# ODD

## Overview

### Purpose

This model aims to model the movement of Noongar travellers across a cost surface representing landscape and cultural variables within Noongar boodjar. It allows the user to explore how different movement decision making priorities and the input of cultural cognitive maps can impact the path that travellers take. Least-cost path tools in GIS work in a similar way, finding the easiest route on a cost surface. However, GIS models work on the assumption that the entire landscape in perfectly known, and that the traveller will always make the best decision when moving. This model is different as the agent does not know the entire landscape in advance, but instead evaluates the best path only locally, and may not always choose the best route. This is thought to better mimic human movement more accurately. Parameters within the model can be easily changed, thus allowing for broader exploration. This model is designed for researchers, namely anthropologists and potentially also Noongar knowledge holders, to explore how people moved throughout the Southwest in the past.

### Entities, state variables, and scales

Each run accommodates one agent – this can either be interpreted as a single Noongar person, or a family group, known as a ‘traveler’ henceforth. Each traveler is assumed to have the same movement capabilities and cognitive map, and the number of travelers in a simulation is unlimited. Only the traveler can move within each run, and can interact with its surroundings. Each traveler also has a goal, which is there for visualization only. The geographical locations of both the traveler and the goal are determined based on given coordinates.

The model runs across a cost surface, which is developed using a fuzzy overlay within GIS. This cost surface incorporates landscape variables such as elevation and vegetation, as well as cultural variables such as avoidance of sacred sites and distance to water. This cost surface also includes obstacles (which represent water bodies) which the traveller cannot step on. This results in a spatially explicit georeferenced cost surface where every cell (patch) contains a cost value. One patch represents approximately 80m. The agent travels one patch every time step, which does not represent any specific time limit as this was not assessed in this study.

The length of the run depends on the distance between the traveller and their goal as well as the tick limit determined by the user. The traveler moves across the landscape towards the goal attempting to find the most optimal route possible, whilst sometimes making ‘imperfect’ decisions. The model stops when the traveler reaches the goal, or if they get stuck and die when the tick limit is reached.

The variables are presented in Table 1.

*Table 1: the global, patch and turtle variables used in this code*

|  |  |  |
| --- | --- | --- |
| Variable Type | Variable Name | Description |
| Global | csr | The cost surface raster used as the landscape on which the agents travel across, imported as an ASCII file from GIS. |
| p-valids | Used in A\* algorithm. Sets patches that can be accessed (i.e. land only) |
| Start | Used in A\* algorithm. Sets the point where the search algorithm begins. This is the same as the start point for travellers. |
| Final-Cost | Used in A\* algorithm. Reports the cost of the final A\* path. |
| Start-point | The start point of the simulation, where the traveller is created. This can be changed by the user. |
| End-point | The end point of the simulation, or the goal. This can be changed by the user |
| a-star-path | The patches underneath the A\* algorithm path. |
| Patch | Father | Used in A\* algorithm. Stores the previous patch in the partial A\* path that passes through it |
| Cost-path | Used in A\* algorithm. Stores the cost from the starting point to the current patch |
| Visited? | Used in A\* algorithm. A Boolean that shows if the patch has been previously visited |
| Active? | Used in A\* algorithm. A Boolean that says if the patch is active in the search process (it has been reached but its children have not been explored yet). |
| Best-path? | Used in A\* algorithm. Sets the patch value to 10 for patches underneath the A\* path. |
| Access | Access value derived from the cost surface raster. |
| popularity | A counter that counts the number of times the patch has been stepped on in each simulation |
| Accessed? | A Boolean that says if the patch has been stepped on in a run. |
| Suitability | The overall attribute assigned to each patch which the agent assesses to decide its next move. It is a sum of access, distance-from-neighbour and distance-to-path variables. |
| Distance-from-neighbour | The Euclidean distance between the patch and the goal patch |
| Distance-from-path | The Euclidean distance between the patch and the closest A\* path (i.e. best-path?) patch |
| Closest-patch | The closest A\* path (i.e. best-path?) patch from the patch |
| A | Normalised access value |
| DFP | Normalised distance-from-path value |
| DTG | Normalised distance-to-goal value |
| Turtle | Goal | The location of the end-point which the agent moves towards |
| Goal-achieved? | A Boolean which says whether the agent has reached the goal |
| Next-patch | The temporary goal of the agent in its Moore neighbourhood if the rnd primitive is successful |
| Unsuccessful-target | The temporary goal of the agent in its Moore neighbourhood if the rnd primitive is unsuccessful |
| Memory | Records the past patches the agent stepped on during the run. Used to remove the path from the patch popularity counter if the agent is unsuccessful in reaching its goal |
| Alive? | A Boolean that says whether or not an agent is alive. |

