Daniel Cheah u5592643

ENGN 2226 Systems analysis personal portfolio

Topic: Systems analysis of main choices a new student to ANU will make if they wish to commute by bicycle.

Executive summary:

Australian National University receives a multitude of enrolling students every year. Of these students many wish to commute to university by bicycle. However, this would impose several constraints and decisions upon the student such as where should they live and what type of bike they should buy to maximize their biking experience. This report aims to alleviate these concerns by performing a systems analysis on these issues. Due to limitations in scope, the problems focused upon are the location of residence, what type of bike should be bought and disposal of the bike. It was determined that the maximum distance to residence should be less than 8km, casual or road bikes and reuse and recycling of the bike as a disposal method are optimal due to advantages in travel time, versatility and energy efficiency.

Contents

Introduction:

Cycling is a healthy and efficient method of travelling and is the chosen transportation method of many people in Canberra. However, there are a multitude of choices that may be made related to cycling, and a wrong choice may make commuting by bicycle undesirable and inconvenient. For people who live in the vicinity of ANU and have biking experience, such decisions may be trivial. However, new students may have more pressing matters to deal with and thus may make an incorrect decision. To alleviate these concerns a systems analysis has been performed on several key decisions that affect cycling experience.

Problem scoping:

To investigate the problem qualitative data was gathered and analysed to determine aspects of key interest. This allowed the problem to be broken down into design requirements and attributes which characterizes the problem and identifies key points of investigation.

Qualitative data:

Existing data from the Australian Bureau of statistics was gathered to identify reasons why people don't commute to work or study by bicycle.

Main reasons for not cycling to work or full time study in the ACT

ID	Reason	Proportion
1	Work/study distance is too far	70.1
2	Need motor vehicle before/during/after hours	24.3
3	Doesn't own bicycle/cannot ride bicycle	13.3
4	Lack of time	11.8
5	Need to carry goods/equipment	6.9
6	Health/physical restrictions	6.9
7	Road safety issues/hazardous	5.5
8	Concerned about personal safety	5.1
9	Other	4.0
10	Climate/weather/seasonal factors	3.5
11	Not interested/no reason/has not considered it	3.0
12	Lack of suitable pathways/end of trip facilities	1.1

Table 1 - Reasons for not commuting via bicycle and the relative proportions of people who gave that reason Note: proportions do not sum to 100 as more than option could be selected. (Australian Bureau of Statistics, 2015)

This implies that the distance is the main factor for not choosing to bike to university by a large proportion. However, lack of time is closely related to distance as they are roughly proportional to each other. As a result it may be possible that people have interpreted a large distance to work as an increased amount of time required to cycle to work. And henc, the importance of a large distance may be overemphasized and the lack of time underemphasized.

Factors or reasons such as needing a motor vehicle at night (2), health and physical restrictions (6), Other (9) and not interested (11) are of lesser importance as the problem is under the assumption that the user has decided to purchase a bike and wishes to cycle to university with it.

Therefore, this census implies that factors that should be considered are: the distance to university, time required to travel to university, not owning a bike, needing to carry goods, health or physical restrictions, safety, climate and a lack of suitable pathways.

This data however applies to the entirety of ACT and is not specific to ANU and thus may not be entirely representative of the student population. However, ANU is within the ACT and thus this data can be considered to be fairly representative.

Defining the problem scope:

A systems boundary chart was drawn to define and limit the scope of the problem. The Endogenous aspects define the features that can be changed and hence were used for the design attributes.

Systems boundary chart

Endogenous	Exogenous	Excluded
Location of residence	Traffic conditions at a given time	Lack of experience with riding a bike
Type of bike bought	User	Time of bicycle use (daytime or night)
Frequency of maintenance	Road infrastructure	Cost of home
Type of maintenance performed	Laws regarding bicycle use	Weather conditions
Disposal of the bike		
Accessories bought		

Table 2– Systems boundary chart

This shows the systems boundary chart for the choices and factors involved that an arriving student may face. Endogenous aspects involve aspects such as location of residence and type of purchased bike as the student can directly decide upon the result. Aspects in the exogenous section mainly include factors that involve road conditions as they are usually outside the users range of influence. Factors included within the excluded section are mainly placed there to simplify the problem or because it has been predetermined that the student wishes to commute by bicycle.

Design requirements

The design requirements which characterize the system were generated and compared to the data gathered in the qualitative analysis via house of quality (see figure 8 and table 7 appendix). This allowed determination of the most important aspects of the system which were distance/time of commute, cost, maintenance, comfort then environmental impact in that order.

Design attributes

	Ranking	Location of residence	Type of bike purchased	Disposal method
Distance and time	56	9	1	
Safety	23.3	3	9	
Cost	9.76		9	1
Time required to perform maintenance	9.18		9	
Comfort	0.9	3	3	
Environmental impact	0.84		3	9
		577	232	17.32

Figure 1– House of quality comparing design requirements and design attributes

The system attributes where retrieved from the system boundary chart and compared to the design requirements by a house of quality. This showed that position of origin, which is the location of the house was the most important attribute followed by type of bike purchased then the disposal method. This report will therefore investigate the location of residence, the type of bike purchased and the disposal method while focusing upon the position of origin. There are other design attributes such as types of accessories purchased and frequency of maintenance however these were excluded from investigation to limit the scope of the problem.

Location of residence:

The location of residence was found to be of critical importance and thus has been investigated through the use of various analysis techniques. Safety considerations were investigated by looking at human factors. Suitable travel distances were investigated by quantitative data and time factors. And the effect of increasing the load on the biking infrastructure was investigated by looking at system dynamics and control. To further investigate the effect of increasing load on the system, a simulation was also produced that drew data from all the aforementioned topics.

Human factors - Safety:

Safety is an important consideration when commuting and was found to be one of the highest ranking design requirements. The safety considerations investigated in this section are safe following distances between bicycles and the relative safety of different types of infrastructure.

There are multiple different types of infrastructure available for cyclists in the ACT. Of particular notice are the bicycle paths, roundabouts, traffic lights and roads. The difference between travelling on the road and on a bicycle path is that cyclists need to adjust their speed and path to avoid other vehicles which may be parked on the side of the road. As such overall speed and safety can be expected to be lower. Additionally, travelling on the road may increase the chances of encountering an intersection that isn't designed to accommodate for cyclists. This is undesirable as intersections such as roundabouts have been reported to be 2-3 times more dangerous than other intersection types (ACT Government 2015). Therefore, bicycle paths and traffic lights should be preferred over roundabouts and roads.

Maintaining a safe following distance is important for safe travel for irrespective of the transportation method. There are multiple recommended ways to determine a safe following distance however the Queensland government recommends that bikers maintain a travelling distance of 2m behind a motor vehicle (Queensland Government, 2015). The difference in stopping distance and reaction time required is unlikely to be significantly different when travelling behind a motor vehicle to a bike and hence a following distance of 2m shall be used.

Quantitative analysis - Travel distance:

To determine the maximum travel distance the maximum time of travel was investigated. Statistics from a 2010 General Social Survey in Canada have shown the results as below.

	Very dissatisfied or dissatisfied	Satisfied	Very satisfied
Less than 15 minutes	4	26	70
15 to 29 minutes	7	55	38
30 to 44 minutes	16	63	21
45 minutes or more	45	46	9

Table 3- satisfaction versus the commute time (Statistics Canada, 2015)

Statistics from Canada show that approximately 45% of people were dissatisfied with a commuting time of 45 minutes or more. As this is a large proportion of the population as well as is a significant

drop of in satisfaction from the previous set (30 to 44 minutes with a dissatisfaction of 16%) thus 45 minutes will be used as an upper limit for commuting time.

However, this data may not be applicable to ANU as it involves a sampled population from Canada. This may change results due to differences in vehicle type and type of terrain. However, it shows that 45 minutes is the limit for satisfaction and thus should be a reasonable limit.

Next the average velocity of a bike was investigated and found to be approximately $10kmh^{-1}$ to $20kmh^{-1}$(Full cycle; Better by cycle; Quora). As such an average speed of $15 \pm 5\ kmh^{-1}$ shall be used as an estimate. However, average speed may be affected by aspects such as slope, weather, terrain and physical fitness. Additionally, the source of this information may be of questionable nature. As such improving reliability by using multiple sources and using a range of values is the best that can be achieved.

Given an upper limit of 45 minutes then the maximum suitable biking distance is approximately 11km ($\frac{45}{60} \times 15 = 11.25 \approx 11$) However, commuting by bicycle generally involves repeated acceleration and deceleration which can be expected to reduce the average speed. Therefore, this figure can be expected to be smaller than what has been calculated.

Time factors – queuing system of bicycle infrastructure:

Commuting by bicycle involves intersections which produces queues which may slow the progress of travellers. This is important as there have been reported cases of congestion on the road causing hours' worth of waiting time (CityLab, 2015). This section therefore investigates the effect of intersections on the expected travel time of cyclists.

The utilized bicycle infrastructure model:

It can be observed that there is a queuing system on the bicycle paths, the roads, intersections and traffic lights. However, the roads and bike paths however behave slightly differently to a conventional queue in that they promote queue jumping if safe. Individually, each road and bike path can be viewed as an individual queue of various number of channels depending on the number lanes present. Overall, the entire road system and bicycle system combined make up a multi-channel, multi-phase system that facilitates travel between 2 points. For simplicity, in this section, queue jumping will be excluded from calculations and road rules traffic light signals will be obeyed at all times.

