Optimizing Distribution of Financial Resources in Presidential Campaigns Using Knapsack Algorithm & Manhattan Ambulance Response Optimization: the Allocation, Dispatch and Routing of Ambulances

Team Control Number: #771 October 18, 2016

Table of Contents

1. Summary

1.1(a) Interpretation of Problem One

The presidential election problem can be interpreted as an optimization problem where the objective is to find a distribution strategy for financial resources that will optimize the number of electoral votes received by a candidate by taking into account the poll results for individual states. Recognizing that the United States presidential election is not won with a simple majority in population, we investigated the United States electoral voting system. In the electoral system, electoral votes are ascribed to each state and the candidate who wins a state wins all the corresponding electoral votes.

1.1(b) Problem One: Methods and Conclusions

We divided the problem at hand into three separate objectives/steps. The first objective is to find the expected result of the campaign at the final election through processing the data provided by existing polls. To accomplish this for each state, we first used data provided by several polls that span over a period of time to find the results and confidence interval for each poll. We then used linear regression curve-fit to find the predicted results at the final election and corresponding confidence interval.

Based on the predicted results, we proceeded to use a Logistics Growth Function to find the impact of financial resources on the support for a candidate in a given state. The Logistics Growth Function is an appropriate model as it is commonly used for analyzing marketing tactics, which is essentially the same as presidential campaigns. The previous predicted results were used to establish the parameters for the Logistics Growth Function. Upon finding the correlation, we were then able to find the financial resources necessary to reach the 50% popularity mark.

With this data, we were then able to apply a 0-1 knapsack model to find the optimal distribution strategy for financial resources. This model is appropriate as it is unreasonable for a candidate to invest in a state if he/she cannot reach the 50% threshold. Hence, the two real options are either to invest or not invest in a particular state. We set the financial resources necessary to reach the 50% threshold as the weight of each state. We also set the electoral votes ascribed to each state as the value of each state. The objective is therefore to maximize the total value of states while ensuring the weight of the states does not exceed the campaign budgets.

Through testing our model with a randomly generated set of data, we were able to help a hypothetical candidate achieve an electoral vote of 274 votes with a campaign budget of $105 million dollars, giving the candidate a slight advantage over the opposing candidate. This proves that our model can effectively help presidential candidates obtain a lead in the election. Upon conducting a sensitivity analysis, we also found that adjustments in the budget had little impact on the results of the election, which showcases the strategic success of our model.

1.2(a) Interpretation of Problem One

Our goal for this question is to form a more efficient ambulance system in Manhattan, because the question tells us to manage “more efficient ambulance routes”. The key point here is efficiency.

We define efficiency in terms of response time. In fact, the question implies that the primary problem with the current ambulance system is its 9-minutes response time. This means that the system’s inefficiency is primarily due to its long response time. Thus, to create the most efficient ambulance system, our goal will be to **minimize the average response time of ambulances**. Manhattan uses a grid structure for its streets. All of its streets intersect perpendicularly, which means that we can consider Manhattan as a giant Cartesian grid.

It is because of this structure that we use Manhattan Distance in our solution. Because you cannot go diagonally across a block, only around it, the Manhattan Distance is then the sum of perpendicular distances from point A to point B.

However, Manhattan is not completely aligned with the longitude and latitude directions; there is an angle between Manhattan’s streets and the longitude and latitude directions. We will factor in this angle into our solution.

Additionally, Manhattan has a total of 23 hospitals, which we will utilize in our solution.

1.2(b) Problem Two: Methods and Conclusions

We attempt to improve the ambulance system through three aspects: allocation, dispatching, and routing.

Allocation refers to the allocation, or distribution, of the ambulances in Manhattan among the hospitals of Manhattan. Since certain areas have had more crash sites in the past than others, the allocation of ambulances should also be unevenly distributed in order to minimize the average response time of all ambulances.

To distribute the ambulances in the best way possible, we use **the Greedy Allocation Algorithm.** The Greedy Algorithm does this by maximizing the benefit of any allocation, or minimizing the response time of any allocation.

Dispatching refers to how ambulances are sent in the case of an emergency; in other words, which hospitals should respond to the accident should an accident occur, in order to minimize the average response time of the ambulances. To form an optimal dispatch strategy, we use the **Genetic Algorithm**. It finds the best strategy by simulating the process of selection, crossover and mutation.

Routing refers to finding the path from the hospital to the location of the car accident for an ambulance. Our routing method attempts to minimize the time taken for the ambulance to move from one point to another, which also means minimizing the response time for ambulances.

To find the optimal path for which the time taken is minimized, we use the **A* pathfinding algorithm.** The A* Algorithm finds a best path for the ambulance by finding a path that incurs the least “cost”, or takes the least time.

2. Introduction

2.1 Problem One: Background Information

The objective of the first problem is to efficiently allocate financial resources for campaigns in order to maximize the probability of winning a presidential election through considering the voting information provided by various polls in different states. Prior to applying any mathematical analysis, we researched the presidential election system in the United States and the concept of electoral votes. The presidential election in the United States is simply based on the percentage of the population but electoral votes instead. Each state is ascribed an electoral vote number based on the population of the state. A state with a higher population will receive a larger number for the electoral vote. In total, there are 51 voting districts in the United States, consisting of 50 states and Washington D.C. The total electoral votes amount to 538 and are distributed among the 51 voting districts. Candidates must receive a simple majority of the electoral votes in order to win the election. For each individual voting district, there is a policy called winner-takes-all. This means that in each voting district, all electoral votes will go to one candidate. Citizens in the voting district will vote for representatives representing two parties and the representatives that receive the most votes in each state will cast their vote for one of the candidate. The representative's vote is equal to the total electoral votes ascribed to the state. For example, in the state of Florida there will be one representative for the democrats and one representative for the republicans. Citizens of Florida will vote for these two representatives, and the representative with a higher support percentage will vote for the designated candidate. Usually, the democrat representative will vote for the democrat candidate and the republican representative will vote for the republican candidate with very rare exceptions. Assuming that the democrat representative receives more votes in Florida, the democrat representative will then cast all of Florida's electoral votes to the democrat candidate. The same process applies for all other voting districts.

2.2 Problem Two: Background Information

In this section, we will be forming a more efficient ambulance system for Manhattan.The question implies that the main problem with the current ambulance's system efficiency is the long average response time of the ambulances in Manhattan. Thus, our goal will be to **minimize the average response time** of ambulances in order to form the most efficient ambulance system.

Manhattan uses a grid structure for its streets. All of its streets intersect perpendicularly, which means that we can consider Manhattan as a giant Cartesian grid, with intersections as grid points. It is because of this structure that we use Manhattan Distance in our solution. Because you cannot go diagonally across a block, only around it, the Manhattan Distance is then the sum of perpendicular distances from point A to point B. However, Manhattan is not completely aligned with the longitude and latitude directions; there is an angle between Manhattan's streets and the longitude and latitude directions. We will factor in this angle into our solution. The angle is approximately 29°. Additionally, Manhattan has a total of 23 hospitals, which we will utilize in our solution.

3. Problem One Solution

3.1 Introduction of Model

According the prompt, we interpreted the problem at hand as an optimization problem where we aim to find the financial resource distribution that will maximize the number of electoral votes. We view the poll results as the basis for funding and the presidential campaigns are supposed to arrange their resources based on the trends in the poll results. Hence, the primary objective is to find the optimal electoral votes while taking into account the finite resources of the presidential campaigns based on the statistics from the poll results. This can be subdivided into three separate objectives. The first objective is to predict the final election results based on past poll results without considering additional funding. The second objective is to find the relation between funding and popularity in each state. The third objective is to consider the relation between funding and popularity found in all states and find the financial resources distribution strategy that will optimize electoral votes.

3.2 General Assumptions and Justifications

1. The representatives elected from each state will faithfully cast their votes to their designated candidate

Justification: While the representative elected in each state is technically allowed to vote for the presidential candidate opposite to the designated candidate, the chances of such scenario occurring has been historically low. Furthermore, such incidences occur completely randomly and are generally irrelevant to the parameters of this problem.

2. All states adapt to the electoral voting system. States such as Nebraska and Maine will not follow their own voting system

Justification: Out of all 51 voting districts, only two voting districts do not follow the electoral voting system and have their own voting system, Nebraska and Maine. Considering that both of these two account for a small percent of the overall electoral vote, we decided to assume that these two states also follow the electoral system, instead of considering them separately.

3. We reduce the current US presidential election from a multi-party system into a bi-party system.

Justification: The actual US presidential election consists of multiple parties that go beyond the Democratic Party and the Republican Party. Adhering to the requirements for the question, we decided to adapt the framework of the US election system while ignoring the presence of other parties for our models. Hence, our models only consider democrat voters, republican voters, and undecided voters.

