KING'S College LONDON

University of London

Agent-based simulation of bicycle theft in London

Final Project Report

Author: Emily Liu

Supervisor: Dr Simon Miles

Student ID: 1744993

May 6, 2020

Abstract

This project aims to create an agent-based model of bicycle theft within London. More specifically, it focusses on deterring bicycle theft within the metropolitan area, by modelling bicycle thefts it may be possible to find interventions to deter thefts of bicycles in the future. The model aims to simulate thieves in a compelling way using agents as actors that have the ability to learn and share knowledge with other thieves.

The goal of the project is to create a simple piece of software that could be used by enforcement agencies with little technical knowledge.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Emily Liu

May 6, 2020

Acknowledgements

I would like to thank my supervisor, Dr Simon Miles, for offering his support and constructive suggestions throughout the duration of the project.

Thank you to Amazon UK who have supported me with my 3 years at King's College London studying for my undergraduate degree. Without their generosity this would not have been possible.

Contents

Chapter 1

Introduction

Exercise is generally accepted to have a positive effect on cardiovascular health, advice from the NHS states that "regular cycling can reduce the risk of chronic illnesses such as heart disease and stroke"[1]. One of the easiest ways to fit cycling into your routine is to use it as a means of transport. Cycling in London has grown year on year with numbers doubling in the capital since 2000 [2], and between 2012 and 2017 featuring an overall 24 per cent increase [3]. In 2018 Transport for London (TfL) estimated that 730,000 trips are made a day by bicycle [4].

Bicycle theft is viewed as a low risk, high reward crime with many 'CRAVED' features, for example they are widely available, often expensive, easy to use and there are numerous channels through which they can be sold [5]. CRAVED features are those that are Concealable, Removable, Available, Valuable, Enjoyable and Disposable [6].

In a report commissioned by the Mayor of London, it was noted that people are generally put off cycling due to the fear of having their bicycle stolen [3]. To reduce the number of bicycle thefts, a possible intervention could be to plant 'bait bikes' designed to detect thieves in the act. However interventions like this are often expensive and the effectiveness is not known prior to commencement.

An agent-based model (ABM) is a computational model where actions and interactions of agents can be simulated, this provides a unique way of simulating hypothesised societal effects. The use of ABMs to model bicycle theft will allow different crime deterrent policies to be tested prior to implementation in the real world to understand potential effects.

1.1 Project aims

The fundamental objective of the project is to develop a simulation of bicycle theft using ABM, focussing on interventions to deter offenders committing theft of bicycles.

Moreover, the ABM will act as a useful tool to evaluate the effectiveness of any interventions prior to real world utilization. An agent's decisions will be strongly influenced by its spacial and social network, a thief is more likely to offend if others within their network also commit theft.

Chapter 2

Background

2.1 Cycling in London and bicycle theft

London has seen a 'cycle revolution' since the Livingston era in 2000. Cycling has been widely recognised as providing benefits to the health of individuals and to society, as a sustainable mode of transport in urban environments. The creation of Cycleways -routes that link communities, businesses and destinations across London in one cycle network [7] - has made cycling safer, encouraging more people to cycle. According to estimates by TfL in 2018 730,000 trips are made by bicycle daily in London [4] compared with only 270,000 trips in 2000 [8]. With the Mayor of London's Cycling action plan, which is primarily focused on expanding cycle routes to connect more people, this number is only going to continue growing.

In recent times, with the declared climate emergency and increased congestion within London the Mayor's plan is to increase the already growing number of journeys made by bicycle to 1.3 million every day by 2024, to be supported by $2.3bn funding[3] in the hopes of transforming London into a cycling city.

Between September 2016 and September 2019 there were 60,552 reported bicycle thefts within the the Metropolitan Police's jurisdiction. It is important to note that the realistic number of bicycle thefts is most likely a lot higher, research has found that only one in every five bicycle theft is reported [5]. It is clear that an intervention will be necessary sooner rather than later as the number of bicycles in London is projected to continually grow. With more bicycles on the streets of London, there is a greater opportunity for offenders to partake in bicycle theft.

Analysis of reported thefts has shown that bicycle theft continues to rise and this is not

limited to London, it is throughout England and Wales, even while there is a decline in overall offending rate. The increase has been linked with the ease of liquidating bicycles whether that be whole or as parts and also it's high utility for door-to-door transportation. Between 2013 and 2016 figures show that the perpetrators of 96 percent of reported bicycle thefts in London were unidentified [9].

2.2 Criminology

There have been several crime opportunity theories associated with theft, one of those being the Routine Activities theory (RCT) [10] which focuses specifically on the characteristics of crime instead of the offenders. RCT is often used in urban settings, the theory states that for a crime to occur, three elements must be present: an available and suitable target; a motivated offender; and no authority figure to prevent the crime from happening [11]. It is through daily routine activity that the aforementioned elements interact and why places that attract high population present opportunities for crime [12]. For instance, the large number of bicycles that are left near or at railway stations for hours at a time are bound to attract thieves because railways provide intermediate transportation for many cyclists. The idea is that if a bicycle is not protected enough, if it is worth the reward, crime will occur- if there is opportunity, there is crime.

Rational choice theory (RCT) describes humans as rational actors who calculate the rationality of committing crime using means and ends, costs and benefits [13]. Higher priced bicycles are more likely to be targeted than lower priced bicycles as a result of offenders selecting objects that maximise their expected utility. Sometimes theft does not take place even when offenders interact with the target, this is because the perceived risk is simply too high [9]. RCT aides situational crime prevention, specifically opportunity reducing measures that can be directed at opportunistic crime- in this case bicycle theft.

2.3 Agent-based modelling

An Agent-Based Model (ABM) is a computer based simulation which comprises of a collection of autonomous decision-making entities called agents, these agents are given rules for their interaction with other agents or entities within the confined environment which is being modelled. It is through these rules and interactions that patterns emerge [14]. Agents can represent individuals, groups or objects. Through the model iterations, each agent is able to assess its

situation and make an informed decision as to its next steps via a set of probabilistic rules.

An ABM comprises of two parts,the first being the system and the component being the second- this being the environment and the agent. The relationship between the system and the component element is key to ABM's. The system defines the environment that the agents are bound by and interact within and also the properties for the whole model. Agents receive input from this environment and this in turn affects the way agents function in terms of their set of rules. With this, agents have the ability to gather information and perform actions in response to the the system's input.

It can be said that the environment and the agents operate on a need-to-know basis where the environment has no knowledge of specific agent behaviours and is not able to calculate the outcome of the model. The environment can however show the overall state of the system, this is by amassing individual agents' properties at a time tick in the model. Individual agents do not have knowledge of the whole system but instead rely on the local environment to make decisions by gathering information about the local environment and responding depending on their input, through interactions like this emergent phenomena can be captured.

The utilisation of agent-based modelling to crime is a relatively new concept, first appearing in the early to mid-2000s [15]. These models were initially used in formalizing theories and more recently used to predict crime in specific places and to understand patterns of crime for example in robbery. Within criminology, ABM's application to investigate situational crime prevention techniques has great potential. It could be used to provide 'what if' analyses which can include the effects that crime reduction plans might have before their implication. [16].

Existing agent-based models in criminology

ABMs have only recently began to be realised in criminology with early models being relatively simple and as a result, studies of direct relevant to crime deterrence are scarce. Some authors have implemented agent-based models of other crime types, for example cyber crime and pick-pocketing. The most relevant work is that of (Malleson et al, 2010)[17] who build an agent-based model of burglary, specifically crime reduction through simulation using the PECS framework focusing on a behavioural model. This model purely focuses on offender behaviour and their need to commit burglary for financial gain.

In a report published in the Journal of Quantitative Criminology, it was concluded that at present, "the lack of model detail reported in publications make it difficult to assess where sufficient evidence exists to support the development of models" [15].

The model proposed in this research will focus predominantly on the behaviour of victims and how victims can utilise preventative measures to reduce the chance of theft occurring through the opportunity theory framework.

2.3.1 Types of simulation systems for agent-based modelling

There are generally two types of simulation systems that can be used to develop agent-based models, these being either toolkits or software. Toolkits are modelling systems that provide a framework for the design of ABMs that provide libraries of software functionality consisting of pre-defined modules, routines and functions specifically designed for ABM. The use of toolkits can make it easier for modellers when faced with programming parts that are content-specific for example a Graphical User Interface and visualisation of the model. As the toolkits have been created and optimised by professional developers the reliability of the model and the efficiency of the algorithm is increased when thinking of the complex parts in models.

However, there are limitations of using toolkits in the development of agent-based models. The modeller will be required to understand how to design and implement models in the toolkit, which can take a considerable amount of effort. Whilst most toolkits do provide demonstration models they can be difficult to understand and also difficult to apply to another purpose. Additional tools may be available from the wider user community but once again, the function may not be applicable for the chosen model purpose. Another issue is the lack of compatibility when dealing with model functions as there can be a lot of conflicts in the later stages of the development process.

As mentioned previously, software is also available for the development of agent-based models- software is useful for rapid development of prototype models. Modellers choosing to use software are typically restricted to the software's design framework, for example some software will have a limited environment to model within or limited number of agents available at any one time. Although there are drawbacks of using software other toolkits, ABM software is sufficient for the aims of the project as stipulated in 1.1 and is what will be focused upon in this paper.

MASON

MASON (Multi Agent Simulation Of Neighbourhood) was developed at George Mason University's Evolutionary Computation Laboratory with the Center for Social Complexity in 2003. MASON is rather complex in comparison with other available open-source software, it is suited

for simulations that require long computations like multi-agent ABMs. Unfortunately technical documentation for MASON is limited and the user group in comparison with other software is somewhat limited. With the lack of debugging and lack of terminal window for users, it can be concluded that it is not user-friendly when compared with others and is most suitable for projects requiring fast computation.

Repast

Repast, the Recursive Porous Agent Simulation Toolkit, was originally developed at the University of Chicago but is now maintained by the Argonne National Laboratory. The tool originally catered for three programming languages from which models could be implemented: Java (RepastJ), Microsoft.Net(Repast.NET) and Python (RepastPy). This was all outmoded by Repast Simphony which provides all the functionality of previous Repast's but exclusively in Java. Although Repast is one of the fastest platforms currently available and feature advanced modelling capabilities, it is also one of the most complex mainly focused towards programming experts with a non user-friendly environment.

AgentSheets

AgentSheets is a proprietary modelling system that allows individuals with limited prior programming experience to effortlessly develop agent-based models. The ease of development of models is due to all development being through a GUI rather than through traditional programming methods. A number of models are available from the system creators through their system website which can be adapted to ones own modelling needs. It must however be noted that models created with AgentSheets are limited in the complexity of the environment, agent behaviours and interactions within the environment.

NetLogo

NetLogo was originally developed by Northwestern University, the Centre for Connected Learning and Computer-Based Modelling as a variant of StarLogo- which is another modelling system developed by Massachusetts Institute of Technology (MIT)- due to its lack of support for Macintosh operating systems. NetLogo is a multi-agent programmable system that allows users to access a large library of sample models that can be adapted to suit different models. The great benefit of NetLogo is the functionality of being able to import image files which can then be used to define the environment that the agents can act within, making it a very effective tool

when developing spatial models. It is a tried and tested software that has been widely used in social sciences and biology, with extensive documentation and models available. For complex models modellers may face limitations regarding execution speed and being constrained with the NetLogo language.

2.4 Machine learning

This section will focus upon machine learning, specifically reinforcement learning that is particularly relevant to this paper.

Reinforcement learning relates to the decision making process of agents within an environment where agent decisions have rewards associated with them - Markov decision process [18]. It is through these rewards that agents are able make value based judgements about their actions by providing a framework for calculating the utility of decisions. Agents aim to take actions that offer the greatest return of reward. By learning, agents form their own expectations of value purely based on previous experiences.

When utilising reinforcement learning, one has to consider three important factors. The first issue is long term reward versus short term reward, this is known as the discount factor. Agents are able to judge actions not just on the expected value of that action but on the expected value of future actions following them. By changing how an agent uses this information, or specifically changing the discount factor in the reinforcement learning, can have significant effects on the simulation. The second issue is the balance between exploration versus exploitation. Exploration is where an agent takes actions that do not offer the greatest reward out of all available actions whereas exploitation is where the agent follows the action that returns the greatest reward. Exploring offers agents the possibility of discovering higher value actions than the ones currently in their knowledge bank. An agent that specifically follows the exploitation route would never explore and would be very limited. A compromise is for agents to use the expected utility of actions to find the probability distribution, the agents will choose high value actions for the majority of the time but will occasionally explore lower value actions to explore.

The third issue with reinforcement learning is the learning rate, this is used to determine how new information affects the knowledge held by an agent. A low learning rate, a value near to 0, results in new information having a negligible effect on an agent's knowledge meaning they do not learn anything. A high learning rate, a value near to 1, results in a great amount of learning. The agent would in effect overwrite old knowledge and use the new knowledge in future instances. Whilst a maximal learning rate would be ideal, in the real world a learning

rate is typically between the two extremes meaning agents learn gradually.

2.5 Concepts

An agent's psychology can be represented by a set of concepts that agents can hold and attempt to spread to other agents within its social and spatial network. These concepts can consequently impact decision making of agents, for example if an agent has not been caught by police for theft previously, the agent would be likely to continue to partake in the action of thieving. As these concepts spread across the spatial and social network, they are strengthened as agents spread concepts to other agents, these concepts can affect an agent's decision making and thus their psychology.

