. This model has an correlation coefficient of 97% (in green).

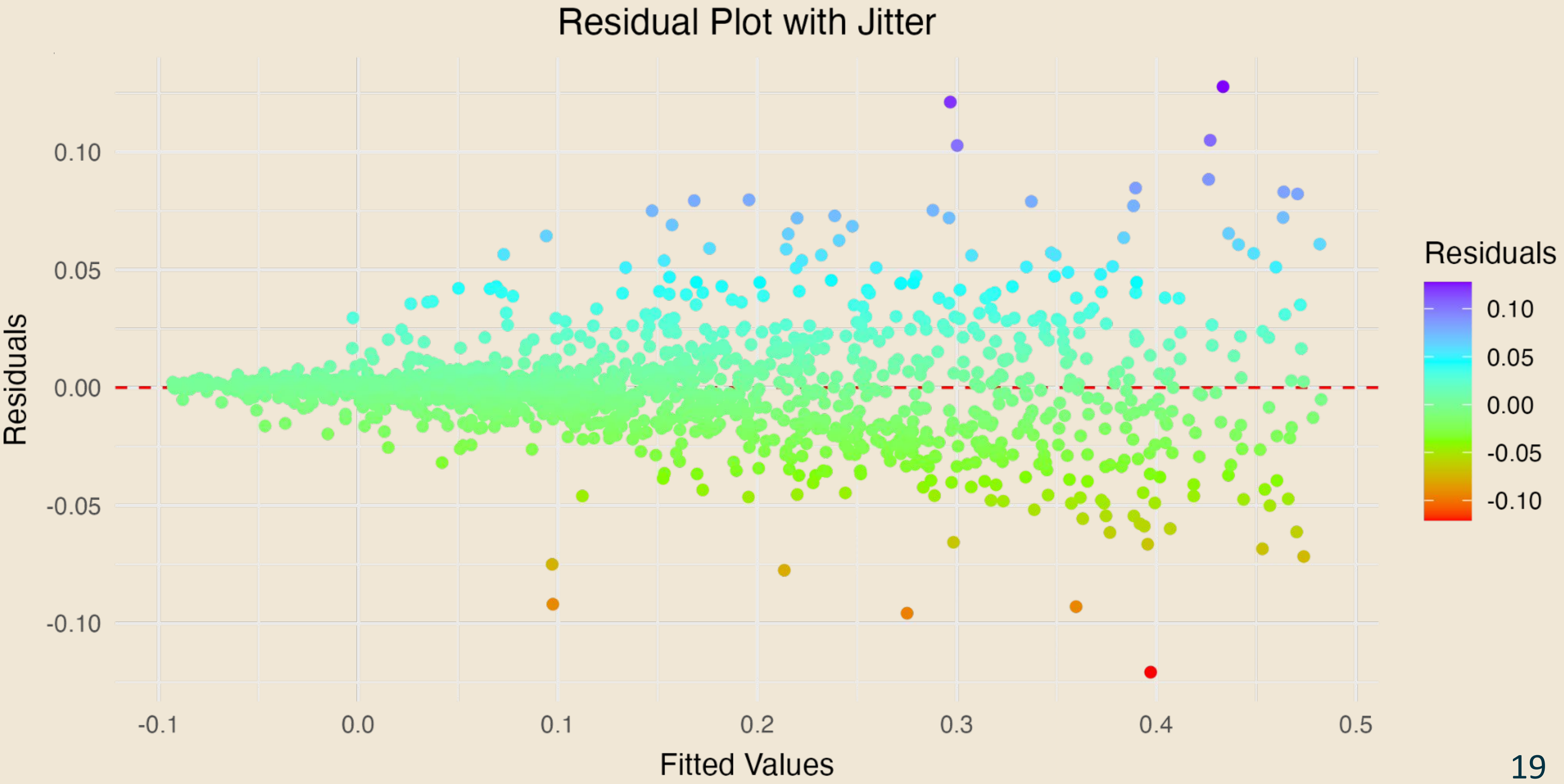
$$E\{Y\}_{\text{added}} = \beta_0 + \beta_1\text{Total.Cost} + \beta_2\text{Traffic.Cost} + \beta_3\text{Number.of.Nodes} + \beta_{31}(\text{Number.of.Nodes})^2$$

```
> numeric_data <- data1[sapply(data1, is.numeric)]
numeric_data <- subset(numeric_data, select = -Elevation.Function)
full.model = lm(Convergence.Time ~ ., data = numeric_data)
stepwise_model <- stepAIC(full.model, direction = "both", k = log(nrow(numeric_data)))
> numeric_data <- subset(numeric_data, select = -Elevation.Function)
> full.model = lm(Convergence.Time ~ ., data = numeric_data)
> stepwise_model <- stepAIC(full.model, direction = "both", k = log(nrow(numeric_data)))
Start:  AIC=-11480.51
Convergence.Time ~ Raw.Cost + Total.Cost + Traffic.Cost + Number.of.Nodes +
    Number.of.Connections + Number.of.Nodes.2


Step:  AIC=-11480.51
Convergence.Time ~ Raw.Cost + Total.Cost + Number.of.Nodes +
    Number.of.Connections + Number.of.Nodes.2

                        Df Sum of Sq     RSS      AIC
- Raw.Cost               1   0.00003 0.69096 -11487.8
- Number.of.Nodes        1   0.00081 0.69175 -11486.1
- Total.Cost             1   0.00104 0.69198 -11485.6
<none>                               0.69094 -11480.5
- Number.of.Connections  1   0.00807 0.69901 -11470.4
- Number.of.Nodes.2      1   1.18528 1.87621  -9989.4

Step:  AIC=-11487.77
Convergence.Time ~ Total.Cost + Number.of.Nodes + Number.of.Connections +
    Number.of.Nodes.2

                        Df Sum of Sq     RSS      AIC
- Number.of.Nodes        1   0.00088 0.69185 -11493.2
- Total.Cost             1   0.00146 0.69243 -11491.9
<none>                               0.69096 -11487.8
+ Raw.Cost               1   0.00003 0.69094 -11480.5
+ Traffic.Cost           1   0.00003 0.69094 -11480.5
- Number.of.Connections  1   0.00809 0.69905 -11477.6
- Number.of.Nodes.2      1   1.20424 1.89520  -9981.6

Step:  AIC=-11493.16
Convergence.Time ~ Total.Cost + Number.of.Connections + Number.of.Nodes.2

                        Df Sum of Sq     RSS      AIC
- Total.Cost             1    0.0023  0.6941 -11495.6
<none>                                0.6918 -11493.2
+ Number.of.Nodes        1    0.0009  0.6910 -11487.8
+ Raw.Cost               1    0.0001  0.6917 -11486.1
+ Traffic.Cost           1    0.0001  0.6917 -11486.1
- Number.of.Connections  1    0.0081  0.6999 -11483.0
- Number.of.Nodes.2      1   21.2150 21.9068  -6317.7

Step:  AIC=-11495.56
Convergence.Time ~ Number.of.Connections + Number.of.Nodes.2

                        Df Sum of Sq     RSS      AIC
<none>                                0.6941 -11495.6
+ Total.Cost             1    0.0023  0.6918 -11493.2
+ Number.of.Nodes        1    0.0017  0.6924 -11491.9
+ Traffic.Cost           1    0.0016  0.6925 -11491.7
+ Raw.Cost               1    0.0005  0.6936 -11489.4
- Number.of.Connections  1    0.0081  0.7022 -11485.5
- Number.of.Nodes.2      1   23.3606 24.0547  -6184.7
>
```