3.3 Solution

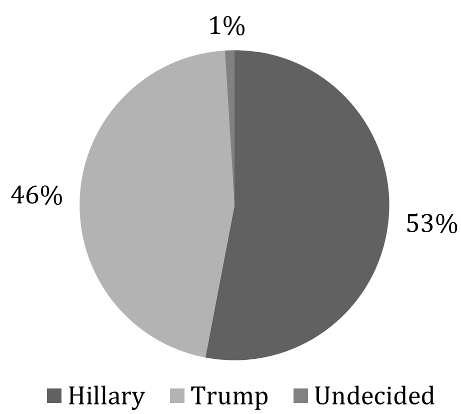
3.3.1 Processing Data in a Poll

The first step of devising an optimal campaign strategy for a presidential candidate is data acquisition. Adhering to the data parameters established by the question, which include the current individual candidate preference and certainty, our sample data comes from 2016 presidential election polls, more specifically, the polls from North Carolina.

We processed the collected data to provide a more explicit analysis. Since there are only two candidates in our theoretical scenario, we excluded votes for other candidates, which accounted for a significant portion in the overall poll.

2016 Presidential Election Poll for North Carolina (9/27-30)	
Hillary Clinton	46%
Donald Trump	40%
Gary Johnson	11%
Other Candidate	2%
Undecided/Don't Know	1%

We will consider Hillary Clinton and Donald Trump to be the two candidates in the theoretical scenario, accounting for 46% and 40% of the poll respectively. In the following example, you will take on the position of Trump, and your opponent will take on the position of Clinton. As seen above, 1% of the sample is undecided and 12% of the sample is voting for candidate aside from the two primary candidates. Hence, it is necessary to tailor the acquired data to our theoretical scenario by scaling the data. We consider the Trump voters, Clinton voters, and undecided voters as the total sample size, and then scale the corresponding percentages. As the sum of Trump voters, Clinton voters, and undecided voters now represent 100%, Hillary's initial 46% is converted into 53%; Trump's initial 40% is converted in 46%; undecided supporters remain at 1%



Scaled 2016 Presidential Election Poll for North Carolina

The sample size of this poll is 660 people, so by multiplying the respective percentages with this sample size, we obtain the following results.

Votes Distribution for 2016 Presidential Election Poll Sample	
Hillary Clinton	349 votes
Donald Trump	303 votes
Undecided	8 votes

Although this number does not represent the actual number of people who voted in the poll as other candidates were excluded, our calculated data is better suited for our theoretical scenario of two candidates.

However, the data acquired does not account for the confidence value, a core component of the problem, of each voter. Hence, the data is not an accurate representation of how voters will actually vote during primary elections. Hence, we devised a method of assigning a numerical probability to these numbers.

We ascribe a value of 1 to each vote for a designated candidate, and a value of -1 to each vote for the opposite candidate. For demonstrative purposes, we will analyze the situation from the perspective of Donald Trump with Hillary representing the opposing side.

As stated in the question, we must consider the confidence value of the voters. We randomly ascribed a decile value between 0.1 and 1 to each voter to represent the confidence value. Currently, the polls only give quantitative values such as “sure” and “not very sure”, which cannot be used for analysis. To improve on the previous metric system, we quantified these confidence values.

Hence the final probability of a person voting for Trump in the future is calculated with the following formula.

$$P(x, y) = 0.5 + 0.5(xy)$$

$\boldsymbol{x}$ is the value of the vote itself, taking the values 1 or -1
$\boldsymbol{y}$ is the confidence value of the voters in their decision

The formula ensures that the probability is within 0 and 1. If the vote is for Trump, with a 50% certainty, the probability of Trump securing a vote is 0.75. If the vote for Trump is casted with 100% certainty, the probability of Trump securing the vote is 1. Conversely, if the vote is for Hillary with a 50% certainty, then the chance of Trump securing the vote is 0.25. Potential voters for Hillary have a less than 50% chance to vote for Trump in the future and vice versa. Hence, our model provides a realistic simulation of future voting probabilities.

The value of y is between 0.1 and 1 because if a voter is 0% sure in his decision, then the final probability is simply 0.5, indicating that the voter is completely undecided. Hence, for voters who have not yet decided, their *x* and *y* value is 0, so their final probability will be 0.5.

	A	B	C
1	x	y	P(x, y)
2	1	0.6	0.8
3	1	0.7	0.85
4	1	0.2	0.6
5	1	0.5	0.75
6	1	0.8	0.9
7	1	0.3	0.65
8	1	0.6	0.8
9	1	0.2	0.6
10	1	0.6	0.8
11	1	0.1	0.55
12	1	0.8	0.9

Snippet of Excel file Used to Calculate P(x,y)

3.3.2 Constructing Confidence Interval

Now, using the processed data from a given poll and the t-distribution, we can construct a 95% confidence interval for the probability of you winning the state.

Justification for the use of a t-interval to construct confidence interval:

In constructing our confidence interval, we chose to find the t-interval for mean instead of the t-interval, as shown by our usage of the t-value instead of the z-value. This is because our data is based on election polls, which only collect data from a sample and not the entire population of a state.

We are able to create a t-interval for mean because the basic assumptions needed to create a t-interval are met:

1. The sample size, or the number of voters, is a number much greater than 30, so the sample follows an approximately normal distribution.
2. Each vote is independent of each other.
3. The data does not give us the population standard deviation. Thus, we cannot use the z-interval which directly uses the population standard deviation. It is more appropriate to use t-interval in this case.

Steps for calculating the confidence interval:

In the following steps, sample calculations are given using processed data from the 9/27-30 North Carolina poll.

First, the population mean can be estimated using the sample mean. This mean is computed using the following formula.

$$\mu = \frac{\sum_{i=1}^{n} p(x_i, y_i)}{n} = 47.30\%$$

The variable *n* represents the sample size; in this case, our sample size is 660. Using our data of 660 voters and the above formulas, we find the average probability is 47.30%. We can also find the standard deviation for confidence interval calculation.

$$\delta = \sqrt{\frac{\sum_{i=1}^{n}(p(x_i, y_i) - \mu)^2}{n}} = 30.30\%$$

Through processing our data, the standard deviation is found to be 30.30%. The margin of error (MoE) can then be derived by the following expression.

$$MoE = t_{\alpha/2} \times \frac{\delta}{\sqrt{n}} = 2.31\%$$

Here, *t* is the critical value. To construct a 95% confidence level, we let $\alpha = 0.05$. The corresponding value of *t*, according to the t-table is 1.96 for a very large *n*. Sample size *n* is considered very large when *n* approaches 1000, and our sample size is well over 600. Using our data, our margin of error is 2.31%. The expression for finding the confidence interval is:

$$\mu \pm t_{\alpha/2} * \frac{\delta}{\sqrt{n}}$$

Hence, the 95% confidence interval for the probability of winning North Carolina is

$$(47.30 \pm 2.31)\% \text{ or } (44.99, 49.61)\%$$

Therefore, according to this data, we can be 95% sure that Trump has a 47.3% chance of winning North Carolina, with a margin of error of 2.31%.

Finding the confidence interval is important because it allows us to consider the expected probability of winning a state and the deviations, which plays an integral role in determining whether a candidate should devote more financial resources. In this example, since the margin of error is 2.31%, Trump's best chance of winning is 49.61%, indicating that Trump is unlikely to win the state. If the confidence interval becomes 5%, then Trump will have a maximum 52.3% of winning, indicating that Trump has a significant chance of winning the state. If the probability for a given state is calculated as 56%, then Trump would be quite likely to win this state, as the smallest probability would be 51% and his largest would be 61%, both probabilities above 50%.

The 95% confidence interval constructed can effectively assess the chances of a candidate winning by taking into account the expected probability and margin of errors, an important indicator for whether the candidate should finance a given state.

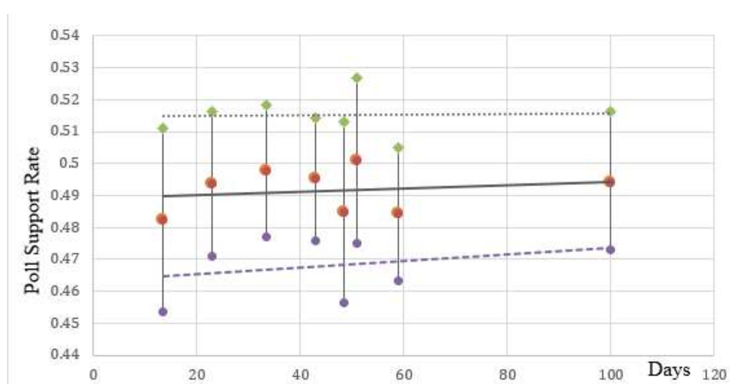
3.3.3 Searching for a Trend in Confidence Intervals

Using the process described in the previous section, a 95% confidence interval for the probability of you, or Trump, winning in a given state can be constructed for every poll. Through collectively assessing the confidence intervals obtained from multiple consecutive polls, we can discover trends and extrapolate the confidence intervals to determine a confidence interval for the probability of winning in the final election. We utilized the several poll results of Florida in August and September to demonstrate this process.