In the field of criminal offending, adult offenders have a proven re-offending rate of 28% - as measured by the Ministry of Justice. A proven re-offence is defined as any offence committed in one-year follow-up from when the offender was initially released from custody, received a non-custodial conviction at court or received a caution [19]. Measuring the true re-offending rate is difficult, they will underestimate the true level because a proportion of crime is left unsolved or undetected in the first instance. Although the figure is not completely accurate, it is plausible to conclude that a substantial amount of offenders stopped their criminal activities after they were cautioned, convicted or in custody.

To follow on, a study by a group in the Netherlands found when examining whether friends' involvement in crime has an influence on people's own involvement found that there was indeed a link. The study indicated that by learning other people's behaviour by observing and imitating peers it is highly likely that individuals if exposed to criminal behaviour would also be involved in crime [20]. This also works conversely, for example if a person involved in crime is caught or ceases their criminal activities it is likely that their friends change their perception of the costs and benefits of crime.

The spread of a concept across a network is known as influence spread or influence diffusion, with the two most prevalent models in computer science being the *independent cascade model* and the *linear threshold model*.

Independent Cascade Model

The Independent Cascade (IC) model can be thought of as the opportunity for an agent to spread concepts associated with each meeting [21]. The model starts with an initial set of active nodes A_0 that hold concepts. When a node v is active at time t, it is possible to activate

an inactive neighbour w, with the probability $p_{\mathrm{v,w}}$ that v will activate w. If v activates w, then w becomes active from the time *t+1*. If v fails to activate w, v gets no further attempts to activate w. This process repeats until there are no more nodes that can activate new nodes [22].

IC models do not consider spread in the network where the agents may spread concepts more than once for example if a bicycle thief tells several other thieves of his negative experience and it is for this reason that IC modelling is not accurate whereby concepts are spread multiple times, as is the case in real-world settings.

Linear Threshold Model

Linear Threshold models is more realistic in that it accounts for multiple spread by agents and the weighting of concepts. When more people hold a concept it is more likely that others nearby will also adopt the concept.

The Linear Threshold (LT) model can be thought of as when an agent is influenced by neighbouring agents to varying weights, or successes, each agent has a minimum influence threshold whereby if the agent has been influenced sufficiently it will adopt the concept. Each node v has a threshold ϑ_{v} that is chosen at random from the interval [0,1]. A node v is influenced by each neighbour w with a weight $b_{\mathrm{v,w}}$ assigned to the network link - the total sum of all network links connected to v cannot be greater than 1. When starting with an initial set of active nodes A_0 that hold the concept at time t, we can say that all nodes that are active at t-1 continue to be active. Any inactive node v will become active when:

$$\sum_{w\ active\ neighbour\ of\ v} b_v, w \geq \theta_v$$

So, once the total weight of active neighbours is greater than the threshold ϑ_{v} v will become active [22].

Maximising spread

In deterring crime it may be preferable for the spread of concepts within a network to be maximised for example by targeting known offenders who are well connected with other offender or have influence over others. To maximise it, it is best to select the set of agents with the greatest influence. The link between concepts can be modelled by looking at the relationship between them, a concept c_1 can either support, inhibit or not affect the adoption c_2. Relationships can

then be used to maximise spread of a concept. With the introduction of a second concept that checks for the first concept, it is possible to maximise adoption of concepts with the use of what is called a boosting concept. A boosting concept improves its probability of influencing others which in turn makes nodes react to an activation faster [23].

Modelling the real-world as proposed presents difficulties in that numerically defining how concepts affect the adoption of further concepts is not simple. Using the aforementioned boosting concept, it is possible to boost a target concept- concepts can then be thought of as part of an agent's decision making. Spreading concepts that boost the target concept strengthens desirable behaviours.

2.6 Conclusion

Several measures have been proposed in the past to reduce bicycle theft, such as better parking facilities and the installation of CCTV's in high risk areas. These measures have been said to have reduced crime levels however bicycle theft has not been studied previously so that cannot be said for certain. Agent-based modelling provides a useful tool to explore bicycle theft in detail, this provides the basis for the proposed model which will explore bicycle thefts with an emphasis on deterrence through spreading concepts to both thieves and bicycle owners.

Chapter 3

Specification

3.1 Brief

The primary goal of this project is to create a software that enables a user without prior programming knowledge to run a simulation of bicycle thefts in London. The main objective is to evaluate deterrent interventions, these interventions will seek to reduce the number of thefts within the environment through agent behaviours.

Whilst this simulation will focus on an area of the City of London, it will be easily adaptable to other areas. The intended users are senior figures within Police forces who will be able to interact with the model to analyse whether certain interventions are viable and may be put into practise with officers on the street.

3.2 Requirements

The software will include the following key parts:

- The simulation,
- The agents,
- The environment.
- NetLogo that will serve as the graphical user interface.

3.2.1 Simulation requirements

- The simulation must run for at least 1 year.

- Agents come and go throughout the day.
- Due to the uniqueness of the City of London whereby the resident population is 7375 with a 56 fold increase in workday population, according to data from the 2011 Census [24], the simulation will ignore weekends and holidays where the footfall is dramatically reduced.
- The simulation must allow the user to call intervention functions to manipulate the simulation world (agents and environment).
- The simulation must return statistics at the end.

3.2.2 Agent requirements

The model will be populated by 3 types of agents:

1. Bicycle
2. Thief
3. Police

- Agents in the simulation must be free to come and go.
- Agents must observe the environment and react to any changes that occur accordingly.
- A bicycle must either be in a locked or stolen state.
- A bicycle must be able to hold concepts.
- A bicycle must be able to spread concepts to other bicycles.
- Thieves must either be active or inactive.
- Thieves are able to steal bicycles.
- Thieves attempt to avoid Police.
- A thief must be able to spread concepts to other thieves.
- Police must either be active or inactive.
- Police must be able to spread concepts to both bicycles and thieves.

3.2.3 Environment requirements

- The environment must represent a real-world physical area within London.
- The environment must support the definition of properties and through-way.
- The number of agents within the environment must be adaptable by the user.
- Bicycle and thief agents must not be able to enter buildings that will be clearly defined in the simulation.
- The environment should be as realistic as possible.

3.2.4 NetLogo requirements

As previously stated in 2.3.1 there are many simulation systems specifically designed for creating agent-based models like the one outlined in this project. These systems make the process of implementing models much simpler and often come with demonstration models that can be adapted. After consideration of the requirements listed above, NetLogo was chosen. Although it somewhat limits modelling when used in complex models, for the requirements of this project, it will suffice.

The model will be constructed in the NetLogo environment due to its relative ease of use when compared with other systems.

- The user must be able to setup, run and stop the model.
- The simulation within NetLogo must create some agents within the environment and interventions to be used in the model.
- NetLogo must run the simulation.
- NetLogo must offer variable changes.
- NetLogo must create some graphs to describe the simulation run.

3.3 Specification

3.3.1 Environment specification

When modelling the movement of humans within the environment the environment will be configured to a 2-dimensional view of the environment with the use of a detailed map. The

environment must provide enough detail to represent the real world but also allow the model to run at a feasible speed with many hundreds of agents simultaneously interacting with one another.

The model environment will be be from a simple PNG image which acts as the map. Each pixel of the image represents one grid space and the colour of each pixel will determine the properties of that grid. Agents are contained within the environment as set out by the image and can only access areas that are not deemed as property.

3.3.2 Agent specification

Actions and Utility

When modelling agent movement, it is best to model them realistically but also in an efficient manner. Agents will be able to move in four directions between grid cells: north, east, south and west. The continuous structure of the environment will show the agents gradually move between points but in reality, agent actions are purely established upon the grid-space they currently occupy and the cells in its immediate vicinity.

With the intention of learning in agents, the agents must be able to determine the utility of an action that they are have taken. To accomplish this, each action must have some sort of reward associated with it, this may be be a static or dynamic reward.

Learning

Learning in agents will take the form of Q-learning in the model with the use of the Q-learning reinforcement learning algorithm. An important factor to account upon is the learning rate and discount factor, these two variables offer great customisability when experimenting with the simulation. When Q-learning is used in modelling a Q-value is used to store the utility of each action.

Agent's will learn from one another through the spread of concepts, through the transfer of Q-values. They will then use their knowledge as expressed in the form of Q-values to attach probabilities to each action. With it being probability based, this will mean that agent's do not take a fully exploitative route but will sometimes choose exploitative actions.

Chapter 4

Design

The design of the software is stipulated with three categories: the design of the agents in the model, the simulation design and finally, the interface design.

4.1 Agents

There are four types of agents within Netlogo:

1. Turtles,
    - These are agents that move around the NetLogo world.
    - Bicycles, thieves and police will act as turtles.
2. Patches,
    - The world is 2D and is divided into a grid of patches, each patch acts as terrain which turtles can move.
3. Links
    - These are agents that connect two turtles.
4. And the observer.
    - The observer does not have a location like the other agents, it acts as an observer over all turtles and patches in the NetLogo world.

4.1.1 Police

Police agents have 3 available modes that can be used in conjunction with one another:

1. Stop suspected thieves
    - Police are able to stop suspects that they believe have stolen bicycles and question them. If after questioning it is found that they have not committed a bicycle theft, they are free to go.
    - If the suspect is a thief but when stopped is not in possession of a stolen bicycle, stopping them now may act as a deterrent in the future when they are thinking about committing a theft of a bicycle.
2. Offer bicycle locking advice
    - Police offer bicycle owners advice on how best to lock their bicycles.
    - This may be done with foot patrols whereby officers leave bicycle locking advice leaflets on bicycles, or if the owner is present face-to-face advice.
3. Recover and return bicycles that have been 'securely marked and registered' [25] .
    - If a bicycle that has been securely marked and registered and the owner can be traced, the bicycle can be returned the rightful owner.
    - If the bicycle is not registered to an owner, there is a very low chance it will ever be returned.

The modes listed above focuses on a small police unit that are dedicated towards bicycle theft and other related crimes. If a mode is not selected, police will not be actively patrolling or intervening and will not be present in the simulation as they will be focused on other crimes taking place.

4.1.2 Thief

Each thief will enter the environment with a crime-probability, this is the probability of the agent partaking in the theft of bicycles. The value will be a float between 0 and 1. If interventions are present and thieves gain concepts from police, these can be spread through the individual thief's spatial network when the concept is gathered or in its wider social network if applicable. Thieves gather concepts depending on the concept's weighting, if the concept is

greater than the set threshold then this concept will be held. It is hoped that by the spreading of negative concepts regarding theft, agents adapt their decision making.

4.1.3 Bicycle

Within the model, bicycle agents can be thought of as the common agent. Thieves see this agent as the 'goal' and police agents see this agent as the agent that they are trying to 'protect'. Their behaviour within the simulation is somewhat limited compared with other agents, as they have no real purpose in the model, they can be said to act as bystanders.

Desirability

This is a value between 0 and 1 that shows how desirable a bicycle is. A bicycle with a high desirability value will be more expensive and desirable than a bicycle with a low desirability value bicycle. A highly desirable bicycle will be resold for a higher price after it is successfully stolen. In some instances a bicycle that is more expensive may have a lower desirability than a cheaper bicycle due to its ease of liquidation. In other words a cheaper bicycle may be easier to sell on once it has been stolen and so it is desirable in that thieves can offload the stolen bicycle quickly reducing their chances of being caught with stolen goods.

Security

This is a value between 0 and 1 that shows how secure a bicycle is. A bicycle with a high security value is locked better or has security features which makes it more time resourceful to steal. As it is more time intensive to steal the bicycle, there is a higher risk of being caught in the act of stealing. This makes the cost of stealing the bicycle in terms of effort is greater as stealing a bicycle that has relatively low security is less costly in terms of effort.

Marked

This is a simple boolean that shows if the bicycle has been security marked. If a bicycle has been security marked, it may be possible for the bicycle to be returned if it is recovered. When a bicycle is stolen and the owner chooses to report the theft giving all identifying features and security marked number, police are able to check if the bicycle is stolen if they suspect it to be so. It is a criminal offence to handle and be in possession of stolen goods. This adds to the risk of stealing a bicycle if the bicycle is security marked.

4.2 Simulation design

4.2.1 Simulation overview

The model executes in 'ticks'. Each tick can be considered as one execution of a program loop. At each tick, methods or procedures as they are called in NetLogo are carried out.

Figure 4.1 shows how the simulation progresses throughout the duration of the simulation run. If interventions are active within the simulation, concepts are spread through agent networks and this in turn affects the decision process in which agents act. The flowchart depicts a simulation which automatically stops after a set amount of time, this has been set this way for ease of use when gathering results. This can of course be easily changed in the simulation to reflect the time-scale that the user would want with a minor change.

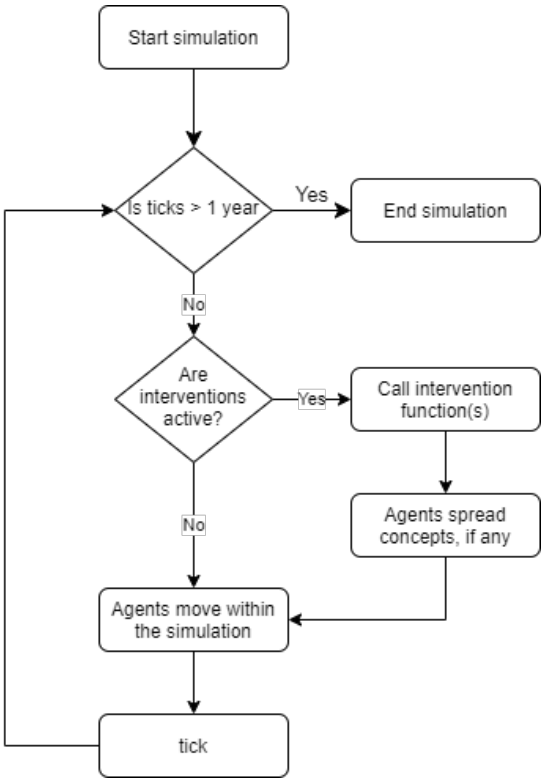


Figure 4.1: Flowchart showing the progression of the simulation.

