QUANT2: AI Generated Transport Scenarios

richard.milton

November 2021

1 Abstract

Using the QUANT travel to work model and a scenario testing framework which models changes as moves in a game, where the object is to maximise the benefits while reducing costs, different computer generated strategies for infrastructure scenarios are tested. The game playing framework also gives an estimate of the complexity of the scenario space under different strategies and how it varies with strategy.

The QUANT model is used to calculate the impacts of scenarios, which uses flows of commuters travelling to work along the road, bus and rail transport networks. The networks compete for commuters, allowing mode change scenarios to be tested, along with employment changes.

By running 60,000 employment scenarios as part of a carbon reduction strategy, a Monte Carlo search algorithm is used to find solutions where working population increases or stays neutral, while the total number of road kilometres driven decreases. This is a road to rail mode switch driven by changing the geographic distribution of jobs.

Changes made to the underlying road, bus or rail networks also give rise to impacts for the working and residential populations. We present a fast, approximate, method for recalculating the gravity model's shortest paths costs on the changed networks. This makes it possible to run ensembles of computer generated network scenarios at scale. The motivation here is to test whether making lots of small changes to networks can generate as much overall impact as a single, large scale, infrastructure project. The comparison with existing infrastructure projects is essential to establish a baseline, so the High Speed 2, Crossrail and CaMKOx projects are introduced. Methods and metrics for comparing real world infrastructure projects with computer generated ones are developed and used to appropriately size the computer generated projects to follow the approximate size of Crossrail.

The final part of the analysis takes the five computer generated network topologies of 'random', 'star', 'ribbon', 'random walk' and 'dense', and runs 1,000 scenarios each. The results of running random, geographically spread, network changes compared to star, ribbon and dense networks is used to show whether the small changes hypothesis is valid.

2 Background

TODO: You really need to write this section [8]. and [1] todo: find refs on geometry of model scenarios and which topologies are best. ribbon development? Shortest paths overlay graphs.

3 Introduction

The basic QUANT model is defined by the following equation 1:

$$T_{ij} = \frac{O_i D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}} \tag{1}$$

where,
i, j origin and destination zone numbers
T_{ij} flow of people between zone i (origin, work location) and j (destination, home location)
O_i is the total number of jobs in a zone, defined as: $\sum_j T_{ij}$
D_j is the total number of residential locations in a zone, defined as $\sum_i T_{ij}$
β willingness to travel model parameter (calibrated)
C_{ij} travel time between i and j in minutes (cost)

There are 8436 zones in the model with i an origin zone and j a destination zone based on MSOA areas in England, Wales and Scotland. The model propagates flows of workers from work zone (i) back to their residential zone j. The total flow of workers between zone i and j is T_{ij} and the travel time between the zones, C_{ij} is used as the travel cost.

Three travel modes are used: road, bus and rail. Each mode has its own T_{ij} trips matrix, C_{ij} cost matrix and β parameter. The O_i and D_j totals for the zones are across all modes, so each travel mode competes for workers. This allows for the testing of employment and transport scenarios by changing the inputs to the model and recalculating T_{ij}.

Changing O_i (total jobs) is an employment scenario. Changing C_{ij} (travel time) is a transport scenario.

Employment scenarios can attract people onto the rail if employment is increased where rail travel is accessible. The results are expected to show net change in jobs resulting in differences in road KM travelled geographically.

The basic network scenario premise is that adding rail network speeds up rail journeys. Car and rail compete with each other for commuters, so faster rail attracts people off of the roads and results in fewer kilometres of road travel.

This is what we are looking for in the results:

Rail network (gives rise to) ⇒ faster travel times (attracts) ⇒ road transport (results in) ⇒ fewer road KM.

In the data the expected result is to correlate amount of network changes with a drop in road KM.

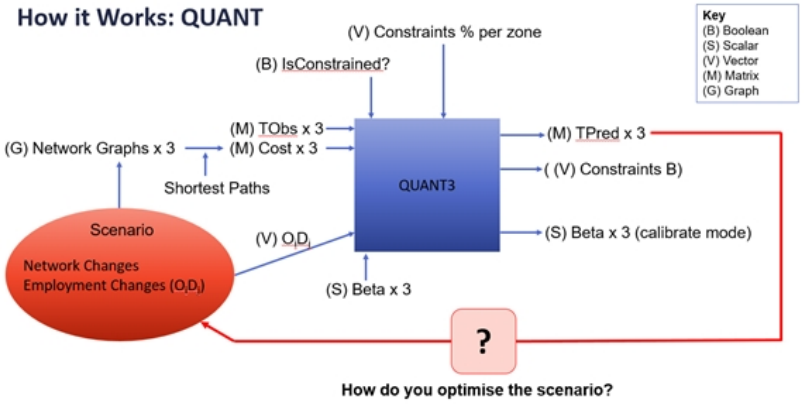


Figure 1: QUANT Dataflow Diagram. The "TObs" matrix is the original observed flow data, which the "TPred", predicted matrix fits via the "Beta" parameters. Once this is calibrated, then a scenario can be run.

3.1 Types of Scenarios

As shown in figure 1, scenarios are expressed in terms of an altered transport network, leading to recalculated trip costs, or alternatively by changes to the residential population or employment. A model is a simplification of real life, so the interface for urban planners to define scenarios must reflect this real world to model world mapping. This not a simple one to one fit, as summarised in table 1.

4 Methodology

The aim is to accelerate the QUANT scenario testing system using the concept of immediate urban modelling [8], allowing thousands of scenarios to be run across the search space. The hypothesis is that this will lead to new discoveries about how to generate new scenarios, while allowing a comparison between existing scenarios, for example HS2, Crossrail and CaMKOx, with the computer generated synthetic ones.

The generation of synthetic scenarios is built around a game framework. The point about seeing it as a game is to make the code re-usable so we can try lots of different ideas using the same framework. This is a single player game, although you can imagine two instances trying to beat each other (GAN).

Figure 2 shows the class diagram for the game interface: *newGame*, *isTerminalState*, *possibleNextMoves* and *makeMove*. Not shown is the game state,

#	*Type*	*Scenario*	*Road*	*Bus*	*Rail*
1	E	Employment (e.g. new airport)	×	×	×
2	N	New Public Rail Infrastructure	×	×	✓
3	N	Mode shift, carbon reduction	✓	✓	✓
4	N	Self-driving cars	✓	×	×
5	N	New motorway speed limit	✓	×	×
6	N	Congestion charging	✓	×	×
7	N	New road bridge or tunnel	✓	(✓)	×
8	E	Covid-19 essential workers, 15% public transport	×	×	×
9	E	Homeworking, telecommute	×	×	×
10	N	Drone transport (new mode)	(✓)	(✓)	(✓)

Table 1: Example scenario types and their requirements to re-run network shortest paths. Road takes approximately 2 hours to run, bus 45 minutes and rail 2 minutes. For scenario type, 'E' denotes an employment scenario and 'N' a network scenario.

which is unique to each type of game.

As explained previously, there are two types of scenario: an employment scenario or a network scenario. While QUANT actually runs both together as a combined scenario, this paper separates the two in order to simplify analysis. Both scenarios use an interface based on 'moves', where a move involves picking a zone from a list of legal moves based on the current game state. The 'maxDepth' field in the interface determines how many moves are made before the scenario is complete and 'isTerminal' returns *True*, triggering the QUANT scenario to evaluate the position. As an example, the 'random 10' network scenario picks 10 random zones, drops the travel cost of the five links connecting them and then re-computes the impact statistics of the scenario.

Listing 1 shows the pseudo-code for running a scenario game. The "possibleNextMoves" function is overloaded to define different move strategies. This allows for comparison between different strategies to determine which is best. At first, both the employment and network scenarios begin with a random strategy in order to set a baseline that alternative strategies can be compared against. Comparisons require a scoring system, which is covered in section 5.2, "Metrics".

The following two sections of "Employment Changes" and "Multi-layer Multimodal Networks" explain the two types of scenarios in more detail and go into the optimisation schemes used. However, combinatorics governing the size of the search space and the move fan-out can be defined directly.

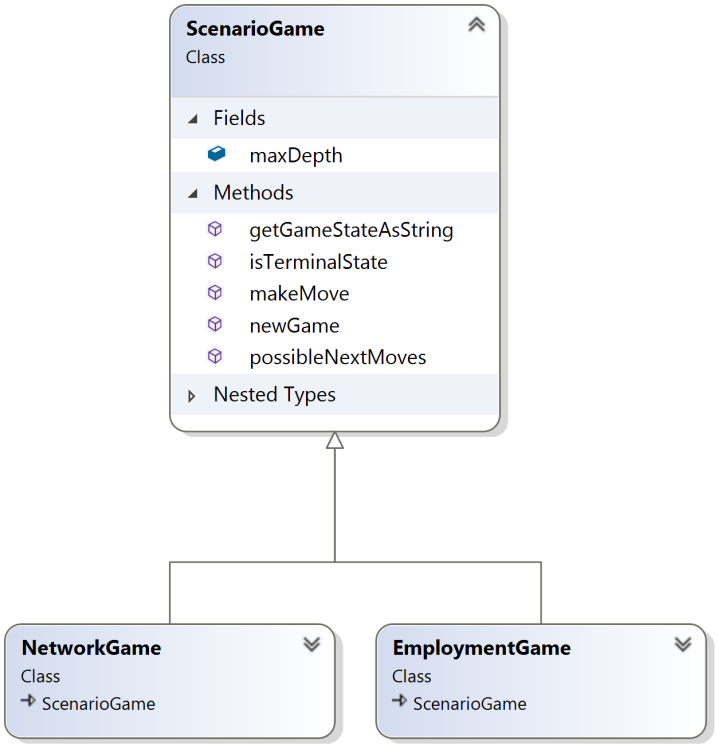


Figure 2: Scenario game class diagram.

A random employment scenario with $maxdepth = 10$ picks a random zone from 8436 zones and either moves the employed population up or down, giving the following formula 2:

"Employment = (2x8436)x(2x8435)x(2x8434)... ten times"

$$S_e = 2^{maxdepth} \times \prod_{m=0}^{maxdepth-1} N_z - m = 1.90392E + 45 \ (maxdepth = 10) \quad (2)$$

A random network scenario with $maxdepth = 10$ picks 10 random nodes, half of which are origins and half destinations, then connects them with five lower cost links, giving the following formula 3:

"Network = (8436x8436)x(8436*8436-1)x(8436*8436-2)... ten/2 times"

$$S_n = \prod_{m=0}^{maxdepth/2-1} N_z^2 - m = 1.82543E + 39 \ (maxdepth = 10) \quad (3)$$

```
game = newGame()
while not(isTerminalState) {
    legalMoves = game.possibleNextMoves()
    #pick a move from legalMoves
    game.makeMove()
}
```

Listing 1: Game Pseudo Code

NOTE#1: the depth of a network scenario refers to the number of nodes picked, which is twice the number of links, therefore $maxdepth/2$ is used in equation 3.

NOTE#2: we're allowing intra-zone changes in networks, which are perfectly valid.

The search space is big. Too big to search exhaustively on most computers.

The plan is to use artificial intelligence and strategy to narrow the search space.

Figure 3 shows the types of network scenario. These are defined as follows:

Random (no limits) pick two random nodes any distance apart and connect them. Repeat for however many nodes are specified by *maxDepth*.

Random (limitr) same as *Random (no limits)*, but limits the distance between the linked nodes. Picks $maxDepth/2$ random pairs of nodes with the constraint that the connecting links between pairs must be less than the radius limit e.g. 30KM apart. NOTE: with a high depth and small limit radius, this is the "Small Changes" scenario with lots of small changes spread across England, Scotland and Wales.

Star picks a single central node and then connects $maxdepth/2 - 1$ nodes randomly, within a set distance from the centre point (radius). Connections are formed as $a \rightarrow b$, $a \rightarrow c$, $a \rightarrow d$ etc. to form the star topology.

Random Walk picks a starting point and then connects randomly to nodes within a distance radius. Connections follow the pattern of $a \rightarrow b$, $b \rightarrow c$, $c \rightarrow d$ etc. to make a continuous line. The line is permitted to cross itself, but not to double back. If the algorithm becomes stuck, then the line is truncated before $maxDepth$ is reached.

Dense picks an initial centroid point and then randomly connects pairs of nodes within a set radius of the centroid, up to the point where max depth is reached.

Ribbon is similar to the random walk, but there is a limit on angle between pairs of nodes so the path forms a ribbon. The angle is recalculated between last node and next node, so the path can bend back on itself. NOTE:

this can cause the algorithm to become stuck. The distance between pairs of nodes is limited to a maximum radius.

Calculating the number of possible combinations mathematically when the radius is limited and different topologies are specified is not feasible as it requires the geography of the UK to be taken into account. However, one of the reasons for using the game framework is that it calculates the number of possible legal moves as the game progresses, giving a direct count of the total number of scenarios possible. For example each scenario starts by picking a starting node from the full set of 8436. Then, the destination move is limited by radius and topology, so might only have a move branching factor of, for example, 300. Then 8436x300 is the total number of possible combinations that this single two move scenario was picked from.

