What is the relationship between the distribution of parking spaces and the local traffic in urban areas of Hong Kong?

Methodology Notes

Abraham Cheung 11B

As I will be investigating how the distribution of parking spaces affect the local traffic, I will be breaking this RQ into two parts:

1. Measurement of traffic
    a. Primary Methods (obtain flow and occupancy)
    b. Secondary Methods (verify flow with ATC and speed with Google Distance Matrix API)
2. Measurement of parking space availability
    a. Buffer zones
    b. Heat maps
    c. Node graphs

1.1 Primary Methods

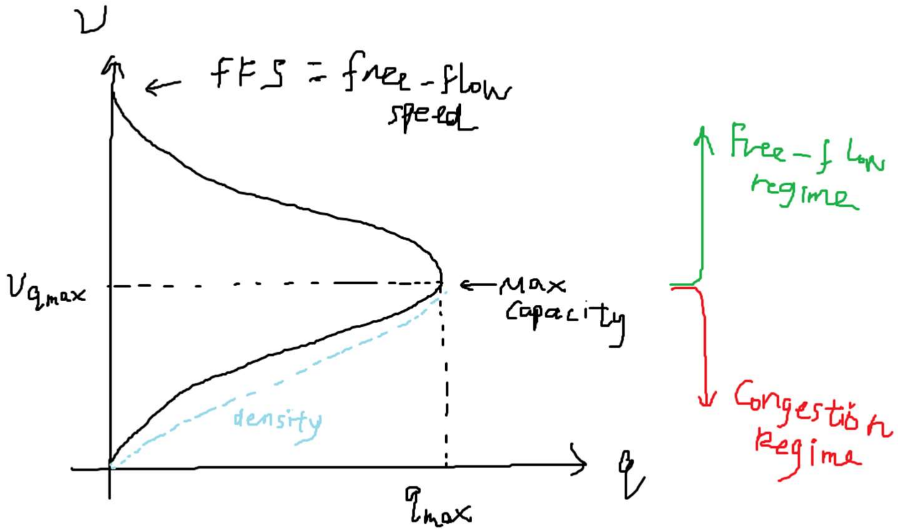
1.1.1 Introduction

Here are some basic variables will be used in the study:

Name	Symbol	Unit
Flow	q	veh/hr
Speed	v	km/hr
Occupancy	o	[dimensionless]
Density	k	veh/km

In short, *flow* is comparable to *current (I)* in a circuit, which is the number of vehicles passing through a single point per unit time, and the *speed* is the distance travelled by a vehicle per unit time.

Below is a graph that shows how they are related:



This graph illustrates that an area with low traffic *flow* does not necessarily mean that the area is congested, rather, it is dependent on the vehicle *speed*.

1. The number of vehicles on the road is low (for example, during off-peak hours / 3a.m.), causing the flow (number of vehicles/time) to be low
2. The area is actually congested – where vehicles are slowed down to the point which increases the time needed to travel, hence causes the flow to be low

1.1.2 Method of surveying

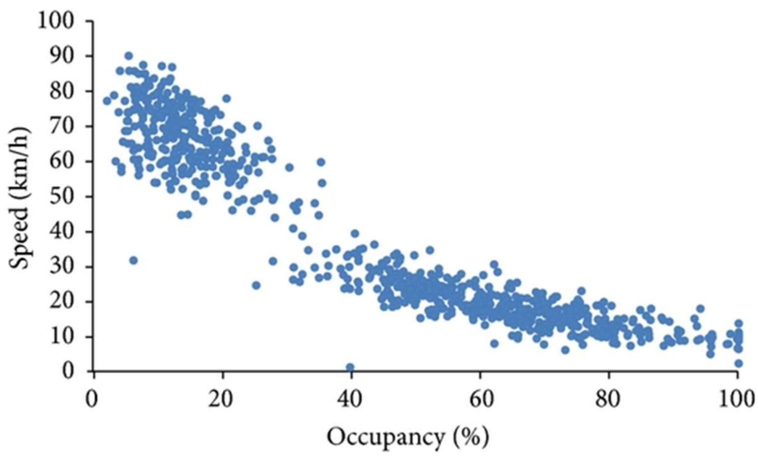
Therefore, to accurately describe how an area is congested, one must know the *flow* and the *speed* of vehicles. Flow can be measured by counting the number of vehicles passing through a single point, but the *speed* cannot be measured directly without professional equipment (LiDAR radar guns / inductive-loop traffic sensors).