Data regarding wait times at intersections

Average length of a bike (General Design Factors, 2007)	1.7m

Estimated average green light phase at an intersection (ACT government territory and municipal services, 2015) and by observation	30 seconds
Estimated average red light phase at an intersection (ACT government territory and municipal services, 2015) and by observation	60 seconds
Average crossing time of a bike on a 2 lane intersection by observation	5 seconds
Safe following distance between bicycles (see human factors - safety)	2m
Normal road lane width (Urban services, 2015)	3.5m
Number of people that can fit at the curb at an intersection by observation	2
Maximum arrival rate per traffic cycle at peak traffic by observation	15

Table 4 – data used to calculate the wait time at intersections

values were gained by observation, estimation and research and thus are subject to error.

Assuming a 2 lane intersection, then the number of people that can cross every traffic light cycle is approximately:

$$\frac{normal\ lane\ width}{length\ of\ bike + safe\ distance} \times number\ of\ people\ at\ curb \times \frac{estimated\ traffic\ light\ green\ phase}{average\ crossing\ time\ on\ bike} = 22.792 \approx 23$$

If a single channel queuing model is assumed then:

$$E(V) = \frac{1}{C\mu - \lambda} = \frac{1}{23 - 15} = 0.125\ minutes = 7.5\ seconds$$

This estimate is however unreasonable as wait times at an intersection are generally larger than 7.5 seconds. This is likely to be caused by the non-uniform service of the traffic lights wherein bicycles are only allowed through on green. However, once green, in most cases all the bikes at the intersection are allowed through due to the large throughput. This effect has been observed at ANU in the morning where waiting bikes generally are all able to pass within one traffic light cycle. Because of these effects, it will be assumed that as long as the system is not overloaded then the average wait time at an intersection is determined by the length of the red phase of the traffic light system. The worst case scenario will be taken which is that the person will need to wait a maximum time of 60 seconds before crossing.

Density of intersections when travelling to ANU

To determine the effect of the calculated wait times at intersections the density of the intersections per kilometre was estimated by taking several sample paths on Google maps:

Origin	Number of	Approximate wait time at	Distance from ANU	Density of wait

	intersections	intersections (minutes)	(km)	time caused by traffic lights (minutes / km)
51 Duffy street	7	7	3.2	2.2
Choku Bai Jo, the farmer's outlet	10	10	4.8	2.0
Ginniderra drive	7	7	6.4	1.1
Anzac Parade	5	5	3.2	1.6

Table 5 – table showing the number of intersections, distance and approximate wait time between various positions and ANU See figure 8 - 11 in appendix.

- Average waiting density = 1.7 $minutes/km$
- Standard deviation $= 0.42$

$$distance\ allowed = \frac{time \times average\ velocity}{1 + average\ density \times average\ velocity} = \frac{45 \times 0.25}{1 + 1.7 \times 0.25} = 7.89^{1.37}_{1.01}$$

$$\approx 8 \pm 1.4\ km$$

This implies that the distance that one can expect to bike to ANU under a reasonable time frame is approximately 8km. The number of intersections is dependent on the road conditions and bicycle infrastructure. This appears as the large uncertainty and the amount the wait time density varies. As such this figure should not be taken as an exact value.

System dynamics and control – effect of overloading the traffic system:

Thus far all calculations have assumed negligible interference by other travellers. However, the effect of this may not be negligible and will be investigated in this section:

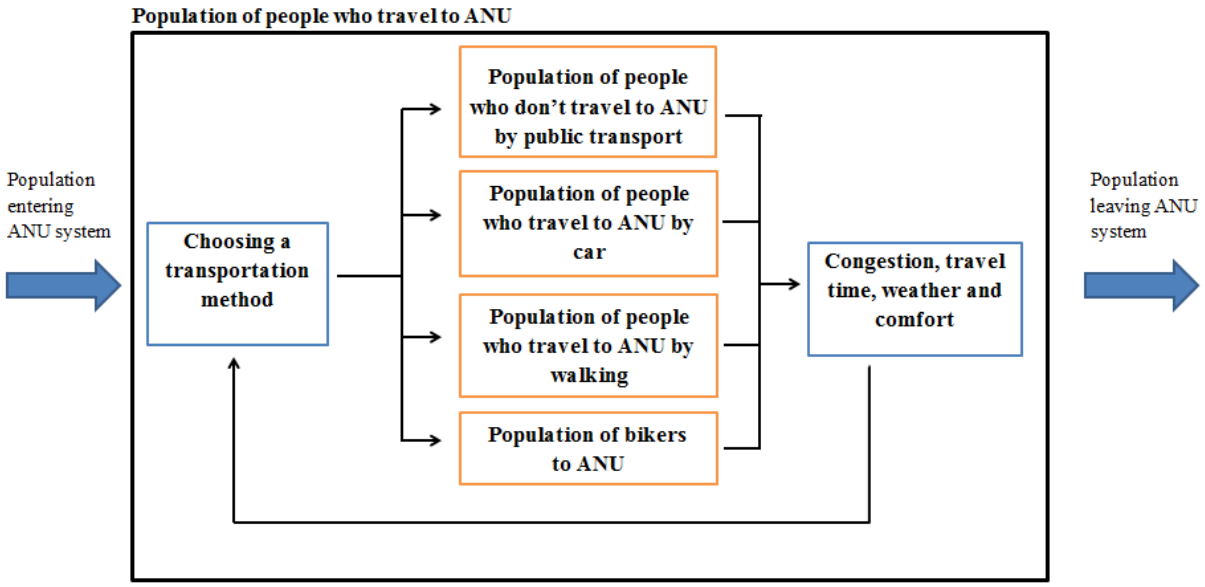

Figure 2 – system dynamics diagram

Above is a diagram that describes the feedback mechanisms of people who choose to commute to ANU. There are people who enter this system such as new undergraduates, and people who leave this system such as post graduates. Within the system includes various method of transport such as by car, public transport such as busses, people who walk and bikers to ANU. However, each method of travel may undergo regular checks to determine efficiency and suitability and if it is unsuitable then another method of travel may be chosen. Examples of this include how walking and biking may be temporarily undesirable due to bad weather conditions.

A longer term complication however may be caused by issues such as congestion and travel time. This is illustrated by the hypothetical situation wherein the entire population that travels to ANU decides to start travelling to university by bicycle. What may result is that the system may be unable to cope with the increased influx to this transportation type which may cause large amounts congestion and increased travel time. There may also be issues with insufficient parking space. Thus, this implies that there is a limit on the 'population' within each travel method that is based upon physical factors such as reasonable travel distance, bicycle path width limiting the throughput of a bicycle path and the total number of parking spaces limiting the maximum number of bikers.

There is therefore expected to be some balancing effect between the population of ANU and the infrastructure that supports biking. This has consequences such as if congestion during peak times is a limiting factor in the number of bikers, then doubling the width and lanes of the bike paths may increase bikers. However, if it is determined by other factors such as comfort and travel time and if the system has already reached saturation, then cycling infrastructure may be unnecessary and redundant.

To more accurately determine the dynamics of the system and its effect on the travel time, a model has been created and simulated:

Simulation

To investigate the dynamics of the system, a simulation modelling the traffic infrastructure was written in Java. This was achieved by approximating the system with a series of nodes and paths. The paths connect two nodes and take time to traverse. Nodes connect multiple paths together and may have a traffic light timing system that limits the throughput of the intersection. Bikes arrive at the nodes and progress through the system until they reach ANU.

To produce the model several additional quantities need to be calculated or estimated including the population density for the arrival rate at nodes and the capacity of a bicycle path.

Quantitative - Population density near to ANU

The population density of North Canberra as recorded in 2010 is 232.2 $Persons/km^2$(ACT government 2011) and thus it can be approximated that if all the people within a 1 km^2 were to commuted by bicycle to ANU or Civic then it can be expected that the rate of arrival per second per km^2 is 0.0645. This can be expected to be an overestimation of the arrival rate if only this region is considered because it relies on the assumption that everybody will commute by bicycle. However, considering that people outside of the calculated area may also utilize this path then it may not be an excessive overestimation of the total arrival rate at peak hour.

Human factors – Capacity of a bicycle path:

To investigate the load that the bicycle paths can take, the size of the pathways that bicycles use was measured and analysed. Several measurements of the width of the bicycle path at the Lyneham wetlands were taken:

Side 1 (m)	Side 2 (m)	Total length (m)
1.3	1.3	2.6
1.4	1.6	3.0
1.6	1.6	3.2
1.6	1.5	3.1
1.55	1.55	3.1
Average		3.0

Table 6 – width measurements taken of bicycle path in the ACT, Approximate values were taken due to the ambiguity of the sides of the bicycle path which was caused by dirt covering the sides.

According to a study done in 1998 the average human male's width between the elbows is approximately 105.6cm if their posture is that of standing with arms spread out(Anthropometry and mass distribution, 1988). However, this figure is expected to be smaller as arms are generally tucked in when cycling. Since the width of the bicycle path is 3.0m on average then this implies that approximately 3 people can bike side to side. However, this rarely happens due to safety considerations. As such bike lanes can be assumed to only allow riders of a single file. This does not take into account the average size of a female however in general female proportions are smaller than males. (Anthropometry and biomechanics, 2015)

Cycle lane width categories for bike paths on the sides of roads in the ACT are: 1.2-2.5m for 60km/hr, 1.8m-2.7m for 80km/hr and 2.0-3.0m for 100km/hr speed limits (Ten year master plan, 2004). This means that bicycle paths on the side of the road are also likely to be designed for a single file of riders.