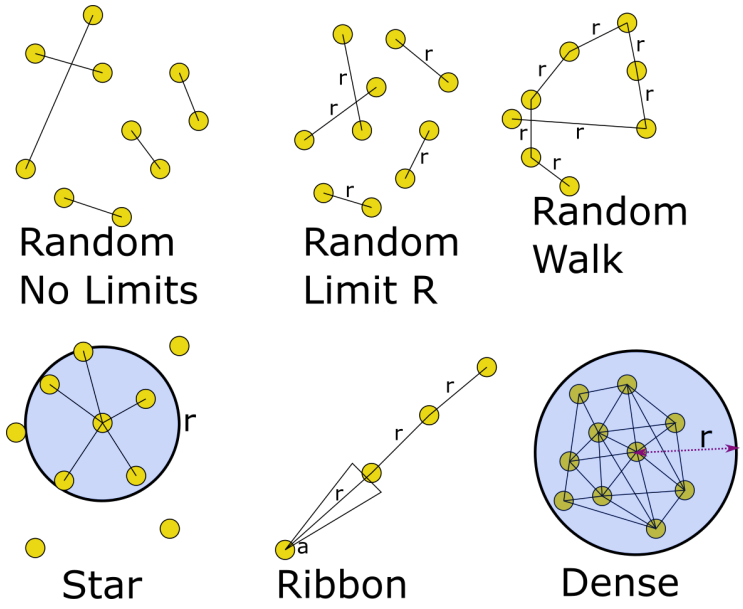


Figure 3: Different network topologies used for testing types of scenarios

The network topologies in figure 3 have been specifically chosen to complement each other. "Dense" and "Star" are similar in that they create the same number of links within a radius of a point, so any differences in impact data are likely to be a result of connected segments, rather than link speed. Similarly, "Random Walk" and "Ribbon" and designed to test whether a geographically long ribbon generates more impact than an identical length random walk around a compact area. Given that the average distance a random walk moves from its starting position after n moves is the root mean square, then we can expect the walk to travel $\sqrt{r * n} = \sqrt{(30 * 20)} = 24.5KM$ for a 30KM radius and 20 moves. However, each move is limited to a maximum of radius of r KM, so the practical value is likely to be less. There is no statistic in the data to show the

distance between the first and last point, but the $LBar$ statistic (equation 12) is a measure of the geographic spread of the points.

The "pure" random network is a control test designed to quantify the impacts of the other network topologies. In addition to this, there is the question of whether adding small changes distributed all across the country can generate more impact than the same number of changes located in a concentrated geographic location. This is the "small changes" hypothesis, which can be answered with a combination of random, dense and ribbon network topologies. Looking at TfL's map of Crossrail, the topology is a ribbon with 33 links plus a couple of spur points, while CaMKOx is similar, but with fewer links. High Speed 2 phase 1 is one long ($> 160KM$) link from London to Birmingham followed by some local links.

If 'limitr', where 30KM links are placed anywhere in the country, has higher impact than 'dense', where links are concentrated in one area, then it is evidence for the 'small changes' philosophy where a single, large intervention like HS2 or Crossrail could be split into many smaller project spread across the whole country to provide the same or greater impact.

Similarly, 'star' and 'ribbon' geometries test whether more impact comes from linking areas along a ribbon development, or linking satellite areas back to a single point of concentration generates the most impact.

The 'randomwalk' geometry in an interesting one as theory states that the random walk will only travel an average distance from its starting point, resulting in a 'dense' type of development, but with the path linked as in the 'ribbon'. The comparison between 'randomwalk' and 'ribbon' tests whether it is the distance of the ribbon that is important, or whether concentration of interventions in one location is favoured over the 'small changes' hypothesis (i.e. 'limitr').

These questions are explored further in the results section.

4.1 Employment Changes

The original QUANT model computed the full T_{ij} matrix comprising 8436×8436 computations of formula 1 every scenario run. When dealing with AI scenarios, only the scenario changes are needed, which leads to a number of performance improvements which are explained in detail here. In real terms, this is the difference between an employment scenario run taking 20 seconds, or being able to do 10 runs per second, an improvement of approximately 100 times.

The scenario delta is the difference between the full scenario run T^s_{ij} with the scenario changes and the baseline model with no changes T^b_{ij}. This is shown in equation 4.

$$T^\delta_{ij} = T^s_{ij} - T^b_{ij} \tag{4}$$

For an employment scenario, T^s_{ij} and T^b_{ij} differ only by a changed O_i value representing total employment in each zone, which we can define as $O^s_i = O_i + \delta_i$.

Then the scenario trips calculation is simplified as the baseline trips (T^b_{ij}, pre-calculated) plus the scenario delta, as follows:

$$
\begin{aligned}
T_{ij}^{s} &= \frac{(O_i + \delta_i) D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}} \\
&= \frac{O_i D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}} + \frac{\delta_i D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}} \\
&= T_{ij}^{b} + \frac{\delta_i D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}}
\end{aligned}
$$

This is the total number of trips for the employment scenario. In order to calculate the difference (δ), using equation 4, this simplifies to:

$$
T_{ij}^{\delta} = \frac{\delta_i D_j e^{-\beta C_{ij}}}{\sum_j D_j e^{-\beta C_{ij}}} \tag{5}
$$

The algorithm to calculate the scenario difference from the baseline can then make use of a further optimisation. By noting that:

$$
\delta_i = 0 \implies T_{ij}^{\delta} = 0 \tag{6}
$$

These zero change i values can be skipped, making a substantial computational saving when $8436 \times 8436 \approx 71\ million\ Tij$ combinations need to be calculated.

The implication of this is that only the row and column where the origin zone employment is changed (i) and the destination zone (j) need to be computed. For the purposes of determining performance, the number of calculations of T_{ij} can be approximated as $8436 \times 2 \times N_z$, where N_z is the number of changed zones. This ignores where rows and columns intersect, but is a useful simplification for $N_z \ll num(zones)$. Taking the Government's Heathrow model as an example, there were 125 employment zone changes. The number of calculations this requires, relative to computing the full T_{ij} matrix are as follows:

$$
\frac{8436 \times 2 \times 125}{8436 \times 8436} = \frac{2,109,000}{71,166,096} (\times 100\%) = 2.96\% \tag{7}
$$

Achieving the same answer in $\approx 3\%$ of the time is a valuable saving when hundreds of thousands of scenarios need to be run to give adequate coverage of the results space.

It follows from this result that any calculations involving differences deriving from trips changes can be optimised in the same way. For example, road KM driven which relates to carbon footprint, or commute time saved (or lost). This is an important result when developing AI algorithms that rely on metrics and gradients to decided if changes are good or bad when optimising scenarios. This results in more potential scenarios being tested in the time available and more coverage of the search space.

4.2 AI and Optimisation

The use of TensorFlow and a matrix representation of the QUANT modelling equation 1 are detailed in [8]. The main point to note is that a T_{ij} calculation is *pure parallel*, as no T_{ij} matrix entry relies on any previous calculations. A GPU with a large number of parallel cores yields an impressive speedup over a multi-core CPU. Calculation times for a T_{ij} matrix are approximately 1 second for an NVIDIA GTX 980, compared to 30 seconds for a CPU version on an Intel i7 2.3GHz implementation with parallel optimisations, maximising all 16 cores. The QUANT code contains unit tests for a naive CPU implementation, which is used as both a benchmark test of speed and to demonstrate correctness of any improved CPU or GPU algorithms.

The speed of computation becomes important when large numbers of scenarios are run in order to test the different optimisation schemes for computer generated scenarios. The "Results", in section 5, presents data showing how effective optimisation schemes are based on following T_{ij}, C_{ij}, O_i and D_j. This is to determine what information is most important to show a human user making a new scenario, but also as a way to narrow down the huge search space of scenarios. The following section details the optimisations and approximations needed in order to make running large numbers of network scenarios feasible. It needs to be stressed that approximations to enable faster traversal of the search space are reasonable, because, once a good solution is found, the full algorithm giving the exact result can be run at leisure in order to verify the approximate result. As long as the approximate search algorithm is following the solution gradient in the same direction as an exact solution would, then the absolute numbers returned are not important as the path followed leads to an improved result.

4.3 Two Layer Multi-Modal Networks

The transport networks used in QUANT are rail, bus and road. A full, all pairs shortest path (APSP), calculation to determine the cost matrix, C_{ij}, takes two hours, while rail takes about 3 minutes (see table 2). By running an approximate shortest paths algorithm, this is reduced to about 30 seconds for all transport modes when there are 20 network changes.

The methodology uses the fact that the gravity model's zonal network, projected onto the full transport network for each mode, is an overlay graph (see [10]). However, having a multi-level overlay graph does not allow for fast recomputation of shortest paths when the underlying network is changed. The fast, modified APSP algorithm, of Demetrescu and Italiano [3] is attractive, but the memory requirements are too high for networks the size of the UK. The key to solving the problem is to accept that an approximate cost change is adequate when the cost only forms part of the gravity model's new T_{ij} calculation. By restricting network changes to be in the zonal overlay graph, the pre-computed all pairs shortest path cost matrix for the zones can be re-used.

Starting with the raw networks for the road, bus and rail, these are derived

from the the Ordnance Survey road network containing road speed limits and the bus and rail timetables. The networks are simple point to point, road, bus or rail segment timings where the weight on a link is the amount of time to traverse the link in minutes. For the timetabled networks of bus and rail, the transit time for a link is simply the transit time for the link averaged over the whole day.

Then the all pairs shortest paths is run on each to get three fully connected networks of $8436 \times 8436 \approx 71\,million$ links, one for each transport mode. These are guaranteed shortest paths between zones which are used in the model. This uses either the Dijkstra algorithm for CPU, or the NVIDIA nvGraph library for GPU.

The problem: we need to change the underlying networks to simulate building a new road, bus or rail link. But, the lower level network changes feed through into the fully connected shortest paths networks at the higher, overlay graph, level.

If we want to run lots of network scenarios, but it takes over two hours per scenario, then the tens of thousands of scenario runs for employment scenarios is impossible.

A faster method is to make the changes at the higher network level, but it's not a straightforward case of changing a single link cost. Changing one link has the consequence of changing many neighbouring shortest path times, often running into the millions. This is shown clearly in figure 4 where the effects of five random changes to the network are plotted. The secondary effects of these five changes are running into the tens of millions, making the optimisation of network scenario algorithms a challenge.

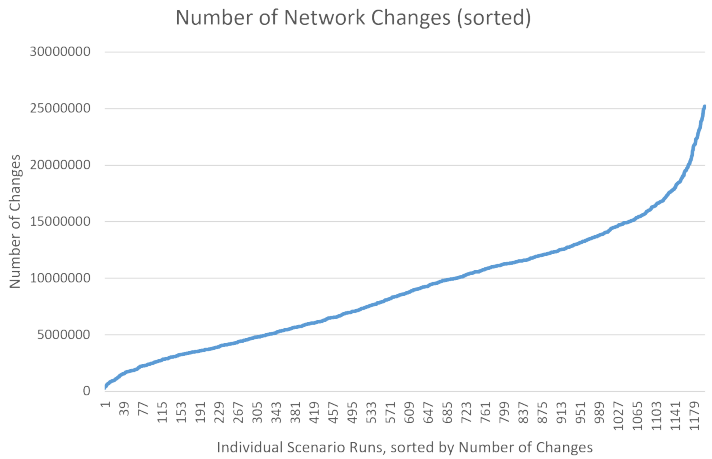


Figure 4: When five random changes are made to the rail network, the secondary effects can run into the tens of millions. This data is for the random network scenario, with all the individual 1200 model runs on the X axis sorted by number of network changes (y axis data).

This is an example of "Modified All Pairs Shortest Paths", where we want to know how all pairs of combinations of shortest paths between nodes is affected when a single link is changed. There are algorithms that can do this, but it's a trade-off between computational speed and storage space.

The solution we use for the QUANT scenarios makes use of the fact that we already have a fully connected shortest paths network calculated.

If the transport network for the modes of road, bus and rail is defined as G_m over V_m vertices and E_m edges, where m is the mode:

$$G_m = (V_m, E_m) \tag{8}$$

This gives us three modes graphs derived from the physical road network, the bus time tables and bus stop locations and the rail time tables and station locations. The link weights are taken as the time in minutes to traverse the link and are directional. While the geographic coordinates of all the vertices are not needed for generating the cost matrices, they are retained in separate lookup tables for when the user needs to edit the network to build a scenario.

The QUANT model requires a cost matrix which contains shortest paths times between all 8436 QUANT model zones. This is computed in between approximately 2.5 hours (road) or 3 minutes (rail) on dual GTX980 GPUs using NVIDIA's nvGraph library. The data in table 2 shows the timings for the three modes on a desktop GPU and compared to a Microsoft Azure virtual machine GPU.

Computer	Type	nodes	edges	SSSP	APSP (est)
Desktop	Rail	9,254	2,595	21.5ms	3.01 mins
Azure	Rail	9,254	2,595	7.89ms	1.1 mins
Desktop	Bus	1,980,701	332,720	1024.89ms	143.48 mins
Azure	Bus	1,980,701	332,720	338.3ms	47.36 mins
Desktop	Road	3,527,606	8,401,778	1038.69ms	146.04 mins
Azure	Road	3,527,606	8,401,778	432.79ms	60.85 mins

Table 2: Single source shortest path (SSSP) and all pairs shortest path (APSP) timings for rail, bus and road for desktop GPU and Azure cloud GPU. APSP timings are estimated from the SSSP timings by multiplying by the 8436 origin zones.

This gives us S_m, which is a fully connected shortest paths graph (matrix).

Even running just rail scenarios in around 3 minutes is limiting in the number of automated scenarios that can be run. Running the 60000 scenarios from the previous employment scenario section would amount to 125 days of computing, while road scenarios are obviously an even bigger problem.

The solution used to accelerate and automate network scenarios is to make the changes in the model space and then relate changes back to the physical world later. In other words, modify S_m directly.

By noting that where S_m is fully connected and its edge weights are guaranteed shortest paths:

$S_m(i,j)$ is minimum for any path of $G_m(i,j)$

If we want to add a new shortest path, maybe to simulate a new road or railway being built, then we define the path as follows:

$X \to Y$ and $t(XY)$

So, directly, the shortest path between the origin (X) and destination (Y) vertices can be changed:

$S_m(X,Y) = t(XY)$ (assuming $t(XY) < S_m(X,Y)$)

However, by modifying this single link there can be consequences for other shortest links.