To solve this, the proxy indicator for *speed* will be *occupancy*, which is defined by:

$$o = \frac{\text{duration of vehicle occupying the point}}{\text{total time}}$$

From this, if the vehicle speed is high, the vehicle will pass through the point within a short period of time/instantly; on the other hand, when the vehicle speed is slow/comes to a standstill, then the *occupancy* will be high.

An occupancy-speed graph confirms this relationship:



Source: *doi:10.1155/2013/254841*

However, measuring the *flow* may not accurately affect the traffic condition because vehicles with long lengths (such as buses and trucks) occupies a larger proportion of the road, and therefore decreases the *flow*. To compensate for this loss, the *flow* will have to be converted into its *PCU*, or passenger-car equivalent unit. Below are the values used in Hong Kong:

	Equivalent Value in pcu's					
	Urban Standard		Rural Standard		Roundabout	Traffic Signal Design
Private Car, Taxi	1.0		1.0		1.0	1.0
Light Goods Vehicle	1.5		1.5		1.5	1.5
Motor Cycle Motor Scooter	0.75		1.0		0.75	0.40
	Terrain					
	Ave	Hilly	Ave	Hilly		
Medium Goods Vehicle	2.0	3.0	2.0	3.0	2.8	1.75
Heavy Goods Vehicle	2.5	3.0	2.5	3.0	2.8	1.75
Bus	2.5	3.0	2.5	3.0	2.8	2.0
Pedal Cycle	0.35		0.5		0.5	0.2
Tram	3.0		-		-	3.5-5.0+
Light Bus	1.5		1.5		1.5	1.5
Special purpose bus	2.0		2.0			
Light Van	1.1		1.1		1.1	1.1
Tractor unit	2.5	3.0	2.5	3.0		