4.2.2 Decision process

Thieves and Police take decisions with the objective of maximising their reward with a values-based learning algorithm, namely Q-Learning [26] - Figure 4.2 shows the steps involved. Bicycles are generally in the same place during weekdays, which is why reinforcement learning was

considered. Reinforcement learning is concerned with learning control policies where agents interact with an environment that is unknown to the agents [27]. This type of learning is widely used in the real world, for example with traffic light controls and in games where algorithms are used to solve games.

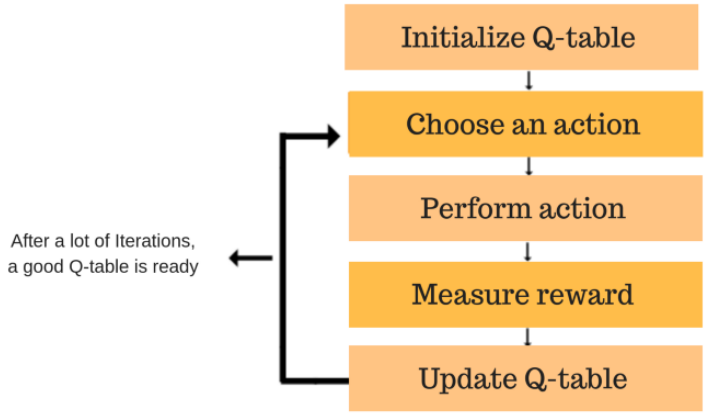


Figure 4.2: Each box shows one step of the algorithm. [28]

A Q-table is used to calculate the maximum expected future reward for actions at each state giving the best action at each state over time. The Q-function:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot (r + \gamma \cdot max(Q(s',a')) - Q(s,a))$$

Where:

- Q(s,a) is the Q value,
- α is the learning rate,
- r is the reward for taking an action in a state,
- γ is the discount factor,
- max(Q(s',a')) is the maximum expected future reward.

The Q function is used to learn each value of the Q-table. Values are initialized to zero to begin with and are updated after an episode, the table acts like a reference table for agents to choose the best action based on q-value. An agent can interact with the environment in 2 ways, they can either exploit or explore. With the exploiting interaction the agent uses the Q-table as a reference and views all the possible actions for a state, the agent then chooses the action that

holds greatest value of those possible actions- uses all information available to make a decision. The latter interaction, exploring, selects actions at random. This type of interaction allows the agent to discover new states that may not have been selected during the agent's exploitation process.

When a thief is in close proximity with another thief, the thief has the option to spread concepts to the other thief. If the concept is strong enough the agent can decide to accept the concept and alter their decision making accordingly, this may mean for example they view a bicycle of a perceived value to be too risky to steal or bicycles of low financial value are not worth the risk of being caught.

Bicycle agents in the model will primarily be in the same place daily as bicycles will be left near to the owner's place of work. Using the average working hours of workers in the City of London we can see that the majority of bicycles arrive into the environment between 800-1000 hours and leave between 1600-1800 hours.

4.2.3 Interventions

The interventions will be defined clearly in the interface. To apply an intervention to the simulation, the user must select the intervention(s) and run the setup procedure once again so the model is updated with the intervention(s).

4.2.4 Output

The simulation must output data that describes what occurred in the simulation run. At each tick a number of statistics will be reported:

- The tick number
- The number of bicycles
- The number of police officers
- The number of thieves
- The number of stolen bicycles

These will be stored in a comma-separate values file that is r....

4.3 Interface design

As the end users will predominantly be decision makers within Police forces, the interface will be as simple as possible. Because of this, the model is constructed in the NetLogo environment and this will be how users interact with the model. Since NetLogo is designed to be accessible to all types of computer users- from advanced programmers to those with few programming skills- users will easily be able to try out the model and design possible interventions and perhaps in the future model new possible interventions.

Although there will be default values for the parameters already, the first step of model setup is to select the model parameters which are available using sliders. This makes it possible to alter the population size, the number of agents and the number of each agent. The user prepares the model to run the simulation by pressing the Setup button. When the Setup button is pressed, it calls the *setup* procedure from the model's source code. By running the *setup* procedure, a map of the modelled area and the agents will begin to load based on the model parameters. If the user changes the value of a parameter after model setup, it is mandatory to run the setup procedure once again to echo the new values for the model parameters.

The model starts and runs by pressing the Go button, this is known as a forever button in Netlogo, since NetLogo continues to loop the *go* command block until a certain criteria is reached which stops the model. Hence, users are only required to click Setup followed by Go to run the simulation.

Chapter 5

Implementation

As previously specified, the model will be implemented within the NetLogo environment which utilises its own language of the same name, NetLogo. NetLogo was chosen purely because of the software's relative ease of use for users who may not have prior programming experience, lowering the barrier of entry for those that would benefit from the model.

An iterative approach was taken in the development of the model with successive versions improving upon the preceding version.

5.1 Version 1

The first version of the model was implemented with a focus towards agents and their movements within the environment whilst omitting interventions. Within the GUI the use of buttons in the model allows the model to be used by users with little or no programming experience. For users who have some experience with programming, they are able to enter commands in the command center which acts as NetLogo's command line.

Setup procedure

The *Setup* procedure, which is initiated once the setup button has been clicked in the interface tab, sets the agents in the model, this being bikes, thieves and police. This is what is known as a *once* button in NetLogo, the instructions associated with the button is executed once at the click of the button.

Setup starts by clearing all variables, turtles, patches, plots and drawings. This clears the whole model so that a new simulation can begin with nothing pre-set. The tick counter is reset

back to zero allowing the model to completely restart from time zero and to update all plots accordingly.

Next, we create the agents that will populate the environment. The environment at this early stage in development will be a simple grid structure with no environmental features. Bicycles, thieves and police officer agents are added to the model with a ratio of 85:10:5 respectively. To make it clear which turtles are which in the model, turtles are given shapes. Bicycles are aptly defined as bicycles with a green bicycle representing a bicycle that is still in the possession of the owner and a red bicycle representing a stolen bicycle. Thieves are the red characters in the model and police are the blue characters in the model.

Bicycles have a random desirability and security between 0 and 1 assigned to them from the start of their agent creation. The desirability value is a fixed value depending on the value of the bicycle and the security value may change depending on interactions during the simulation which affects the way the bicycle is secured. Thieves enter the world with a crime-probability value, this is given randomly but can change throughout the simulation.

After the agents have been created, the model is ready. To note, agents begin the simulation with no knowledge about the environment and are in an exploration phase where they randomly explore the environment. Over the duration of the simulation the agent would build up knowledge from reaching grid cells and be successful with their goals.

Go procedure

The *Go* procedure is what makes the model start running. This is where agent interaction is controlled. First we deal with the interaction between thieves and bicycles, bicycles are stationary for the majority of their lives and thieves travel within the environment scouting bicycles to thieve. Each thief enters the environment with a crime probability, that is the likelihood that they commit a theft, if the thief's crime probability is greater than the threshold for theft and the thief is in a patch with a bicycle that is poorly secured with a low security value, the bicycle will be stolen. All agents are able to navigate freely within the world, this includes police agents who patrol the world in search of thieves.

The model has been set with each tick representing one day, 24 hours. With the agents interacting as they should we can now add more functions to the model.

5.2 Version 2

The second version was a continuation of the first version but with a focus on the interface and the decision making of agents.

Agents are now able to leave the environment forever for example if a thief feels the risk in the location is too great and stealing bicycles is too risky and they choose to thieve elsewhere or due to policing they decide to no longer thieve.

Interface

Instead of a fixed number of agents in the previous version, the user is now able to change the population size of the model and the ratio of the individual agents via sliders - the sliders are a quick way to change variables. With this, users are able to compare results of the simulation easily without recoding the procedure every time.

The interface now also includes a map of the modelled environment which is initiated in the *setup* procedure, the environment being a small area of the City of London. The area is 1265 x 429 which gives a total of 542,685 patches. Obviously not all of these patches will be available for the agents, the patch could be a building or a road.

The previous version of software featured ticks that were equivalent to one 24 hour period of time. Whilst the simulation can be slowed down manually in NetLogo it was clear than one tick was too quick and the model simulated time too quickly. As a result, the time of 1 tick was changed to one hour- meaning each tick was equivalent to one hour of time, 24 ticks models 1 day. As previously said the unique nature of the City of London whereby population drastically declines on weekends and public holidays, the model will only model 261 calendar days. With each tick representing 1 hour of a day, the model will run for 6264 ticks.

Agent decision making

In the previous version, agents roamed around the world looking for bicycles with no set plan- this is where q-learning is involved. For each type of agent, patches have different q-values for example, a thief with the next patch being a bicycle will have a higher q-value than a police agent. This is because the value that is gained from taking that step is greater for a thief than other agents. For thieves a patch that contains another thief has the q-value of 0, given that they have nothing to gain, a q-value of 10 if the patch contains a bicycle and a q-value of -10 if the patch contains a police officer. This is similar for police agents, a patch that contains a thief has the q-value of 10, a q-value of -10 for a patch with another police agent and a q-value

of 2 for a patch containing a bicycle.

With the patches now having q-values dependent upon the agent that is looking at the patches we can now finalise the q-learning algorithm by computing the q-values looking at north, east, south and west to decide upon which patch to move to next. We use the q function as stated in 4.2.2 to calculate the q-value for all four possible directions and the greatest q-value is then taken as the optimal action from that patch.

From quick testing, it was apparent that using variable rates in regards to the learning rate and discount factor of agents within the q-learning caused poor performance and fluctuating results. As a result, moving forwards it was decided that fixed constants would be used.

Interventions

For simplicity for the user, possible interventions are activated with the use of buttons within the interface. The user simply activates the intervention and set the model up ready for simulation.

Police give advice to thieves

Police agents give advice to suspected thieves to stop stealing bicycles. Agents choose whether or not they want to listen to the advice given and their crime probability is reduced accordingly if they do listen. If the agent's crime probability reaches the low threshold, the agent then leaves the environment to either steal bicycle's elsewhere where they feel it is safer for them to do so or they stop stealing altogether.

Police offer bicycle locking advice

Whilst patrolling the environment, police agents offer advice on how to lock bicycles making them more secure whilst left unattended. This can be done without the owner being present through the attachment of leaflets to bicycles. Once again, agents can decide whether or not to take this advice on board but if the agent does there bicycle will have a higher security- which should in theory make it more difficult for thieves to steal and in turn deterring thieves to steal the bicycle.

Encourage bicycle marking

By having a bicycle security marked if a thief was caught with a registered bicycle that is reported stolen the thief can be arrested and the bicycle can be returned to the rightful owner.

Due to the threat of arrest if caught with a marked bicycle, this acts as a visible deterrent to possible thieves. The bicycle has a lower desirability to thieves due to the increased risk. This can be done similarly to offering locking advice through leafleting and bicycle agents interacting with one another.

Advise the removal of expensive parts from bicycles when left unattended

Bicycles frequently have parts to them that make them particularly attractive to thieves for example saddle and accessories. By removing these parts when the bicycle is left unattended, the value of the bicycle is reduced and the bicycle may be seen as less desirable in the eyes of a thief. In the case of removing a saddle from a bicycle, this makes it more difficult for the thief to transport the stolen good if they were planning on door-to-door transport to sell.

5.3 Version 3

The completion of the previous version, version 2, marked the completion of core functionality of the model however improvements can still be made. The previous versions were too slow to be used practically, taking a considerable amount of time to complete the simulation even in the fastest setting in NetLogo. The source code for the simulation was profiled to identify the slow parts and bottlenecks in the simulation were discovered.

Performance improvements

When an agent is in close proximity to another agent, instead of just interacting with that said agent the agent would instead interact with all of the same agents within the environment to then check who was nearby and in a close distance. This was rectified by limiting the radius in which an agent can interact.

It was also clear that the agent decision making process as set out in Section 4.2.2 was slow. There was an issue whereby there were too many agents on one grid space and new agents that wanted to enter that grid space were forced to wait for that grid space to become available. One grid space in the environment is the equivalent to roughly a few square meters, in the real-world bicycles and people are not that physically close and so by limiting the amount of agents per grid space agents were able to move quicker by being forced to go elsewhere if there choice of grid space is unavailable.

In previous versions, bicycles would enter the environment whenever they wished- this is

unrealistic, with bicycles sometimes entering the environment at the equivalent to 0300 hours. The peak time of travel within the City of London is between 0800-0930 hours in the morning and 1630-1900 hours in the evening. To account for this, the majority of bicycles are set to enter and leave between those times in the equivalent time in ticks.

There were clear bottlenecks in the interventions, this was down to interventions not being targeted to agents and instead targeted to all agent types in the model. This was easily rectified by only calling the needed procedures when certain conditions were met, so only suspected thieves would receive advice from police agents and bicycles do not.

5.4 Testing

By choosing to follow an agile development cycle, testing of functionalities in the model were done iteratively with each subsequent version and little testing was required to ensure that the software was performing as required. Nonetheless time was required to test individual variables in the model, as minor adjustments of these numerical values would cause severe changes in agent behaviour. There was a great deal of trial and error testing involved in seeking default values for the model so that the model would produce a realistic and balanced simulation.