Noting that S_m contains only guaranteed shortest paths between zones, then any new shortest paths, created as a consequence of modifying $S_m(X,Y)$ can be identified as follows.

For all pairs of $a \to b$ in zones:

$$t(a,X) + t(XY) + t(Y,b) \quad < t(a,b) \tag{9}$$

Alternatively, using the pre-computed shortest paths matrix:

$$S(a,X) + t(XY) + S(Y,b) \quad < S(a,b) \tag{10}$$

The algorithm for this is shown in listing 2.

```
for l in newlinks:
    S(l.x,l.y) = l.t #update new XY link time
    for a=0 to n:
        for b=0 to n:
            #testing going from a to b via our
            #new link l which is l.x to l.y
            newT = S(a,x) + l.t + S(y,b)
            if newT < S(a,b):
                #update with new shortest time
                S(a,b) = newT
            endif
        endfor
    endfor
endfor
```

Listing 2: Shortest Paths Pseudocode

Although this is n^2 improved path tests per link added, all of the computationally intensive shortest path calculations are removed due to the fact that this is pre-computed in $S(a,b)$. This must be repeated for each link changed, so, in terms of speed, it scales linearly with the number of links added, $\mathcal{O}(l)$, and is $\mathcal{O}(n^2)$ in number of zones.

There is one major caveat to this method, though. It only finds *improved* links, so, for example, a scenario involving a closed road bridge where the link time is changed to infinity is not possible.

An auxiliary question is how changes made at the overlay graph level correspond to real world changes in the transport network graph. However, QUANT is a high level planning tool, which is deisgned to answer the question of what would happen if, for example, the time between London and Birmingham was reduced to 45 minutes, without worrying about the the lower level detail of where exactly to put new track or roads, or how to modify timetables. While the link between MSOA zones at the overlay level can be put into the real transport network directly and a full all pairs shortest path algorithm run to compute the new cost matrix, this will give different results from listing 2 above. The search algorithm does not care about the absolute numbers, only about improvements. The approximate algorithm shows where the improvements are being made, while it can be proved that the real network changes necessary would yield equal or greater benefit[1]. For a non-perfect search of large numbers of possible scenarios, this method is adequate.

From the point of view of computer performance, one fact that can often be overlooked is the format and storage of the networks in memory and how much time is required to make changes to them. The structure used by NVIDIA's nvGraph library is a packed memory structure containing the node and link data. In order to change this to add a new link then the structure either needs to be re-created from scratch, or use a process of moving large blocks of data around in memory in order to make space is required. This is used rather than a linked pointer structure because the block of network definition data needs to be moved to the GPU cache memory in order for the nvGraph library to be able to run on chip. The alternative method of storing the data as an adjacency matrix is impractical as it would require a $v \times v$ matrix of links, where v is the number of vertices, which can be 8 million. For sparse networks like road, bus and rail, this type of adjacency matrix is not used.

Finally, after a cost function, C_{ij} has been calculated, there is the calculation of a new trips matrix, T_{ij} which is used to compute the impacts of the new scenario. This is not optimisable in the way that the employment scenarios were, because changes are not limited to a small number of zones (see the graph in figure 4). It is possible to track which zones are affected by changes and which are not via the algorithm in listing 2, but the speedup achieved is not worth the effort. It is simpler to run the full T_{ij} calculation for all the zones and rely on the GPU gravity model code to speed up the computation to around 1 second (or around 30 seconds for the CPU equivalent). While this is a valuable speedup over the naive approach, the implication is that the nature of network scenarios means that they are always going to be an order of magnitude slower

[1]To go from a link in the shortest paths overlay to the real world, simply place the direct MSOA to MSOA link into the real transport network and re-run the APSP to get a new shortest paths overlay. However, a direct link may not be feasible in the real world, so physical constraints and additional connections will cause the two shortest path cost solutions to diverge.

than employment scenarios.

One final point which has not been mentioned so far is the idea of using a priority sorted T_{ij} queue to speed up an approximate version of the gravity model calculation. Many of the changes being made are very small and so do not have much bearing on the final output. If it were possible to compute the network changes to C_{ij} and resulting trips changes to T_{ij} in a priority sort order, then time can be saved computing small changes that have no bearing on the final results. This is future work.

5 Results

Employment scenarios and transport scenarios were run using different strategies to see how the outcomes were affected. Both focused on reducing the number of road trips, which are seen as a proxy for carbon reduction.

The logic behind the experiment is as follows: for a rail scenario, there is a mode switch, where more attractive rail costs attract people to the faster routes, which causes a reduction in the number of road KM driven. The amount of network KM added to the rail network is a proxy for cost, while reduced road KM is a proxy for benefit, as it represents a net gain in low carbon road transport.[2]

Geography is an important factor which needs to be taken into consideration. As will be explained shortly in section 5.1, the number of possible scenarios is extremely high, but, by placing sensible geographic limits on scenarios, the total number of combinations can be reduced. For employment scenarios, this means concentrating changes within a geographic location, for example, not accepting solutions where employment in London is being transferred to Glasgow. With the network scenarios, this means limiting the topology and distance that links can be made.

The rest of the results section is structured as follows. Section 5.1 covers the combinations and permutations of network scenarios and shows how this can be reduced to manageable levels. Then, section 5.2 lists the metrics and impact statistics produced by the scenarios to measure the cost, benefit and geographical restrictions. In addition to this, the section also covers how to make comparisons between real world scenarios like HS2 and Crossrail and computer generated scenarios, which gives some context to the scenarios that are being generated. The chapter finishes with the final results from the employment scenarios, the transport network scenarios and comparisons between different topologies of network scenarios.

5.1 Total Number of Network Scenarios

Table 3 shows the total number of combinations obtained for each topology after running the model game 30,000 times. The figure given is an average over all the 30,000 runs, so ignores the inherent variations. It is important to note that the full spatial interaction model does not need to be run to calculate the move combinations, only the game framework which is choosing the moves. By using the game framework to split the work in this way it is possible to run over 100 tests per second, when a single, full, spatial interaction model run including impacts would take around 30 seconds.

The numbers for the move combinations are too large to visualise, so the branching factor[3] is used instead. This is defined as the average number of legal

[2]NOTE: a conversion of cars from diesel and petrol to electric vehicles achieves the same aim, but is a different type of scenario that QUANT could also run.

[3]An alternative term, which is sometimes used in place of "branching factor", is "fan out".

moves that are possible along the full sequence of the 20 moves that make this set of scenarios.

Topology	Move Combinations (y)	Branching Factor (b)
no limits	3.33e+78	8436
dense	6.76e+60	1100.27
limitr	1.79e+64	1631.50
ribbon	0.37e+20	11.13
rndwalk	1.34e+33	45.32
star	2.29e+33	46.56

Table 3: Move combinations and branching factor for 20 move scenarios.

NOTE: we assume that the total number of moves is $y = b^m$ for an average branching factor, b, which is equal across the whole move depth and yields a total number of combinations of y. Then the average branching factor is given as $b = y^{(1/m)}$ for $m = 20$ moves.

The "no limits" topology simply picks two random nodes to connect, regardless of distance. The branching factor is 8436, which is the number of zones in the model. The total number of combinations of $3.33e + 78$ is the maximum possible when picking 20 random nodes from 8436, which is 8436^{20}. All the limited topologies are lower than this, based on how limiting the constraints are. The "ribbon" topology is the lowest, because its 30 degrees constraint on ribbon angle results in fewer legal nodes per move, when "rndwalk" and "star" can pick from the full 360 degrees. The "limitr" and "dense" topologies are just subsets of "no limits", but with link distance limits applied.

What the figures in table 3 demonstrate is that, even with a fast computer and an even faster QUANT model algorithm, there is no hope of ever testing all the possible scenarios to find the best ones. There has to be a reliance on AI to find the best moves, in a similar way to how AI is used in complex games like Chess and Go. For reference, Chess branching factor is about 35, while the Go branching factor starts at 361 and reduces with play [9, p. 185]. Branching factor is important in determining how good various heuristics are when presented with a large search space of moves.

5.2 Metrics

One of the reasons for speed is the choice of metrics, which are embedded into the main calculation. Rather than running a baseline, then a scenario, subtracting one from the other and running the impact statistics, the metrics are calculated directly as *delta* differences from the baseline. This is all that is needed in order to know if one scenario is better or worse than other ones.

The real life projects, used for comparison, are in the following table 4. These include two variants of HS2, the Cambridge, Milton Keynes Oxford (CaMKOx) arc and Crossrail. The "HS2 Phase 1" project is the initial Euston/Old Oak

Common/Birmingham Interchange/Birmingham Airport phase of High Speed 2. The "HS2 IRP" project is the Integrated Rail Plan, published by the UK Government in November 2021, which includes phase 1, plus other links between Liverpool, Manchester and Leeds, as part of a much larger infrastructure plan [4].

In table 4, *Length (KM)* is calculated from the links added to the network graph as straight lines between stations, so will be less than the real built track totals. The *Saved Mins* column is calculated by matching changed links from the scenario with existing links in the current rail network and differencing the link transit times to obtain how many travel time minutes the scenario will save. This is purely on geometry and does not take into account how many commuters use the links. The *Saved Mins per KM* is then *Saved Mins* divided by *Length (KM)*.

Table 5 shows the impact statistics for the infrastructure projects in the previous table. L_k is the distance travelled on mode k in KM, with the table showing $deltaL_k$, which is the difference between the scenario and the baseline totals. $deltaL_k = L_k^{scenario} - L_k^{baseline}$. $deltaD_k$ is similar and counts the difference in the number of commuters using each mode. For both impact metrics, a negative number means fewer people or KM travelled as a result of the scenario changes.

Table 6 shows a selection of the metrics generated by the computer generated scenarios. Equations 11 and 12 show how the number of network changes (netChg) and geographic spread (LBar) metrics are calculated.

$$netChg = count(C_{ij}^{scenario} \neq C_{ij}^{baseline}) \tag{11}$$

$$LBar = \sum_{i \in nodes} \sum_{j \in nodes, i \neq j} dist(i,j)/perm(i,j) \tag{12}$$

The number of "networkSavedMinutes" is being used as a cost of building the scenario. This turns out to be a complicated metric to compute for the real scenarios, as it involves differencing the changes to the network made by the scenario from the existing network graph. Figure 5 shows the logic behind how this is achieved. In the first example (left), a new link is being built where none existed before, or where new nodes have been constructed. This results in the whole link transit time being added to the "networkSavedMinutes" total cost because "b" new minutes of track are being added. The middle example shows a new link upgrading an existing link time. In this situation, the result is "a-b" minutes because the scenario drops the link transit time by this amount, so this is taken as the cost of building. The third example (right) shows a new long link replacing an existing path of two links. Here, the total saved minutes added is the entire cost of the new link, "b" minutes. The reason for this is that it is assumed to be a new fast link being built parallel to an existing link and so requiring the entire path to be built faster, rather than an upgrade to an existing path. This is not a perfect solution, as it could be argued that the new path should be differenced with the existing shortest path between the

Project	Length (KM)	Saved Mins	Saved Mins per KM	Cost £M[4]
HS2 IRP	2999.5	2328.7	0.78	32,995.0
HS2 Phase 1	329.2	176.7	0.54	3,621.2
CaMKOx	283.9	448.1	1.58	3,122.8
Crossrail	229.1	480.0	2.10	2,520.1

Table 4: Geometric metrics

	deltaDk			deltaLk		
Project	Road	Bus	Rail	Road	Bus	Rail
HS2 IRP	-110,259	-34,967	145,226	-1,775,735	-418,498	28,054,970
HS2 Phase 1	-16,881	-4,396	21,278	-248,026	-41,743	6,284,993
CaMKOx	-27,729	-14,746	42,476	-610,188	-321,758	2,786,490
Crossrail	-11,053	-2,151	13,204	-179,924	-22,893	809,813

Table 5: Impacts

Metric	Description
netChg(mode)	Total number of network link changes after the shortest paths is run. Measures the extent of the scenario network changes. There are a maximum of 8436×8436 changes per link added. See equation 11
netSavedMins(mode)	Network saved minutes - primary changes (cost) - added up from network changes compared to the original link transit times that the scenario is upgrading. See figure 5
savedMins(mode)	Primary and secondary changes - benefit - computed on whole network after the shortest paths is re-run
NetworkKM(scenario mode)	Total amount of network added - proxy for cost
deltaDk(mode)	Change in number of workers on the given mode
deltaLk(mode)	Change in KM travelled by workers on the given mode
LBar	Mean distance between changed nodes - geographic spread. See equation 12

Table 6: Metrics used with computer generated scenarios for comparison with real life.

two nodes. However, if "networkSavedMins" is intended to be used as a cost of building, or size metric, then the first method is favourable. This situation exists with the HS2 development, where London to Birmingham is a valid route prior to HS2 being built, with HS2 taking 45 minutes off the time. However, HS2 requires a complete new high speed track to be built over the entire route, so taking the whole time as a building cost makes sense.

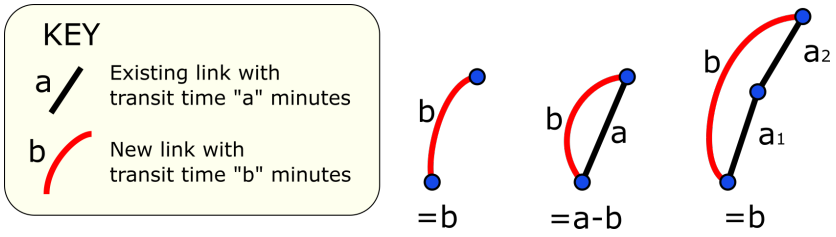


Figure 5: Logic used for computing minutes of saved time on a changed rail network. Example 1 (left) shows a new link where none existed before or a new node, resulting in "b" minutes added. The centre example shows a new link adding speed to an existing link, resulting in "b-a" minutes of saved time. The third example (right) shows a new link short-cutting two existing, resulting in "b" saved minutes added.