Processed Poll Data for Trump in Florida: Probability of Winning					
Date	Mean	St. Dev.	MoE	Upper	Lower
8.12-15	0.482214	0.294824	0.02882	0.511034	0.453394
8.22-24	0.49368	0.288918	0.022651	0.516331	0.471029
8.29-9.7	0.497766	0.291336	0.020699	0.518465	0.477067
9.10-9.14	0.495098	0.289085	0.019243	0.514341	0.475855
9.16-9.19	0.48475	0.29048	0.028466	0.513216	0.456284
9.19-9.21	0.5011	0.295514	0.025902	0.527002	0.475198
9.27-9.29	0.484146	0.303598	0.02078	0.504926	0.463367

*St. Dev.: Standard Deviation; MoE: Margin of Error

Upon finding the mean value, or the winning probability reported in each poll, and the margin of error, we decided to use a linear regression lines to predict the confidence interval for the final election results. Using linear regression lines is appropriate because the corresponding data in each of the polls were similar, with only small fluctuations. Using a linear regression line can provide a relatively good prediction. The data above is graphed with the x-axis representing time in days and the y-axis representing the probability of winning. In order to ensure that time increments are proportionally factored into our forecast, we defined July 31st as x=0 with each day considered as an increment of 1. To process the time spans of each poll, we used the average of the beginning date of the poll and the end date of the poll to represent the time of the poll. For instance, the poll from August 12th and 15th will be denoted by a single point at x=13.5.



As seen in the graph above, the point where x=100 represents the date of the final election which is November 8th. Using three linear regression lines for mean, upper bound, and lower bound, we found that the confidence interval for the probability of Trump winning in Florida is (0.4729, 0. 5164) or (47.29, 51,64)%, indicating that while there is a chance of Trump winning in Florida, there is an even higher chance of him losing. It should also be noted that the same methodology can also be employed to find a confidence interval using non-linear regression lines, if this model is to be improved in the future.

By repeating the steps above, 51 confident intervals for the probability of winning.in the final election can be found for each of the 50 states and the District of Columbia.

Strengths and Weakness:

Strengths: One of the biggest strengths of this model is its ability to find not only the expected probability of winning in the future but also the confidence interval for the expected probability of winning in the future. As opposed to a single probability, a range of probability will allow for well-informed decisions so candidates can more effectively assess their chances. Furthermore the model can also be adapted for non-linear regression lines to model cases with more fluctuation

Weaknesses: Although the model can effectively forecast the final results of the elections for the majority of cases, it will not perform at its best level if the poll results show major fluctuations. Although the chances of poll results experiencing drastic fluctuations are low, some polls may turn out to be extremely unstable and difficult to accurately predict.

3.3.4 Logistics Growth Function

In the previous section, we demonstrated how to construct a confidence interval for the probability of winning in a given state in the final election, according to recent poll results. We assume that when

$$\mathrm{kR} = \mathrm{R'}$$

where

$\mathrm{R} =$ the amount of financial resources you input into the state

$\mathrm{R'} =$ the amount of financial resources you expect your opponent to input into the state

$\mathrm{k} =$ how effectively you spend each unit of financial resouce compared to your opponent

the percentage of popular votes you get P will be the lower bound of the confidence interval, to create a safe estimate. As the amount of financial resources that you input into the state increases, the percentage of popular votes that you get also increases, until it reaches an upper bound a.

In this case, it is suitable to use a logistics growth curve to determine the relationship between P and R. The logistics growth curve was first published by Pierre Verhulst in 1845 to describe the growth in a population and has since been applied to many fields. Our use of the logistics growth curve can be justified using the following two points:

1) One common application of the logistics growth curve is in business marketing. A logistics growth curve can be used to determine the effect of advertising on the growth of sales of a product. In a Presidential Election, financial resources are mainly devoted to TV, telephone, mail, and face-to-face advertising. The product being advertised is the Presidential candidate, and the growth of sales is the growth of popularity among voters.

2) A major feature of the logistics growth curve is that it is increasing with an upper bound. This matches our situation completely. The percentage of voters voting for you is upwards bounded because there will always be voters who have already pledged their vote to your opponent and your opponent's party. It is unlikely that their votes will be swayed even if you input a significant amount of resources.

The logistics growth curve for our situation can be derived as follows:

$$\frac{dP}{dR} = mP(1-P), \text{ where m is positive constant}$$

$$\frac{1}{P(P-1)}dP = -mdR$$

$$\frac{1}{(P-1)}\,dP - \frac{1}{P}dP = -mdR$$

$$\ln\left(\frac{P-1}{P}\right) = -mR + C$$

$$P = \frac{1}{1-Ce^{-mR}}$$

Suppose when $R = 0,\ P = P_0 = \frac{1}{1-C}$, $C = 1 - \frac{1}{P_0}$

$$P = \frac{1}{1+(\frac{1}{P_0}-1)e^{-mR}}$$

This form can then be rewritten as

$$P = \frac{a}{1+be^{-\frac{kR}{R'}}}$$

where *a*, *b*, *k*, and *R'* are constants. The variables and constants are defined below:

P: The percentage of vote you obtain in a given state ($0 \leq p < a$)

a: Upper bound of popularity that you can gain in the state ($0 < a \leq 100$)

b: Coefficient representing other factors that affect your popularity in a state such as the dispersion speed of advertisements in media, the effectiveness of your campaign strategy, and trends in historic data of past democrat candidates running in the state

R: The amount of financial resources that you input (in millions of $)
R′: The amount of financial resources you expect your opponent to input, assumed to be a given value (in millions of $)
k: The effectiveness of you spending each unit of financial resource compared to your opponent

The goal of this model is to find an optimal value for R, which we denote as R^* such that $P = 50$, meaning that you need to input R^* millions of dollar in a state to guaranteed that you obtain at least 50% of the votes in that state. The steps to finding R^* are listed below, with the state of Florida used as an example.

First, it can be noticed that

$$\text{when } R \to \infty, \qquad p = \lim_{R\to\infty} \frac{a}{1 + be^{-\frac{R}{R'}}} = \frac{a}{1 + b(0)} = a \quad (1)$$

Here, a represents the upper bound of the percentage of votes that you can gain in a given state, when you input a significant amount of financial resources into the state. We define

$$a = \text{\% of democrat voters + \% of undecided voters}$$

Democratic voters refer to all voters that selected the democrat candidate, you, in recent polls. Undecided voters refer to all voters who were undecided in recent polls. We assume that all republican voters who voted for the republican candidate, your opponent, in recent polls will also vote the same in the final election. This may be a slight underestimate of the value of a , but overall it is a reasonable estimate because historic data indicates that the majority of voters who vote for a candidate in polls also vote for that candidate in the final election.

To demonstrate how a is calculated, we can use data from Florida, a traditional swing state, as an example and examine how funding will impact the poll results in Florida.

The following table contains data on the number of Democrat, Republican, and undecided voters in several polls that took place from August to September.

Table 2.1: Florida Poll Results: Data on Voters

Poll Dates	**8/12-15**	**8/22-24**	**8/29-9/7**	**9/10-14**	**9/16-19**	**9/19-21**	**9/27-29**	**Total**
Democrat Voters	170	285	361	401	178	235	374	2004
Republican Voters	210	299	362	401	200	229	410	2111
Undecided Voters	22	41	38	65	22	36	36	260

Total Voters	402	625	761	867	400	500	820	4375

Using the data above, the percentage of each type of voter can be obtained.

Table 2.2: Florida Voters in Percentages	
Type of Voter	Percentage of Total Voters (%)
Democratic Voters	45.81
Republican Voters	48.25
Undecided Voters	5.94

Hence,

$$a = 45.81 + 5.94 = 51.75$$

After a is calculated for a state, there are two cases:

1) $a \leq 50$

This means that your chance of winning in the state will never reach 50% no matter how much financial resources is inputted. Under such a situation, we define $R^* = 0$, since there is no chance of you getting any electorate votes from that state. You should not input any resources into that state.

2) $a > 50$

The following steps can be used to calculate R^*. It can be noticed that

$$\text{when } kR = R', \qquad P = \frac{1}{1 + be^{-\frac{kR}{R'}}} = \frac{1}{1 + be^{-1}} = \frac{a}{1 + \frac{b}{e}} \qquad (2)$$

As explained previously, when $kR = R'$ in a state, P should equal the lower bound of the confidence interval for your probability of winning in that state in the final election. In Florida, such confidence interval is (47.29, 51,64)%. Therefore,

$$\frac{a}{1 + \frac{b}{e}} = 47.29$$

Now, there are two equations in total, and the values for “a” and “b” can be solved out. For Florida, the equations are

$$\begin{cases} a = 51.75 \\ \frac{a}{1+\frac{b}{e}} = 47.29 \end{cases} \text{With the solution } \begin{cases} \mathrm{a} = 51.75 \\ \mathrm{b} \approx 0.2563 \end{cases}$$

Only the values of R' and k remain to be found.

According to ABC News, Hillary Clinton who represents your opponent is expected to spend \$36.6 million in Florida, so we can let $R' = 36.6$.