Chapter 6

Professional Issues

Throughout the implementation of the project, great effort was taken to abide by the British Computer Society (BCS) Code of Conduct & Code of Good Practice. Any Open-Source code or libraries used in the project has been made explicitly clear. This project consists of my own work and Open-Source code and where Open-Source code has been used, it has been modified to suit the requirements of the project.

It is important to note that this project does not act as an accurate simulation of bicycle theft in real life. Due to the under-reporting of bicycle thefts, it is impossible to accurately validate this model. The intended purpose of this model is to show how thieves *may* be deterred in carrying out bicycle theft.

Chapter 7

Results

Creating a model that would be able to simulate possible interventions to judge viability was an important part of this project. This following section will outline experiments performed using the model to demonstrate the model's utility as a tool and express the results from said experiments.

Experimentation can be split into two distinct types with this model, with the first being model experimentation. This type of experimentation is used to explore changes in the simulation by changing the way the agents work, focussing upon the model behaviour rather than trying to replicate real-world scenarios. By changing variables such as the learning and discount rate it is possible to affect the performance of the agents. Testing this would be useful in trying to make the model behave similarly to the real world and using it to replicate other real world phenomena.

The second type of experimentation will focus on the interventions set out with the use of scenario experimentation, this will be similar to what the end user would use the model for. The end user would not be concerned with the specifics of the software, for example how each agent works, how each agent interacts within the model and how they are initiated in the model, but instead focus on the testing variables and attempting to apply the model to the real world.

7.1 Testing parameters

To ensure consistency of results, elements that are used in testing that are consistent throughout are explicitly defined. They are as follows:

Parameter	Value
Population size	12000
Ratio of thieves	0.01
Ratio of bicycles	0.25
Ratio of police	0.10
Minimum agent learning rate	0.1
Maximum agent learning rate	0.9
Agent discount factor	0.8

The values above were decided through development of new functionalities. The result of these values produces a balanced behaviour in agents.

There is only a clear 'win condition' for thief agents within the model, it is in their best interest to steal as many bicycles as possible. Due to the nature of interventions model runs simply conclude after a fixed number of ticks as is already done in the model, 6264 ticks was decided upon after the exclusion of weekends and public holidays where footfall in the real-life environment drastically declines.

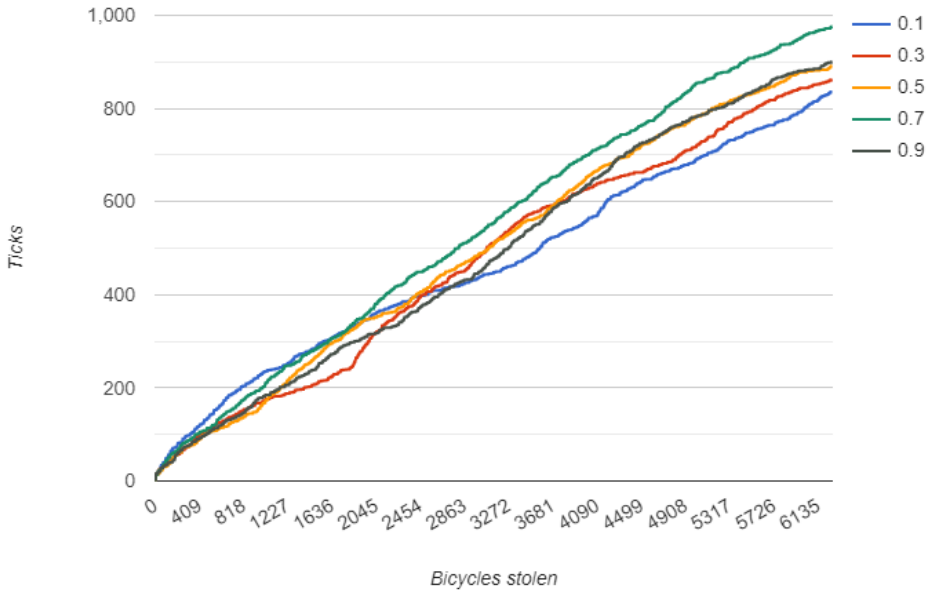
There is only one value that is important in the model and that is the total number of bicycles that are stolen over the duration of the simulation. Using this value we are able to see how 'well' agents performed and evaluate the effectiveness of interventions. The higher the value, the 'better' thieves are doing- this value shows the success of both thieves and police whilst also looking at changes within bicycles as thieves are in direct competition with all other agents.

7.2 Model experiments

7.2.1 Considering the learning rate of agents

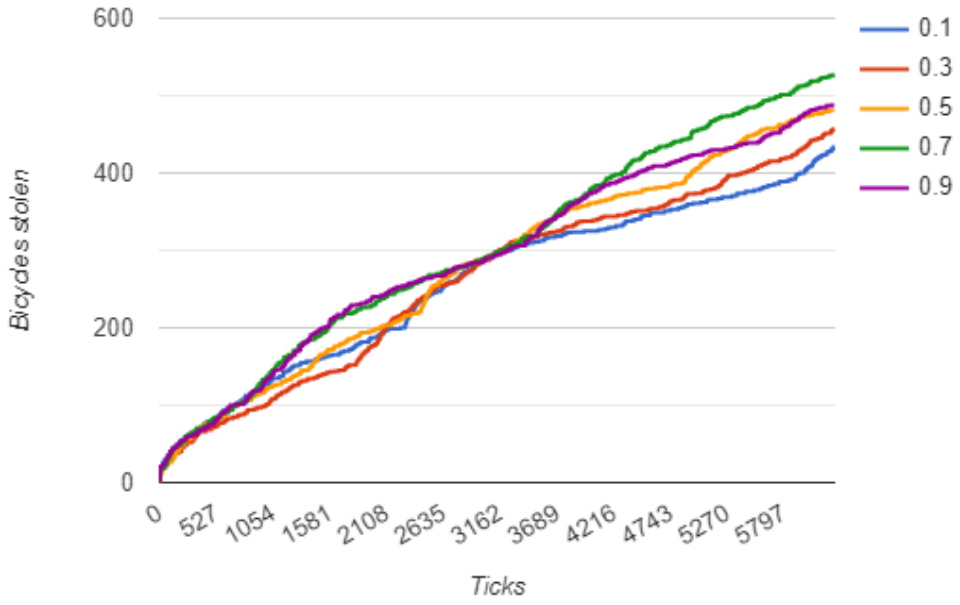
Learning rate determines the speed at which agents are able to take on new information, this usually takes the form of a constant or a variable. The following experiment aims to examine how changing the learning rate affects the number of bicycles stolen.

Learning rate with no interventions



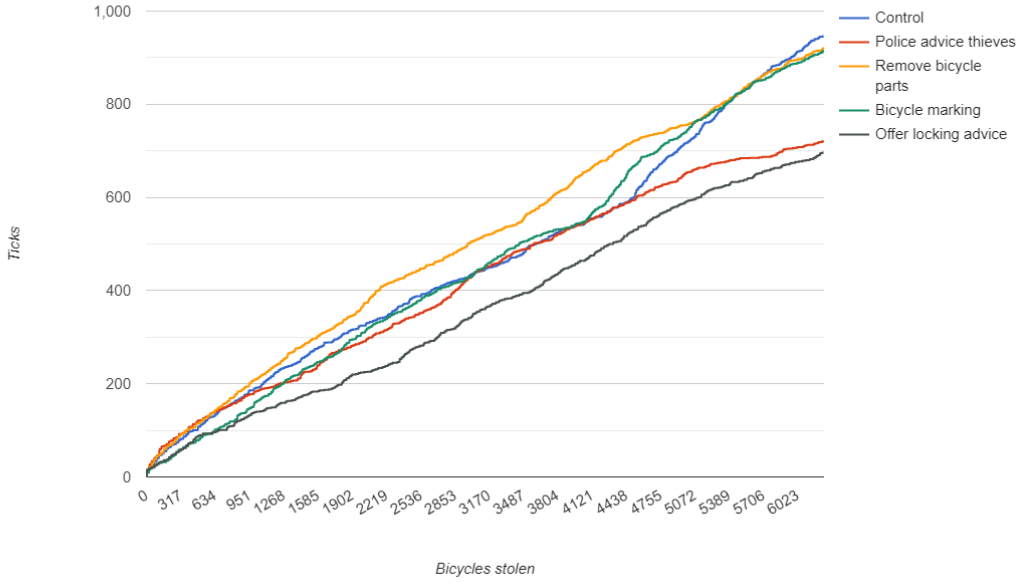
Whilst not substantial, there is an apparent difference in agent effectiveness when using differing learning rates. With no interventions present the agents have the greatest opportunity to steal bicycles. A constant low learning rate in the agents performed as expected and unsurprisingly the experiment with a learning rate of 0.1 gave the worst results. Agents utilising lower learning rates were less effective at stealing bicycles than compared with greater learning rates suggesting a low rate is not preferable. The higher rate of 0.9 is more effective however peaks at certain points and fluctuates throughout, this is expected as with high learning rates that are close to 1 the agents ignore all prior knowledge and solely consider the most recent information via exploration. The results show that a learning rate between 0.7 and 0.9 is most optimal with the greatest performance in theft, this value acts as a compromise between a high value and a consistent value.

Learning rate with interventions



With the addition of interventions, the difference in agent effectiveness is made more clear. This experiment uses all of the possible interventions available in the model. As before, the low learning rate returned the worst results and the trend is a higher learning rate delivered greater results. Once again there reaches a point where a high learning rate no longer delivers and results start to suffer, the results shown here reinforces the results seen previously. A learning rate of 0.9 initially performs better than the learning rate of 0.7 but is less favourable as we reach the latter stages of the run. We can confidently say that the model is most optimal with a learning rate between 0.7 and 0.9.

7.2.2 Intervention experimentation



The above graph shows results with interventions present. There are varying results with the interventions with some interventions showing promise whilst others not so much. A control is included which shows theft with no interventions present and is used as a comparison against results with interventions present.

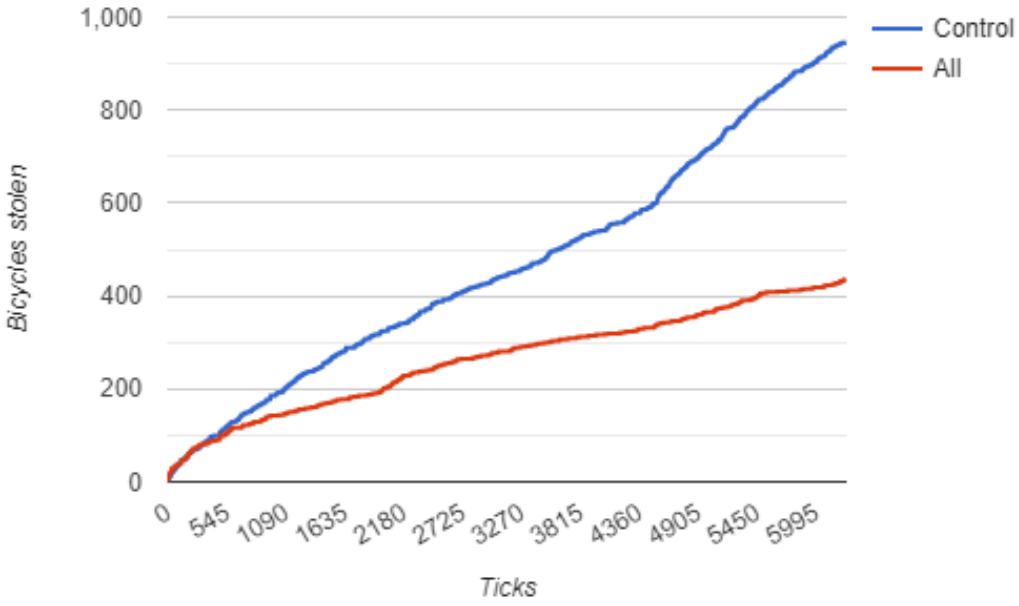
As it can clearly be seen in the graph, removing bicycle parts from a bicycle when leaving it unattended does little to deter theft and at points even attracts thieves to steal it. This may be due to the fact that as there are missing parts, thieves may think that the bicycle is abandoned anyway and stealing it when it has been abandoned is seen as less risky. Another intervention which is worth noting is bicycle marking. In theory by security marking a bicycle, thieves should be deterred from stealing them as if they are caught with them and they have been reported as stolen it is an offence. The results show that bicycle marking makes little difference by itself but it is also worth noting that if the bicycle is found eventually, there is a chance that the bicycle can be recovered and returned to its owner.