### Process overview and scheduling

At each tick, the traveler evaluates the direction in which they need to travel to reach their goal, taking into consideration the accessibility of their surrounding patches and their position in the landscape. At all times, the traveler ha a temporary local target that attempts to bring them closer to the final goal. At the beginning of each tick, the traveler evaluates if they have reached their final goal. If not, the traveler then assesses if its final goal is within its surrounding patches and if it is moves towards it. If the goal is not visible the agent looks at its surrounding 7 patches (Moore neighbourhood) and identify which one it will choose.

At every step, the traveler evaluates its Moore neighbourhood and identifies the suitability of each patch (discussed further below). A roulette wheel selection is applied, where the traveler has the highest chance of moving towards the goal with the highest suitability value but may choose a lower suitability patch. Once the traveler has moved to its chosen patch, the patch gains a popularity value of 1. Once a patch has been stepped on, it cannot be stepped on again for the simulation. This is so that the agent does not get stuck and continues moving forward toward its goal. The path followed by the traveller is represented by a coloured line.

## Design Concepts

### Theoretical and Empirical Background

This model tries to mimic human behaviour in a realistic environment, whilst going beyond the GIS least-cost path approach that assumes perfect knowledge to create the one best path between two points. It relies on the assumption that the perceived difficulty of moving across the landscape can be simulated using a cost surface; the criteria chosen in the cost surface was based off literature review and interviews with Noongar knowledge holders. The agent’s decision-making model is based on multiple hypotheses:

1. The agent does not have perfect knowledge of the landscape, and thus can only assess its local Moore neighbourhood
2. The agent possesses a cognitive map. These assumptions are based of established theories within Traditional Ecological Knowledge (TEK) and psychology (e.g. Simon, 195x), as well as through discussions with Noongar knowledge holders:  
   a. It knows the location of its final goal and will attempt to move towards it at all times   
   b. It has a knowledge of obstacles in the landscape and how to avoid / get around them. This is included via an A\* obstacle avoidance algorithm included in the model.
3. The agent will not always make the optimal decision as in a realistic environment travellers are not “perfect maximisers”. Therefore a weighted roulette selection is included in the model.
4. The agent cannot cross over obstacles and therefore can not step on obstacle patches in any circumstances.

### Individual Decision Making

Each individual traveler makes a decision based on multiple levels:

1. The suitability of each surrounding patch for movement
2. The distance of each patch from the final goal
3. The optimization of each patch to navigate around obstacles.
4. If the patch has been stopped on before in that simulation by the same agent.

There is a bounded rationality behind the agent’s decision making in the model, so agents will select a decision that is satisfactory rather than optimal. Agents make their decisions through a weighted random choice, where any decision may be chosen, but those that are weighted higher (based on its suitability attribute) will be more likely to be chosen. The landscape does not change throughout the simulation so the agents do not adapt their behaviour to any endogenous or exogenous variables. Cultural values play a role in the decision process, as they are included in the access value of the cost raster and also are accounted for in the form of internal cognitive maps within each traveler. Spatial aspects play a role in the decision process as the agent is aware of their position in space and also the location of the goal, and aim to move towards. Temporal aspects were not explicity included in this model.

Uncertainty was included at each time step in the decision making process, as the agent uses a stochastic weighted random decision model to decide which patch to move to.

### Learning

No individual or collective learning is included in the decision process

### Individual Sensing

The traveller is aware of the suitability (i.e. the cost to move to) of each patch in its surrounding Moore neighbourhood as well as the location of the goal in the landscape. It is also aware of patches it has stepped on in the past. However, it is not aware of the actions of other travellers. The mechanisms by which agents obtain information are modelling explicitly through the cost surface raster and limiting information to only the travellers immediate surroundings. The sensing process is not erroneous. The costs for cognition and gathering information are not included in this model.