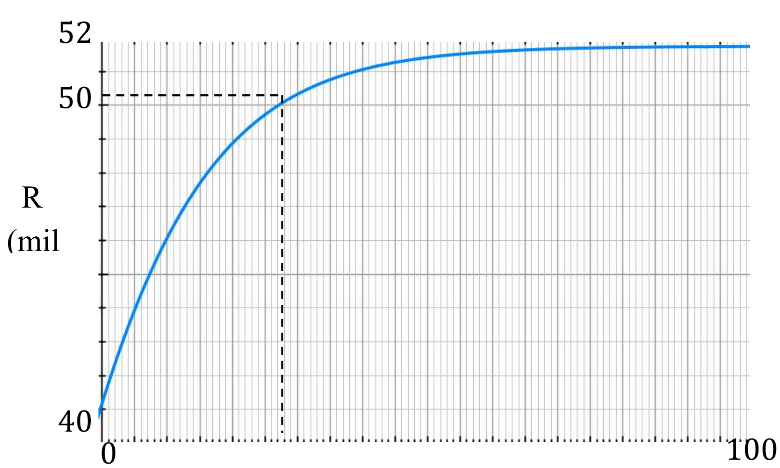
k represents how effectively you use each unit of financial resource compared to your opponent. After conducting research online, we found that based on poll data and the financial records of Hillary Clinton and Donald Trump, Hillary Clinton spends approximately \$13.97 to obtain each vote, while Donald Trump only spends approximately \$5.19 to obtain each vote. Hence we can calculate a value for k

$$k = \frac{13.97}{5.19} \approx 2.69$$

After evaluating the values of *a, b, k* and R' in the original function for the Logistics Growth Model, we get

$$\mathrm{P} = \frac{51.75}{1 + 0.2563e^{-\frac{2.69R}{36.6}}}$$

Now, we can use the graph of this function to obtain the value of R^*



R (millions of $)

When $P = 50$, $R = -\frac{36.6}{2.69}\ln(\frac{\frac{51.75}{50}-1}{0.2563}) \approx 27.1$. This means that if you want to obtain all the electorate votes in Florida, you should input at least $27.1 million. Hence,

$$R^{*}_{\text{Florida}} = 27.1$$

Using the same procedure above, a value of R^* can be obtained for each of the 50 states and and the District of Columbia.

Strengths and Weaknesses:

Strengths: The adapted Logistic Growth model is very versatile and can be directly applied to all states without making significant additional adjustments. The adapted Logistics Growth Model only requires a poll result that shows the percentage of different types of voters and the expected funding of the opposing campaign to be constructed. The versatility of this model will allow campaign managers to quickly assess the relationship under unified criteria for all voting districts.

Weaknesses: One of the major assumptions that this model relies on is that expected campaign investments from the opposing campaign is given and not subject to change. In reality, although it is possible to accurately estimate the financial resources of the opposing campaign, the actual financial resources may change over time as the other campaign may be implementing a similar strategy. The opposition's strategy cannot be considered in this model if it involves major fluctuations.

3.3.5 0-1 Knapsack Problem

Upon finding the financial resource input necessary to reach 50% popularity in each state, we realized that we were essentially dealing with a "knapsack problem". Traditionally, the knapsack problem refers to a situation where a traveler needs to select several items to put in a knapsack. The objective of the knapsack problem is to select the items such that the total weight of the selected items does not exceed the maximum weight capacity and the total value of the items in the knapsack is maximized. The presidential election is similar to this situation. The candidates only have finite financial resources, or in terms of the knapsack problem, finite weight capacity, but must maximize the number of electoral votes they obtain to win the election. Hence, each individual voting district is an item, R^* is the weight of each item, and the number of electoral votes for each state is the value of the item. There are 51 voting districts in total, composed of the 50 states and the District of Columbia, which has 3 electoral votes.

For each voting district *k*, the weight is R^{*}_{k} with the unit millions of dollars and the value is EV_k, the total number of electoral votes for that voting district.

More specifically, the knapsack problem that we are dealing with here is a “0-1 knapsack problem,” meaning that the only options are to choose an item entirely or leave out an item. In terms of the presidential election, the only two options are to input the optimal level of financial resources R^* to reach the 50% popularity mark, or input no financial resources. We believe that limiting the options to these two is reasonable, since it is meaningless to input financial resources into a state unless the chance of you winning reaches 50%. The winner-take-all rule will render the financial resources useless. It is equally meaningless to continue inputting financial resources once you reach a 50% chance of winning, since the excess financial resources inputted do not benefit you in any way.

Realizing that the problem of distributing financial resources can be simplified into a “0-1 knapsack problem,” we can use dynamic programming to solve the “0-1 knapsack problem” and find the optimal distribution strategy. The algorithm for solving the “0-1 knapsack problem” is a recursive in nature. The steps are listed below.

First, we define an objective function:

$$F(51, B)$$

This function gives the maximum value of the electoral votes under the conditions: 1) The total financial resource inputted does not exceed the total campaign budget B, and 2) All 51 election districts will be considered in the decision making process of where to input financial resources. Now, we can consider a general case of the function above

$$F(k, b) \qquad (1 \leq k \leq 51,\ 0 \leq b \leq B)$$

This function also gives a maximum value of the electoral votes, but the conditions for this function are redefined into a general form: 1) The total financial resources inputted does not exceed the total campaign budget b, and 2) Only the first k voting districts will be considered in the decision making process.

For the function $F(k, b)$, there exists two possible cases. The first case is that there are financial resources inputted into k^{th} voting district. Hence,

$$F(k, b) = F(k-1, b-{R_k}^*) + EV_k$$

As shown above, in this case that there financial resources inputted into voting district k, so the maximum amount of electoral votes will be the maximum electoral votes that can be obtained from the first k-1 voting districts plus the total amount of electoral votes for district k. The maximum amount electoral votes that can be obtained from the first k-1 voting districts can be denoted as $F(k-1, b-{R_k}^*)$ because the budget cap for the first k-1 voting districts is reduced to $b - {R_k}^*$ after ${R_k}^*$ millions of dollars is inputted into voting district k.

The second case is that there are no financial resources inputted into voting district k.
The second possible scenario is where the financial resource input for the voting districts has already exceeded the budget limitation, and thus no additional financial resources can be spent on district k. This scenario can be represented as follow.

$$F(k,b) = F(k-1,b)$$

Here, the yield for $F(k,b)$ and $F(k-1,b)$ are equal. This indicates that k will not be considered for financial resources since considering the k states for the function yields the same result as considering k-1 states. This suggests that financing state k will cause the financial input to exceed *b*.

By combining the two scenarios we can redefine the function $F(k,b)$ as follow.

$$F(k,b) = max\{F(k-1,b), F(k-1,b-R_k{}^*) + EV_k\}$$

In order to solve for the value of $F(k,b)$, we can process our recursive algorithm by substituting the value of $F(k-1,b)$ with $max\{F(k-2,b), F(k-2,b-R_{k-1}{}^*) + EV_{k-1}\}$. We can continuous this process by substituting the first component of the function with another recursive function of a smaller degree. Each time a substitution appears, there is a decision made for a new state. As the recursion continues through k-1, k-2, k-3, and so on, the max function will essentially make a decision of whether a candidate should in invest in the voting district or not. Hence, k, k-1, k-2, k-3, and so on, can essentially be interpreted as states. As this recursion process is continued, we will eventually arrive at our two base cases shown below.

$$Base\ Case: \begin{cases} F(1,b) = \begin{cases} EV_1, R_1{}^* \le b \\ 0, \ R_1{}^* > b \end{cases} \\ F(k,0) = 0 \end{cases}$$

In the first base case, all states have been accounted for except for voting district 1 and there remain financial resources to be considered. If the remaining financial resources are enough to support voting district number, in this case $R_1{}^* \le b$, then the candidate will earn additional electoral votes from voting district 1, denoted as EV_1. If the remaining resources are not adequate to support voting district 1, when $R_1{}^* > b$, then the candidate will not receive a any electoral votes from voting district 1. Similarly, if all financial resources are depleted before taking into account the situation of voting district 1, then the candidate will definitely not be able to receive any electoral votes from voting district one as seen in the second base case, where the remaining budget is 0 and the function yield is also 0.

By changing the value of k to 51 and the value of b to B, the 0-1 knapsack model can also be applied to the actual US presidential election.

Sample Calculation:

In this section, a sample calculation will be used to demonstrate how the algorithm listed above can be applied to a specific situation.

Suppose the following data is obtaining using the Logistics Growth Model:

Index	Voting District	R*	EV
1	Alabama	14.4	9
2	Alaska	3.4	3
3	Arizona	6.7	10
4	Arkansas	8.7	6
5	California	28.0	55
6	Colorado	12.5	9
7	Connecticut	3.7	7
8	Delaware	1.6	3
9	D.C.	5.6	3
10	Florida	27.1	27
11	Georgia	19.1	15
12	Hawaii	2.8	4
13	Idaho	3.7	4
14	Illinois	26.0	21
15	Indiana	8.8	11
16	Iowa	8.8	7
17	Kansas	8.9	6
18	Kentucky	5.0	8
19	Louisiana	6.8	9
20	Maine	6.2	4
21	Maryland	1.9	10
22	Massachusetts	3.9	12
23	Michigan	13.6	17
24	Minnesota	19.5	10
25	Mississippi	10.4	6

Index	Voting District	R*	EV
26	Missouri	20.8	11
27	Montana	1.3	3
28	Nebraska	3.6	5
29	Nevada	8.4	5
30	New Hampshire	1.1	4
31	New Jersey	21.9	15
32	New Mexico	3.0	5
33	New York	13.7	31
34	North Carolina	13.3	15
35	North Dakota	1.8	3
36	Ohio	6.1	20
37	Oklahoma	6.3	7
38	Oregon	13.7	7
39	Pennsylvania	4.1	21
40	Rhode Island	1.8	4
41	South Carolina	6.5	8
42	South Dakota	4.4	3
43	Tennessee	0.8	11
44	Texas	0.3	34
45	Utah	0.5	5
46	Vermont	2.3	3
47	Virginia	19.2	13
48	Washington	15.6	11
49	West Virginia	9.7	5
50	Wisconsin	6.3	10
51	Wyoming	5.4	3

Suppose we also know that the budget $B = \$105$ million.