Scope of the simulation:

To limit the scope of the simulation only a slice of the traffic system has been modelled. While this reduces the accuracy of the model, it is necessary due to time constraints:

Figure 3 – Chosen modelled section of the Canberra bicycle infrastructure ($1.5 \times 3\ km$)

To model the system, the several assumptions and constraints were made based upon research and results found in previous sections. A full list of assumptions can be found in the appendix.

Inflow rates:

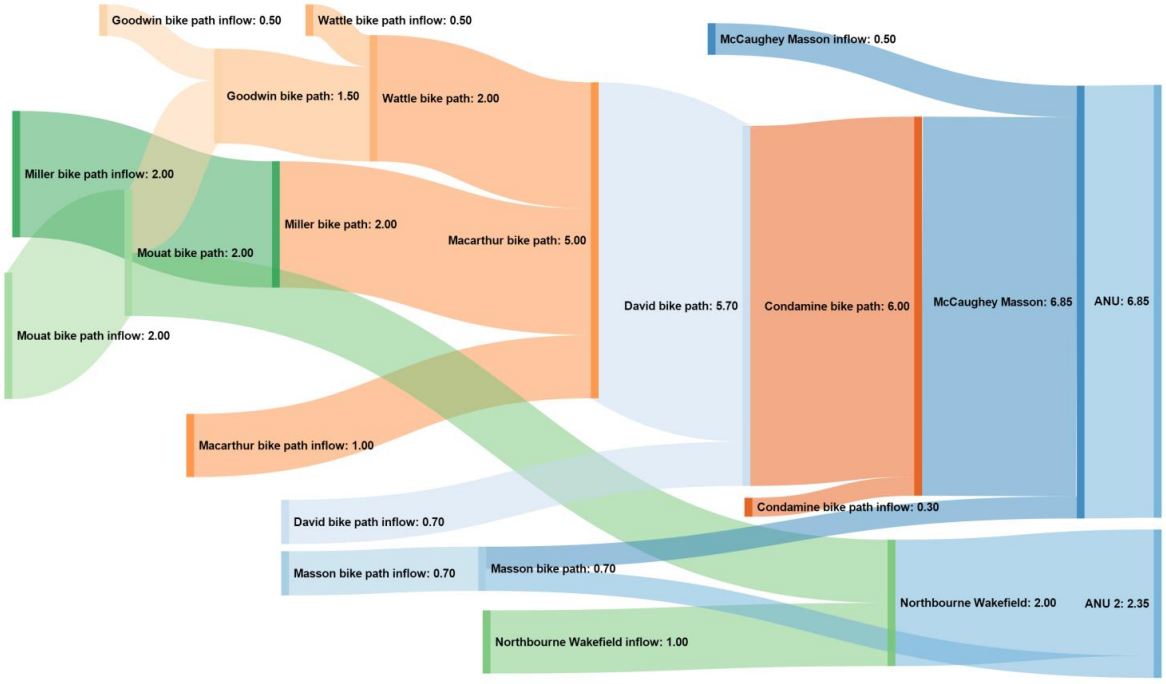

Figure 4 – Sankey diagram of flow rates of model

In the Sankey diagram above are the intersections and flow rates utilized in the model. As the population density is assumed to be evenly distributed, to more accurately model the rate at which

the population enters the system the inflow at each intersection was scaled to the relative distance to another node.

Simulation results:

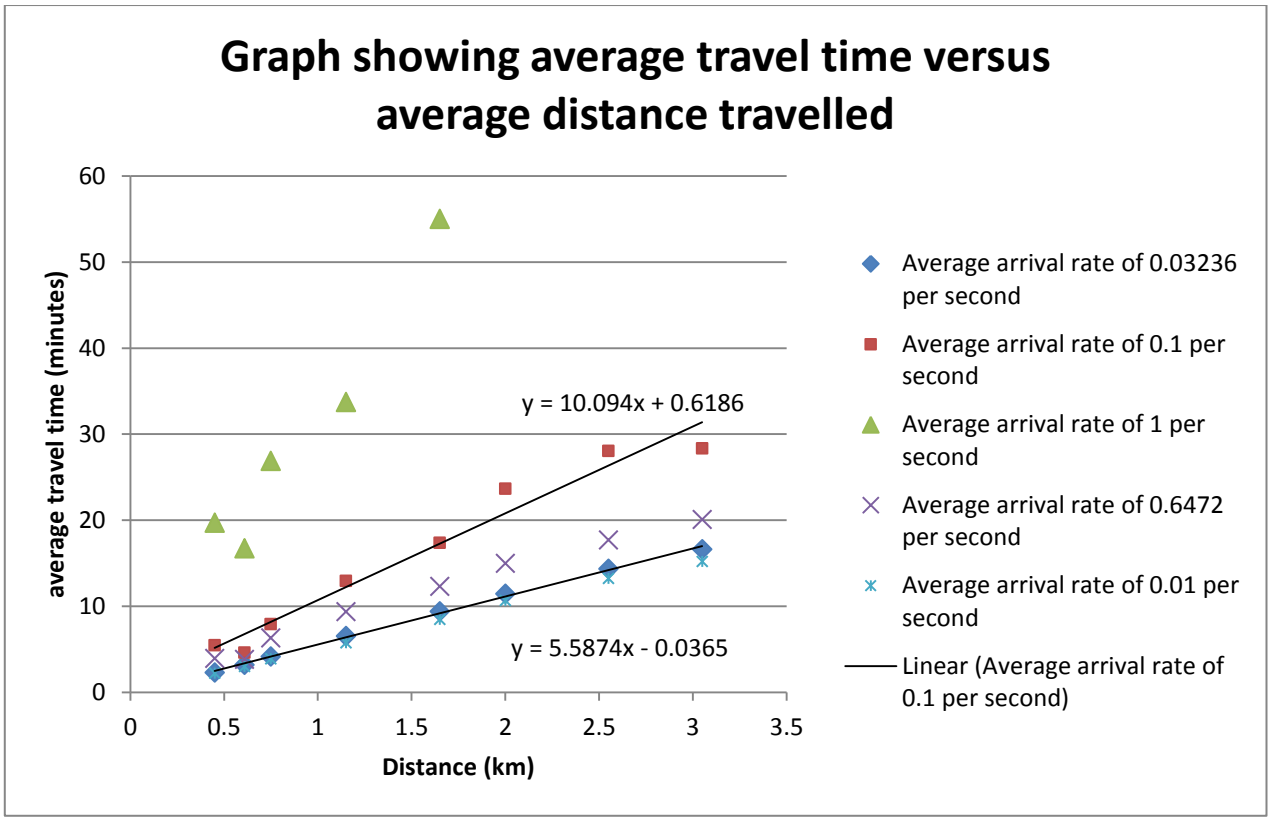

Figure 5 – Graph showing average travel time versus average distance travelled

The relatively linear nature of the results implies a directly proportional relationship between distance and average travel time. It also implies that as the arrival rate increases, the gradient at which the average travel time increases with distance also increases. By projection, it can be estimated that for an arrival rate of each node between the ranges of $0.01\ to\ 0.65$ the average maximum distance is approximately 9km ($\frac{45}{5.59} = 8.05 \approx 8km$). However, as the arrival rate increases, the intersections start to overload thereby increasing time due to extended queues. As a result, the efficiency of the system drops. For an arrival rate of 0.1 per second, the maximum recommended distance to commute to university is approximately 4.5km ($\frac{45}{10.09} = 4.45 \approx 4.5$km) and for an arrival rate of 1 per second the maximum recommended distance is 1.5km ($\frac{45}{30.676} = 1.466 \approx 1.5$). However, an arrival rate of 1 unrealistic because it implies an average of 32400 entering the system within an hour.

The value 0.03225 ($0.0645 \times \frac{1.5 \times 3}{9}$) was derived from the arrival rate derived from the population density over the nodes in the system. This implies that the infrastructure is currently sufficient for cyclists and that the maximum recommended distance of residence remains 8km.

This result however is based upon an uncertain set of data and an investigation of limited scope and thus further investigation should be performed to verify the results and extend conclusions beyond the limited scope chosen for the simulation.

Bike disposal:

While disposal is not an immediate concern, it will need to be addressed eventually. Additionally, end of life is an issue concerning universities considering how one of the disposal methods that students choose includes abandoning bikes on campus. This is wasteful as well as environmentally unfriendly especially if the bikes end up in a landfill (Transportation services; The red and black). Therefore, materials and energy factors were used to investigate this end of life issue.

Material factors and energy factors – recycling and reusing:

More environmentally friendly methods opposed to disposing by landfill include reducing, reusing and recycling. Reducing can be achieved by constant maintenance to ensure the bicycle lasts as long as possible. However, this only delays the inevitable and eventually disposal is necessary. There are several methods of achieving this including companies which reuse the bicycle through resale or recycling. Two of these companies include the green shed and the Canberra environment center (Ecoaction; The green shed).

However, reusing only applies if the bicycle is still in good condition. The lifetime of a new bicycle is generally longer than 4 years. Therefore, if a new bicycle is bought then it is likely that it will last for longer than the degree the student is studying. However, if the bicycle is bought second hand then it is further along its lifetime and thus more likely to fail due to wear-out failure according to the bath tub curve. Out of the causes of failure, crank shaft failure appears to be the most common (Broken Bike Bits, 2015). As this is only a small part of the bicycle then it can be replaced or the broken bicycle recycled. According to the Pareto principle, if this is taken care of well then the university student is likely to have a fully functional bicycle at the end of their degree.