Of all the interventions, offering locking advice shows the most promise and with that a considerable drop in thefts. By offering locking advice bicycles can be left more secure whilst unattended, whilst the bicycle can still be stolen by a determined thief there is more risk on the thief agent stealing a bicycle that is secured well. More time will need to be spent with said bicycle to free it from its locked state which may arouse suspicion of nearby people in the

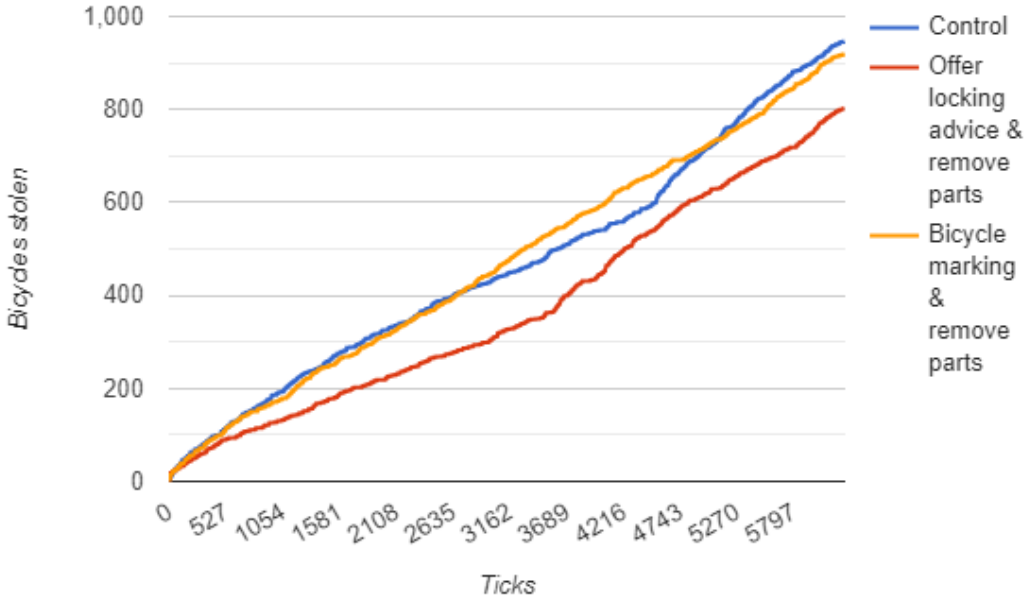
real-world. When observing the model visually, the thief agents could be seen going past secure bicycles and instead targeting bicycles that are less secure.

Combination of interventions

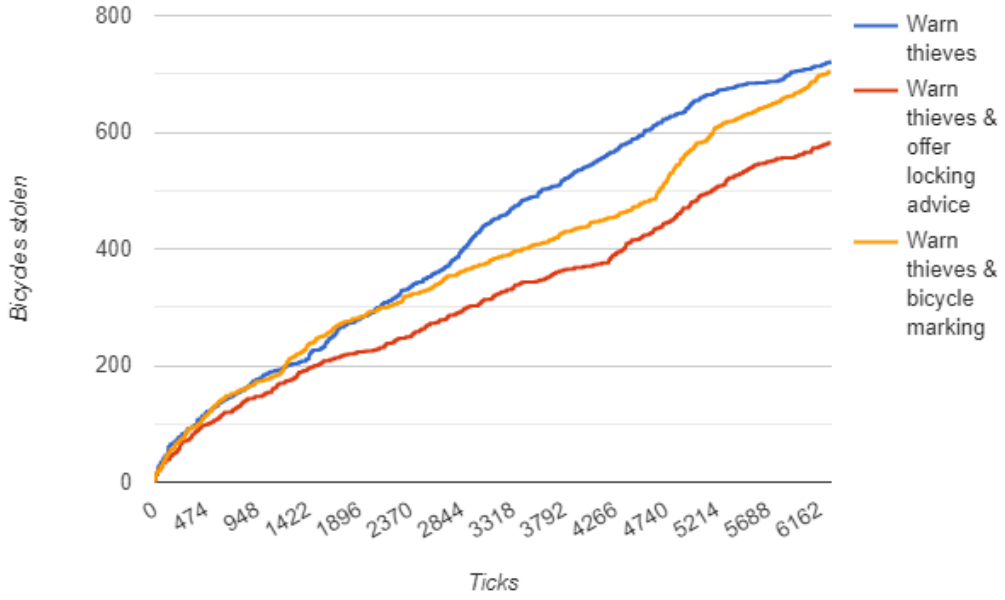
For interventions to make an impact, it would be best to use a combination of interventions.



The blue line shows bicycle theft with no interventions present whereas the red line shows all interventions being used. In an ideal world where all possible interventions could be implemented the results are positive and there is a clear drop in the projected bicycles being stolen as a result of thieves being deterred to thieve in the modelled location and also thieves deciding that the risk no longer outweighs the reward. This is wholly unrealistic given the many interventions that are being modelled. The modelled simulation includes police advising suspected thieves against thieving, encourage the removal of expensive bicycle parts, encourage bicycle security marking and offering secure locking advice.



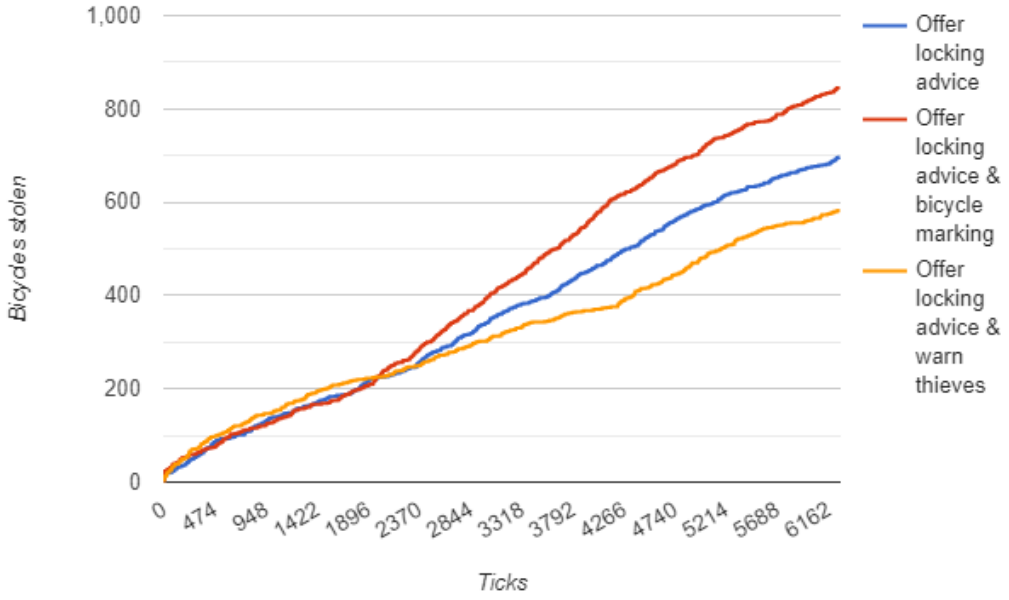
This experiment followed on from the experiment with individual interventions. With the combination of interventions we are able to notice how one intervention may work well with other interventions. From the graph we can see the control, which is the simulation with no interventions present. Encouraging bicycle security marking and encouraging owners to remove expensive parts from their bicycles makes little difference to overall theft. From this we are able to conclude that by removing bicycle parts it fails to act as a deterrent and should not be used as an intervention moving forwards. Offering locking advice is a more suitable intervention with a clear decrease in thefts, this may be due to the increase in bicycles being locked securely which makes it more difficult for thieves to steal.



Police giving advice to suspected thieves to stop stealing bicycles was discovered to be effective in experimentation. This may be due to the fact that thieves are aware of the fact that their actions have been noticed and that there may be a greater risk in the future of being caught whilst committing a crime, in effect increasing the risk of committing bicycle theft.

When experimenting the warning thieves intervention with other possible interventions, it was found that when combined with encouraging bicycle security marking thefts there was little effect overall. It is clear that warning thieves against stealing bicycles is effective in some form but bicycle marking makes little difference in deterring thieves from stealing bicycles.

Offering advice on how to securely lock bicycles shows some promise when in combination with warning thieves, there is a decrease in thefts when compared with just offering locking advice by itself.



From the initial experiment it was found that offering locking advice on how best to secure a bicycle resulted in the greatest drop in thefts. This experiment focused on the potential implementation of other interventions alongside it. In earlier experiments we found that bicycle marking offered little in the act of deterring thefts but instead aided the return of bicycles if successfully recovered. In combination with bicycle security marking and offering bicycle locking advice, results were surprisingly poor. Over time, thefts of bicycles after the combination of interventions were made actually suffered than compared with just offering locking advice. We can speculate that this could be down to the fact that the bicycle is security marked it is perceived to be secure and that there is no need to securely lock it.

As we found earlier the best reduction in thefts came from offering bicycle locking advice and police warning suspected thieves individually. Combining these two interventions showed the most promise with a drastic reduction in bicycle thefts. With bicycles more securely locked when left unattended and thieves being made aware of the risks of being caught with stolen bicycles, the risk vs reward is changed significantly. Whilst the reward of the bicycle remains the same, the risk is adversely changed for the greater.

Chapter 8

Evaluation

8.1 Requirements

To ensure that the developed software is complete, it was evaluated against the earlier described requirements in Section 3.2.

8.1.1 Simulation requirements

- **The simulation must run for at least 1 year.** Due to the unique nature of the City of London it was decided that it best to only model 261 calendar days.
- **Agents come and go throughout the day.** At each time tick agents in the model are free to move within or even leave the environment if they so do wish.
- **Due to the uniqueness of the City of London whereby the resident population is 7375 with a 56 fold increase in workday population, according to data from the 2011 Census [24], the simulation will ignore weekends and holidays where the footfall is dramatically reduced.** As explained above, the model ignores weekends and holidays and models 261 calendar days.
- **The simulation must allow the user to call intervention functions to manipulate the simulation world (agents and environment).** The user is able to call interventions by simply selecting the intervention buttons in the interface.
- **The simulation must return statistics at the end.** Statistics can be seen in the monitors in the interface, graphs can be exported to see more in-depth statistics.

8.1.2 Agent requirements

- **Agents in the simulation must be free to come and go.** Each agent in the model has its own decision making and is able to move within the environment or to leave altogether.
- **Agents must observe the environment and react to any changes that occur accordingly.** As described in the design chapter, agents have a utility function whereby they are able to observe the environment around them and it is this surrounding environment that affects decision making.
- **A bicycle must either be in a locked or stolen state.** Each bicycle agent can either hold the variable whereby it is locked or a state where it is stolen.
- **A bicycle must be able to hold concepts.** Each bicycle has a security variable attached to them, this variable can be affected by the adoption of concepts from other bicycles.
- **A bicycle must be able to spread concepts to other bicycles.** When a bicycle is within close proximity to another bicycle they are able to spread and adopt concepts.
- **Thieves must either be active or inactive.** Each thief can either be in state where they are active or inactive.
- **Thieves are able to steal bicycles.** For thief agents it is their goal in the model to steal bicycles, it can be said that if they do not steal a bicycle they have not succeeded.
- **Thieves attempt to avoid Police.** As described in the design, the agent has a utility function that it uses to decide where to go, if they are near a Police agent they will attempt to take a different route.
- **A thief must be able to spread concepts to other thieves.**When a thief is within close proximity to another thief they are able to spread and adopt concepts.
- **Police must either be active or inactive.** Each police agent can either be in state where they are active or inactive.
- **Police must be able to spread concepts to both bicycles and thieves.**When a police agent is within close proximity to suspected thieves or bicycles they are able to spread concepts.

8.1.3 Environment requirements

- **The environment must represent a real-world physical area within London.** With the use of a PNG image, the model models an area of the City of London.
- **The environment must support the definition of properties and through-way.** The PNG image as discussed above sets out the roads and buildings with the use of different colours. Within the model itself, agent movement is restricted to areas that are seen as roads or pavements with the use of patches. Areas that are deemed as buildings from the PNG through the colour of the patch are not accessible by agents.
- **The number of agents within the environment must be adaptable by the user.** The user is able to adjust the number of agents by changing the overall population within the environment or individually change the ratio of each agent.
- **Bicycle and thief agents must not be able to enter buildings that will be clearly defined in the simulation.** Through the use of the coloured PNG image of the area, agents are restricted to only the public roads and footways.
- **The environment should be as realistic as possible.** The environment is a small area within the City of London.

8.1.4 NetLogo requirements

- **The user must be able to setup, run and stop the model.** Through the use of buttons within the NetLogo interface, the user is able to simply setup, run and stop the model.
- **The simulation within NetLogo must create some agents within the environment and interventions to be used in the model.** At the setup stage of the simulation, agents propagate in the model, possible interventions are pre-set and can be used at a click of a button.
- **NetLogo must run the simulation.** This can be simply achieved by setting up the model and pressing *Go*.
- **NetLogo must offer variable changes.** Changes to variables can be made with the use of the sliders. If the user wishes to change the learning rate or discount rate in the agent decision making this can be done by changing the variables in the source code.

- **NetLogo must create some graphs to describe the simulation run.** A graph is outputted to the user as the simulation progresses but due to NetLogo's restrictive nature this will need to be exported to a CSV so that more in-depth analysis can take place.

8.2 Experimentation

The results chapter of this report shows how the software could be used in real-world application and showed promising results that were interesting and could be explained intuitively. With the results from the experiments more interventions can be be tried with the model and greater combinations to find the most optimal solution.

Chapter 9

Conclusion

9.1 Conclusion

This project relies on behaviours of agents and it is clear greater research needs to be done to understand the psychology of a criminal so that modelling their behaviour can be more accurate and reliable.

The project aimed to create a piece of software that modelled bicycle theft in London that could be easily used by someone with little technical knowledge. In this way it can be said that the project was quite successful, however using NetLogo was rather restrictive at times and if another system was used implementation could have been made simpler and agent behaviour and interactions could have been made more complex to simulate it similarly to real-life.

9.2 Future Work

Many improvements and additions can be implemented in future iterations of the model. With the ease of additions in NetLogo, this would be relatively easy if necessary.

When thinking about the environment, a clear improvement would be to add street-level features like CCTV and lighting to the model that would affect agent decision making. This will obviously take a large amount of time to accurately map the locations of these environmental features however analysing the influence of readily available public safety elements would be useful.

Another major improvement that could be made would be the addition of weather modifiers, this would allow the user to be able to simulate how weather affects agent behaviour. Perhaps

fewer bicycles being on the streets in addition to poor weather detract thieves from going to an area to steal.

Agent-based modelling is a very useful tool and there are many opportunities to develop this project further to achieve fascinating results.