Then the objective function is $F(51,105)$. After implementing the algorithm for the knapsack model in Matlab, and solving for this specific example, the following results are obtained.

$$F(51,105) = 274$$

The total amount of financial resources spent to obtain this number of electoral votes is $\$104.8$ million, $\$0.2$ million less than the budget.

The method for distributing the financial resources is given in the table below. A "1" means that the optimal amount of financial resources should be inputted into the voting district; a "0" means that no financial resources should be inputted into the voting district

Alabama	0	**Kentucky**	1	**North Dakota**	1
Alaska	0	**Louisiana**	1	**Ohio**	1
Arizona	1	**Maine**	0	**Oklahoma**	0

Arkansas	0	**Maryland**	1	**Oregon**	0
California	1	**Massachusetts**	1	**Pennsylvania**	1
Colorado	0	**Michigan**	0	**Rhode Island**	1
Connecticut	1	**Minnesota**	0	**South Carolina**	0
Delaware	1	**Mississippi**	0	**South Dakota**	0
D.C.	0	**Missouri**	0	**Tennessee**	1
Florida	0	**Montana**	1	**Texas**	1
Georgia	0	**Nebraska**	1	**Utah**	1
Hawaii	1	**Nevada**	0	**Vermont**	0
Idaho	0	**New Hampshire**	1	**Virginia**	0
Illinois	0	**New Jersey**	0	**Washington**	0
Indiana	0	**New Mexico**	1	**West Virginia**	0
Iowa	0	**New York**	1	**Wisconsin**	1
Kansas	0	**North Carolina**	0	**Wyoming**	0

In this example, if you distribute your financial resources in the most optimal manner, you can obtain 274 electoral votes in total giving you a slight lead over your opponent who obtains 265 electoral votes in total, and you can successfully become the President of United States of America.

3.3.6 Sensitivity Analysis

A sensitivity analysis can be done to see how sensitive the maximum amount of electoral votes is in response to a change in budget. Let the budget in the previous example be

$$B_0 = \$105 \text{ million } EV_{max} = 274$$

Change in B_0	New Value of B	New Value of EV_{max}	Change in EV_{max}
-25%	78.75	236	-13.87%
-20%	84	243	-11.31%
-15%	89.25	251	-8.39%
-10%	94.5	260	-5.11%
-5%	99.75	266	-2.92%
0%	105	274	0%
+5%	110.25	280	+2.19%
+10%	115.5	286	+4.38%
+15%	120.75	294	+7.30%
+20%	126	300	+9.49%
+25%	131.25	306	+11.68%

As seen in the table above, the percentage change in EV_{max} is approximately half the percentage change in B_0. The results of the sensitivity analysis appear to be fairly reasonable. The change in EV_{max} is not too sensitive to a change in B_0. For the same percentage of change in B_0, a positive change results in a slightly greater change in EV_{max} than a negative change. This sensitivity analysis has two implications:

1) If the current amount of electoral votes that you expect to obtain is less than 270, the number of electoral votes necessary to guarantee your victory, you may have to increase your budget significantly to be elected President. For example, if the current amount of electoral votes that you expect to obtain is short of 270 by 5%, you may need to increase your budget by 10% or more.

2) If the current amount of electoral votes that you expect to obtain is more than 270, you may be able to spend much less than your budget and still to be elected President. For example, if the current amount of electoral votes that you expect to obtain exceeds 270 by 5%, you may be able to decrease your spending by 10% or more and still be elected President.

Strengths and Weaknesses

Strengths: The biggest strength of using the 0-1-knapsack model is its ability to cater the calculations to all states and hence provide a holistic assessment of which states are worthy for financial resources. The recursive algorithm goes through all 51 voting districts and ensures that the final result is optimized.

Weaknesses: It should be noted that the threshold popularity defined in this model is at 50%, which is only a bare minimum for securing the electoral votes of a state. Hence, while this model is able to provide an accurate assessment for the general funding required, it should be noted that realistically additional funding may be required to fully secure a state.

3.4 Conclusion of Solution to Problem One

Upon introducing several models that will be used to solve the presidential campaign problem, we can summarize the overall procedure as follow.

After acquiring the data from the poll results, we must first convert the poll results into a bi-party system through scaling the data to only include two parties and the undecided voters, which adheres to the requirement for the question.

Based on the data acquired we then calculated the confidence interval and winning probability for each poll in each state which will allow us to forecast the final results in the final election. After acquiring the winning probability and the confidence interval we proceeded to use a linear regression line to predict the final winning probability during the final election as well as the predicted confidence interval during the final election.

After finding the predicted results for the final election as well as the confidence interval, we can apply our Logistics Growth Model to find the relation between campaign funding and popularity. The confidence interval will help determine the parameters for the Logistics Growth

Model when applied to each individual state. Upon determining the relation between financial resource input and popularity, it is possible to find the financial resources needed to reach 50% popularity and secure a voting district.

Knowing the financial resources required to in each voting district as well as the electoral votes for each voting district, we then used a 0-1 Knapsack model to find the optimal financial resource distribution strategy through considering the financial resources needed to reach 50% popularity as the weight and the electoral votes as the value for each voting district.

We conducted sample calculations with a randomly simulated set of data and found that based on our mechanisms, we can obtain 274 electoral votes, which gives a slight advantage over the opposing campaign. We then conducted a sensitivity analysis by altering the value of the budget and examined the corresponding effect on the value of EV. We found that if the electoral votes are below 270 by 5%, it is necessary to increase the budget by 10% or more to ensure victory. The sensitivity analysis indicates that the model can be applied without significant changes under different circumstances.

4. Problem Two Solution

4.1 Introduction of Model

For the allocation of the ambulances, we use the **Greedy Allocation Algorithm**. This Algorithm minimizes the average response time for a certain allocation of ambulances. For this model, we assume that the amount of traffic along all the streets in Manhattan is uniform. This is because we are concerned more with the distribution of ambulances in this model than the actual routes of ambulances in the case of an emergency. Traffic levels along the streets of Manhattan vary from day to day, so it would be more practical to assume a constant traffic level than to keep it variable when calculating an optimal distribution of ambulances.

For the dispatching of ambulances, we use the **Genetic Algorithm.**

For the routing of ambulances, we will use the **A* Algorithm.** This algorithm selects the best path by finding a path that incurs the least "cost", or in our case, takes the least time.

4.2 General Assumptions and Justifications

1. We assume that all ambulances will drive with the intent of reaching the location of the accident in the shortest time.

   Justification: Our overarching goal in this paper is to minimize the response time of the ambulances. Therefore, the ambulances should also attempt to achieve this goal.

2. Assume there are enough ambulances for all crash sites in a day

   Justification: In Manhattan, the number of ambulances far exceed to number of accidents in a day, so it would be plausible to assume that we have enough ambulances for any emergency.

3. We assume that all ambulances are the same in terms of travelling capacity.

   Justification: This paper is more concerned with an ambulance system than individual ambulances, so there is no point is excessively complicating calculations by factoring in another variable of the capacity of the ambulances. In real life Manhattan, most ambulances are the same model anyway, so not much accuracy is lost through this assumption.

4.3 Solution

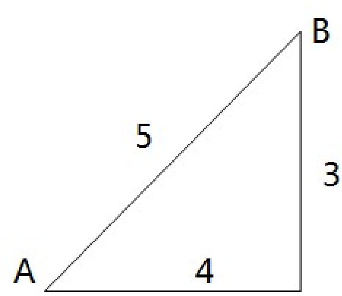
4.3.1 Greedy Algorithm for Allocation of Ambulances in Hospitals

4.3.1(a) Defining Manhattan Distance

Before getting started on the algorithm itself, there is one fundamental variable to our Greedy Allocation Algorithm that we must define clearly: distance. Distance will be one of the core components in our solution.

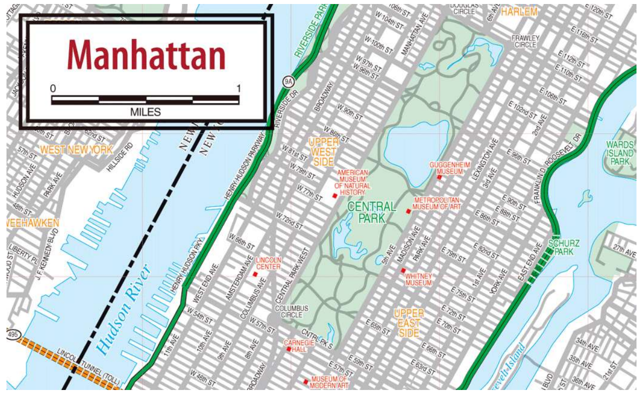
In this paper, we define distance d to be the Manhattan distance: the sum of the perpendicular distances between two grid points A and B. A grid point is simply an intersection point between two perpendicular streets in Manhattan.