What we're doing is taking a real network scenario where roads or railways are being built to change the network time between nodes, giving rise to a change in time between MSOA centroids, and comparing this to the direct change between centroids for the synthetic scenario. The network changes are not identical, but relative sizes are comparable. For example, Crossrail has 229KM of network changes resulting in 480 saved minutes on the network. We make the assumption that this is comparable with a network change at the MSOA network level.

In terms of comparisons, a random network with $maxDepth = 20$ and $radiusLimit = 30KM$ constructs 10 links of between > 0 and 30 kilometers. This sets a maximum limit of $300KM$ on the scenario, which is approximately equivalent to the network changes made by the real-life Crossrail project, which has $229KM$ of changes. The data in figure 6 shows this for 1200 random scenarios. This confirms that the size of a 20 node scenario with 30KM radius is close to the Crossrail scenario (480 mins) and CaMKOx secenario (448.1 mins), if averaging a slightly lower value (mean=331.5 mins).

While network saved minutes is an indicator of benefit, in order to allow comparison of different sized scenarios, there needs to be a way of normalising by size. One attempt at this is to divide by the network KM value, which is derived from the network geometry being changed by the scenario. Figure 7 shows this plotted for the same sample as figure 6. Comparing to the data from table 4, HS2 Phase 1 is the lowest at 0.54 mins/KM, while Crossrail is highest at 2.1 mins/KM. The mean of figure 7 is 1.84 mins/KM, somewhere between CaMKOx and Crossrail, so there is confidence that these parameters result in

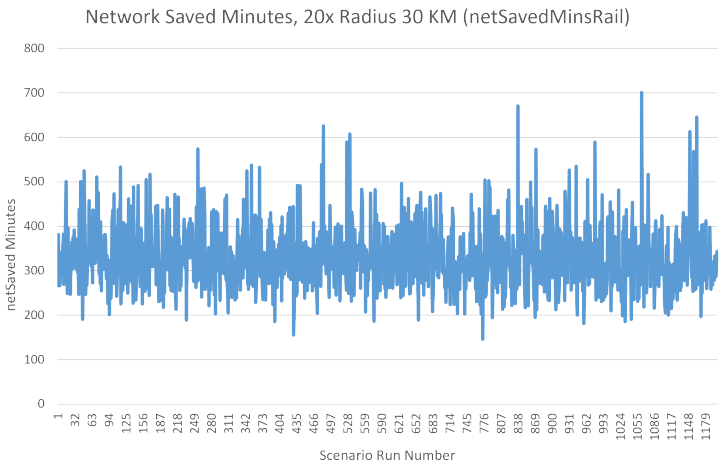


Figure 6: Minutes saved by the network scenario, calculated by differencing the network graph changes. These are primary changes only and do not include faster times arising from connections to the existing network. This is for a random allocation of 20 links with maximum link radius of 30KM. The sample size is 1200 scenarios. This data closely matches the size of the Crossrail and East West Rail (CaMKOx) scenarios. Mean network saved minutes for this sample is 331.5 minutes.

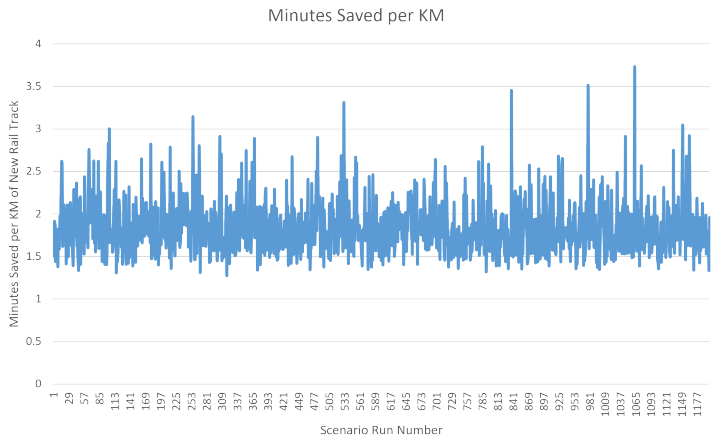


Figure 7: Minutes saved per Kilometre of scenario links for a random allocation of 20 links with maximum link radius of 30KM. The sample size is 1200 scenarios. This data closely matches the Crossrail and East West Rail (CaMKOx) scenarios.

scenarios that are the same relative sizes and are comparable to real life ones.

A star network topology with $maxDepth = 20$ and $radiusLimit = 30KM$ also results in 10 links being created of at most $30KM$, so is directly comparable to a random network with identical parameters. This enables comparisons to be made between different network topologies.

As no method of direct comparison between employment scenarios and network scenarios has yet been devised, these types of scenarios are treated individually in the following two sections. In QUANT scenarios, these can be combined easily, but there is no comparison with a real life scenario to be made.

5.3 Employment Scenarios

There are 7201 MSOA zones in England and Wales together with 1235 Intermediate (IZ) zones in Scotland, giving 8436 zones in total. The *moveDepth* in employment scenarios refers to the number of changed zones. No zone can be picked more than once and a zone can either have its total jobs increased or decreased by 1000. In the event that there are less than 1000 jobs in a zone, it is hard limited to 0. The scenario runner will randomly pick 10 zones ($maxDepth = 10$) for increase or decrease, before running the QUANT scenario model to determine the impacts.

We want to answer the following question:

> "Is it possible to reduce the amount of road KM driven by selectively adding and removing jobs without decreasing the total working population?"

Then, as a secondary problem:

> "Can this be done in a way that minimises jobs migration? Are there neighbouring zones where swapping jobs results in benefits?"

The graph in figure 8 answers the first question. This is a random search of employment changes. The top left quadrant ($x < 0, y > 0$) shows that solutions exist where road transport reduces, but jobs either remain stable or actually increase. The most striking feature of this graph is that it fits a linear relationship between jobs and commuter kilometres being driven. The part of the line cluster that shows positive jobs for road KM reduced, in other words a mode change, is the most important part.

In figure 9, the graph shows a Monte Carlo Tree Search algorithm [2] which is designed to search the solution space of only the top left quadrant. This is where the game framework used to make the moves that generate a scenario is beneficial. This is a one player game where the aim is to maximise benefit and minimise cost. The Monte Carlo search uses statistics generated at previously visited nodes of the search tree in order to determine where best to explore next. A "rollout" is then performed from the chosen location, where a sequence of random moves are played out to the end of the game to determine the score. Here, the score is based on minimising jobs lost and maximising road KM reductions. One additional concession that needed to be added was to bias the score towards scenarios that are geographically localised (minimise $LBar$), otherwise changes end up being made all over the UK. The 30,000 scenarios in figure 9 were found in under an hour, demonstrating the viability of this approach.

The Monte Carlo Tree Search algorithm is a tree search of the input space using random sampling and scoring of scenarios to find the best moves. The algorithm has four phases: Selection, Expansion, Rollout (or Simulation) and Backpropagation. The method of operation is to perform multiple *iterations* over the current game tree in order to expand the tree and improve its current information. This uses a combination of how many times a node has been visited

and an upper confidence bound of how much information is known about that branch of the tree (equation 13). After the selection phase has found a branch to expand, then the rollout phase plays the game from that point in the tree, making random moves until the terminal state is reached at the end of the game. Then, the final score is used in the backpropagation phase to update all the nodes along the path of moves leading to this final state. Each node records a score and a number of visits that is then used to calculate the upper confidence bound during the selection phase of the next iteration. In this way, multiple iterations build up a tree of moves and a confidence value based on how many times that part of the tree has been sampled.

$$UCB = Vi + 2 \times \sqrt{\frac{\log(N)}{n_i}} \tag{13}$$

where
V_i = `Average reward value of all nodes below`
N = `Number of parent visits`
n_i = `Number of child i visits`

However, in this game the order of moves has no importance, so a strict ordering of the selection process is enforced to eliminate move cycles. Zones are labelled from 0 to 8435, then a selection branch starting with zone 0 picks following zones from the set of 1..8435. A zone 1 move then picks from 2..8435, continuing in this triangular pattern until all the possible combinations of move are accounted for. This guarantees that the search will never visit the same combination of moves, but in a different order. This type of search can be viewed as identifying just the top level zones that generate the greatest number of "good" scenarios, but every rollout run results in a scenario and a data point. In a game like Chess or Go, this type of search would be used to find a potential next move, which the computer would make and wait for its opponent to make their move, before clearing the search tree and starting again[5]. These search statistics can then be plotted as a map to give an overview of where the most effective scenarios are being discovered.

The reason that the total jobs added isn't discrete on boundaries of 1000 when we're only adding or removing 1000 ten times is that a zone can have less than 1000 jobs so it's capped when being removed. This gives us the continuous spread of total jobs. The total number of KM driven each day, obtained by summing all the road trips in the QUANT model, is 130,000,000 KM. To put this into perspective, the -100,000 total KM saved limit on the extreme left edge of the graph in figure 8 represents 0.077% of the UK total.

[5]The search tree might not need to be completely cleared, as we might have some information on the computer move and opponent move part of the tree from the previous iterations, so the tree could just be pruned and moved upwards.

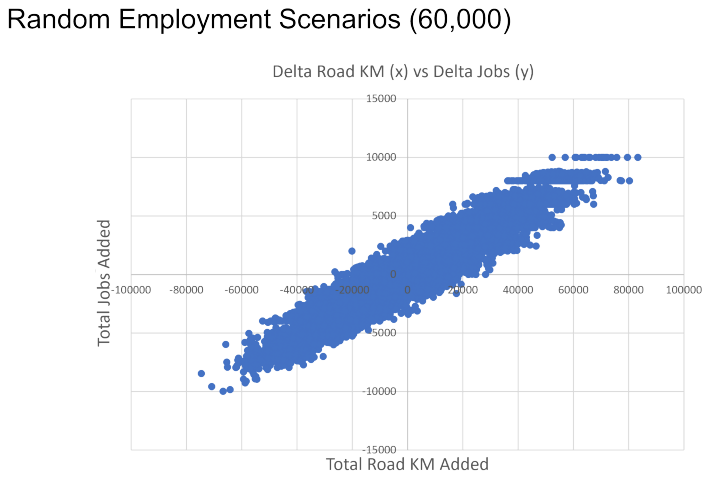


Figure 8: 60,000 employment scenarios using random choices. Employed population is increased or decreased by 1000 in ten zones. X axis is change of road KM. Y axis is the change in total number of jobs. Based on 10 employment location changes.

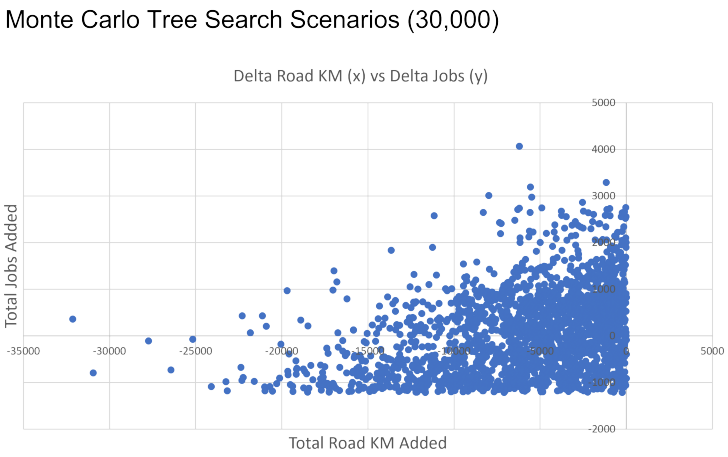


Figure 9: Employment scenario. Choices based on Monte Carlo Tree Search to cluster changes geographically (LBar), result in 0 or positive change in jobs and reduce KM driven on the road network by commuters. X axis is change of road KM. Y axis is the change in total number of jobs. Based on 10 employment changes.

Each dot on the graphs in figures 8 and 9 has a geography. The employment increase or decrease in the ten chosen zones gives rise to a change in the employment and commuting pattern of the UK. Some examples of these are shown in figures 10 to 13.

The first map, in figure 10, shows jobs being removed from the areas in blue around the southwest of London, while jobs are increased around Peterborough and Cambridge. There is also an increase in the very north of Scotland, while other changes in the Edinburgh and Glasgow areas seem to be having little effect. The blue and red coloured squares show the actual zones where employment changes have been made, while the red and blue areas on the maps shows where the biggest effects of these changes are produced. Red means an increase in jobs, while blue is a decrease. The net total of 596 jobs lost overall is because the algorithm is aiming for net zero change, but will tolerate a loss of up to 1,000. This is for a scenario with 10 zonal changes, amounting to a maximum of 10,000 being re-distributed. The aim is to maximise the road KM saved, while minimising the number of jobs being moved around the country.

The next map in figure 11 shows a migration of jobs from various parts of England and Wales, moving to Sheffield and Scotland. This type of scenario could be categorised as "levelling up", but the ideal way of running this type of scenario would be to define a different set of optimisation targets. This one has saved 15,966 road KM and lost 596 jobs.

The next map in figure 12 is an interesting re-distribution of jobs from the central England corridor to both the west into North Wales and east into Lincolnshire. The possibility is that this solution is attracting road commuters away from the M1 motorway by seeding new jobs to the east and west. This saves 4,324 road KM at the expense of 505 jobs.