Bibliography

[1] NHS. *Cycling for beginners*. [Online; accessed 30-November-2019]. URL: https://www.nhs.uk/live-well/exercise/cycling-for-beginners/.

[2] London Assembly Transport Committee. *London's cycling infrastructure*. 2018. URL: https://www.london.gov.uk/sites/default/files/londons_cycling_infrastructure_0.pdf.

[3] Transport for London. *Cycling action plan, Making London the world's best city for cycling*. 2018. URL: http://content.tfl.gov.uk/cycling-action-plan.pdf.

[4] Transport for London. *Travel in London, Report 11*. 2018. URL: http://content.tfl.gov.uk/travel-in-london-report-11.pdf.

[5] A. Sidebottom. *Bicycle (bike) theft*. [Available from www.jdibrief.com]. JDiBrief Series, London: UCL Jill Dando Institute of Security and Crime Science. 2012.

[6] Gohar Petrossian and Ronald Clarke. "Explaining and Controlling Illegal Commercial Fishing: An Application of the CRAVED Theft Model". In: *British Journal of Criminology* 54 (Jan. 2014), pp. 73–90. DOI: 10.1093/bjc/azt061.

[7] Transport for London. *Cycleways*. [Online; accessed 07-December-2019]. URL: https://tfl.gov.uk/modes/cycling/routes-and-maps/cycleways.

[8] Transport for London. *Travel in London, Report 11*. 2012. URL: http://content.tfl.gov.uk/travel-in-london-report-5.pdf.

[9] Lucy Waruguru Mburu and Marco Helbich. "Environmental Risk Factors influencing Bicycle Theft: A Spatial Analysis in London, UK". In: *PLOS ONE* 11.9 (Sept. 2016), pp. 1–19. DOI: 10.1371/journal.pone.0163354. URL: https://doi.org/10.1371/journal.pone.0163354.

[10] Lawrence E. Cohen and Marcus Felson. “Social Change and Crime Rate Trends: A Routine Activity Approach”. In: *American Sociological Review* 44.4 (1979), pp. 588–608. ISSN: 00031224. URL: http://www.jstor.org/stable/2094589.

[11] Lawrence E. Cohen and Marcus Felson. “Social Change and Crime Rate Trends: A Routine Activity Approach”. In: *American Sociological Review* 44.4 (1979), pp. 588–608. ISSN: 00031224. URL: http://www.jstor.org/stable/2094589.

[12] Lawrence W. Sherman, Patrick R. Gartin, and Michael E. Buerger. “Hot Spots Of Predatory Crime: Routine Activities And The Criminology Of Place*”. In: *Criminology* 27.1 (1989), pp. 27–56. DOI: 10.1111/j.1745-9125.1989.tb00862.x.

[13] Derek B. Cornish and Ronald V. Clarke. “Understanding Crime Displacement: An Application Of Rational Choice Theory”. In: *Criminology* 25.4 (1987), pp. 933–948. DOI: 10.1111/j.1745-9125.1987.tb00826.x.

[14] Andrew Crooks, A.J. Heppenstall, and Nick Malleson. “Agent-Based Modeling”. In: Dec. 2017. ISBN: 9780124095489. DOI: 10.1016/B978-0-12-409548-9.09704-9.

[15] Elizabeth R. Groff, Shane D. Johnson, and Amy Thornton. “State of the Art in Agent-Based Modeling of Urban Crime: An Overview”. In: *Journal of Quantitative Criminology* 35.1 (2018). DOI: 10.1007/s10940-018-9376-y.

[16] Nick Malleson and Andrew Evans. “Agent-Based Models to Predict Crime at Places”. In: *Encyclopedia of Criminology and Criminal Justice*. Ed. by Gerben Bruinsma and David Weisburd. New York, NY: Springer New York, 2014, pp. 41–48. ISBN: 978-1-4614-5690-2. DOI: 10.1007/978-1-4614-5690-2_208. URL: https://doi.org/10.1007/978-1-4614-5690-2_208.

[17] Nick Malleson, Alison Heppenstall, and Linda See. “Crime reduction through simulation: An agent-based model of burglary”. In: *Computers, Environment and Urban Systems* 34.3 (2010), pp. 236–250. ISSN: 0198-9715. DOI: https://doi.org/10.1016/j.compenvurbsys.2009.10.005. URL: http://www.sciencedirect.com/science/article/pii/S0198971509000787.

[18] Eugene A Feinberg. “Total reward criteria”. In: *Handbook of Markov decision processes*. Springer, 2002, pp. 173–207.

[19] Ministry of Justice. *Proven reoffending statistics*. Jan. 2020. URL: https://www.gov.uk/government/collections/proven-reoffending-statistics.

[20] Josja J Rokven et al. “How friends’ involvement in crime affects the risk of offending and victimization”. In: *European Journal of Criminology* 14.6 (2017). PMID: 29187805, pp. 697–719. DOI: 10.1177/1477370816684150. eprint: https://doi.org/10.1177/1477370816684150. URL: https://doi.org/10.1177/1477370816684150.

[21] Paulo Shakarian et al. “The Independent Cascade and Linear Threshold Models”. In: *Diffusion in Social Networks*. Cham: Springer International Publishing, 2015, pp. 35–48. ISBN: 978-3-319-23105-1. DOI: 10.1007/978-3-319-23105-1_4. URL: https://doi.org/10.1007/978-3-319-23105-1_4.

[22] David Kempe, Jon Kleinberg, and Éva Tardos. “Influential Nodes in a Diffusion Model for Social Networks”. In: vol. 3580. July 2005, pp. 1127–1138. DOI: 10.1007/11523468_91.

[23] Konstantinos Liontis and Evaggelia Pitoura. “Boosting Nodes for Improving the Spread of Influence”. In: *ArXiv* abs/1609.03478 (2016).

[24] *2011 Census: The workday population of England and Wales - An alternative 2011 Census output base*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/articles/theworkdaypopulationofenglandandwales/2013-10-31#differences-between-the-usually-resident-and-workday-populations.

[25] Metropolitan Police. *How safe is your bike?* [Online; accessed 27-November-2019]. URL: https://www.met.police.uk/cp/crime-prevention/theft-of-a-bicycle/how-safe-is-your-bike.

[26] Christopher J. C. H. Watkins and Peter Dayan. “Q-learning”. In: *Machine Learning* 8.3-4 (1992), pp. 279–292. DOI: 10.1007/bf00992698.

[27] Matthew Hausknecht and Peter Stone. “Deep Recurrent Q-Learning for Partially Observable MDPs”. In: *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15)*. Arlington, Virginia, USA, Nov. 2015.

[28] freeCodeCamp.org. *An introduction to Q-Learning: reinforcement learning*. Aug. 2018. URL: https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/.

[29] Lin Larry (model id 3986) – netlogo modeling commons. *Q-Learning in MDPs*. URL: http://modelingcommons.org/browse/one_model/3986#model_tabs_browse_procedures.

Appendix A

User Guide

A.1 Environment setup

A.1.1 Installing NetLogo

In order to run the simulation you must first install NetLogo, the simulation was developed in Version 6.1.1 (September 2019) and may not be backwards compatible with earlier versions of NetLogo. To do this visit https://ccl.northwestern.edu/netlogo/6.1.1/ and download the relevant application depending on your current operating system.

A.1.2 NetLogo

- Interface tab : This is the main tab where the user interacts with the model and agents behaviour is visualised. It contains buttons, sliders, switches, monitors and plots.
- Information tab: This contains information for the user on how to use the model.
- Code tab: This contains all the source code.

A.1.3 Running the simulation

With NetLogo now installed, you are now able to go to the root folder of the simulation and open the nlogo file. Once opened, you will be presented with the main view in the centre and various buttons, sliders and monitors to the left of the main view as seen in figure A.1.

The sliders allow you to change the population of the area that is being simulated and the ratio of each of the agents. The possible interventions are at the bottom, it is possible to run the simulation with no interventions, certain interventions or a combination of the interventions.

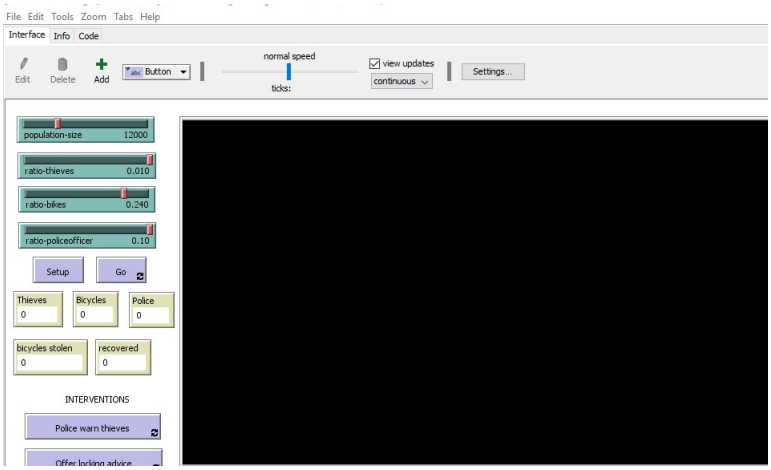


Figure A.1: The NetLogo interface

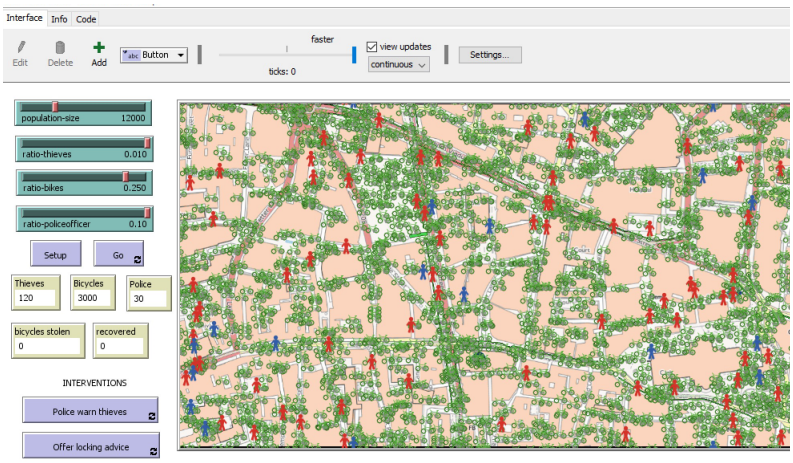


Figure A.2: The NetLogo interface once setup has been completed

Once that is decided click the Setup button to setup the model, agents will now appear in the main view as denoted by the respective characters in the simulation. If there are any changes to the interventions or the variables, the model needs to be setup again with the use of the Setup button. To run the simulation, click the the Go button- this will run the simulation for one calendar year in the respective number of ticks. To note, in the simulation one tick is equivalent to one hour of time.

The speed of the simulation can be changed using the speed slider in the upper toolbar, it is set at normal speed by default. It is important to ensure that the view updates box is applied, this ensures that updates can be seen visually in the interface.

With the model now setup, the interface should look like figure A.2.

Appendix B

Source Code

B.1 Avowal

I verify that I am the sole author of all code, except for a minor section in the Q-learning in the decision making of agents whereby open-source code was modified. The author of the original code is Larry Lin and the code can be found on Modelling Commons [29]. This has been made explicitly clear.