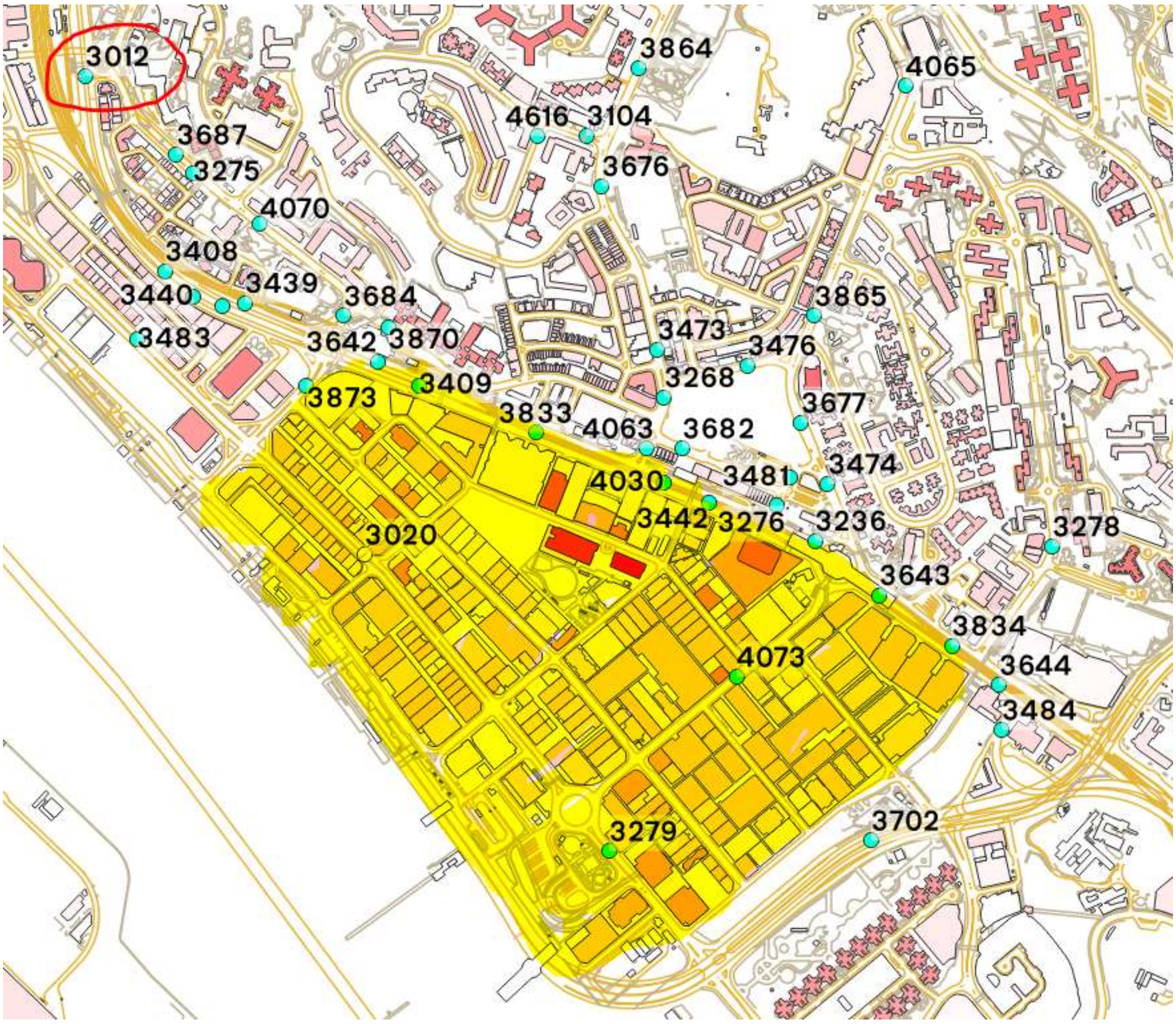
Source: *Transport Planning & Design Manual (TPDM) Volume 2*

The actual count will be multiplied according to the values in the table above, as an example, 1 bus = 3 pcu.

1.2 Secondary Methods

1.2.1 Simple Validation on *flow* data

Regarding the study area, I have chosen an area enclosed within Kwun Tong Road, Wai Yip Street and Wai Fai Street in Kwun Tong (highlighted in yellow). This area has been chosen because it is renowned for its poor traffic performance during peak traffic hours, especially along Kwun Tong Rd, where many arterial roads merge that causes traffic issues. Below is a map of the area:



Source: *iB1000 Maps by LandsD*

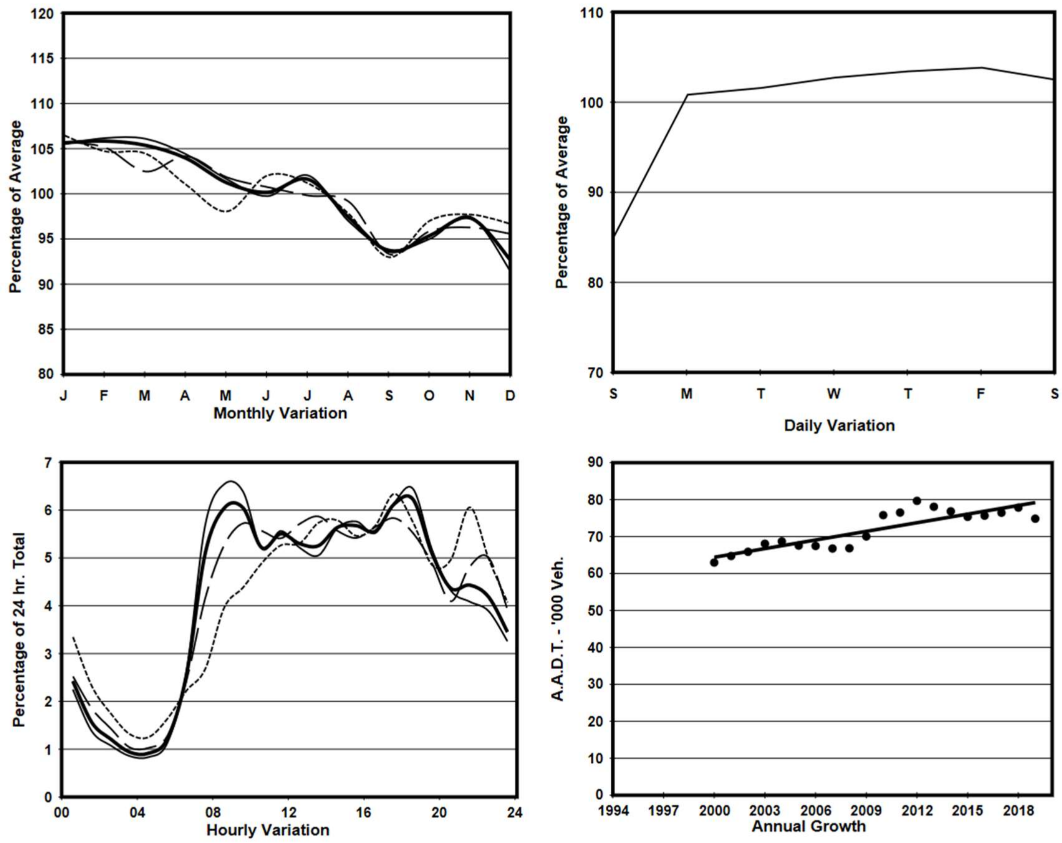
The color on the buildings denote its height (white: 0m, red: 200m), which serves as a rough indicator on the traffic demand of the location.

To verify the primary data, it will be compared against the expected traffic flow, which is a part of a territory-wide annual traffic census (ATC), performed by the Transport Department.

There are two types of survey stations that store different levels of detail:

1. **Core** – has annual, monthly, daily and hourly variation of annual average daily traffic (AADT) (e.g. #3012, circled red)
2. **Basic** - only has the annual variation of AADT (e.g. #4073, blue)

Therefore, to estimate the flow at any arbitrary point, I will have to first find the closest **basic** station (#4073) as well as the closest **core** station (#3012). Below is a graph of the details at #3012:



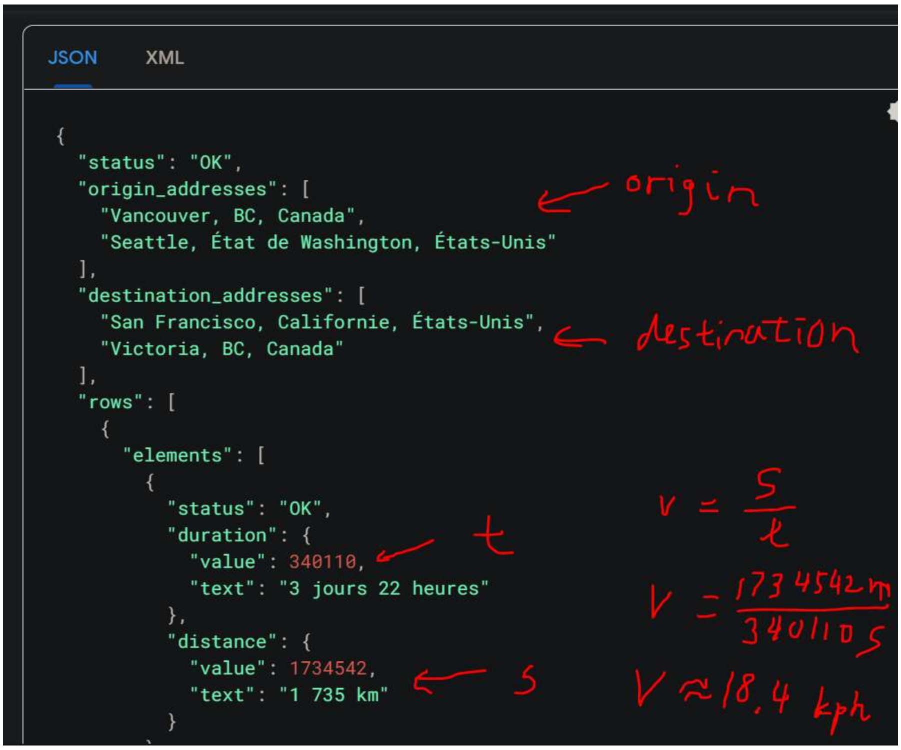
Source: https://www.td.gov.hk/filemanager/en/content_5018/Appendix%20A2/S3012.pdf