The borough of Manhattan possesses a grid system; intersecting streets intersect perpendicularly, forming a grid of streets. Thus, the only way is to get from grid point A to B in Manhattan is to travel in perpendicular paths; you cannot go diagonally across a block. It is only possible to travel along the sides of the blocks. This is why the Manhattan Distance is defined as the sum of the perpendicular distances between grid point A and grid point B.



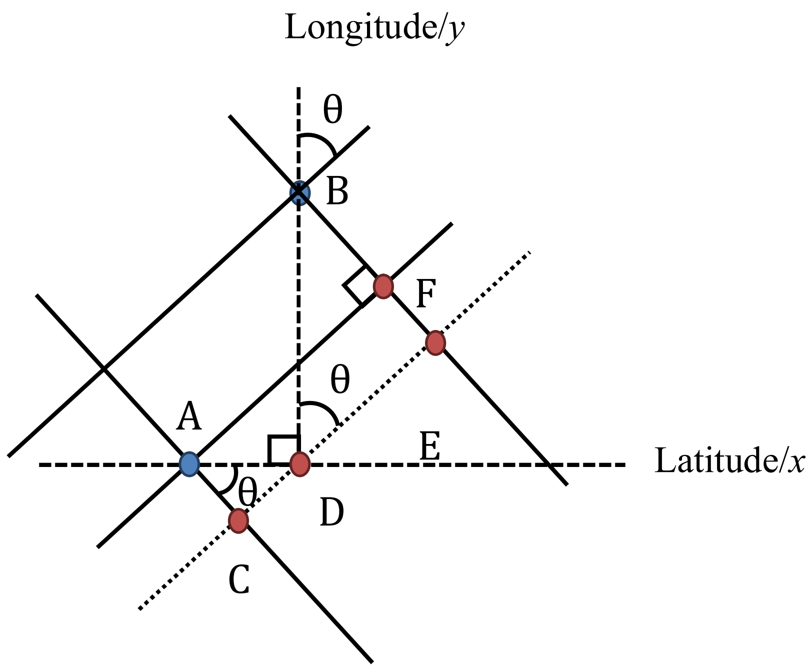
The Manhattan distance is more clearly explained with this diagram. On a Cartesian plane, the distance between A and B would simply be the hypotenuse of this triangle, 5, but the Manhattan distance would be the sum of the x and y distances, which is 7.

Therefore, when using the variable d, we are referring specifically to the Manhattan distance.



However, there is one problem with the Manhattan grid: its streets aren't in the latitude and longitude directions. If the streets *were* in the latitude and longitude directions, we could simply use the difference in the longitudinal and latitudinal coordinates to find the Manhattan distance between two grid points. However, the streets were at an angle to the longitudinal and latitudinal axes, so we must factor this angle into our calculation of the Manhattan distance between two grid points.

To factor in the angle between the streets and the axes, we first draw a simple diagram of the Manhattan grid:



This diagram presents one block in the Manhattan grid, with two pairs of parallel streets perpendicular to each other. We want to get from grid point A to B on the diagram. The Manhattan distance d in this case would be the sum of the parallel distances: distance AF and distance FB. However, there is an angle θ between the streets of the Manhattan grid and the longitude/latitude grid as shown on the diagram. Since we only possess the longitude and latitude coordinates for grid points A and B, we must find a way to express distance AF and FB in terms of these coordinates.

To do so, we use three imaginary lines: one in the direction of true north, denoted N on the diagram (this means in the y direction), one in the x direction, and one parallel to a side of the block (line CE).

We use the x axis to represent the latitude direction, and the y axis to represent the longitude direction. The x and y lines allow us to make use of our longitudinal and latitudinal coordinates, or our xy coordinates. Using our coordinates, we can easily find distance AD and distance BD. We shall denote AD as Δx and BD as Δy, because they represent the changes in the x and y coordinates.

Using our third imaginary line CE, we can see that $\Delta x sin\theta = \text{CD}$, and that $\Delta y cos\theta = \text{DE}$. Thus:

$$\text{CE} = \text{AF} = |\Delta x sin\theta + \Delta y cos\theta|$$

and

$$\text{AF} = |\Delta x sin\theta + \Delta y cos\theta|$$

We can also see that $\Delta x cos\theta = \text{AC}$, and $\Delta y sin\theta = \text{BE}$. Thus:

$$\text{BF} = \text{BE} - \text{FE} = \text{BE} - \text{AC} = |\Delta x cos\theta - \Delta y sin\theta|$$

and

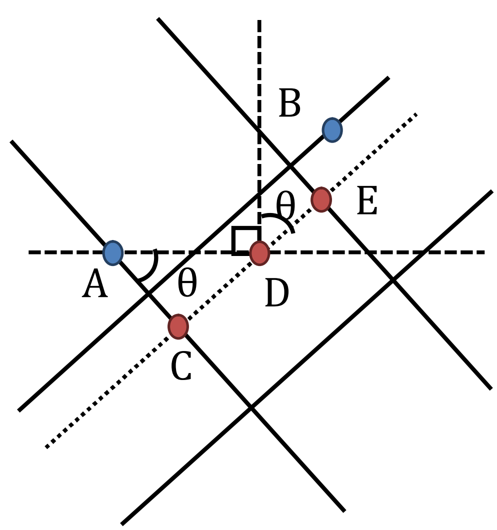
$$\text{BF} = |\Delta x cos\theta - \Delta y sin\theta|$$

Finally, Manhattan distance d equals to the sum of AF and BF:

$$d = |\Delta x sin\theta + \Delta y cos\theta| + |\Delta x cos\theta - \Delta y sin\theta|$$

This holds true for any two grid points A and B on the Manhattan grid.

We will now consider the case in which A and B are not grid points; A and B can also be on the streets and not at the intersection of streets. This will be shown with the following diagram:



As shown, A and B are points not located at the intersection of two streets. This is a very special case, because it is under this case that our formula for Manhattan distance *d* given above does not apply. If we try to use the formula here:

$$d = |\Delta x sin\theta + \Delta y cos\theta| + |\Delta x cos\theta - \Delta y sin\theta|$$

Δx is distance AD, and Δy is distance DB. Therefore:

$$\mathrm{CD} + \mathrm{DE} = |\Delta x sin\theta + \Delta y cos\theta|$$

Which is correct; however:

$$|\Delta x cos\theta - \Delta y sin\theta| = \mathrm{AC} - \mathrm{BE}$$

Which is a problem, because our Manhattan distance *d* in this case should be:

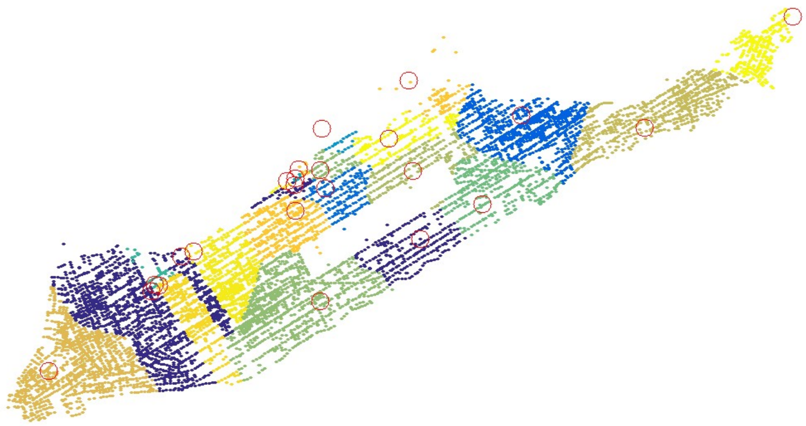
$$d = \mathrm{CD} + \mathrm{DE} + \mathrm{AC} + \mathrm{BE}$$

Therefore, there will be a small error in the Manhattan distance when A and B are not grid points. However, this error will not be very significant unless both points are right in the middle of their respective streets. We can still use our formula for Manhattan distance with adequate accuracy; the resulting distance will still be a good estimate of the actual distance. If needed, we can improve upon our solution with more complex programming to solve this issue. Thus, we will still use our formula for Manhattan distance *d* in our solution.

4.3.1(b) Data Processing

As mentioned in the introduction, some modifications must be made to the given data before we can actually start using our algorithm.