As both methods reduce the environmental impact, it is ideal to reuse and then recycle the bicycle. However, this may not always be possible as reuse may result in excessive rust. Therefore, a case wherein only recycling or reusing will be considered by use of $I = PAT$.

Firstly, the population is expected to be the same between different end of life models and thus will not be considered. Secondly, between recycling and reusing, recycling can be expected to have a larger affluence because reusing reduces the number of new bicycles into the system. The lifetime of a bicycle can be estimated to be 6 years. As the average length of a degree is 4 years, then the affluence is approximately 1.5 times larger. For calculation of technology, the improved efficiency

due to recycling will be considered. For a bicycle, the majority of the mass of a bicycle comes from its frame which may be composed of various materials such as titanium, aluminium and steel. Of these types aluminium is a common material used. When recycling aluminium, 20 times less energy is used and hence the technology factor of recycling is 20 times better. This advantage may be reduced due to other factors such as energy cost in transporting, dismantling and recycling other bike parts. The significant difference implies that recycling is better than reusing if only one method is available.

Type of bike purchased

Cost factors – comparison of different:

Cost was considered relative to the type of bike purchased. Below is a table of the approximate cost or price range of different types of bikes.

Type of bike	Approximate cost or approximate price range ($)	Operating cost ($)	Approximate maintenance cost ($)	Disposal cost ($)	Residual cost ($)
Road bike	800	$0	$18 / 2years	$0	$0
Comfort bike	400				
Recumbent	2000 – 4000				
Second hand	100 – 500				

Table 7– cost of different types of bikes (Greenspeed; Anaconda; Gum Tree; 99 Bikes; London cyclist)

This implies that the cost of different types of bikes is dependent on the capital cost rather than maintenance as it is highly unlikely for a bicycle to last over 10 years without several major replacements. Therefore if cost is only considered then the second hand or comfort bikes are the better choice of bicycle.

Evaluation matrix

This cost data was then combined with data researched and gathered from other parts of this report and weighted against design requirements to compare each type of bike in an evaluation matrix.

Time of travel

The time of travel of each type of bicycle can be expected to be fairly similar as it is dependent on personal ability. Cases where this may differ may be in the second hand bike which may not be equipped with gears and thus may not operate at optimal efficiency at certain speeds.

Time required for maintenance:

The time required to perform maintenance is also likely to be similar, however the second hand bike may require more maintenance due to how it is further in its life.

Safety and comfort

In terms of safety and comfort, each bike has its own set of benefits such as how recumbent bikes can result in health benefits to wrists, hands, arms, shoulder, neck and back (Paul K. Nolan, 2015). Additionally, Recumbents are also more aerodynamic than casual bikes (Ignacio et al, 2012)., However they also have a set of disadvantages as well such as how recumbent bikes have a lower field of vision and thus may be more likely to be involved in crashes. Second hand bikes may also be in worse condition and thus may be less likely to perform adequately.

Environmental impact

If environmental impact is considered then comparing between different types of bikes, the population, affluence is the same and the technology of each type of bike can be expected to be fairly similar, therefore the type of bike chosen can be expected to have little difference in impact on the environment. However, second hand bikes can be considered to be slightly improved due to their reuse.

	Importance	Road bike	Recumbent bike	Comfort bike	Second hand bike
Time of travel	56	3	3	3	2
Safety	23.3	3	1	3	2
Cost	9.76	2	1	3	4
Time required to perform maintenance	9.18	3	3	3	2
Comfort	0.9	3	4	4	3
Environmental impact	0.84	3	3	3	4
Rating		290	235	301	222

Figure 6 – evaluation matrix comparing different types of bicycles.

This implies that comfort bikes are the optimal choice for the type of bike used to bike to university. This choice is however subject to personal opinion as relative levels of importance were determined using census statistics. The reason why comfort bikes was shown to be optimal is likely because there is not significant advantage of any other bike over the comfort bike in terms of any of the criterion and is instead well rounded whereas other types of bikes fail are better at a few design requirements however are worse off at the rest leaving them worse off overall.

Additionally the viability of the road bike has been determined to be close to the comfort bike and with a sensitivity analysis the results may change. Thus, it is recommended to purchase road or comfort bikes.

Conclusions:

Systems analysis has been performed on the issue of various choices an ANU student must make if they wish to commute to university by bicycle. The main results were that:

- The location of residence should be within 8km of ANU to optimize the distance and travel time.
- For safety, when choosing a cycle path, bicycle paths and traffic lights should be preferred over roads and roundabouts.
- When disposing a bicycle it is optimal to allow reuse by donation then recycling. However, if the bicycle is overly rusted then it is more environmentally friendly to recycle than to reuse as it is much more efficient to recycle than to create a new bike.
- A casual or road bike is the optimal type of bike to purchase because it is well rounded. However choice may depend on personal opinion and whether its only purpose is to commute to ANU.

References:

99 Bikes, n.d., Lubricants and grease, Accessed 14/10/2015, Available from: http://www.99bikes.com.au/accessories/tools-maintenance/lubricants-grease

ACT government 2011, Population and residential density in Canberra, Available from: ACT Government Environment and planning directorate Planning

ACT government territory and municipal services, 2015, Drakeford Drive, Accessed 4 October 2015, Available from: http://www.tams.act.gov.au/roads-transport/traffic/trafficsignals/drakeford_drive

ACT Government 2015. Traffic signals - Territory and Municipal Services. [ONLINE] Available at: http://www.tams.act.gov.au/roads-transport/traffic/trafficsignals. [Accessed 17 August 2015].

Anaconda, 2015, road bikes, Accessed 4 October 2015, Available from: http://www.anacondastores.com.au/cycling/bikes/c/bikes

Anthropometry and biomechanics. 2015. Anthropometry and biomechanics. [ONLINE] Available at: http://msis.jsc.nasa.gov/sections/section03.htm. [Accessed 17 August 2015].

Anthropometry. And Mass Distribution . For Human Analogues. 1st ed. 1988. Web. 16 Aug. 2015.

Australian Bureau of Statistics, 2015. Environmental Issues: Waste Management, Transport and Motor Vehicle Usage, Mar 2012. [ONLINE] Available at: http://www.abs.gov.au/AUSSTATS/abs@.nsf/DetailsPage/4602.0.55.002Mar%202012?OpenDocument. [Accessed 10 August 2015].

Better by cycle, 2015. How accurate are Google Maps cycling time estimates? ~ Better By Bicycle. [ONLINE] Available at: http://www.betterbybicycle.com/2014/09/how-accurate-are-google-maps-cycling.html. [Accessed 10 August 2015].

Broken Bike Bits, 2015, Accessed 4 October 2015, Available from: http://www.pardo.net/bike/pic/fail-001/000.html

Bicycle Habitat, n.d. A simple bike maintenance chart, Accessed 14/10/2015, Available from: http://bicyclehabitat.com/how-to/a-simple-bike-maintenance-chart-pg366.htm

London cyclist, 2015. Bike lube: What's all the fuss?. [ONLINE] Available at: http://www.londoncyclist.co.uk/bike-lube/. [Accessed 01 September 2015].

CityLab, 2015, How to Create a 26-Lane Traffic Jam in Delhi, Accessed: 11/10/2015, Available from: http://www.citylab.com/commute/2015/09/how-to-create-a-26-lane-traffic-jam-in-delhi/404509/

Ecoaction, n.d., Canberra environment center, Accessed 4 October 2015, Available at: http://www.ecoaction.com.au/

Full cycle, 2015. Venus Bike Club – Average Speed Article : Full Cycle Bikes, Bicycle Shop in Boulder, Colorado, Service, Sales & Rentals. [ONLINE] Available at: http://www.fullcyclebikes.com/venus-bike-club/average-speed-article/. [Accessed 10 August 2015].

General Design Factors, 2007, Mn/DOT Bikeway Facility Design Manual, Accessed 5 October 2015, Available at: http://www.dot.state.mn.us/bike/pdfs/manual/Chapter3.pdf

Greenspeed, 2010, Recumbent trikes, Accessed 4 October 2015, Available from: http://www.greenspeed.com.au/trikes.html

Gum Tree, 2015, Hybrid bikes in Canberra region, ACT, Accessed: 5 October 2015, Available at: http://www.gumtree.com.au/s-canberra/hybrid+bike/k0l3002977

José Ignacio Íñiguez, Ana Íñiguez-de-la-Torre and Ignacio Íñiguez-de-la-Torre (2012). Human-Powered Vehicles - Aerodynamics of Cycling, Applied Aerodynamics, Dr. Jorge Colman Lerner (Ed.), ISBN: 978-953-51- 0611-1, InTech, Available from: http://www.intechopen.com/books/applied-aerodynamics/human-poweredvehicles-aerodynamics-of-cycling

Logan waste services, 2015, Aluminium recycling, Accessed 5 October 2015, Available at: http://www.logan.qld.gov.au/__data/assets/pdf_file/0009/8775/aluminiumrecycling.pdf

Paul K. Nolan, 2015, Medical Benefits of Recumbent Bicycles, Accessed 5 October 2015, Available from: http://www.bikeroute.com/Recumbents/BentMedBenefits.php

Queensland Government, 2015, Bicycle road rules and safety, Accessed 14/10/2015, Available from: https://www.qld.gov.au/transport/safety/rules/nonpowered/bicycle/

Quora, 2015. What is the average speed of a bicycle in MPH? - Quora. [ONLINE] Available at: http://www.quora.com/What-is-the-average-speed-of-a-bicycle-in-MPH. [Accessed 10 August 2015].