B.2 Source

```
;each tick is representative of 1 hour

;6264 ticks = 261 days
;model only models working days Mon-Fri (footfall in the City dramatically reduces on weekends)

;singular and plural
;TO NOTE: bicycles and bikes are used interchangably
breed [bikes bike]
breed [thieves thief]
breed [policeofficer police]

;global variables that all agents own
globals[
  count-total
  enforce?
  hide?
  police-thief?
  police-bike?
  bikemarking?
  removeparts?
  recovered
]

;variables that all patches own
patches-own[
  accessible?
  q-val-north
  q-val-south
  q-val-east
  q-val-west
]

;variables that thief agents own
thieves-own[
  crime-probability
  birth-tick
  hidden?
]

;variables that bicycles own
bikes-own[
  desirability
  security
  stolen?
  birth-tick
  show?
  marked?
  advised?
  remove?
]

;creates the map of the City of London
to init-map
  import-pcolors "./images/london.png"
  ask patches[
    ;set all patches in the image (grid space) as accessible
    set accessible? true
    ;patches that are pinkish in the image are buildings and therefore are inaccessible
    if pcolor = 28.6 [set accessible? false]
  ]
```

```
end

;sets bicycles in the model
to setup-bikes
  create-bikes ratio-bikes * population-size [
  spawn-bike

]
end

;sets thieves in the model
to setup-thieves
  create-thieves ratio-thieves * population-size [
    spawn-thief
  ]
end

;sets police in the model
to setup-policeofficer
  create-policeofficer ratio-policeofficer * population-size / 40[
    spawn-police
  ]
end

;setup procedure that is called once the setup button is clicked
to setup
  __clear-all-and-reset-ticks
  init-map
  setup-bikes
  setup-thieves
  setup-policeofficer
  set hide? false
end

;go procedure that is called once the go button is clicked
to go
  ;thieves are not seen in the model at all times, if hidden is false they are
seen, if hidden is true they are hidden
  ask thieves with [hidden? = false][
    set hide? false
  ]
  ask bikes[
    ;bicycles leave during 5-7ish
    let leave (8 + random 2)
    ;bicycles arrive during 7-9ish
    let arrive (22 + random 2)

    ;ensures that bicycles come and go during the working day
    ;hides and shows the turtles
    if (ticks - birth-tick = arrive) or (ticks - birth-tick = arrive + 1)[
      ;restarts the time when bicycles arrive
      set birth-tick ticks
      set show? true
      st
    ]
    if (ticks - birth-tick = leave) or (ticks - birth-tick = leave + 1)[
   ; or (ticks - birth-tick = leave + 1)[
      set show? false
      ht
    ]
  ]

  ;thieves move about if they are in the area
```

```
  if hide? = false[move-thief]

  ;if the bike marking intervention has been clicked, this is run
  if bikemarking? = true [
    bike-bikemarking
    ask bikes with [bikemarking? = true and show? = true][
      bikes-talk
    ]
  ]
  ;if the advise removal of bicycle parts intervention has been clicked, this is run
  if removeparts? = true [
    remove-parts
    ask bikes with [remove? = true and show? = true][
      bikes-talk
    ]
  ]
  ;calls the procedure that moves police agents
  move-police

  ask thieves[
    ;like bicycles, thieves do not spend all their time at the location
    if (ticks - birth-tick) = random 8 + 3 [hide-turtle set hidden? true]
    if (ticks - birth-tick) = 15 + random 5 [show-turtle set hidden? false set birth-tick ticks]
    if random 1000 = 5 [die]
    if random 1000 = 5 [thief-enter]
    ;low threshold, once this is reached thief-elsewhere procedure is called
    if crime-probability < 0.1 [
      thief-elsewhere
    ]
    ]
  ;new thieves enter the environment
  if count thieves < ((ratio-thieves * population-size ) - (ratio-thieves * population-size * 0.4)) [create-thieves random (ratio-thieves * population-size) [spawn-thief]]
  ;new people decide to cycle to work
 if count bikes < ((ratio-bikes * population-size) - (ratio-bikes * population-size) * 0.15) [create-bikes (ratio-bikes * population-size * 0.1 ) [spawn-bike]]

tick
  ;model stops once 261 days has been reached
  if ticks = 6264 [stop]
end

;sets out the value of patches for thieves
;as each patch is 1 pixel in the image, a radius of 5 is used to make it as close to real life as possible
to thief-patches
  ask patches in-radius 5 with [(accessible?) and (count turtles-here > 1)][
    if (any? thieves in-radius 5 with [accessible?])[
        set q-val-north 0
        set q-val-east 0
        set q-val-south 0
        set q-val-west 0
      ]
    if (any? bikes in-radius 5 with [accessible?])[
        set q-val-north 10
        set q-val-east 10
        set q-val-south 10
        set q-val-west 10
      ]
    if (any? policeofficer in-radius 5 with [accessible?])[
```

```
        set q-val-north -10
        set q-val-east -10
        set q-val-south -10
        set q-val-west -10
      ]
    ]
end

;sets out the value of patches for police
to police-patches
  ask patches in-radius 5 with [(accessible?) and (count turtles-here > 1)][
    if (any? thieves in-radius 5 with [accessible?])[
        set q-val-north 10
        set q-val-east 10
        set q-val-south 10
        set q-val-west 10
      ]
    if (any? bikes in-radius 5 with [accessible?])[
        set q-val-north 2
        set q-val-east 2
        set q-val-south 2
        set q-val-west 2
      ]
    if (any? policeofficer in-radius 5 with [accessible?])[
        set q-val-north -10
        set q-val-east -10
        set q-val-south -10
        set q-val-west -10
      ]
    ]
end

;movement of all agents
to move
  ;gets the current x and y co-ordinates
  let current-xcor xcor
    let current-ycor ycor
  ;25% chance of moving north, east, south, west
    set heading ((random 4) * 90)
      let probability random-float 1
      ifelse (probability < 0.8)[ ;80% chance going in intended  direction (MDP)
      ][
        ifelse (probability < 0.9)[
          set heading (heading + 90)];go left of intended direction
        [set heading (heading - 90)];go right of intended direction
      ]
  fd 10 ; direction is set, go forwards 10 patches
  ;accidently entered an area that is marked as a building, reverse
      if pcolor = 28.6[
    bk 10
  ]
  ;set the q-values
      set-qvalue current-xcor current-ycor heading xcor ycor
end

;movement of thief agents that are active
to move-thief
  ask thieves with [hidden? = false][
    move
    if crime-probability > 0.2[
    steal-bike
    ]
    sell-bike
```

```
243   ;]
244     thief-patches
245   ]
246 end
247
248 ;police need to check for stolen bikes with trackers
249 to move-police
250   ;ifelse enforce? = true[
251   ifelse (police-thief? = true) or (police-bike? = true) or (bikemarking? = true)[
252   ask policeofficer[
253     move
254       if police-thief? = true [police-thief]
255       ;policebike recover bicycles
256       if (police-bike? = true) or (bikemarking? = true)[police-bike]
257      ; if bikemarking? = true [police-recover]
258   police-patches
259   ]
260   ]
261   [ask policeofficer[hide-turtle]]
262
263 end
264
265 ;thief steals a bicycle
266 to steal-bike
267  let randomnumber random-float 1
268  let randomnumber2 random-float 1
269 ;  ask bikes in-radius 6 with [(hidden? = false) and (shape = "bike") and (color =
    green) and (not stolen?)][
270   ask bikes in-radius 2 with [(show? = true) and (shape = "bike") and (color =
    green) and (not stolen?)][
271     ;if (shape = "bike") and (color = green) and (not stolen?)[
272       if (desirability > randomnumber) and (security < randomnumber2)[
273         set count-total count-total + 1
274         hatch-bikes 1 [set stolen? true set color red]
275         die
276       ]
277     ]
278  ; ]
279 end
280
281 ;bicycles are sold outside the area
282 to sell-bike
283   ask bikes with [color = red and show? = true][
284   ;ask bikes with [stolen? = true][
285     ;if ticks - birth-tick > localvariable2 + random 10 [show-turtle set hidden?
    false set birth-tick ticks]
286     ;bike is sold in the area
287     if ticks - birth-tick = random 40 [hatch-bikes 1 die]
288     ;bike is sold outside of the area
289     if ticks - birth-tick > random 50 [die]
290   ]
291 end
292
293 ;thief comes to area
294 to thief-enter
295   hatch-thieves 1 [spawn-thief]
296 end
297
298 ;thieves detered from stealing/go to another area
299 to thief-elsewhere
300   die
301 end
302
```

```
;police interaction with thieves
to police-thief
  ask thieves in-radius 5 with [hidden? = false][
    if random-float 1 > 0.5 [
    set crime-probability crime-probability - (crime-probability * 0.25)
    thieves-talk
    ;thieves go elsewhere/stop thieving here
    if crime-probability < 0.2 [die]
  ]
  ]
end

;police have greater 'vision' of the environment
to police-bike
  ask bikes in-radius 50 with [show? = true and advised? = false][
    ;if the giving advice intervention is active,  advise them on safer locking
through verbally or leaflet
    if police-bike? = true[
    let chance random-float 1
    if chance > 0.5 [set security security + 0.2 set advised? true]
    ]
    ;if the bike marking intervention is active and a bicycle is found, take it.
    if bikemarking? = true[
    if any? bikes in-radius 3 with [(stolen? = true) and (marked? = true)][
        if random-float 1 > 0.8[
          let time ticks
          set recovered recovered + 1
          die
          ;bicycle is returned to the owner
          if ticks = time + 48 [
      hatch-bikes 1 [spawn-bike set marked? true set security random-float 1 +
0.2]
        ]
        ]
  ]
  ]
  ]
end


;bikes have bikemarking trackers- if recovered, possible to return to owner
to bike-bikemarking
  ask bikes with [(show? = true) and (random-float 1 < 0.02)][
    if random-float 1 < 0.02 [
    set marked? true
    set desirability desirability - 0.1
    set security security + 0.05
  ]
  ]
end

;bicycle parts are removed if the owner has finally decided upon
to remove-parts
  ask bikes with [(show? = true) and (random-float 1 < 0.02) and (remove? = not
true)][
      set remove? true
      set desirability desirability - 0.1
  ]
  end

;thieves talk to other thieves
to thieves-talk
  ask thieves in-radius 15 with [(hidden? = false) and (random-float 1 > 0.5)][
```

```
    set crime-probability crime-probability - 0.05
  ]
end

;bike owners talk to other bike owners
to bikes-talk
  ask bikes in-radius 5 with [(show? = true) and (random-float 1 > 0.2)][
    if bikemarking? = true [set marked? true]
    if removeparts? = true [remove-parts]
  ]
end

;open source code that has been adapted from Larry Lin
to set-qvalue[current-xcor current-ycor current-heading new-xcor new-ycor]
  ; Q(s',a') optimal future value
  let optimal-f-val 0

  ;compute optimal future value
  ;finds the maximum reward possible (north, east, south, west)
  ask patch new-xcor new-ycor[
     set optimal-f-val (max (list q-val-north q-val-east q-val-south q-val-west))
  ]

  ;computed q-values
  ask patch current-xcor current-ycor[
    let alpha 0.8 ;learning rate
    let gamma 0.8 ;discount factor
    let reward 10
  if(current-heading = 0)[
    ;; north
    set q-val-north (precision (q-val-north + alpha * (reward + (gamma * optimal-f-val) - q-val-north)) 1)
    ]
  if(current-heading = 90)[
    ;; east
    set q-val-east (precision (q-val-east + alpha * (reward + (gamma * optimal-f-val) - q-val-east)) 1)
    ]
  if(current-heading = 180)[
    ;; south
    set q-val-south (precision (q-val-south + alpha * (reward + (gamma * optimal-f-val) - q-val-south)) 1)

    ]
  if(current-heading = 270)[
    ;; west
    set q-val-west (precision (q-val-west + alpha * (reward + (gamma * optimal-f-val) - q-val-west)) 1)
    ]
]
end

;sets all characteristics and variables of police
to spawn-police
  set shape "person"
  set color blue
  set size 20
  move-to one-of patches with [accessible?]
end

;sets all characteristics and variables of bicycles
to spawn-bike
  set shape "bike"
```

```
  set color green
  set size 20
  set desirability random-float 1
  set security random-float 1
  set stolen? false
  set birth-tick ticks
  set marked? false
  set show? true
  set advised? false
  move-to one-of patches with [accessible?]
end

;sets all characteristics and variables of thieves
to spawn-thief
  setxy random-xcor random-ycor
  set shape "person"
  set color red
  set size 20
  set crime-probability random-float 1
  move-to one-of patches with [accessible?]
  set birth-tick ticks
  set hidden? false
end
@#$#@#$#@
GRAPHICS-WINDOW
217
21
1490
459
-1
-1
1.0
1
10
1
1
1
0
1
1
1
-632
632
-214
214
0
0
1
ticks
30.0

BUTTON
34
196
98
229
Setup
setup
NIL
1
T
OBSERVER
NIL
```

```
NIL
NIL
NIL
1

SLIDER
14
21
186
54
population-size
population-size
5000
30000
12000.0
1000
1
NIL
HORIZONTAL

SLIDER
15
66
188
99
ratio-thieves
ratio-thieves
0
0.01
0.01
0.001
1
NIL
HORIZONTAL

SLIDER
16
108
187
141
ratio-bikes
ratio-bikes
0
0.3
0.25
0.005
1
NIL
HORIZONTAL

BUTTON
113
196
176
229
Go
go
T
1
T
OBSERVER
NIL
NIL
```

```
NIL
NIL
1

MONITOR
154
240
211
285
Police
count policeofficer
17
1
11

SLIDER
16
153
188
186
ratio-policeofficer
ratio-policeofficer
0
0.1
0.1
0.01
1
NIL
HORIZONTAL

MONITOR
10
299
103
344
bicycles stolen
count-total
17
1
11

BUTTON
24
391
194
424