### Individual Prediction

The traveller can look at patches within its Moore neighbourhood, which implies that they can predict the cost of the next 100m or so. The agent can also predict upcoming obstacles (through the A\* algorithm) and navigates around said obstacle. No other broad-scale prediction is included in this model.

### Interaction

There is no interaction between agents, and the agent only interacts indirectly with patches to calculate their suitability for movement.

### Collectives

No collectives.

### Heterogeneity

Whilst all travelers are the same in their navigational and cognitive capabilities, they are heterogeneous in their decision-making process, as due to the stochasticity included in the weighted random walk model, each agent may choose a different patch at each time step.

### Stochasticity

The weighted random walk is assumed to be partly random, as whilst patches with high suitability values are more likely to be picked, the roulette wheel selection means that any cell can hypothetically be chosen.

### Observation

The interface provides several outputs that help follow model dynamics. *Final-cost* calculates the cost of the A\* algorithm, which can be used to provide an approximate estimation of the distance between the two points, accounting for obstacle avoidance. *Ticks* show the length of each run for each traveller. *Success-rate* reports the number of travelers that successfully reached the goal. Each patch has a popularity counter that counts the number of agents that stepped on that patch. Only agents that reached the goal had their path counted towards the patch popularity score. After a predetermined number of travellers had reached the goal, the simulation is considered finished and the map is saved *(store-output)* as an ASCII file which shows the popularity of each patch. This is exported to ArcGIS for further analysis. The patch popularity can also be visualized within the interface using the *show-popularity* variable.

As more travellers are simulated moving between the two points, patterns of popularity tend to emerge. Whilst there is an element of randomness from the weighted random walk, over time more popular patches emerge that may demonstrate the most likely path, whilst also accounting for the stochasticity of imperfect human decision making.

## Details

### Implementation Details

This model was created in NetLogo (V.6.3.0). It was created over a 7 month period between March and September, 2023. The code can be accessed here: <https://github.com/annachenko/NoongarAB-LCP>

### Initialisation

The landscape is set on a grid matrix with the origin located at the bottom left corner, and the world does not wrap around. The patch size can be manipulated to the users requirements. The access value of each patch in the landscape is determined by the CSR map which is unique for each study site. The map is in ASCII format within the data folder. The world is resized to fit the imported map, and access values of patches are visualized on a colour scale. The start and end points (i.e. significantly cultural sites within the landscape) are imported either as a shapefile or manually. The start point and end point are manually selected by changing the colour of the respective patch. The start and end points change depending on user requirements, but the rest of the initialization stays the same

### Input data

The CSR and site coordinates (optional) are imported into the model in ASCII and shapefile format respectively. Both should be in the same projection.

### Sub-models[[1]](#footnote-1)

***To setup:***  
As explained above, this sets up the initial initialization of the model as well as sets up the variables required for A\* algorithm. Valid patches are set up to be land only, so the algorithm excludes obstacles as valid patches. The start is set as the start point for the simulation, and a turtle is created to draw a path at the end for visualization purposes.

***To-report Total-expected-cost [#Goal]***

This calculates the expected cost for the path to reach the goal from the start point that is calculated by:

*Cost-path + Heuristic*

In every moment every patch has a total expected costs to reach its goal. The *cost-path* is the cost of the real path that goes from the starting point to the patch, and the heuristic is the expected cost provided by the heuristic function, explained below

***To-report Heuristic[#Goal]***

A heuristic search is where all nodes are explored to find that one that is nearest to the goal. The heuristic report is something that is adapted for each patch, and calculates the Euclidean distance (how Netlogo calculates distance) from each patch to the goal.

***To-report A\****

A\* algorithm uses the heuristic and the total expected cost to calculate the total cost of each patch. This is repeated at each time step until the most optimal path is created that successfully navigates around obstacles (See Figure 1).

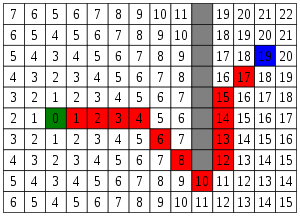


Figure : Example of the A\* algorithm. Image Credit: Swift, 2017

This algorithm receives as input the starting point of the search, the goal of the search and the set of patches that the path is able to visit. These are first cleared to reset the information.