The final map in figure 13 is another "levelling up" scenario, where jobs are reduced in the southwest of England and East Anglia, to be re-distributed in Sheffield. The loss of jobs across the Glasgow to Edinburgh belt makes it a "levelling down" for Scotland, though. The jobs increase in the Orkneys is probably not a realistic scenario. One observation that has been made after looking through all the thousands of scenarios is that there are a lot of possible swaps of jobs in the Glasgow to Edinburgh belt that result in fewer road KM, with no harm to the overall numbers of jobs. This particular scenario saves 29,908 road KM with a loss of 834 jobs, making it one of the bigger carbon reduction solutions discovered.

Returning to the idea of "small changes", that was introduced earlier, it is easy to see how the number of job zone changes can be increased, while the change in job numbers is decreased. This would have the effect of letting the computer make lots of small changes across the whole country, producing many "micro scenarios", which would hopefully lead to large overall impacts when added together. This is the continuing focus of the research.

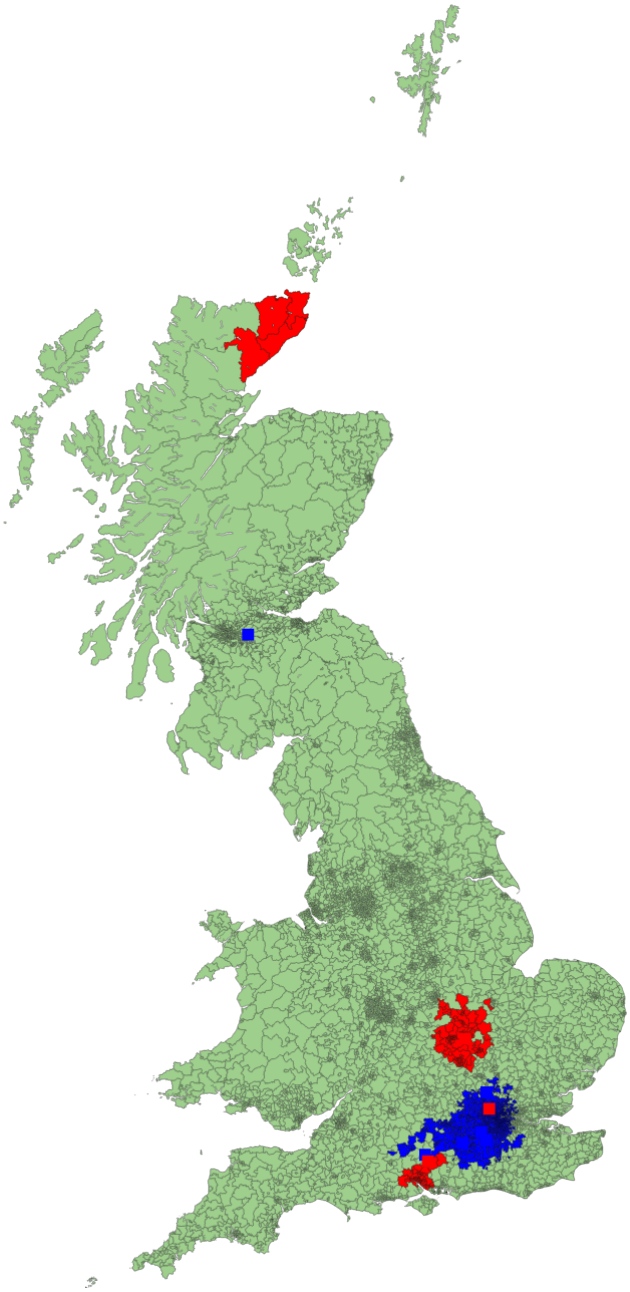


Figure 10: Employment scenario. Choices based on Monte Carlo Tree Search. Based on 10 employment changes. Road KM saved: 18,327KM with total net jobs created: -481.

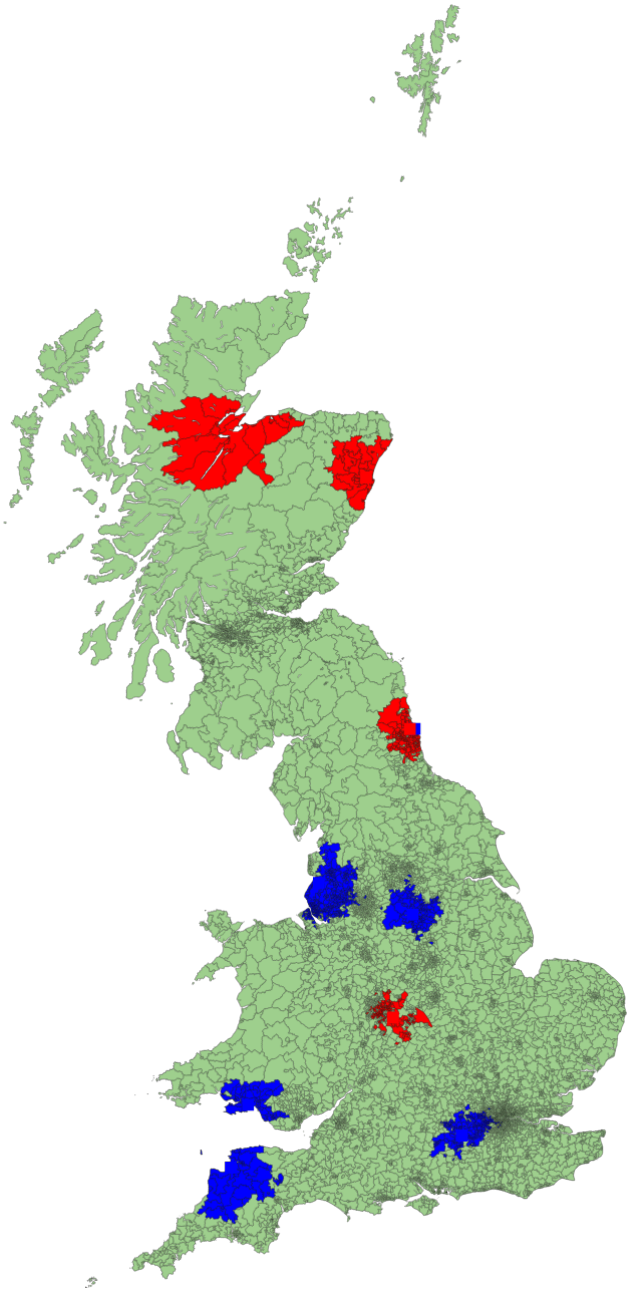


Figure 11: Employment scenario. Choices based on Monte Carlo Tree Search. Based on 10 employment changes. Road KM saved: 15,966KM with total net jobs created: -596.

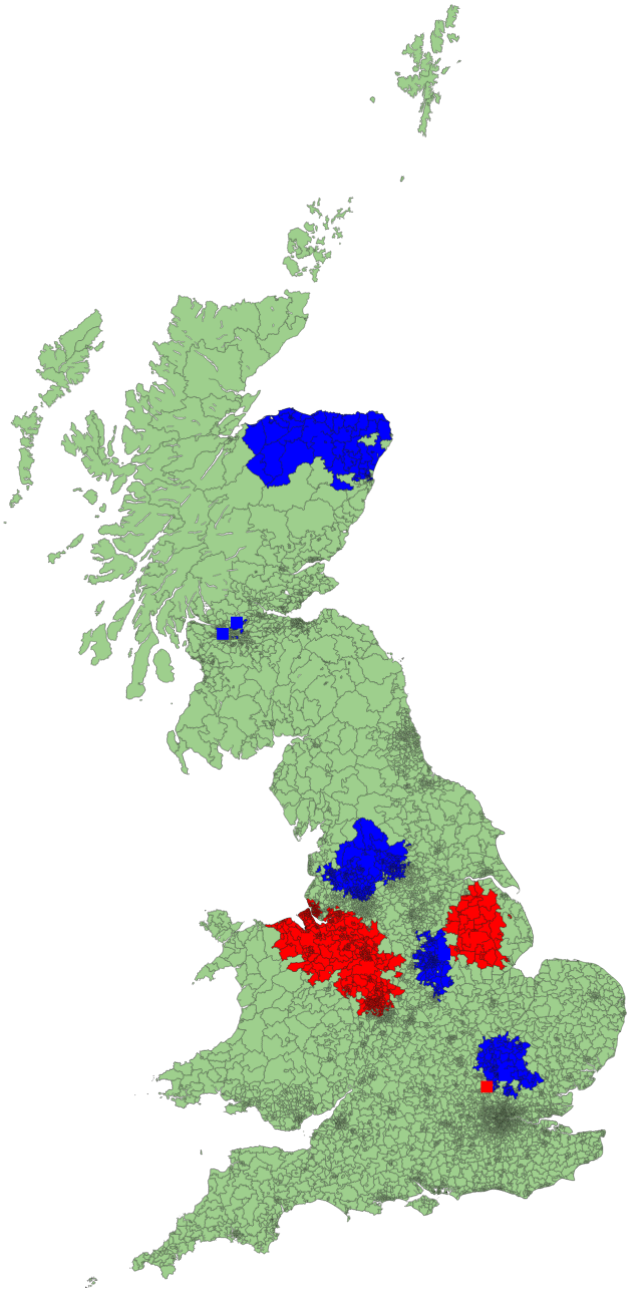


Figure 12: Employment scenario. Choices based on Monte Carlo Tree Search. Based on 10 employment changes. Road KM saved: 4,324KM with total net jobs created: -505.

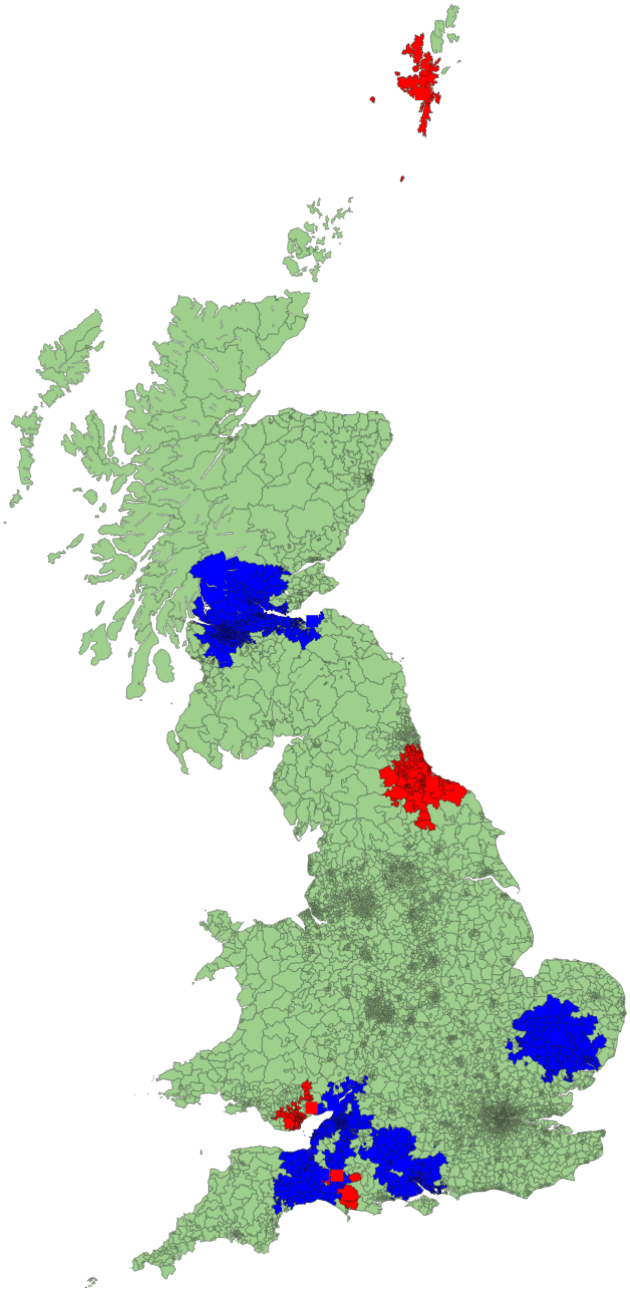


Figure 13: Employment scenario. Choices based on Monte Carlo Tree Search. Based on 10 employment changes. Road KM Saved: 29,908KM with total net jobs created: -834.

5.4 Transport Network Scenarios

To begin with, 1200 runs of 10 random network changes were run, followed by the same number of 20 and 40 random changes and then a more limited number of 200 random changes. This takes approximately 25s per scenario (random 10), 30s per scenario (random 20) or 120s (random 200), while an employment scenario is 100 times faster. That's why there are only 1200 points on these network graphs and 60,000 on the employment ones. This initial run of random scenarios did not place any limit on the length of the link, so connections between London and Edinburgh are possible compared to shorter links between neighbouring zones. When a link is upgraded by the scenario, its transit time is divided by two, simulating the building of a high speed line. While this might be unrepresentative in specific real life situations, it is treated like a test probe in that all the synthetic scenarios are comparable with each other.

The first analysis is a cost/benefit metric. Amount of rail network KM added is a proxy for cost, while benefit is measured as reduction in road KM driven. Alternatively, the change in number of road users, or saved rail commuter KM or saved minutes could be used. The results are consistent whichever metric is used.

Figure 14 shows random links with no limit on the length for different sized scenarios, while figure 15 shows the same random sets of scenarios, but with a 30 KM radius limit constraining the links. These graphs show how the benefits in terms of a mode change from road to rail increase with increasing size of the rail network scenario. More expensive scenarios produce more benefit, but there is substantial overlap between them, suggesting that solutions exist for lower cost scenarios that provide the same benefit as more costly ones.

The change in road KM driven by commuters against network KM built is plotted in figure 16 as this shows a mode change from road to rail and is a carbon reduction metric.