Police warn thieves
set police-thief? true
T
1
T
OBSERVER
NIL
NIL
NIL
NIL
1

BUTTON
24
435
196
468
```

```
Offer locking advice
set police-bike? true
T
1
T
OBSERVER
NIL
NIL
NIL
NIL
1

PLOT
271
493
1077
806
Bicycle thefts over time
time (in ticks)
stolen bicycles
0.0
2545.0
0.0
800.0
true
false
"" ""
PENS
"default" 1.0 0 -16777216 true "" "plot count-total"

BUTTON
23
482
197
515
Bike marking
set bikemarking? true
T
1
T
OBSERVER
NIL
NIL
NIL
NIL
1

MONITOR
112
299
182
344
recovered
recovered
17
1
11

MONITOR
84
239
141
284
```

```
Bicycles
count bikes with [show? = true]
17
1
11

MONITOR
10
239
72
284
Thieves
count thieves with [hidden? = false]
17
1
11

TEXTBOX
75
367
225
385
INTERVENTIONS
11
0.0
1

BUTTON
15
529
214
570
Advise removal of expensive parts
set removeparts? true
T
1
T
OBSERVER
NIL
NIL
NIL
NIL
1

@#$#@#$#@
## WHAT IS IT?

This model is a 2D agent-based model of bicycle theft within a small area in the City of London specifically focussing upon interventions that may be implemented to deter bicycle thefts.

## HOW IT WORKS

Agents are broken down into 3 types: bicycles, thieves and police. It is the sole aim for thieves to steal bicycles within the environment, bicycles are naturally in the environment and police are there to try and limit thefts.

## HOW TO USE IT

Click on the SETUP button to set up the environment. Set the NUMBER slider to change the number of agents within the environment. Click on GO to start the agents moving. Interventions can be activated by clicking on the button of the intervention, remember to setup the model again with the new intervention.
```

```
730
731
732
733 ## THINGS TO TRY
734
735 Change the number of agents within the environment and experiment with the
    interventions provided. To note, multiple interventions are allowed in the model.
736
737 ## EXTENDING THE MODEL
738
739 The inclusion of environmental features for example street lighting, CCTV and bus
    stops.
740 Greater interventions.
741
742
743
744 ## CREDITS AND REFERENCES
745
746 Emily Liu
747
748 Q-learning: Larry Lin, Modelling Commons.
749 @#$#@#$#@
750 default
751 true
752 0
753 Polygon -7500403 true true 150 5 40 250 150 205 260 250
754
755 airplane
756 true
757 0
758 Polygon -7500403 true true 150 0 135 15 120 60 120 105 15 165 15 195 120 180 135
    240 105 270 120 285 150 270 180 285 210 270 165 240 180 180 285 195 285 165 180
    105 180 60 165 15
759
760 arrow
761 true
762 0
763 Polygon -7500403 true true 150 0 0 150 105 150 105 293 195 293 195 150 300 150
764
765 bike
766 false
767 1
768 Line -7500403 false 163 183 228 184
769 Circle -7500403 false false 213 184 22
770 Circle -7500403 false false 156 187 16
771 Circle -16777216 false false 28 148 95
772 Circle -2674135 false true 24 144 102
773 Circle -16777216 false false 174 144 102
774 Circle -2674135 false true 177 148 95
775 Polygon -2674135 true true 75 195 90 90 98 92 97 107 192 122 207 83 215 85 202 123
    211 133 225 195 165 195 164 188 214 188 202 133 94 116 82 195
776 Polygon -2674135 true true 208 83 164 193 171 196 217 85
777 Polygon -2674135 true true 165 188 91 120 90 131 164 196
778 Line -7500403 false 159 173 170 219
779 Line -7500403 false 155 172 166 172
780 Line -7500403 false 166 219 177 219
781 Polygon -2674135 true true 187 92 198 92 208 97 217 100 231 93 231 84 216 82 201
    83 184 85
782 Polygon -2674135 true true 71 86 98 93 101 85 74 81
783 Rectangle -16777216 true false 75 75 75 90
784 Polygon -2674135 true true 70 87 70 72 78 71 78 89
785 Circle -7500403 false false 153 184 22
786 Line -7500403 false 159 206 228 205
```

```
787
788 box
789 false
790 0
791 Polygon -7500403 true true 150 285 285 225 285 75 150 135
792 Polygon -7500403 true true 150 135 15 75 150 15 285 75
793 Polygon -7500403 true true 15 75 15 225 150 285 150 135
794 Line -16777216 false 150 285 150 135
795 Line -16777216 false 150 135 15 75
796 Line -16777216 false 150 135 285 75
797
798 bug
799 true
800 0
801 Circle -7500403 true true 96 182 108
802 Circle -7500403 true true 110 127 80
803 Circle -7500403 true true 110 75 80
804 Line -7500403 true 150 100 80 30
805 Line -7500403 true 150 100 220 30
806
807 butterfly
808 true
809 0
810 Polygon -7500403 true true 150 165 209 199 225 225 225 255 195 270 165 255 150 240
811 Polygon -7500403 true true 150 165 89 198 75 225 75 255 105 270 135 255 150 240
812 Polygon -7500403 true true 139 148 100 105 55 90 25 90 10 105 10 135 25 180 40 195
    85 194 139 163
813 Polygon -7500403 true true 162 150 200 105 245 90 275 90 290 105 290 135 275 180
    260 195 215 195 162 165
814 Polygon -16777216 true false 150 255 135 225 120 150 135 120 150 105 165 120 180
    150 165 225
815 Circle -16777216 true false 135 90 30
816 Line -16777216 false 150 105 195 60
817 Line -16777216 false 150 105 105 60
818
819 car
820 false
821 0
822 Polygon -7500403 true true 300 180 279 164 261 144 240 135 226 132 213 106 203 84
    185 63 159 50 135 50 75 60 0 150 0 165 0 225 300 225 300 180
823 Circle -16777216 true false 180 180 90
824 Circle -16777216 true false 30 180 90
825 Polygon -16777216 true false 162 80 132 78 134 135 209 135 194 105 189 96 180 89
826 Circle -7500403 true true 47 195 58
827 Circle -7500403 true true 195 195 58
828
829 circle
830 false
831 0
832 Circle -7500403 true true 0 0 300
833
834 circle 2
835 false
836 0
837 Circle -7500403 true true 0 0 300
838 Circle -16777216 true false 30 30 240
839
840 cow
841 false
842 0
843 Polygon -7500403 true true 200 193 197 249 179 249 177 196 166 187 140 189 93 191
    78 179 72 211 49 209 48 181 37 149 25 120 25 89 45 72 103 84 179 75 198 76 252 64
    272 81 293 103 285 121 255 121 242 118 224 167
```

```
Polygon -7500403 true true 73 210 86 251 62 249 48 208
Polygon -7500403 true true 25 114 16 195 9 204 23 213 25 200 39 123

cylinder
false
0
Circle -7500403 true true 0 0 300

dot
false
0
Circle -7500403 true true 90 90 120

face happy
false
0
Circle -7500403 true true 8 8 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Polygon -16777216 true false 150 255 90 239 62 213 47 191 67 179 90 203 109 218 150 225 192 218 210 203 227 181 251 194 236 217 212 240

face neutral
false
0
Circle -7500403 true true 8 7 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Rectangle -16777216 true false 60 195 240 225

face sad
false
0
Circle -7500403 true true 8 8 285
Circle -16777216 true false 60 75 60
Circle -16777216 true false 180 75 60
Polygon -16777216 true false 150 168 90 184 62 210 47 232 67 244 90 220 109 205 150 198 192 205 210 220 227 242 251 229 236 206 212 183

fish
false
0
Polygon -1 true false 44 131 21 87 15 86 0 120 15 150 0 180 13 214 20 212 45 166
Polygon -1 true false 135 195 119 235 95 218 76 210 46 204 60 165
Polygon -1 true false 75 45 83 77 71 103 86 114 166 78 135 60
Polygon -7500403 true true 30 136 151 77 226 81 280 119 292 146 292 160 287 170 270 195 195 210 151 212 30 166
Circle -16777216 true false 215 106 30

flag
false
0
Rectangle -7500403 true true 60 15 75 300
Polygon -7500403 true true 90 150 270 90 90 30
Line -7500403 true 75 135 90 135
Line -7500403 true 75 45 90 45

flower
false
0
Polygon -10899396 true false 135 120 165 165 180 210 180 240 150 300 165 300 195 240 195 195 165 135
Circle -7500403 true true 85 132 38
```

```
Circle -7500403 true true 130 147 38
Circle -7500403 true true 192 85 38
Circle -7500403 true true 85 40 38
Circle -7500403 true true 177 40 38
Circle -7500403 true true 177 132 38
Circle -7500403 true true 70 85 38
Circle -7500403 true true 130 25 38
Circle -7500403 true true 96 51 108
Circle -16777216 true false 113 68 74
Polygon -10899396 true false 189 233 219 188 249 173 279 188 234 218
Polygon -10899396 true false 180 255 150 210 105 210 75 240 135 240

house
false
0
Rectangle -7500403 true true 45 120 255 285
Rectangle -16777216 true false 120 210 180 285
Polygon -7500403 true true 15 120 150 15 285 120
Line -16777216 false 30 120 270 120

leaf
false
0
Polygon -7500403 true true 150 210 135 195 120 210 60 210 30 195 60 180 60 165 15
135 30 120 15 105 40 104 45 90 60 90 90 105 105 120 120 120 105 60 120 60 135 30
150 15 165 30 180 60 195 60 180 120 195 120 210 105 240 90 255 90 263 104 285 105
270 120 285 135 240 165 240 180 270 195 240 210 180 210 165 195
Polygon -7500403 true true 135 195 135 240 120 255 105 255 105 285 135 285 165 240
165 195

line
true
0
Line -7500403 true 150 0 150 300

line half
true
0
Line -7500403 true 150 0 150 150

pentagon
false
0
Polygon -7500403 true true 150 15 15 120 60 285 240 285 285 120

person
false
0
Circle -7500403 true true 110 5 80
Polygon -7500403 true true 105 90 120 195 90 285 105 300 135 300 150 225 165 300
195 300 210 285 180 195 195 90
Rectangle -7500403 true true 127 79 172 94
Polygon -7500403 true true 195 90 240 150 225 180 165 105
Polygon -7500403 true true 105 90 60 150 75 180 135 105

plant
false
0
Rectangle -7500403 true true 135 90 165 300
Polygon -7500403 true true 135 255 90 210 45 195 75 255 135 285
Polygon -7500403 true true 165 255 210 210 255 195 225 255 165 285
Polygon -7500403 true true 135 180 90 135 45 120 75 180 135 210
Polygon -7500403 true true 165 180 165 210 225 180 255 120 210 135
```

```
Polygon -7500403 true true 135 105 90 60 45 45 75 105 135 135
Polygon -7500403 true true 165 105 165 135 225 105 255 45 210 60
Polygon -7500403 true true 135 90 120 45 150 15 180 45 165 90

sheep
false
15
Circle -1 true true 203 65 88
Circle -1 true true 70 65 162
Circle -1 true true 150 105 120
Polygon -7500403 true false 218 120 240 165 255 165 278 120
Circle -7500403 true false 214 72 67
Rectangle -1 true true 164 223 179 298
Polygon -1 true true 45 285 30 285 30 240 15 195 45 210
Circle -1 true true 3 83 150
Rectangle -1 true true 65 221 80 296
Polygon -1 true true 195 285 210 285 210 240 240 210 195 210
Polygon -7500403 true false 276 85 285 105 302 99 294 83
Polygon -7500403 true false 219 85 210 105 193 99 201 83

square
false
0
Rectangle -7500403 true true 30 30 270 270

square 2
false
0
Rectangle -7500403 true true 30 30 270 270
Rectangle -16777216 true false 60 60 240 240

star
false
0
Polygon -7500403 true true 151 1 185 108 298 108 207 175 242 282 151 216 59 282 94
175 3 108 116 108

target
false
0
Circle -7500403 true true 0 0 300
Circle -16777216 true false 30 30 240
Circle -7500403 true true 60 60 180
Circle -16777216 true false 90 90 120
Circle -7500403 true true 120 120 60

tree
false
0
Circle -7500403 true true 118 3 94
Rectangle -6459832 true false 120 195 180 300
Circle -7500403 true true 65 21 108
Circle -7500403 true true 116 41 127
Circle -7500403 true true 45 90 120
Circle -7500403 true true 104 74 152

triangle
false
0
Polygon -7500403 true true 150 30 15 255 285 255

triangle 2
false
```

```
0
Polygon -7500403 true true 150 30 15 255 285 255
Polygon -16777216 true false 151 99 225 223 75 224

truck
false
0
Rectangle -7500403 true true 4 45 195 187
Polygon -7500403 true true 296 193 296 150 259 134 244 104 208 104 207 194
Rectangle -1 true false 195 60 195 105
Polygon -16777216 true false 238 112 252 141 219 141 218 112
Circle -16777216 true false 234 174 42
Rectangle -7500403 true true 181 185 214 194
Circle -16777216 true false 144 174 42
Circle -16777216 true false 24 174 42
Circle -7500403 false true 24 174 42
Circle -7500403 false true 144 174 42
Circle -7500403 false true 234 174 42

turtle
true
0
Polygon -10899396 true false 215 204 240 233 246 254 228 266 215 252 193 210
Polygon -10899396 true false 195 90 225 75 245 75 260 89 269 108 261 124 240 105 225 105 210 105
Polygon -10899396 true false 105 90 75 75 55 75 40 89 31 108 39 124 60 105 75 105 90 105
Polygon -10899396 true false 132 85 134 64 107 51 108 17 150 2 192 18 192 52 169 65 172 87
Polygon -10899396 true false 85 204 60 233 54 254 72 266 85 252 107 210
Polygon -7500403 true true 119 75 179 75 209 101 224 135 220 225 175 261 128 261 81 224 74 135 88 99

wheel
false
0
Circle -7500403 true true 3 3 294
Circle -16777216 true false 30 30 240
Line -7500403 true 150 285 150 15
Line -7500403 true 15 150 285 150
Circle -7500403 true true 120 120 60
Line -7500403 true 216 40 79 269
Line -7500403 true 40 84 269 221
Line -7500403 true 40 216 269 79
Line -7500403 true 84 40 221 269

wolf
false
0
Polygon -16777216 true false 253 133 245 131 245 133
Polygon -7500403 true true 2 194 13 197 30 191 38 193 38 205 20 226 20 257 27 265 38 266 40 260 31 253 31 230 60 206 68 198 75 209 66 228 65 243 82 261 84 268 100 267 103 261 77 239 79 231 100 207 98 196 119 201 143 202 160 195 166 210 172 213 173 238 167 251 160 248 154 265 169 264 178 247 186 240 198 260 200 271 217 271 219 262 207 258 195 230 192 198 210 184 227 164 242 144 259 145 284 151 277 141 293 140 299 134 297 127 273 119 270 105
Polygon -7500403 true true -1 195 14 180 36 166 40 153 53 140 82 131 134 133 159 126 188 115 227 108 236 102 238 98 268 86 269 92 281 87 269 103 269 113

x
false
0
Polygon -7500403 true true 270 75 225 30 30 225 75 270
```

```
Polygon -7500403 true true 30 75 75 30 270 225 225 270
@#$#@#$#@
NetLogo 6.1.1
@#$#@#$#@
@#$#@#$#@
@#$#@#$#@
@#$#@#$#@
@#$#@#$#@
default
0.0
-0.2 0 0.0 1.0
0.0 1 1.0 0.0
0.2 0 0.0 1.0
link direction
true
0
Line -7500403 true 150 150 90 180
Line -7500403 true 150 150 210 180
@#$#@#$#@
0
@#$#@#$#@

```