The scaling factor based on each type of temporal variation can then be found using the values on the graph above: 102.03081775640% (monthly) * 103.83043670654% (daily) * 5.6878409385681 (hourly) = <u>**6.03%**</u> at that specific given time.

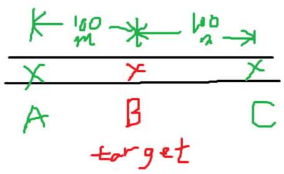
By multiplying this scaling factor with the AADT at #4073 (basic), the hourly flow at that arbitrary point is estimated to be: 16460veh/24hr * 6.03% = **992veh/hr**.

1.2.2 Obtaining Speed with Google Distance Matrix API

As mentioned previously in 1.1.2, the occupancy is used as a proxy indictor for vehicle speed. However, the vehicle speed can be estimated using Google's Distance Matrix API between two points, as shown below:



Source: https://developers.google.com/maps/documentation/distance-matrix/overview



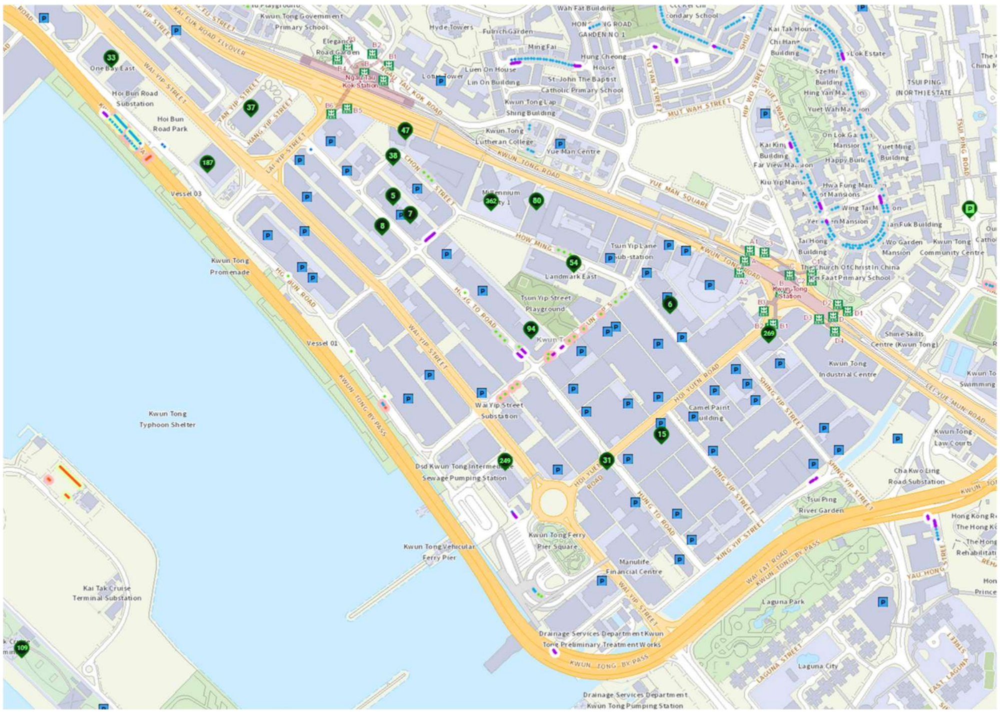
Details:

1. To find the vehicle speed at B, find two points A, C that are colinear and 100m away from B.
2. Get time needed to travel from A to C.
3. Speed can then be determined using the formula $speed = \frac{distance}{time}$.

2.1 Measurement of parking space availability

Now that the method of measuring traffic has been proposed, I will now focus on ways to quantify/measure the locational advantage of a spot.

Below is a map of the study area, with blue squares representing off-street parking lots, and green-outlined placemarks represent off-street parking lots with real-time availability data. However, because a large proportion of parking spaces do not offer real-time availability data, to ensure that data fairness, the **absolute number of parking spaces the place has** will be used instead.



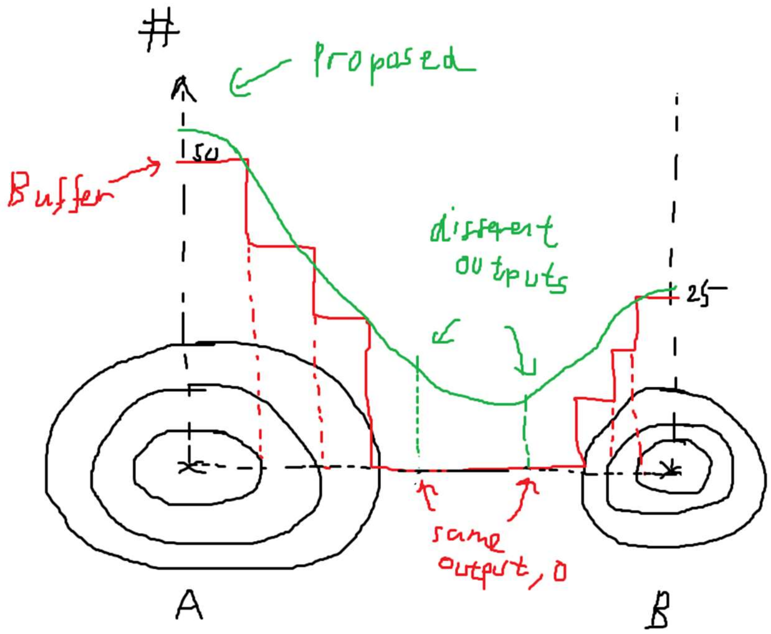
Source: Hong Kong eMobility Website

The original proposed method was to use buffer zones, which is the process of drawing buffers, or "spheres of influence" around each parking lot, with its radius proportional to the number of parking spaces. Data will then be collected at specific intervals along each "ring".

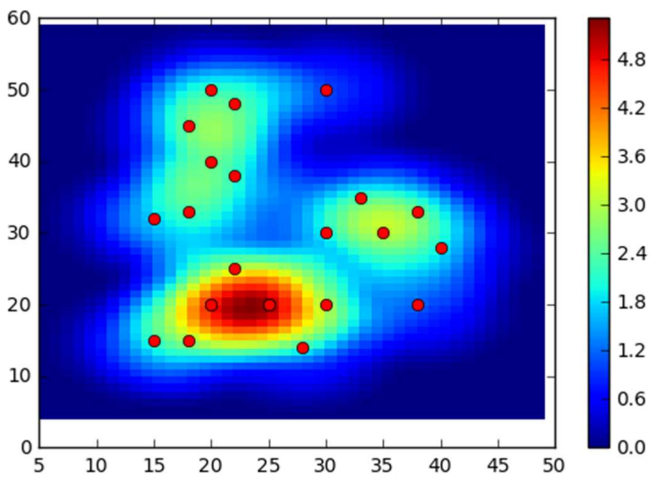
However, buffer zones may not be suitable in this context because parking spaces are rather evenly distributed and therefore may not exhibit a significant difference in traffic flow. I've proposed two methods that I am currently exploring:

1.2.2 Heatmap

Suppose Parking Lot A, which has 50 parking spaces, and Parking Lot B, which has 25 parking spaces:

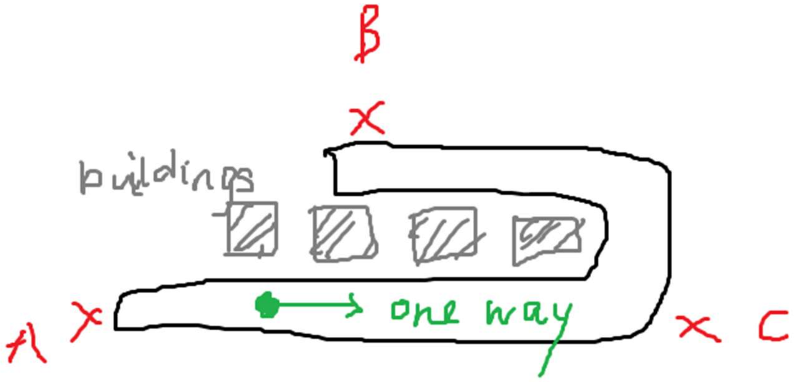


With buffer zones, 3 circles with **discrete boundaries** are drawn away from the centre of the parking space. However, a limitation is that when trying to compute a "score" for which there are no circles "overlapping", it can become a significant problem because its output value is 0. Therefore, the proposed method will be to interpolate a smooth curve between A and B, so to more accurate depict how the influence of parking lots influence each other. This can be done with Python packages such as heatmap.



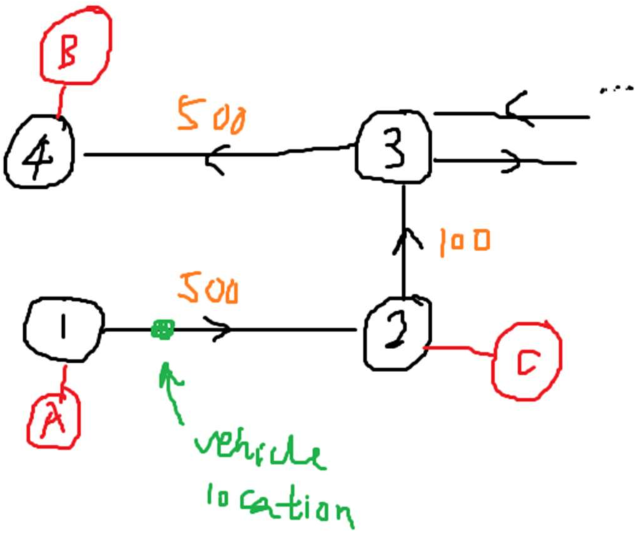
1.2.3 Node graphs

A major flaw for the previous methods is that it assumes that the area is a flat plane, and that it will ignore any geographical constraints that prevents it from taking a direct path towards the parking lot. Take the example of a vehicle (in green):