In contrast to plotting how adding rail network influences road KM driven (figure 16), the graph in figure 17 plots the same data, but based on the ratio of the change of rail commuter numbers to road and bus. This graph shows how this is a closed system and so any commuters taken off of the road by the attraction of the new rail scenario must be balanced. The almost 45 degree line for the road commuters is caused by the effect of having three modes (road, bus and rail), so the rail scenario is attracting people from both road and bus, but mainly from the road. A smaller rail scenario attracts fewer people, so $Y = 0$ corresponds to no scenario changes, with scenario size increasing as the Y value increases. Large rail scenarios have little effect on bus numbers, which increase comparatively little as Y increases for rail.

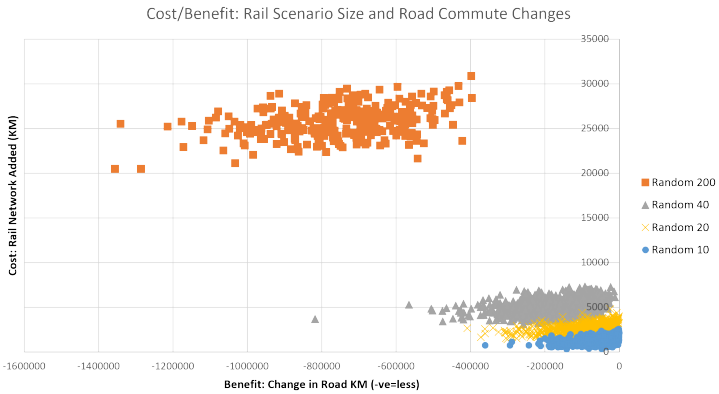


Figure 14: Change in Road KM (X axis, negative means a reduction) against rail network KM added (y axis). Greatest benefit (X) is closest to the left axis. Lowest cost (Y) is closest to the bottom axis. Plotted for 10, 20, 40 and 200 random links with no link length limit.

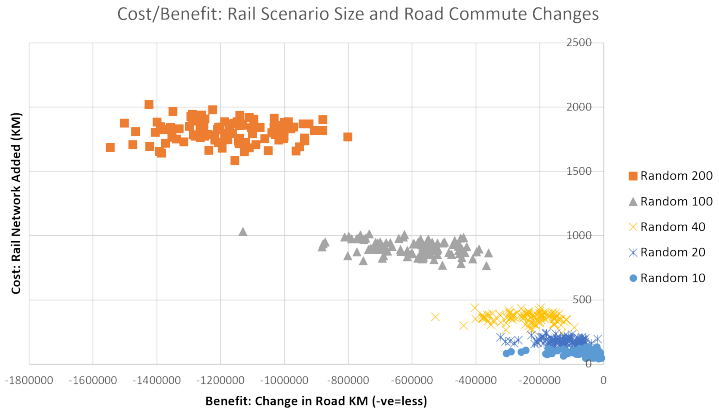


Figure 15: Change in Road KM (X axis, negative means a reduction) against rail network KM added (y axis). Greatest benefit (X) is closest to the left axis. Lowest cost (Y) is closest to the bottom axis. Plotted for 10, 20, 40, 100 and 200 random links with a 30KM link length limit radius.

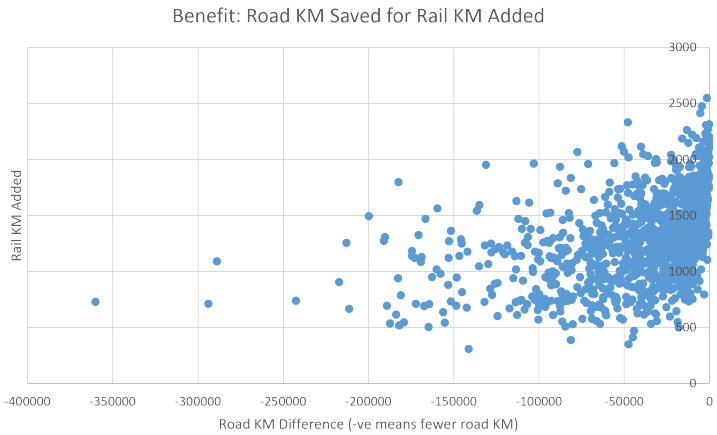


Figure 16: Change in Road KM (X axis, negative means a reduction) against rail network KM added (y axis). Greatest benefit (X) is closest to the left axis. Lowest cost (Y) is closest to the bottom axis.

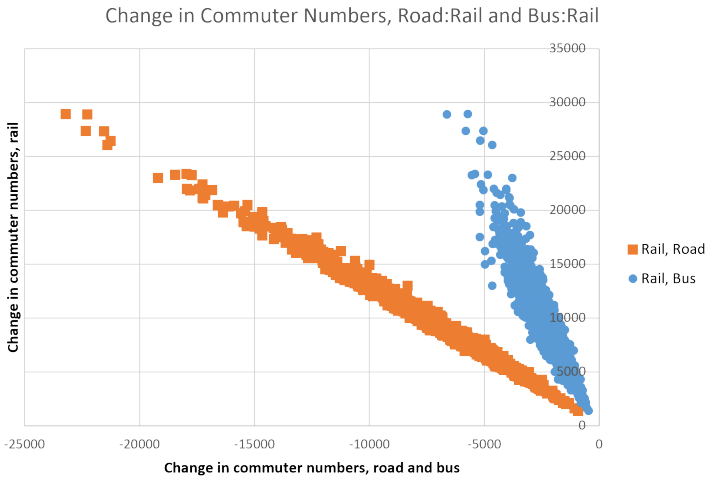


Figure 17: Change in number of road commuters (orange) and bus commuters (blue) against the change in the number of rail commuters for a random scenario with 30KM limit on link radius. Increasing Y values (rail) correspond to increasingly bigger scenario sizes, leading to reductions in road and bus commuters (X axis).

The purpose of the results presented so far is to establish that building new rail links attracts commuters away from road, while trying to determine what factors drive the size of the change.

Looking at the previous graphs, there is no simple relationship between rail network KM added and the reduction in road KM driven due to commuters switching mode. In general, more rail network changes can be seen to generate a larger attraction, giving rise to a mode switch from road to rail, but there is a large variation between similar sized scenarios. The results are down to the underlying network. Increasing the amount of railway built does not necessarily increase the benefit.

Adding rail links to the existing network adds kilometres of track and saved travel minutes, but the "Number of Network Changes" metric comes from running a shortest paths algorithm with the changes, so cannot be predicted. This can be seen when plotting "netSavedMins" or "networkKM" against "number of network changes", or "deltaDkRoad" (or any other model delta). There is no correlation. We cannot predict the network benefits because it depends on the shortest paths and the residential/work population patterns. Figure 16 of "deltaLkRoad" against "networkKM (rail)" shows this quite effectively. There is a general trend is towards smaller or mid-rang rail interventions producing greater benefits. However, it is worth bearing in mind that 1200 Monte Carlo samples of scenarios is a very small percentage of the total number, which is $(8436 \times 8435)^{linksdepth}$, to a very good approximation, or roughly 1.8x10e39 for depth 5 links.

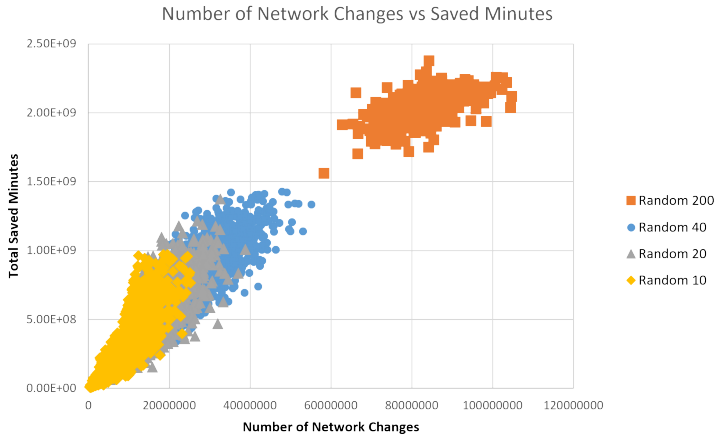


Figure 18: Comparison of network scenarios with 10 random nodes connected in pairs of links, compared to 20, 40 and 100. The plot shows the number of network changes against total number of minutes saved in travel time by rail commuters.

The second analysis is to try to determine a reason behind why some scenarios result in a higher impact than others. Figure 18 compares the number of

rail network changes made (X axis) to the total number of travel minutes saved by rail commuters (Y axis). The four groups show scenarios where 10 random zones were connected by links, along with 20, 40 and 200. So, for the "random 10" scenario, the computer picks links from the 8436 zones, as origin followed by destination until it has taken 10 nodes. These are connected by 5 links in the order the 10 origin plus destination nodes were picked, resulting in 5 *probably* completely isolated changes of unlimited length.

The "random 200" scenario adds 100 improved rail links all across the country, often resulting in more than 100% of the approximately 71 million links being changed. This happens because the algorithm adds each link separately to the network and re-computes all the changes. This means that links can be changed and counted multiple times.

This shows that the number of network changes correlates highly with the number of rail minutes saved. So, the total benefit is linked to how much effect the network changes have on the shortest paths graph. The only other factor to take into consideration is geography, specifically the geographic spread of the links and where they are concentrated in the country with respect to working and residential populations. This is covered in the next section.

5.5 Network Geometry

The network geometry experiment is designed to test whether the topology of the links determines the final outcome. The topologies being tested are: *dense*, *limitr*, *ribbon*, *rndwalk* and *star*.

To put the data into context, figure 19 compares a ribbon topology with 20 links and radius limit of 30KM with its equivalent random network from the previous section. Similarly, figure 20 is a comparison of the random walk network against the ribbon. Both graphs show change in road KM against rail network KM added, so are cost versus benefit plots. Although it is difficult to discern the spread of scenario results for the two graphs, there is a good correspondence between the random scenarios and the ribbon scenario from the first graph, which gives confidence that the results are comparable. However, there is no obvious, immediate, difference between the results being produced by the random or the limited area topologies. The random scenarios can put limited sized links in all over the country, while the limited topology ones concentrate on a central location. These two graphs provide some limited evidence that the "small changes" hypothesis might be correct.

For reference, figure 21 shows an example of each of the six basic topologies used for the following analysis, taken from the real data. Next, the probability densities of different scenario outcomes is plotted for the different topologies in figure 22. These charts all show the road KM saved (benefit) divided by the rail network KM added (cost) as binned counts. The first one (figure 22a) is for unlimited link lengths, while all the others limit the link lengths to a maximum of 30KM. Apart from the link limit, figures 22a and 22b are identical as they place random links all over the country. The remaining topologies of dense, ribbon, star and random walk all group changes around a geographical

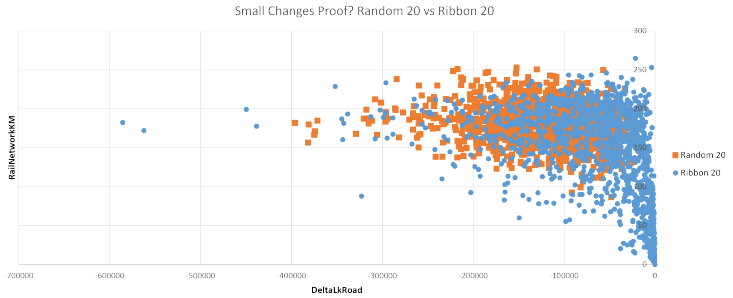


Figure 19: Comparison of a Random scenario with 20 nodes to a Ribbon scenario with 20 nodes. The plot shows road KM difference (benefit) against rail network KM added (cost). The sample size is 1200 scenario runs for each, with a link length limit of 30KM.

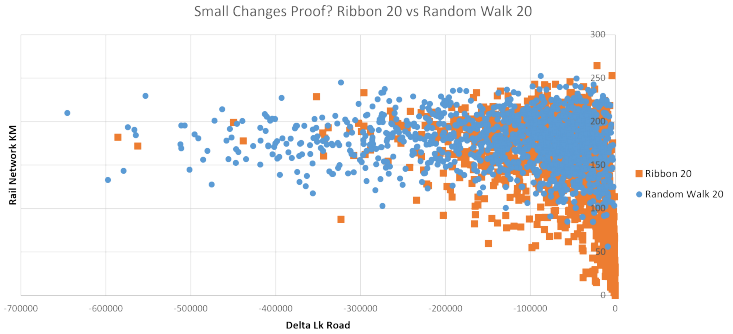
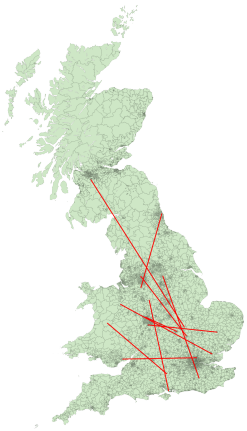


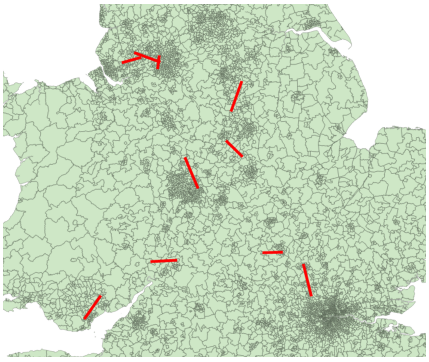
Figure 20: Comparison of a Random Walk scenario with 20 nodes to a Ribbon scenario with 20 nodes. The plot shows road KM difference (benefit) against rail network KM added (cost). The sample size is 1200 scenario runs for each, with a link length limit of 30KM.

area, although ribbon obviously builds a line of linked segments that heads away from its starting point. The x-axis shows the benefit:cost ratio, which are all fixed to span -8000 to 0. There are no units as it is road KM divided by rail KM and -8000 covers the full range of outliers, with the star network having the greatest score of -7537. The mean, standard deviation and max value are all printed on the graphs.