A searching loop is then activated, where the patch looks at all its surrounding valid and active patches in the Moore neighborhood and adds them to an open list, calculating the heuristic and total cost. The original patch then becomes the father patch, and the algorithm moves to the patch with the lowest F cost on the active list and deactivates the father patch. The process is repeated until the goal is reached.

If a path exists, then the list of patches in the path are extracted by using the fathers of every patch. The *cost-path* of the successful path is then reported in the interface.

***To Look-For-Goal***

The path is calculated and the drawer turtle is used to highlight the path using the stamp primitive. The patches under which the A\* path runs are given a value of 10 for the *best-path?* patch variable.

The *best-path?* variable was then set as the *a-star-path*, which were patches with a *best-path?* score of 10.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4.3 | 5 |  |  | 7.2 | 7.8 | 8.6 | 9 | 10 | 9 |
| 5 | 5.8 |  |  | 7.8 | 8.6 | 9 | 10 | 9 | 8.6 |
| 5.6 | 6.4 | 7.2 | 7.8 | 8.6 | 9 | 10 | 9 | 8.6 | 7.8 |
| 6.4 | 7.2 | 7.8 | 8.6 | 9 | 10 | 9 | 8.6 | 7.8 | 7.2 |
| 7.2 | 7.8 | 8.6 | 9 | 10 | 9 |  | 7.8 | 7.2 | 6.4 |
| 8 | 8.6 | 9 | 10 | 9 | 8.6 | 7.8 | 7.2 | 6.4 | 5.8 |
| 9 | 9 | 10 | 9 | 8.6 | 7.8 | 7.2 | 6.4 | 5.8 | 5 |
| 10 | 10 | 9 | 8.6 | 7.8 | 7.2 | 6.4 | 5.8 | 5 | 4.3 |

All patches in a ten patch radius from the *a-star-path* were given a *distance-from-path* value, which calculates the Euclidean distance from their location to the closest a-star-path patch and inverts it, so that the patches further from the A\* variable had a lower *distance-from-path* value (See Figure 2).

If the patch was in a radius greater than 10, the *distance-from-path*  variable was set to 0.

***To set-up-weighted-walk***

The start and end points are defined as patches. At the *start-point,* one traveller is created and set the goal as the end point.

**To weighted-walk**

Figure 2: Example of the distance-from-path variable. As the patches get further away from the A\* path, the variable value decreases. The red path symbolises the A\* algorithm path, and black shading represents obstacles.

If the agent does not reach its goal in the set tick limit (set in the interface), its path is removed from the popularity patch-set (*memory*) and the agent dies.

As the traveller moves along the landscape, the patch popularity *(popularity)* increases by 1 on each patch within the patch-set that the traveller steps on.

If all the travellers in the landscape have reached their goal, or if all the travellers are stuck, the simulation stops. If the traveller is on a patch next to the goal, the agent moves directly to the goal and then dies.

If the agent is not next to the goal, then the traveller undertakes a weighted random decision weighting. Each patch has a suitability/weight attribute that consists of three variables:

(1)

*Where:  
S = Suitability  
A = Access (value derived from CSR)  
DFP = Distance from A\* path (patches)  
DTG = Distance to goal (patches)*

These three variables are derived either from input data, or from the agents position in the landscape. To ensure all variables are weighted evenly, they were all normalized to a 0 to 1 scale:

1. A = Access. Already on a 0 to 1 scale so no further normalization needed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | **Distance to goal** = + 1.4 |
|  | **Turtle to goal** Distance = 10  **Distance from neighbour** Distance = 8.6 |  |  | G |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | 😊 |  |  |  |  |
|  |  |  |  |  |  |

1. DFP = distance from A\* path (described above). As the value ranges were from 0 to 10, each value was divided by 10.
2. DTG = Distance to goal (Figure 3. This consisted of two variables:
   1. *Turtle-to-goal* which is the Euclidean distance from the agent to the goal
   2. *Distance-from-neighbour* which is the Euclidean distance from the potential **patch** to the goal.