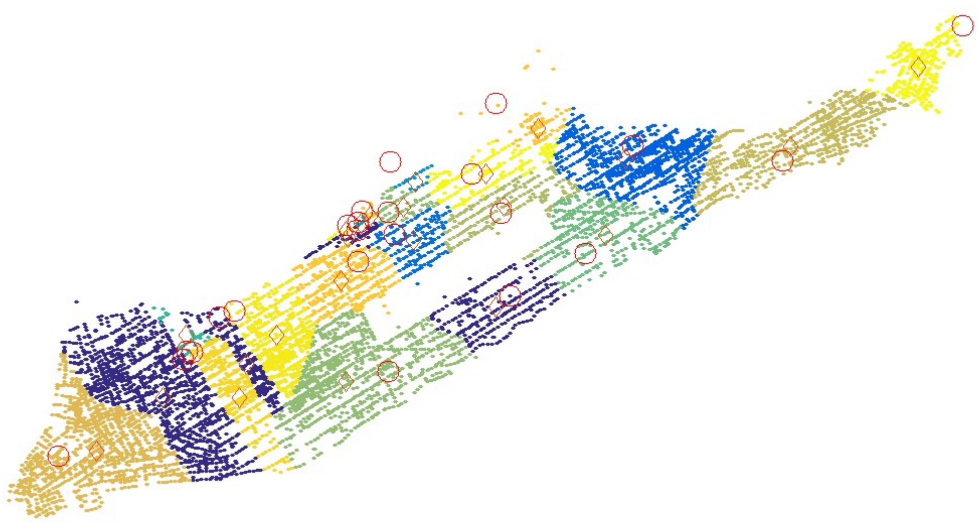
First, we divide Manhattan into several districts. We do this because using all 17,000 crash sites to calculate the average time is simply too much workload for the algorithm to be practical.



This is a map of Manhattan, separated into districts by a program written by us. Each color represents a district. The red circles represent hospitals.

We separate Manhattan into districts based on the location of the hospitals. All the points in the same district are closer, in Manhattan distance, to a single hospital than to any other hospital. A district is composed of one hospital and all the crash sites that are closest to it.

We then average all the crash sites in a single district into one point, using the center of mass formula:



Using a program, we find a point in each district that is the average of all the positions of the crash sites in that district. On our map, these points are represented by diamonds.

Thus, we reduce the total number of crash sites to a much more manageable number.

Justification of separating Manhattan into districts:

If we do not separate Manhattan into districts, we would have to consider all 17,000 crash sites each and every iteration of the Greedy Allocation Algorithm. The workload is simply too large for our algorithm to be practical, as it would require a long period of time to complete its calculations.

By separating Manhattan into districts using our method, we still are able to preserve a great degree of accuracy.

4.3.1(c) The Greedy Allocation Algorithm:

We will now provide a detailed step-by-step walkthrough of our Greedy Allocation Algorithm to demonstrate how we arrive at the best allocation of ambulances in Manhattan.

We begin by defining some key terms and variables:

A, a certain allocation of ambulances

a, one ambulance

T, defined as the average response time of a certain allocation of ambulances A. We define response time as the time taken for an ambulance from any hospital to reach a crash accident location.

U, a function that is defined as the total benefit received from using a certain allocation A.

δ_U, a function defined as the marginal benefit received from adding an ambulance a to a certain allocation A.

k, the total number of ambulances in Manhattan.

Step 1:

The first step in our Greedy Allocation Algorithm is to assume an allocation A with no ambulances. We then consider adding one ambulance a to this allocation in this fashion:

-Consider adding a to Hospital 1, and record the T of this particular allocation A of ambulances.

-Repeat step one for all hospitals in Manhattan.

-Pick out the allocation with the smallest T, and add ambulance a to the corresponding hospital.

We want the allocation with the smallest T because our goal is to minimize the average response time to accidents in Manhattan; this is the condition for an ideal allocation of ambulances.

Now we have an allocation A with one ambulance in one hospital.

Step 2:

Now that we have an allocation A with one ambulance in a particular hospital, we consider adding another ambulance a to the allocation A.

We go through the same process:

-Consider adding *a* to Hospital 1, and record the *T* of this particular allocation *A* of ambulances.

-Repeat step one for all hospitals in Manhattan.

-Pick out the allocation with the smallest *T*, and add ambulance *a* to the corresponding hospital.

Step 3:

-Repeat this entire process *k* times, until there are no more ambulances to add.

-after the *k*th iteration, we will have an allocation *A* with the response time *T* minimized

-This means we will have reached our goal of finding an allocation of ambulances that minimizes the average response time to accidents in Manhattan

Utility in The Greedy Allocation Algorithm:

Previously, we defined the function *U*, the total benefit received from using a certain allocation A, and the function δ_U, the marginal benefit received from adding an ambulance *a* to a certain allocation A.

We also said that the Greedy Allocation Algorithm uses a certain formula earlier in the introduction:

$$\delta_U(a|A) = U(A \cup a) - U(A)$$

Which means the marginal benefit received from adding an ambulance *a* to a certain allocation *A* is equal to the total benefit of the new allocation $A \cup a$, which is the current allocation along with the ambulance *a*, minus the total benefit of the current allocation *A*.

What the Greedy Allocation Algorithm does is maximize this marginal benefit, or maximize the difference between the two allocations.

Because our goal is to minimize the average response time, we measure our benefit in terms of time. Since our average response time will stay the same or decrease when we add an ambulance to the current allocation of ambulances, we can rewrite the formula as:

$$\delta_U(a|A) = |T(A \cup a) - T(A)|$$

Which means that the marginal benefit is equal to the absolute value of the difference between the average response time with the new allocation $A \cup a$ and the current allocation *A*.

This formula is more suitable for the context of this paper, because it expresses benefit in terms of time.

As mentioned n the previous section, our goal is to minimize *T*. This formula also explains why we must minimize $T(A \cup a)$ when determining where to put the next ambulance. The smaller $T(A \cup a)$ is, the bigger the difference will be, and so the marginal benefit will be maximized. Thus, in our algorithm, we maximize the marginal benefit of adding an ambulance to the current allocation by determining the minimum average response time for $T(A \cup a)$.

Justification of the Greedy Allocation Algorithm:

As we add ambulances, *T* will either stay the same or decrease, because it is not possible that adding another ambulance will increase average response time to accidents in Manhattan. This guarantees that each time we add an ambulance, the effectiveness of the allocation of ambulances will never get worse. The effectiveness of the allocation can only get better.

As mentioned in the introduction, the Greedy Allocation Algorithm is **non-submodular.** We justify that even though the Greedy Allocation Algorithm will ignore possible allocations due to its non-submodularity, it will nevertheless guarantee that our current allocation will never get worse when we add an additional ambulance. As mentioned above, average response time *T* will either stay the same or decrease, because adding an additional ambulance will never increase the average response time of an allocation.

In addition, the algorithm will look for **the least average response time *T* in each iteration**, meaning the degree by which the current allocation will be improved is always maximized as much as possible. So the Greedy Allocation Algorithm is nevertheless an effective algorithm in maximizing improvement of an allocation of ambulances.

Furthermore, attempting to include all the possible allocations simply isn't feasible. If we do attempt to include all possible allocations, we would need 23^k iterations to consider every possible allocation (23 is the number of hospitals in Manhattan, and *k* is the total number of ambulances).

This is because:

-After we finish allocating the first ambulance, we now attempt to allocate the second ambulance.

-There are 23 hospitals you can allocate this particular ambulance to, but you also have to consider the 23 possible hospitals that the first ambulance could have been allocated to.

-This means: We iterate 23 times through each hospital for the second ambulance, under the condition that the first ambulance is in Hospital 1.

-We iterate 23 times again through each hospital for the second ambulance, under the condition that the first ambulance is in Hospital 2

-This process is repeated until we finish considering the case that the first ambulance is in Hospital 23.

-Therefore, the total number of iterations is $23 \times 23 = 23^2$.

-We can imagine that by the time we finish allocating all the ambulances, we will have iterated 23^k times.

This workload is enormous and cannot be completed within a short period of time by a computer. If the workload cannot be computed in a short period of time, then it isn't feasible or practical to use such a system. In an actual case of emergency, computers must be able to produce useful data within a short period of time; thus, attempting to explore every possible allocation is not a suitable course of action in this situation.

We can then look at the workload of our Greedy Allocation Algorithm:

For the Greedy Allocation Algorithm that builds upon itself, the total number of iterations required is $23k$.

This is because:

-After we finish allocating the first ambulance, we now attempt to allocate the second ambulance.

-The position of the first ambulance is fixed (or we can say the current allocation of ambulances is fixed), meaning we only have to consider the 23 hospitals that we can allocate the second ambulance to.

-This goes for each ambulance; for each ambulance, we only need to consider the 23 possible different choices for that ambulance, because the current allocation is also fixed.

-Therefore, the final number of iterations will be $23k$, since we consider 23 hospitals for each ambulance.

This workload is much smaller and much more manageable by a computer than the previous workload of 23^k. Furthermore, the Greedy Allocation Algorithm still provides incredibly useful data, because it maximizes the improvement in the allocation of ambulances. Thus, the Greedy Allocation Algorithm is a highly suitable choice for this particular situation, as it requires a workload that can be done in a short period of time, while at the same time providing highly useful information.