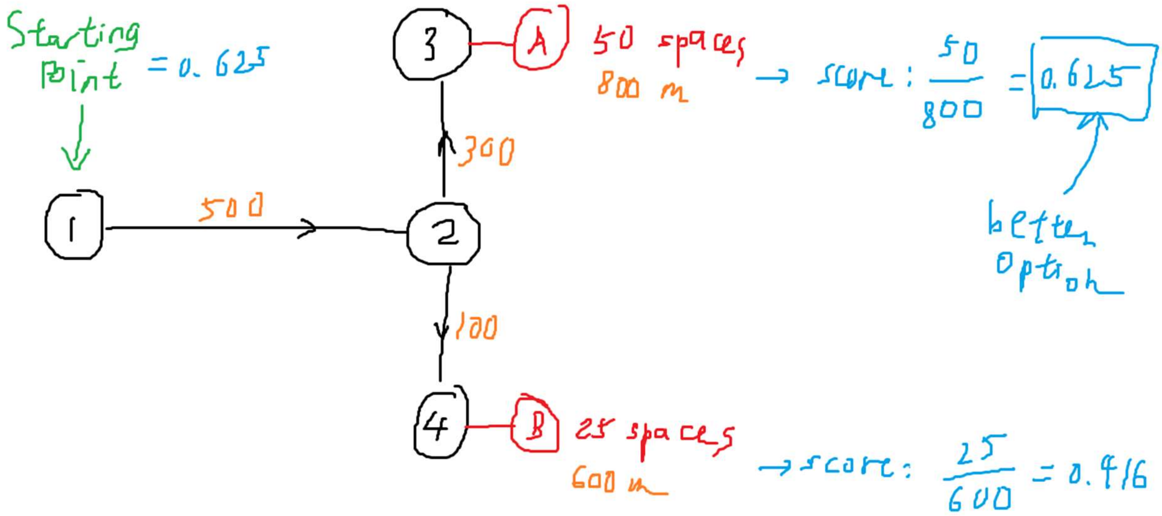


In the previous two methods, A and B are considered the most favourable parking lots as they have the lowest direct distances, but it is not true because the one-way road prevents the vehicle from backtracking to A, and buildings have blocked the access to B. Hence, the algorithms above fails to recognise that C is the "best option".

A new model that eliminates this issue is to create a custom "road network" consisting of nodes (which acts as intersections), with lines connecting each node (acting as roads).



With this method, it is relatively easy for computer programs to find the best route and guarantee that each route is valid and "drivable". Here is a demonstration on how the computer determines the score for any location along the line:

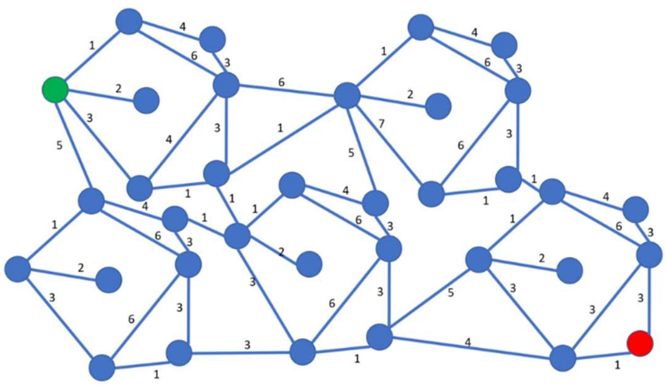


The function that determines the locational advantage score will be:

$$\text{score} = \frac{\text{number of parking spaces}}{\text{driving distance}}$$

The reason for this is essentially, we want the score to be higher as the driving distance is decreased, and when the number of parking spaces increases. With this fact, by varying the location of the vehicle along the line, we will be able to generate a set of "scores" easily.

Just to demonstrate the viability of this option and its expansibility to a larger number of interconnected nodes, there are many algorithms that are designed specifically to solve such complex node graphs, such as the Dijkstra's algorithm:



In conclusion, I believe that this model is better than the previous methods because it not only gives a better representation on the locational advantage "score", but also allows me to explain phenomena easier. In this example, explaining the concept of traffic backlog as a result on increased demand at B will cause congestion at Intersection 2, which overall slows the traffic down.

More about node graphs in traffic management:
https://res.mdpi.com/d_attachment/sustainability/sustainability-12-02056/article_deploy/sustainability-12-02056.pdf

In case Google Maps fails / runs out of quota – Bing Maps API provides a similar alternative, with information such as free-flow speed: https://docs.microsoft.com/en-us/rest/api/maps/traffic/gettrafficflowsegment