Road safety commission, 2013, Keeping a safe distance behind other vehicles, Accessed: 4 October 2015, Available at: http://rsc.wa.gov.au/Documents/Road-Rules/ors-road-craft-following-distances.aspx

Statistics Canada, 2015. Table 3 Satisfaction with time spent commuting to work, 2010. [ONLINE] Available at: http://www.statcan.gc.ca/pub/11-008-x/2011002/t/11531/tbl003-eng.htm. [Accessed 10 August 2015].

Ten year master plan for trunk cycling and walking path infrastructure 2004-14. 1st ed. 2004. Web. 17 Aug. 2015.

The red and black, 2014, Abandoned bikes on campus left to rust, Accessed 4 October 2015, Available at: http://www.redandblack.com/uganews/campus/abandoned-bikes-on-campus-left-to-rust/article_b2fa6e54-2e30-11e4-946e-001a4bcf6878.html

The green shed, n.d., the green shed, Accessed 4 October 2015, Available at: http://www.thegreenshed.net.au/index.php/11/Bikes

Transportation services Texas university, 2011, Over 2,000 abandoned bikes on TAMU campus local bike company has answers, Accessed 4 October 2015, Available at: https://transport.tamu.edu/about/news/2011/11_9_30MaroonBikes.aspx

Urban services, 2015, Design standards for urban infrastructure, Accessed 4 October 2015, Available at http://www.tams.act.gov.au/__data/assets/pdf_file/0011/396857/ds03_roaddesign.pdf

Appendix:

Design requirements

Attribute	Metric	Direction of improvement
Distance of travel	km	Decrease
Cost	$	Decrease
Time required to perform maintenance	Minutes	Decrease
Environmental impact	Joules (embodied energy and energy waste)	Decrease
Comfort	Rating	Increase

Table 8

House of quality comparing design requirements and qualitative analysis results

	Ranking	Position of home	Cost	Time required to perform maintenance	Environmental impact	Comfort
Work/study distance is too far	70.1	9				
Doesn't own bicycle/cannot ride bicycle	13.3		9		1	
Lack of time	11.8	9		9		
Need to carry goods/equipment	6.9	3	3			
Safety issues	10.6	9	1	3		
Climate/weather/seasonal factors	3.5	3		1		3
Lack of suitable pathways/end of trip facilities	1.1	3				3
Importance		867	151	142	13	14

Figure 7

Time factors google maps images:

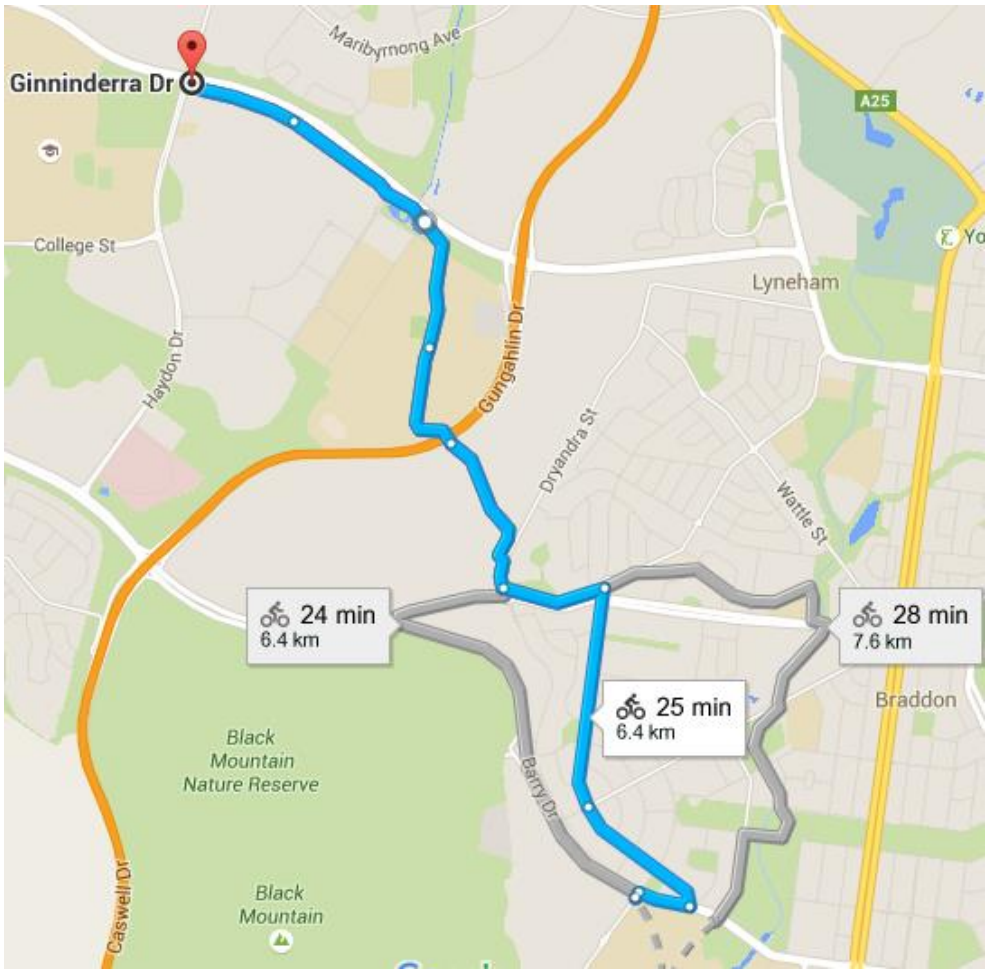

Figure 8

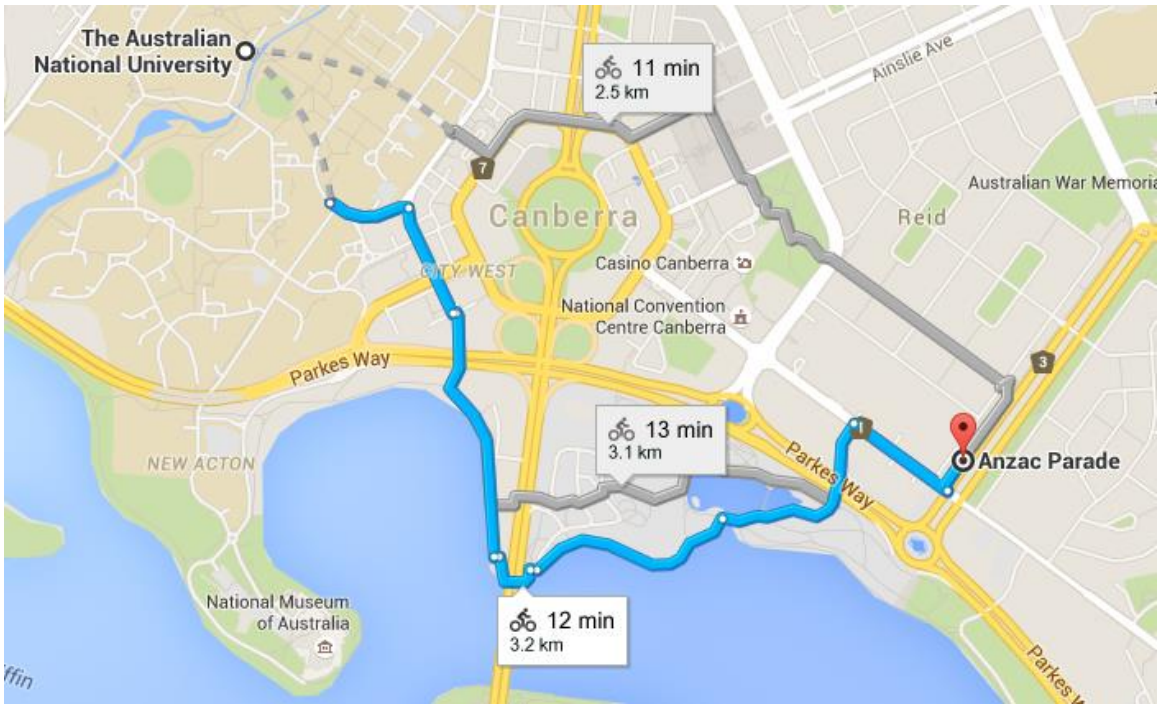

Figure 9

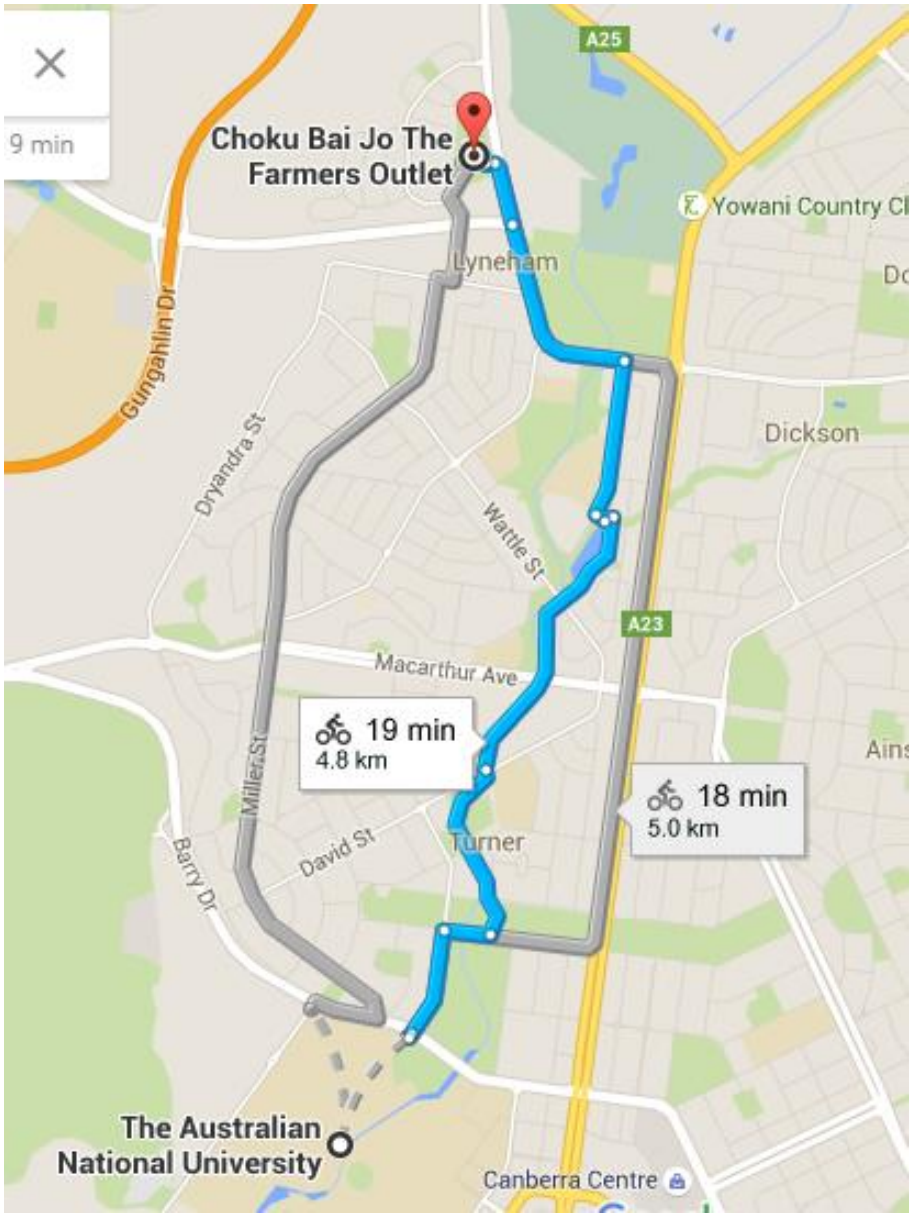

Figure 10

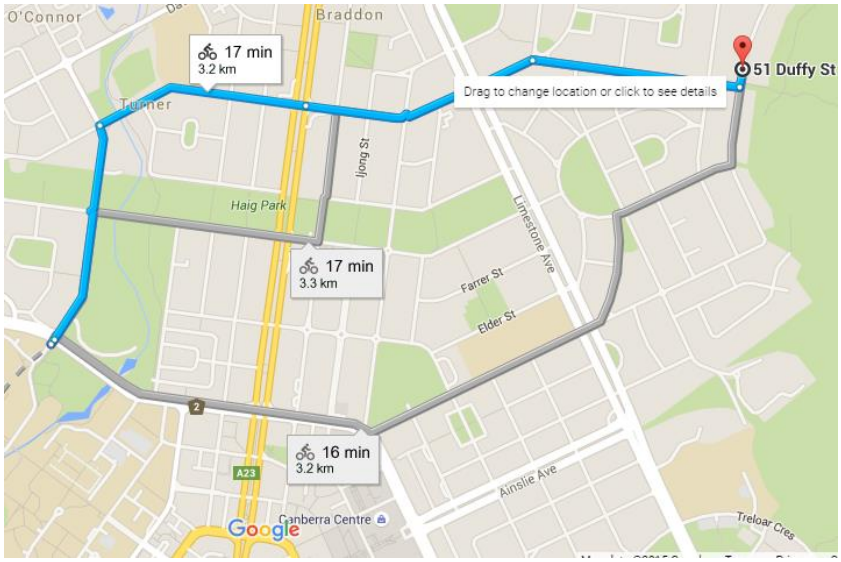

Figure 11

Assumptions made for simulation:

- The safety distance between bicycles as found by human factors has been set to 2m
- The throughput of an intersection as found by time factors has been approximated to be 1 bicycle per second. (for a 30 second cycle)
- A position has been invalidated for travel if travel to ANU takes longer than 45 minutes
- As found by human factors the bicycle paths are set to 1 bike at a location at a single point in time. The exception being when a bicycle is overtaking another.
- Traffic lights at intersections are obeyed at all times and queues at nodes are first in first out.
- Traffic light timings have been standardized to an average 30 seconds on 60 seconds off.
- Input of bikes into the system only occurs at the nodes and can be modelled by a Poisson distribution.
- Population density is assumed to be even within this given area
- The speed of an individual bicycle has been approximated with a even distribution around the average speed. This was decided to improve efficiency of the simulation. The average velocity of a bicycle has been set to $15kmh^{-1}$ as found by quantitative data.
- Bikes will overtake another bike if they are more than $1ms^{-1}$ faster.
- Peak hour of travel is estimated to occur only for a single hour

Simulation data:

Node location	Approximate average distance travelled (km)	Arrival rate per node: 0.01			Arrival rate per node: 0.3236			Arrival rate per node: 0.1			Arrival rate per node: 1			Arrival rate per node: 0.6472		
		Number of arrivals from location	Average travel time (minutes)	Average speed	Number of arrivals from location	Average travel time (minutes)	Average speed	Number of arrivals from location	Average travel time (minutes)	Average speed	Number of arrivals from location	Average travel time (minutes)	Average speed	Number of arrivals from location	Average travel time (minutes)	Average speed
Intersection between MaCaughey and Masson street	0.45	20	2.23	3.37	52	2.29	3.27	135	5.45	1.37	617	19.69	0.38	105	3.93	1.91
Masson bicycle path	0.61	20	2.98	3.43	61	3.18	3.22	149	4.59	2.23	772	16.71	0.61	103	3.81	2.69
David bicycle path	1.15	17	5.74	3.34	60	6.52	2.94	148	12.93	1.58	139	33.7	0.57	119	9.34	2.05
Condamine bicycle path	0.75	9	3.87	3.23	28	4.16	3.00	79	7.89	1.58	195	26.87	0.47	45	6.29	1.99
Macarthur bicycle path	1.65	30	8.46	3.24	93	9.41	2.92	164	17.36	1.58	11	54.98	0.50	164	12.29	2.24
Wattle bicycle path	2.00	9	10.65	3.13	51	11.43	2.92	77	23.65	1.41	0	NA	NA	86	14.98	2.23
Goodwin bicycle path	2.55	19	13.23	3.21	38	14.35	2.55	65	28.02	1.52	0	NA	NA	64	17.68	2.40
Miller bicycle path	3.05	51	15.20	3.34	164	16.59	3.05	233	28.31	1.80	0	NA	NA	246	20.08	2.53
Intersection between Northbourne ave and Wakefield street	2.20	27	9.62	3.81	88	10.47	2.20	285	11.28	3.25	902	26.92	1.36	176	11.13	3.29
29 Mouat Street	3.88	53	17.49	3.52	142	18.92	3.79	332	23.83	2.71	67	48.38	1.38	276	21.57	2.93

Table 9 – results from simulation

Java code for simulation:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

/**
 * Created by Daniel on 10/10/2015.
 */
public class base {
    public static int current_time = 0;
    public static int number_of_bikes = 0;
    public static ArrayList<String> list_of_intersection_names = new ArrayList<>();

    public final static double INFLOW_PROPORTION = 1;

    public static void increment_current_time(){
        current_time ++;
    }

    public static int increment_and_retrieve_number_of_bikes(){
        return number_of_bikes++;
    }

    public static boolean[] generate_lights_list(int red, int green){
        boolean[] lights = new boolean[red + green];

        for(int i = 0; i < red; i++)
            lights[i] = false;
        for(int i = red; i < lights.length; i++)
            lights[i] = true;

        return lights;
    }

    public static void main(String[] args) throws Exception {
        Map<String, intersection_node> intsect_map = new HashMap<>();
        Map<String, destination_node> dest_map = new HashMap<>();
        Map<String, path> path_map = new HashMap<>();

        // add destinations
        add_destination(dest_map, "ANU");
        add_destination(dest_map, "ANU_2");

        // add intersections
        add_intersection(intsect_map, "McCaughey Masson"); // McCaughey 30 to 31
        add_intersection(intsect_map, "Masson path"); // Masson 26 to 38 or Masson 26 to
Watson
        add_intersection(intsect_map, "David path"); // David 39 to Condamine 54
        add_intersection(intsect_map, "Condamine path"); // Condamine 54 to McCaughey 30
        add_intersection(intsect_map, "Macarthur path"); // Macarthur 72 to David 39
        add_intersection(intsect_map, "Wattle path");// Wattle 31 to Macarthur 74
        add_intersection(intsect_map, "Goodwin path"); // Goodwin 42 to Wattle 42
        add_intersection(intsect_map, "Miller path"); // Miller 80 to Macarthur 74
        add_intersection(intsect_map, "Northbourne Wakefield"); // Wakefield to Watson
        add_intersection(intsect_map, "Mouat 29 path"); // Mouat 29 to Wakefield or Mouat
29 to Goodwin 42
        //add_intersection(intsect_map, "");