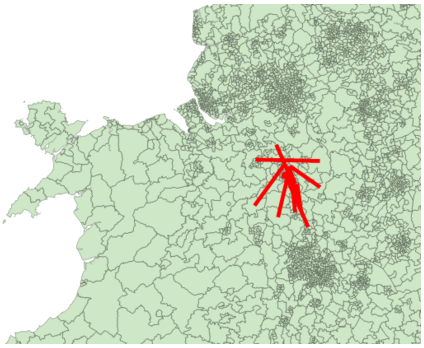
The "small changes" hypothesis pits "random radius 30" (22b) against topologies dense (22c), ribbon (22d), star (22e) and random walk (22f). For small changes across the country to produce equal or more benefit than changes concentrated in one location, then 22b should be producing similar results. While it is not producing some of the outliers of the other topologies, its mean value of -696 is second best of: dense (-564), ribbon (-456), star (-644) and random walk (-744). Not forgetting that larger negative numbers are more benefit per cost. The small changes scenario would also appear to be producing fewer *bad* scenarios of close to zero score compared to the others, and has a lower standard deviation, suggesting it is a more consistent methodology. Finally, it needs to be understood that these are small random samples of what is a very large scenario space. The random links, or small changes scenario, has a much larger solution space than the others because of the way the other topologies are geographically limited, so this could be affecting the results (see combinatorics table 3).



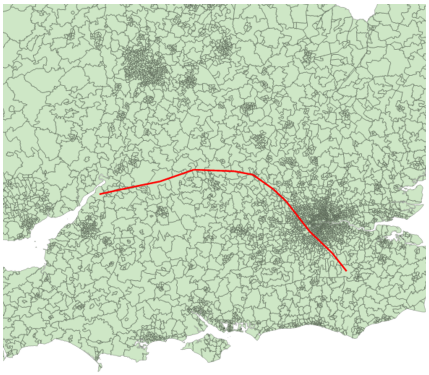
(a) Example of “random 20 nolimit”.



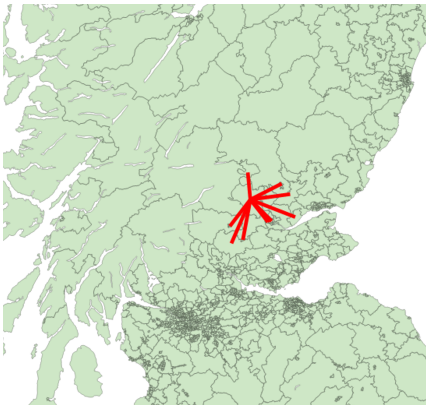
(b) Example of “random 20 radius 30”.



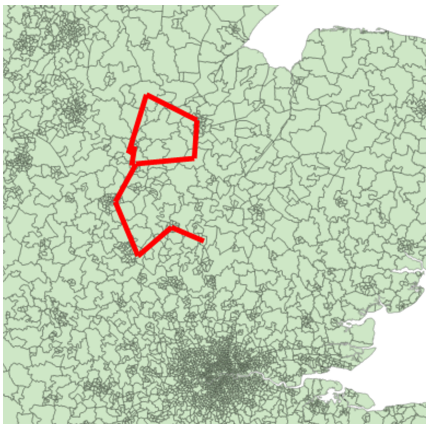
(c) Example of “random dense 20 radius 30”.



(d) Example of “random ribbon 20 radius 30”.

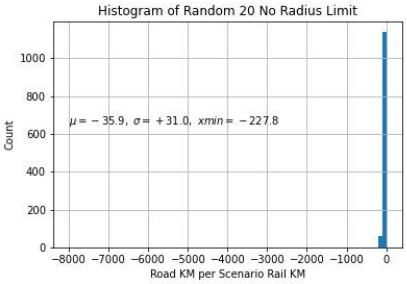


(e) Example of “random star 20 radius 30”.

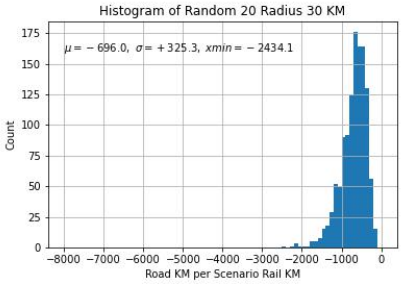


(f) Example of “random walk 20 radius 30 with 30 degree angle limit”.

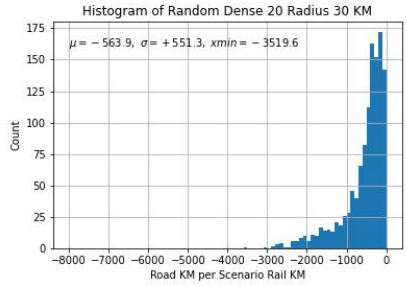
Figure 21: Topology examples showing actual scenarios from the results data for each of the different network topologies.



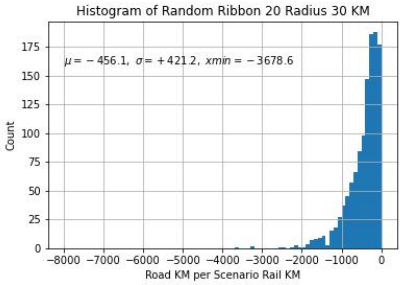
(a) Probability histogram of "random 20 nolimit".



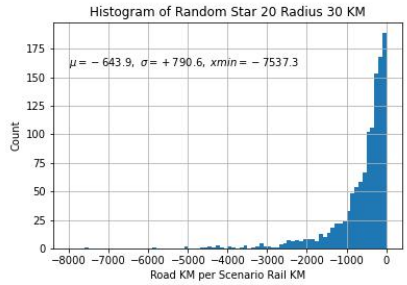
(b) Probability histogram of "random 20 radius 30".



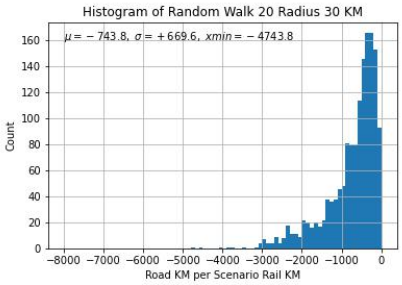
(c) Probability histogram of "random dense 20 radius 30".



(d) Probability histogram of "random ribbon 20 radius 30".



(e) Probability histogram of "random star 20 radius 30".



(f) Probability histogram of "random walk 20 radius 30".

Figure 22: Probability histograms showing the count of the number of scenarios of different benefit size. (a) has unlimited length links, while all the others are limited to a maximum of 30KM links. The x-axis range is consistent across all charts, but the y-axis varies. The metric plotted on the x-axis for all of them is road KM saved (benefit) divided by rail network KM added (cost). All are for the same 20 node scenario size. See text for full details. Positions match the maps in the previous figure 21.

The next factor to investigate is whether scenarios vary geographically across the country. Figure 23 shows the result of taking all the data from all the scenario runs (limited by 30KM links) and producing a mean cost benefit score (road KM saved / rail network KM added) for every zone.

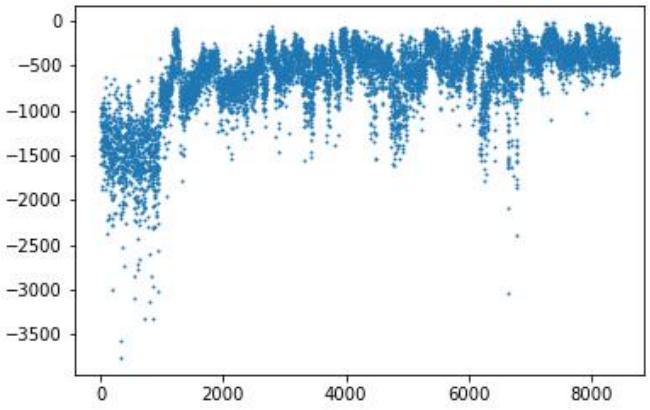


Figure 23: Scatterplot of mean network KM Road Saved (LK, y axis) vs zone number (x axis). The mean is taken of all the scenarios containing the x axis zone number, wherever they occur. All scenario types are summed together. NOTE: London zones occupy zone numbers 0..1000, while Scotland zones are 7201..8436.

The methodology behind how the graph in figure 23 is created needs some explaining. Firstly, all the data for every type of scenario is put together in a single table and the score for each scenario is calculated as $score = deltaLkRoad/networkKM(rail)$. Each line in the data table also contains the *gamestate*, which is simply a list of the 20 nodes that defined the scenario. These are 10 origin/destination pairs forming the links, regardless of what the actual topology is. The consistent origin/destination gamestate format makes it easy to compare scenarios, although wasteful if you consider a star topology, which is $a \rightarrow b$, $a \rightarrow c$, $a \rightarrow d$ or a ribbon $a \rightarrow b$, $b \rightarrow c$, $c \rightarrow d$ etc. The next step is to produce table of zone numbers and their mean scores. The mean score is defined as the mean of all the scenario scores where a particular zone number was used as part of that scenario. In other words, for zone 0, take all the scores where zone 0 appeared in the gamestate and average them. This is what is plotted in figure 23 and then as a map in figure 24.

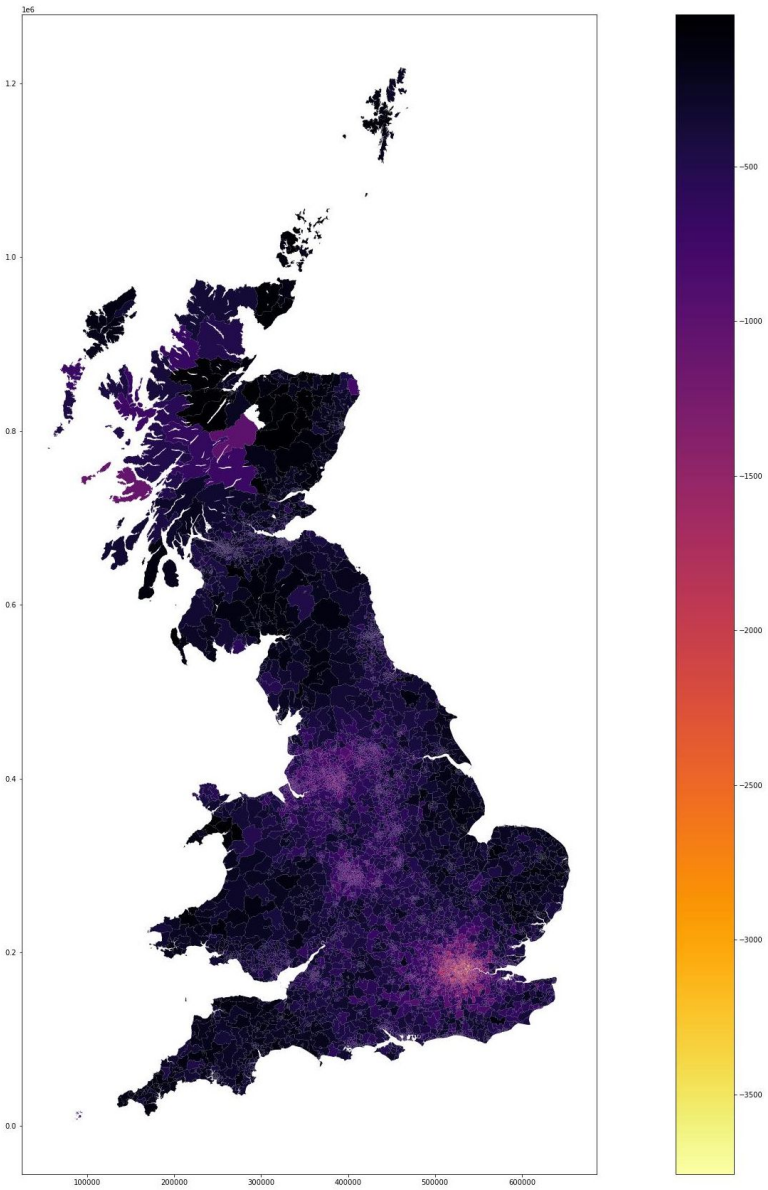


Figure 24: Map of mean of 100,000 network scenario runs showing KM saved for road by region. Same data as the previous histogram, but mapped. Lighter colours produce more scenario benefits per unit cost.

5.6 Network Search Optimisation

Random scenarios with a large search space give background context, but the ultimate aim is to be able to generate optimised scenarios without having to randomly search the space. Comparisons with real life scenarios are important in this section, as it is essential to be able to compare the computer generated scenarios with the human created scenarios. The question is whether the computer can beat the human planner.

By changing the scenario search process to take the highest D_j value instead of being completely random, the results shown in table 7 are obtained when the five different topologies are re-run. Two factors are derived from the data and used as impact measures: "change in road KM/rail network KM added" and "change in road KM/rail network minutes added". The numerator of each fraction is the amount of road KM being driven by commuters, where a negative number means a reduction. The denominator is the amount of physical rail track added in the scenario and the amount of saved rail minutes that each scenario creates respectively. This is for comparison with the real scenarios in table 5. The problem with real scenarios is that they include upgrades to track, which the two impact statistics of "road KM per rail KM" and "road KM per rail minutes" are intended to mitigate against.