By subtracting distance-from-neighbour (dfm) from turtle to goal (ttg) from, you get a discrete value of either   
-1.41421356, -1, 0, 1, +1.41421356 based on Euclidean distances, so this was normalized to a 0 to 1 scale using the following equation:

Figure 3: Visual representation of the distance to goal variable. The closest patch to the goal will have the largest distance to goal value based on Euclidean distances.

*((ttg - dfm) + 1.41421356) / (1.41421356 \* 2))*

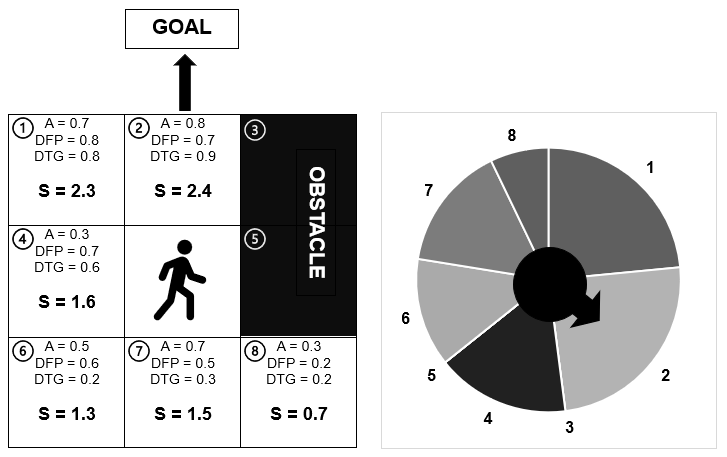
Once all the attributes were normalized, the suitability equation (equation 1) was set, incorporating weight variables that could be set from the interface. Three weighting models were tested in this study (Table 2)

|  |  |  |
| --- | --- | --- |
| Name | Weights | Description |
| Null Model | Access = 1 Distance to goal = 1 Distance from A\* path = 1 | This model places equal weighting on the access variable and the distance to goal variable – the agent will choose a patch equally accounting for its access variable and the closeness to the goal |
| Access Model | **Access = 2** Distance to goal = 1  Distance from A\* path = 1 | This model prioritises the easiest route across a landscape. It has a greater weighting on access than distance to goal – that is, the agent will more likely pick a patch with a higher access value, even if it is further from the goal. |
| Goal Model | Access = 1 **Distance to goal = 2**  Distance from A\* path = 1 | This model prioritises the quickest route across a landscape. It has a greater weighting on the distance to goal than access values– that is, the agent will more likely pick a patch closer to the goal, even if its access values are lower. |

Table 2: The three models created in Netlogo’s Behaviourspace to investigate the most important priority for Noongar travellers across the landscape and assess the most accurate model.

The next-patch (the next patch the agent would move to) was decided using the reporter primitive *rnd:weighted-one-of neighbors with,* only choosing from valid patches that had not been accessed before. This primitive is part of the rnd netlogo extension, which allows for the agent to select its next patch with partial randomness using a roulette wheel selection (Figure 4). If for some reason the agent is unable to use this primitive, then the agent moves to the neighbour with the maximum suitability value. At each time step where the agent moves successfully, it sets the patch as accessed to ensure it does not step on it again in the simulation.

Figure 4: Visual demonstrating the weighted targeted walk as a function of the suitability attribute. Each independent attribute is added together to create an overall suitability value for each patch (Figure 3a). Patches with an overall higher suitability value will be more likely to be picked from a roulette than those with lower values, but any patch (except for patch 3 and 5, which have a suitability value of 0 due to being obstacles), has a chance of being picked (Figure 3b)



**A**.

**B.**

***To show-popularity***

Used to visualize the popularity in the interface by setting the patch colour on a colour scale based on popularity values.

***To-report success-rate***

Reports the popularity of the patch underneath the goal patch. This reporter tells the user the number of agents that have successfully reached the goal

***To-report boorna-gnamma-popularity***

This was only used in initial testing and sensitivity analyses. Same reporter that success-rate, that tells the user the number of agents that have stepped on a certain patch (in this case, the patches underneath Boorna Gnamma location coordinates)

***To store-output***

Stores the output of the patch popularity as an ASCII file.

1. All A\* submodel explanations are referencing Caparrini, 2018. [↑](#footnote-ref-1)