4.3.2 Genetic Algorithm for Dispatching Ambulances

Now that we have calculated the optimal allocation strategy of ambulances, we need to consider how to dispatch the ambulances in the event of multiple car accidents. In other words, ambulance at which hospital should respond to a specific accident, assuming there are enough ambulances, to minimize the average response time of ambulances. If there is only one car accident reported in a certain time frame, the best dispatch strategy is obviously to send the ambulance from the hospital that is closest to the crash site, which is the one in the same district. However, if multiple accidents are reported at approximately the same time, the dispatch strategy is no longer so straightforward, because there may be concentrated accidents in a single district and ambulances need to be dispatched from other districts.

Therefore, we need to consider all possible dispatch strategies before obtaining the optimal dispatch strategy with the lowest response time. Unfortunately, the number of dispatch strategies grows exponentially with the number of crash sites. Considering all strategies is thus computationally NP hard.

As a result, we decided to use Genetic Algorithm to arrive at a relatively optimal dispatch strategy.

Now, we will introduce how Genetic Algorithm works in our case.

Step One:

Generate 20 random dispatch strategies, assigning a random ambulance to each crash site. Let’s assume there are j crash sites and k ambulances labeled 1~k available. Generate a random permutation of 1~k, taking the first j elements as one dispatch strategy.

$$\{10,34,2,4,7,1\ldots\}$$
$$\{5,1,25,3,1,2\ldots\}$$
……

Step Two:

Calculate the average response time for each strategy and assign a score for each strategy. The lower the response time, the higher the score. Then we choose one out of the twenty strategies with replacement for 20 times. The probability of selection is proportional to the score. The purpose of selection is to ensure strategies get better with each iteration.

Step Three:

Randomly pair up the twenty strategies. For each pair, there is a probability of crossover, after which part of the strategy is swapped with each other. For example,

$\{10,34,2,4,7,1\ldots\} \rightarrow \{10,34,25,3,1,2\ldots\}$
$\{5,1,25,3,1,2\ldots\} \rightarrow \{5,1,2,4,7,1\ldots\}$

Crossover is a process of increasing the variety of strategies to avoid being stuck in local optimum.

Step Four:

There is a probability of mutation for each strategy, in which an element of the strategy is changed. For example,

$\{10,34,25,3,1,2\ldots\} \rightarrow \{10,34,25,4,1,2\ldots\}$

Mutation is also aiming to avoid the algorithm being stuck in a local optimum.

Step two to step four is repeated until the scores of the strategies no longer improve. With each iteration, the score of the strategies generally get a bit higher and the average response time lower. Finally, the strategy with the highest score is the optimal dispatch strategy we need.

4.3.3 A* Algorithm for Routing Ambulances

After we have completed allocating the ambulances to minimize the average response time, we now consider the best possible routes for ambulances when an actual emergency happens.

To find the best possible route, we use a pathfinding algorithm called the A* Algorithm, or the "Least Cost Algorithm". In general terms, the A* Algorithm selects the path that minimizes:

$$f(n) = g(n) + h(n)$$

Where *n* is the next node on the path, *g* is the cost of moving from the current node to node *n*, and *h* is an estimation of cost of the cheapest path from node *n* to the goal node.

This abstract algorithm can be implemented for the ambulances in Manhattan

Each grid point will be a node, which make sense because the grid structure of Manhattan streets forces ambulances to move from grid point to grid point.

The *g* function, or the cost of moving to a node will be determined by how much crash sites will be passed by in moving to that node. Crash sites will increase the "cost", because crash sites are areas where accidents are more likely. Accidents will result in congestion, which will slow the ambulance's response time down. This is a "cost", because our goal is to minimize response time. Hence, we can define "cost" as an "increase in response time". Additionally, the level of congestion in the street will also contribute to the cost of moving to a node. Finally, the distance from the current node to node *n* will also determine the cost, because a longer distance means a longer time spent moving.

The *h* function will simply be the Manhattan distance from node *n* to the location of accident. It's viable because distance corresponds to time, but also because this estimate is admissible, meaning the cost will never be overestimated. Because we only factor distance and not congestion nor number of nearby crash sites, we are in fact estimating the least possible cost from node *n* to the location of accident.

The Algorithm in Detail:

We will now go through a step by step process of the A* Algorithm in the context of an ambulance:

-When the ambulance receives notice of a nearby accident, its computer will receive the coordinates of the accident

-The accident's location will be replaced by the nearest grid point, because our nodes are defined as grid points. This will simplify the process of path-finding while still keeping the information useful, because a car accident can never happen too far from a grid point.

-We now have two grid points: its current position, and the accident location.

-Now we consider all the possible grid points we can move to from our position, and select the one that will result in the lowest *f* when moving to it. We record which node we have chosen.

-From the second grid point, we repeat the process of finding a cheapest node to move to.

-We repeat until we reach the goal grid point, which is the accident location.

-We backtrack through all the recorded nodes, and output a route to the ambulance.

Justification of the A* Algorithm:

The A* Algorithm is extremely well suited to our particular situation because it is a pathfinding algorithm that **minimizes cost**. Our goal is, similarly, to reduce the average response time, which is also a form of minimizing cost.

Additionally, it is quite easy to define nodes in our situation due to the structure of Manhattan. Cost is quite easy to define as well; anything that serves to increase the time taken to reach the accident location is a cost, because our goal is to reduce time. So the g function can be smoothly implemented into our situation with factors such as number of likely crash sites near the street, congestion within the streets, and distance. The h function can be implemented just as easily using Manhattan distance.

Overall, the A* Algorithm serves as a suitable solution due to its affinities with our situation and its ease of implementation to our case.

4.3.4 Sensitivity Analysis

Our model for problem two is roughly divided into allocation, dispatch and routing of ambulances. Our algorithm for allocation of ambulances is a general strategy applicable to any situation and it is based on the collection of historical data from the past few years, so it is extremely insensitive to noise. A single car accident happening in a specific site will not affect the allocation strategy at all. For out dispatch algorithm, it is extremely sensitive to any change in location and number of crash sites, reflecting the dynamic nature of ambulance dispatch. The dispatch strategy for each situation should certainly be different. Moreover, our routing algorithm is also quite sensitive to the change in location and number of crash sites, also representing the dynamic nature of routing. The route of ambulances can change according to real time conditions.

4.3.5 Strengths and Weaknesses

Strengths:

Our solution is very suitable to the situation at hand. Our A* Algorithm is a perfect fit for helping ambulances route under the condition that we want to minimize response time, because the A* Algorithm is perfect for minimizing costs, or in this case, time. Our Greedy Allocation model is an adequate model for distributing the ambulances because it guarantees improvement for any current allocation of ambulances. The allocation can also be implemented smoothly in this situation by defining marginal benefit in terms of response time. In addition, our separation of Manhattan into districts makes the computational workload practical. This means our solution can be implemented as a functional solution.

Weaknesses:

Our weaknesses mainly stem from that fact that we reduce certain numbers and avoid certain algorithms in order to make calculation possible. By separating Manhattan into districts and using a center of mass calculation method to average the positions of the crashes to make our calculation possible, we lose some degree of accuracy. Our use of the Greedy Allocation Algorithm is also an estimate, because the algorithm itself is non-submodular. We ignore some possible allocations that may have a smaller average response time in exchange for the calculation of the solution to be possible. So while the allocation guarantees an improved allocation, it may not produce the absolute best allocation.

4.4 Conclusion of Solution to Problem Two

In this problem, we attempted to improve the ambulance system through three aspects: allocation, dispatching, and routing.

Allocation refers to the allocation, or distribution, of the ambulances in Manhattan among the hospitals of Manhattan. Since certain areas have had more crash sites in the past than others, the allocation of ambulances should also be unevenly distributed in order to minimize the average response time of all ambulances.

To distribute the ambulances in the best way possible, we use **the Greedy Allocation Algorithm.**

Dispatching refers to how ambulances are sent in the case of an emergency; in other words, which hospitals should respond to the accident should an accident occur, in order to minimize the average response time of the ambulances. To form an optimal dispatch strategy, we use the **Genetic Algorithm**.

Routing refers to finding the path from the hospital to the location of the car accident for an ambulance. Our routing method attempts to minimize the time taken for the ambulance to move from one point to another, which also means minimizing the response time for ambulances. To find the optimal path for which the time taken is minimized, we use the **A* pathfinding algorithm.**

5. Citations

1. Duke University. (n.d.). *Logistic Growth Model*. Retrieved from Duke University: https://services.math.duke.edu/education/ccp/materials/diffeq/logistic/logi1.html
2. *List of State Electoral Votes*. (n.d.). Retrieved from 1keydata: http://state.1keydata.com/state-electoral-votes.php
3. Muhamma, R. (n.d.). *Dynamic-Programming Solution*. Retrieved from Kent State University: http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Dynamic/knapsackdyn.htm
4. Politico. (n.d.). *2016 Election Campaign*. Retrieved from Politico: http://www.politico.com/news/2016-elections
5. Stanford University. (n.d.). *Introduction to A**. Retrieved from Stanford University: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
6. Stephen Wolfram. (n.d.). *Genetic Algorithm*. Retrieved from Wolfram Mathworld: http://mathworld.wolfram.com/GeneticAlgorithm.html
7. Stephen Wolfram. (n.d.). *Logistic Equation*. Retrieved from Wolfram Mathworld: mathworld.wolfram.com/LogisticEquation.html