Residual Analysis






Residual Plot with Jitter
Residuals
0.10
0.05
0.00
-0.05
-0.10
-0.1
0.0
0.1
0.2
0.3
0.4
0.5
Fitted Values
Residuals
0.10
0.05
0.00
-0.05
-0.10

Accuracy Evaluation

```
def elevation_func_1(x, y):
    return np.sin(x) * np.cos(y) + 3

def elevation_func_2(x, y):
    return 0.5*np.sin(0.5*x) * np.cos(0.5*y) + 3

def elevation_func_3(x, y):
    return 1/10*(np.sin(x)*np.cos(y) + np.cos(x*y))

def elevation_func_4(x, y):
    return np.sin(x)*np.cos(y) + (x**2 - y**2)/10 + (np.sin(2*x)*np.cos(2*y))/4

def elevation_func_5(x, y):
    return x**2 -3*x*(y**2)

def traffic_func(num_nodes):
    return np.random.rand(num_nodes, num_nodes) * 10
```

Convergence Time	Raw Cost	Total Cost	Traffic Cost	Path	Number of Nodes
0.0037376880645751953	0.0	0.6491956938147314	0.6491956938147314	[1, 5]	10
0.0032958984375	0.0	1.1292326537737407	1.1292326537737407	[0, 7]	10
0.0037000179290771484	0.0	2.2662553241230454	2.2662553241230454	[0, 2]	10
0.0032739639282226562	1.3623146290293664	2.0825239040643684	0.720209275035002	[3, 9, 4]	10
0.003016233444213867	0.0	2.6032591610066538	2.6032591610066538	[2, 5]	10
0.0044972896575927734	4.77474504852364	6.246369582097671	1.471624533574032	[3, 4, 0, 8]	10
0.0030350685119628906	0.0	1.3574462230022433	1.3574462230022433	[3, 6]	10
0.0037810802459716797	0.0	2.458571578712465	2.458571578712465	[3, 6]	10
0.004003047943115234	0.0	6.291822052767635	6.291822052767635	[8, 9]	10
0.004552125930786133	0.6483094127366043	2.3144688967706886	1.6661594840340843	[3, 7, 6]	10

Accuracy Evaluation

	Convergence Time	Total Cost	Traffic Cost
# of Nodes: 10			
elevation_func_1	0.003563	4.01752	2.05694
elevation_func_2	0.002854	1.63114	0.892274
elevation_func_3	0.003685	2.80337	1.37104
elevation_func_4	0.003837	3.39295	3.07367
elevation_func_5	0.003877	3.45934	1.47638
	Absolute Error		
elevation_func_1	0	0	0
elevation_func_2	0.000709	2.38638	1.16466
elevation_func_3	0.000122	1.21415	0.6859
elevation_func_4	0.000274	0.62457	1.01673
elevation_func_5	0.000314	0.55818	0.58056
	Relative Error		
elevation_func_1	0	0	0
elevation_func_2	0.19899	0.59399	0.56621
elevation_func_3	0.03424	0.302213	0.333456
elevation_func_4	0.076901	0.155461	0.494292
elevation_func_5	0.0881	0.138936	0.282244

- From the data gathered, given 10 nodes and the same numbers of connections, we're able to determine the absolute and relative errors based on perturbations on the first elevation function.
  - Elevation function 3 is most accurate to the reference data in relation to convergence time.
  - Elevation function 5 is most accurate to the reference data in relation to both total and traffic cost.

Conclusions

- The A* algorithm is both optimal and finds the shortest distance in certain conditions.
- It combines the strengths of Dijkstra’s algorithm(which guarantees the shortest path) with greedy best first search(which aims toward a goal).
- There is an interesting interlink between the number of nodes, the number of connections, and the convergence time as in general an increase in the nodes will result in an increase in the convergence time due to the larger search space.
- An increase in the number of connections does not necessarily lead to an increase in the convergence time

Thank you!