        // add paths
        add_path(path_map, "McCaughey 30 to 1"); // 450 ANU
        add_path(path_map, "Condamine 54 to McCaughey 30"); // 300 McCaughey Masson
        add_path(path_map, "Masson 26 to 38");//180 McCaughey Masson
        add_path(path_map, "Masson 26 to Watson"); // 600 ANU_2
        add_path(path_map, "Condamine 54 to Masson 24"); // 400 Masson path
        add_path(path_map, "David 39 to Condamine 54"); // 400 Condamine path
        add_path(path_map, "Macarthur 72 to David 39"); // 500 David path
        add_path(path_map, "Wattle 31 to Macarthur 74"); // 350 Macarthur path
        add_path(path_map, "Goodwin 42 to Wattle 42");// 550 Wattle path
        add_path(path_map, "Miller 80 to Macarthur 74"); // 1400 Macarthur path
        add_path(path_map, "Wakefield to Watson"); // 2200 ANU_2
```

```
        add_path(path_map, "Mouat 29 to Wakefield"); // 1800 Northbourne Wakefield
        add_path(path_map, "Mouat 29 to Goodwin 42"); // 950 Goodwin path

        // set intersections
        intsect_map.get("McCaughey Masson").set_path_of_intersection(path_map, new
String[]{"McCaughey 30 to 1"});
        intsect_map.get("Masson path").set_path_of_intersection(path_map, new
String[]{"Masson 26 to 38", "Masson 26 to Watson"});
        intsect_map.get("David path").set_path_of_intersection(path_map, new
String[]{"David 39 to Condamine 54"});
        intsect_map.get("Condamine path").set_path_of_intersection(path_map, new
String[]{"Condamine 54 to McCaughey 30"});
        intsect_map.get("Macarthur path").set_path_of_intersection(path_map, new
String[]{"Macarthur 72 to David 39"});
        intsect_map.get("Wattle path").set_path_of_intersection(path_map, new
String[]{"Wattle 31 to Macarthur 74"});
        intsect_map.get("Goodwin path").set_path_of_intersection(path_map, new
String[]{"Goodwin 42 to Wattle 42"});
        intsect_map.get("Miller path").set_path_of_intersection(path_map, new
String[]{"Miller 80 to Macarthur 74"});
        intsect_map.get("Northbourne Wakefield").set_path_of_intersection(path_map, new
String[]{"Wakefield to Watson"});
        intsect_map.get("Mouat 29 path").set_path_of_intersection(path_map, new
String[]{"Mouat 29 to Wakefield", "Mouat 29 to Goodwin 42"});

        // set paths
        path_map.get("McCaughey 30 to 1").set_path_data(intsect_map, dest_map, 450,
"ANU");
        path_map.get("Condamine 54 to McCaughey 30").set_path_data(intsect_map, dest_map,
300, "McCaughey Masson");
        path_map.get("Masson 26 to 38").set_path_data(intsect_map, dest_map, 180,
"McCaughey Masson");
        path_map.get("Masson 26 to Watson").set_path_data(intsect_map, dest_map, 600,
"ANU_2");
        path_map.get("Condamine 54 to Masson 24").set_path_data(intsect_map, dest_map,
400, "Masson path");
        path_map.get("David 39 to Condamine 54").set_path_data(intsect_map, dest_map,
400, "Condamine path");
        path_map.get("Macarthur 72 to David 39").set_path_data(intsect_map, dest_map,
500, "David path");
        path_map.get("Wattle 31 to Macarthur 74").set_path_data(intsect_map, dest_map,
350, "Macarthur path");
        path_map.get("Goodwin 42 to Wattle 42").set_path_data(intsect_map, dest_map, 550,
"Wattle path");
        path_map.get("Miller 80 to Macarthur 74").set_path_data(intsect_map, dest_map,
1400, "Macarthur path");
        path_map.get("Wakefield to
Watson").set_path_data(intsect_map,dest_map,2200,"ANU_2");
        path_map.get("Mouat 29 to
Wakefield").set_path_data(intsect_map,dest_map,1800,"Northbourne Wakefield");
        path_map.get("Mouat 29 to Goodwin 42").set_path_data(intsect_map, dest_map, 950,
"Goodwin path");

        // set exact timings for relevant intersections
        intsect_map.get("McCaughey Masson").set_timing_data(0.5, generate_lights_list(60,
30), constants.STD_OUTFLOW_RATE);
        intsect_map.get("Masson path").set_timing_data(0.5, generate_lights_list(60, 30),
constants.STD_OUTFLOW_RATE);
        intsect_map.get("David path").set_timing_data(0.7, generate_lights_list(60, 30),
constants.STD_OUTFLOW_RATE);
        intsect_map.get("Condamine path").set_timing_data(0.3, generate_lights_list(60,
30), constants.STD_OUTFLOW_RATE);
        intsect_map.get("Macarthur path").set_timing_data(1, generate_lights_list(60,
30), constants.STD_OUTFLOW_RATE);
        intsect_map.get("Wattle path").set_timing_data(0.5, generate_lights_list(60, 30),
constants.STD_OUTFLOW_RATE);
        intsect_map.get("Goodwin path").set_timing_data(0.5, generate_lights_list(60,
30), constants.STD_OUTFLOW_RATE);
        intsect_map.get("Miller path").set_timing_data(2, generate_lights_list(60, 30),
constants.STD_OUTFLOW_RATE);
        intsect_map.get("Northbourne Wakefield").set_timing_data(1,
```

```
generate_lights_list(60, 30), constants.STD_OUTFLOW_RATE);
        intsect_map.get("Mouat 29 path").set_timing_data(2, generate_lights_list(60, 30),
constants.STD_OUTFLOW_RATE);

        ArrayList<connector> list_of_connectors = new ArrayList<>();
        list_of_connectors.addAll(dest_map.values());
        list_of_connectors.addAll(intsect_map.values());
        list_of_connectors.addAll(path_map.values());

        long start = System.nanoTime();
        // main loop
        for(int i = 0; i < constants.SIMULATION_LENGTH; i+=constants.UNIT_TIME){
            list_of_connectors.forEach(temp -> temp.simulate());
            increment_current_time();
            if( i % 100000 == 0)
                System.out.println("Simulation is at " + i + "s");
        }

        long end = System.nanoTime();

        System.out.println("\n\nBikes that have arrived at ANU:");
        //dest_map.get("ANU").bikes_at_node.forEach(temp_bike ->
System.out.println(temp_bike));
        System.out.println("Number of bikes at end:" +
(dest_map.get("ANU_2").bikes_at_node.size()));
        System.out.println("\n\nBikes that have arrived at ANU)2:");
        //dest_map.get("ANU").bikes_at_node.forEach(temp_bike ->
System.out.println(temp_bike));
        System.out.println("Number of bikes at end:" +
(dest_map.get("ANU").bikes_at_node.size()));

        System.out.println("\nTime taken for simulation:" + ((end - start)/ 1000000) +
"ms");
        System.out.println("Number of simulated seconds:" + constants.SIMULATION_LENGTH);


        // calculate average waiting tmes for each node
        Map<String, Double> average_times = new HashMap<>();
        Map<String, Integer> number_of_occurances = new HashMap<>();
        Map<String, Double> average_distance_travelled = new HashMap<>();

        list_of_intersection_names.forEach(temp_intersection -> {

            average_times.put(temp_intersection,0.0);
            number_of_occurances.put(temp_intersection,0);
            average_distance_travelled.put(temp_intersection,0.0);

            dest_map.values().forEach(
                    temp_dest -> temp_dest.bikes_at_node.forEach(
                            temp_bike -> {
                                if (temp_bike.origin.equals(temp_intersection)) {
                                    number_of_occurances.put(temp_intersection,
number_of_occurances.get(temp_intersection) + 1);
                                    average_times.put(temp_intersection,
average_times.get(temp_intersection) + (double) temp_bike.travel_time);
                                    average_distance_travelled.put(temp_intersection,
average_distance_travelled.get(temp_intersection) + temp_bike.distance_travelled);
                                }
                            })
            );

            average_times.put(temp_intersection, average_times.get(temp_intersection) /
number_of_occurances.get(temp_intersection));

average_distance_travelled.put(temp_intersection,average_distance_travelled.get(temp_inte
rsection) / number_of_occurances.get(temp_intersection));

            System.out.println("For intersection:" + temp_intersection + ", \nnumber of
arrivals:" + number_of_occurances.get(temp_intersection) +
                    ", \naverage travel time:" + (average_times.get(temp_intersection) /
60) + " minutes" + ", \naverage distance travelled:" +
```

```
                    (average_distance_travelled.get(temp_intersection) / 1000) + " km" +
            "\n average speed of travel:" +
(average_distance_travelled.get(temp_intersection) /
average_times.get(temp_intersection)) + "\n\n\n");

        });


    }

    public static void add_intersection(Map<String, intersection_node> intsect_map,
String id) throws Exception {

        if(id == null)
            throw(new Exception("no identifier in intersection node"));

        list_of_intersection_names.add(id);

        intsect_map.put(id,new intersection_node(id));
    }

    public static void add_destination(Map<String, destination_node> dest_map, String id)
throws Exception {

        if(id == null)
            throw(new Exception("no identifier in destination node"));

        dest_map.put(id,new destination_node(id));
    }

    public static void add_path(Map<String, path> path_map, String id) throws Exception {

        if(id == null)
            throw(new Exception("no identifier in path"));

        path_map.put(id, new path(id));
    }

}



import java.util.Random;

/**
 * Created by Daniel on 10/10/2015.
 */
public class bicycle {
    int start, end, travel_time;
    double distance_to_node, speed, distance_travelled;
    String origin, destination;
    int bike_id;

    public String toString(){
        return "bike:" + bike_id + ", origin:" + origin + ", travel_time:" + travel_time
+ ", distance travelled:" + distance_travelled + ", distance to node:" +
distance_to_node;
    }

    public bicycle(int time_entered, String node_id, int bike_id){

        this.start = time_entered;
        this.origin = node_id;
        distance_travelled = 0;
        this.bike_id = bike_id;
        // nextDouble used instead of nextGaussian because Gaussian caused it to run 53
times slower compared to
        // assigning the average bike speed whereas nextDouble only caused it to run 2
times slower
        speed = (constants.rnd.nextDouble() * 2 - 1) * constants.BIKE_STD_SPEED +
constants.BIKE_AVG_SPEED;
```

```
    }

    public void set_end(int end_time, String destination){
        this.end = end_time;
        this.travel_time = end - start;
        this.destination = destination;
    }

    public void set_distance_to(double dist){
        distance_to_node = dist;
    }

    public boolean move_bike_forwards(double pos_of_bike_in_front, double
speed_of_bike_in_front){
        double old_dist = distance_to_node;
        double new_dist = distance_to_node - speed;

        if(speed_of_bike_in_front > (this.speed -
constants.BIKE_SPEED_OVERTAKING_MAGNITUDE)) {
            if (pos_of_bike_in_front + constants.SAFETY_DIST < distance_to_node) {
                if (pos_of_bike_in_front + constants.SAFETY_DIST > new_dist) {
                    this.distance_to_node = pos_of_bike_in_front + constants.SAFETY_DIST;
                } else {
                    this.distance_to_node = new_dist;
                }
            }
        } else {
            this.distance_to_node = new_dist;
        }

        if(distance_to_node <= 0) {
            this.distance_travelled += old_dist;
            return true;
        }

        this.distance_travelled += old_dist - this.distance_to_node;
        return false;
    }

}



import java.util.ArrayList;

/**
 * Created by Daniel on 10/10/2015.
 */
public abstract class connector {
    ArrayList<bicycle> bikes_at_node = new ArrayList<>();
    String identifier;

    public connector(String id){
       identifier = id;
    }

    public void add(bicycle bike){
        bikes_at_node.add(bike);
    }
    public void simulate(){
    }
}



/**
 * Created by Daniel on 10/10/2015.
 */
public class destination_node extends connector {

    public destination_node(String id) {
```

```
        super(id);
    }

    @Override
    public void add(bicycle bike){
        bike.set_end(base.current_time,identifier);
        super.add(bike);
    }

    public String toString(){
        return "(destination_node): id:" + identifier;
    }
}



import java.util.ArrayList;
import java.util.Map;

/**
 * Created by Daniel on 10/10/2015.
 */
public class path extends connector{

    double length_of_path;
    connector output_connector;

    public path(String id, double dist, connector output_connector){
        super(id);
        length_of_path = dist;
        bikes_at_node = new ArrayList<>();
        this.output_connector = output_connector;
    }

    public String toString(){
        return "(path): id:" + identifier + ", length:" + length_of_path + ", Connects
to:" + output_connector;
    }

    public path(String id){
        super(id);
    }

    public void set_path_data(Map<String, intersection_node> insect_map, Map<String,
destination_node> dest_map, double dist, String output_connector) throws Exception {
        connector temp_connector = insect_map.get(output_connector);
        if(temp_connector == null) {

            temp_connector = dest_map.get(output_connector);

            if(temp_connector == null)
                throw (new Exception("nodes inputted in incorrect order, output node:" +
output_connector + "not found for node:" + identifier));
        }


        length_of_path = dist;
        bikes_at_node = new ArrayList<>();
        this.output_connector = temp_connector;
    }

    @Override
    public void add(bicycle bike){
        //System.out.println("Adding to path:" + bike);
        bike.set_distance_to(length_of_path);
        bikes_at_node.add(bike);
    }
    @Override
    public void simulate(){

        /*
```

```
        System.out.println("contents of bike path:");
        bikes_at_node.forEach(temp -> System.out.println(temp));

        System.out.println();*/

        double pos_of_bike_in_front = -9999;
        double previous_bike_speed = 99;

        for(bicycle temp_bike : bikes_at_node){
            temp_bike.move_bike_forwards(pos_of_bike_in_front,previous_bike_speed);
            pos_of_bike_in_front = temp_bike.distance_to_node;

            // reset to - 9999 to ensure that bikes can queue at intersections
            if(pos_of_bike_in_front <= 0)
                pos_of_bike_in_front = - 9999;

        }

        // pass along all bikes that have reach the end

        while((bikes_at_node.size() > 0) && bikes_at_node.get(0).distance_to_node < 0){
            output_connector.add(bikes_at_node.get(0));
            bikes_at_node.remove(0);
        }



    }
}



import java.util.ArrayList;
import java.util.Map;
import java.util.Random;
import org.junit.Test;

import static org.junit.Assert.assertTrue;

/**
 * Created by Daniel on 10/10/2015.
 */
public class intersection_node extends connector {
    double pput;
    boolean[] lights;
    connector[] output_paths;
    int outflow_rate;
    int current_light_timing;
    int number_of_output_paths;

    public intersection_node(String id, double pput, boolean[] light_timings, connector[]
outputs, int outflow_rate){
        super(id);

        //System.out.println("Creating intersection with id:" + id);
        this.outflow_rate = outflow_rate;
        this.pput = pput * base.INFLOW_PROPORTION;
        this.lights = light_timings;
        this.output_paths = outputs;
        this.number_of_output_paths = outputs.length;
        this.current_light_timing = 0;
        this.bikes_at_node = new ArrayList<>();

        assertTrue("in intersection_node, connector outputs has null", outputs.length > 0
&& outputs[0] != null);
    }

    public intersection_node(String id){
        super(id);
    }
```

```
    public void set_path_of_intersection(Map<String, path> connectors_map, String[]
outputs) throws Exception {
        //set_intersection_data(connectors_map, 0, new boolean[]{true}, outputs,
constants.STD_OUTFLOW_RATE);

        connector[] output_connectors = new connector[outputs.length];

        for (int i = 0; i < outputs.length; i ++){
            output_connectors[i] = connectors_map.get(outputs[i]);
            if(output_connectors[i] == null)
                throw(new Exception("nodes inputted in incorrect order, output node:" +
outputs[i] + " not found for node:" + identifier));
        }

        this.outflow_rate = constants.STD_OUTFLOW_RATE;
        this.pput = 0;
        this.lights = new boolean[]{true};
        this.output_paths = output_connectors;
        this.number_of_output_paths = outputs.length;
        this.current_light_timing = 0;
        this.bikes_at_node = new ArrayList<>();

    }

    public void set_timing_data(double pput, boolean[] light_timings, int outflow_rate){
        this.outflow_rate = outflow_rate;
        this.pput = pput * base.INFLOW_PROPORTION;
        this.lights = light_timings;
        this.current_light_timing = 0;
        this.bikes_at_node = new ArrayList<>();
    }


    private void add_bikes_from_inflow(){

        int number_of_new_bikes = poisson(pput);

        for(int i = 0; i < number_of_new_bikes; i++){
            bikes_at_node.add(new
bicycle(base.current_time,identifier,base.increment_and_retrieve_number_of_bikes()));
        }
    }

    private void allow_through_intersection_if_green(){
        if(this.lights[current_light_timing]){
            for(int i = 0; i < this.outflow_rate; i++){
                if(this.bikes_at_node.size() > 0){

output_paths[constants.rnd.nextInt(this.number_of_output_paths)].add(this.bikes_at_node.g
et(0));
                    this.bikes_at_node.remove(0);
                }else
                    break;
            }
        }
    }

    @Override
    public void simulate(){
        add_bikes_from_inflow();
        allow_through_intersection_if_green();


        current_light_timing ++;
        if(current_light_timing >= lights.length)
            current_light_timing = 0;
    }


    private static int poisson(double mean) {
        int r = 0;
```

```
        Random rnd = new Random();
        double a = rnd.nextDouble();
        double p = Math.exp(-mean);

        while (a > p) {
            r++;
            a = a - p;
            p = p * mean / r;
        }
        return r;
    }
}
import java.util.Random;

/**
 * Created by Daniel on 10/10/2015.
 */
public class constants {

    public static Random rnd = new Random();

    public final static int SIMULATION_LENGTH = 3600;
    public final static int UNIT_TIME = 1;
    public final static double SAFETY_DIST = 2;

    public final static int STD_OUTFLOW_RATE = 1;
    public final static double BIKE_AVG_SPEED = 4.16;
    public final static double BIKE_STD_SPEED = 1;
    public final static double BIKE_SPEED_OVERTAKING_MAGNITUDE = 1;
}
```