One slight change is made to the greedy D_j optimisation scheme, because, otherwise, it will always pick the same network paths for each zone. The D_j values have a small random element added to them to ensure that the same ones aren't picked every time. Then, the algorithm is run for greedy optimisation of D_j, O_i, T_{ij} and C_{ij} for 300 simulation runs.

	deltaLk(Road)/networkKM(Rail)			deltaLk(Road)/netSavedMins(Rail)		
Geometry	Minimum	Mean	StdDev	Minimum	Mean	StdDev
RndWalk	-23313.12	-1805.24	3092.86	-17270.25	-1403.23	2292.86
LimitR	-3205.5	-1081.94	543.83	-2370.78	-674.26	372.74
Ribbon	-6022.81	-998.76	986.86	-4046.29	-613.75	680.66
Dense	-5720.4	-783.84	1044.77	-3725.81	-535.60	708.34
Star	-9151.91	-756.21	1418.97	-4685.31	-445.96	731.72

Table 7: Greedy Dj, 300 runs each, all 20 network nodes and 30KM link limit, ordered by mean Lk road per net KM rail (column 3). NOTE: 'LimitR' is the random topology with a radius limit added.

The comparison of the results in table 7 show values in line with the real world project impacts in table 8. The biggest surprise looks like the emergence of "RndWalk" as the geometry that consistently produces the highest impact scenarios.

This data is plotted graphically in the bar charts of figure 25 (minimum) and figure 26 (mean). These are from the same 300 simulation runs as before. The minimum is the best case outlier from all the runs, while mean averages the good

Project	deltaLk(Road)/networkKM(Rail)	deltaLk(Road)/netSavedMins(Rail)
Crossrail	-785.35	-374.84
CaMKOx	-2148.55	-1361.72
HS2P1	-753.42	-1403.66
HS2IRP	-592.01	-762.54

Table 8: Real world infrastructure projects with metrics that are comparable to the computer generated ones.

and the bad. It can be argued that the minimum is a more important statistic as the point about this type of directed search is to find the best cases. Average is irrelevant if you only need one high impact example. The minimum outlier from the D_j optimisation of the random walk topology is the most significant measure to come from the data, along with the generally good performance of random walk overall.

The different optimisation schemes being used in figures 25 and 26 show how optimising for the highest cost routes (C_{ij}) and highest flow (T_{ij}) both perform consistently worse than for O_i and D_j, even if some results are a close call, while for minimum "star", T_{ij} is the winner. It looks like the information that needs to be presented to the user in the scenarios interface is the O_i and D_j values of the links in and out of each node making up the scenario.

The map in figure 27 shows the solution which gives rise to the best minimum of -23313.12 LkRoad/netKMRail in table 7 and the graph in figure 26. This immediately shows a problem with how the random walk logic works. If the initial random position is in London, then it quickly finds the optimal D_j zone to walk to, but ends up trying to walk backwards and forwards between two optimal D_j zones, which results in the path being terminated prematurely before 10 links have been followed. Looking at all the top values for random walk, this path truncation is present in all of them, with the artificially short length of the paths leading to high benefit/cost ratios due to the low costs. The average network KM added for random walk with D_j optimisation is 123.3KM, while for the equivalent "limitr" algorithm, this figure is higher at 176.2KM. Without the optimisation (fully random choices), the random walk average network KM added is 176.3KM, while the equivalent "limitr" algorithm is 180.3KM. This highlights the difficulty in comparisons between different scenarios.

Given the problems with the path length of the random walk algorithm, it is apparent that the "small changes" scenario of placing random links anywhere in the country does out-perform the geographically dense topologies and produce more impact for the same cost.

One final point needs to be make about sample sizes, as it is not entirely clear how big a sample size is required for it to be representative. Work is continuing to optimise the scenarios code so that more scenarios impacts runs can be produced in a shorter period of time.

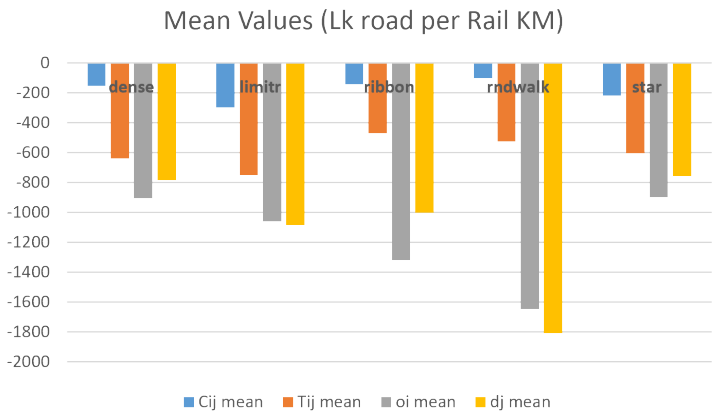


Figure 25: Mean Lk (road) per KM Rail Network for topologies and different optimisation schemes - Cij, Tij, Oi and Dj.

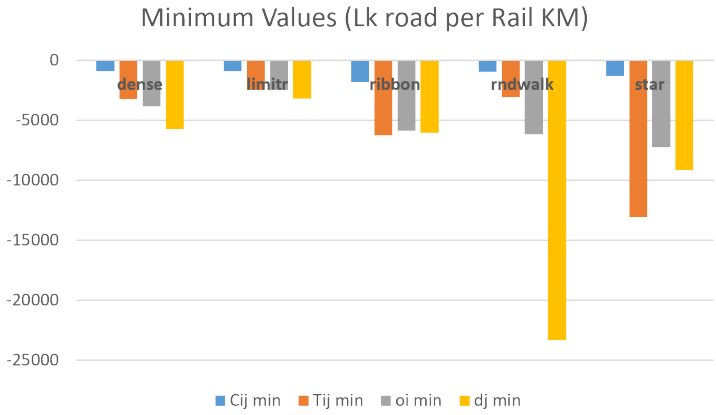


Figure 26: Min Lk (road) per KM Rail Network for topologies and different optimisation schemes - Cij, Tij, Oi and Dj.

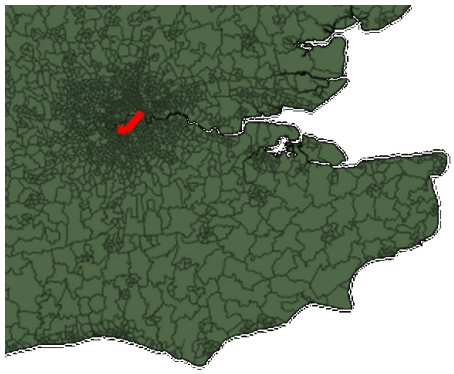


Figure 27: Min Lk (road) per KM Rail Network for Random Walk, Dj optimisation, minimum (-23313.12) solution. This amounts to just two links in central London before the algorithm gets stuck and terminates prematurely with a short path. It's not a bad scenario, it's just not immediately equivalent to all the others due to its artificially low cost.

6 AI and Urban Modelling

The employment scenario data and the network scenario data that has been created in the previous chapters is now useful in linking AI and Urban Modelling. With current AI, for example in speech and vision, there are large repositories of data which are used as training sets. This is lacking in the urban modelling world. Datasets like ImageNet ([6]), MNIST ([7]) and CIFAR ([5]) are well-known benchmarks for computer vision experiments, so the aim here is to create an equivalent for gravity models, travel to work models and other types of urban modelling.

Tables 9 and 10 show three lines of data generated from three employment scenarios. Table 9 shows the impacts which are derived from running the gravity model, while table 10 shows the "gameState", which is ten index numbers of zones, where '+' means the jobs totals were increased in that zone, while '-' means a decrease. todo: This data is available in full online at the following address: PUT UCL CASA ARCHIVE HERE - OR OSF?

score	deltaLkRoad	deltaDjRoad	deltaTotalJobsCreated	LBar
-0.032264724	24141.65869	1522.405282	3126	280.74078
-0.005727778	4776.932006	302.0974299	1000	313.808
-0.003492613	-11423.63044	-703.2356679	-1591	327.07974

Table 9: Data generated by running an employment scenario (Part 1 - impacts data).

GS0	GS1	GS2	GS3	GS4	GS5	GS6	GS7	GS8	GS9
889	-995	1113	1294	2932	-3957	-6861	7446	-8061	8062
-1244	-1524	1884	2503	3259	-3807	-3908	4585	7327	-7437
1834	2402	-2763	-3424	4131	-4177	4873	-6709	-7517	-7919

Table 10: Data generated by running an employment scenario (Part 2 - scenario definition). NOTE: "GSx" = gameState.

Similarly, for the network scenarios, a set of 19 impacts is generated by the gravity model in response to a set of 40 network node inputs. TODO: this is also available to download from here: xxx

These datasets form the training data that can be used to train an AI system by learning to predict the results indicator from the input set of zone changes.

Then run some AI search to find solutions that beat the previous ones.

This is future work.

7 Conclusion

Drawing comparisons between computer generated scenarios and real world scenarios exposes problems with how the costs and benefits are measured. Crossrail is 300KM track, while HS2 could be 300KM or 3000KM, depending on how track upgrades and improvements are counted. Methods were introduced based on network time saved which mitigate these problems to some extent, but do not solve them completely. The idea of building computer generated scenarios using different topologies which are self-referential by design opens up the possibility of comparisons between types. However, although the number of links, nodes and radius are the same, there is still the question of how comparable the different topologies are with each one another. The '10 node star' has 5 links, but only 6 different nodes connected. By contrast, the '10 node dense' topology has 10 different nodes connected. Despite these differences, though, the results presented in the previous section do show consistent differences, although more scenario runs would be desirable to prove this conclusively. The direction for this part of the research must be to speed up the model algorithm.

One unexpected result that came out of examining the generated scenario maps in detail is that the random scenarios know nothing about geography. Figure 28 shows a good scenario which has found a favourable solution by crossing the Bristol Channel.

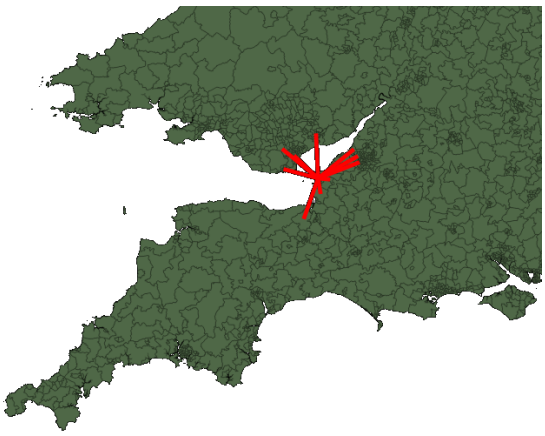


Figure 28: A random star network with a maximum link radius of 30KM crossing the Bristol Channel. In the 1200 'star' samples that this example was taken from, 15 were found to have crossed the Bristol Channel at least once.

While this might look like the difficult engineering involved in a shortcut across the water might be a problem, it has to be recognised that the normal routes between these nodes would go around via a land route. This means that the scenario network speed up is relative to the real life transit time anyway. The times used for the cost are based on already built roads and rail and existing bridges, while any new infrastructure added by the scenario is more ambiguous

and has to be based on straight line distances.

Looking at the data for this scenario, the impacts statistics show that it saves -82203.5 KM of road transport (deltaLkRoad) for 155.5 KM of rail track built, although, admittedly, this track figure is based on a straight line distance between zone centroids, not on a land-based distance around the water. This is the type of physical constraints problem that future work could target with additional layers of physical data. The simplicity of the approach taken here has its merits, though, as the question being asked is whether building a fast link between these zones would bring any benefits. If the answer is "yes", then the physical constraints layer and increased physical cost can be handled as a follow-up analysis. This is now starting to encroach into the more complex area of "Geodesign" [11], while the idea of running lots of scenarios across a vast solution space requires a simpler question to answer.

The "small changes" hypothesis, which is behind the design of the topologies for testing coverage, density, connectedness and geographic length, is hard to prove conclusively. Take any regular scenario and there's probably always a "small changes" scenario that beats it, if you look hard enough. That's just the nature of the problem space. The question is whether there are more small changes scenarios than there are ribbon, star or dense etc. Making comparisons between the dense and connected against the geographically sparse is shown to be a valid technique. The small size of the 1200 model runs completed for each topology dataset could be increased in order to reduce variability in the results. The standard deviation scores included with the data are difficult to interpret given the very large search space. Geography and 30KM limited radius links certainly reduces the search space, but the number of possible scenarios remains huge. Optimisation of the search provides the expected jump in performance, but more work is needed in this area and an AI directed algorithm seems like an interesting prospect for future work.

The big combinatorial numbers of possible moves do not tell the whole picture. It is not completely random as similarity is seen across the country because of similar employment patterns and transport links. If this is a repeated pattern that is being sampled randomly, then the results would appear to be stable on the small samples taken here. The results are also limited by geography, as this is not a regular grid of zones. The large number of London scenarios reflect the high density of MSOA zones clustered in London, which can be seen in all the data, for example figure 23 and the map of figure 24. This geography is limiting the geometry of possible scenarios at the edges of the coastline and around the islands and other large waterways. An open question is whether there are optimisation heuristics which work equally well across the whole geography, being based on workplace population, residential population, cost and flow. This makes sense given that the same model equation is being used, although the effects of topology are not easy to theorise about. This is an important question when it comes to training an AI algorithm to do the optimisation and whether this is an example of "transferable AI" or not.

The whole question of cost and benefit has been central to this work, with an essential element being the comparison of real world scenarios to the computer

generated ones. The doubling of the link speeds in the computer generated scenarios is rather an arbitrary decision, but it was shown to work and generate benefits in line with Crossrail, HS2 and CaMKOx. Any future work should examine this in more detail, along the lines of having infrastructure levels. For example, a high speed rail link is 300KPH and has a set cost per kilometre of track. However, different levels of infrastructure investment multiplies the search space, and the simple methodology used here has worked well as a 'probe' into where there might be benefits gained from improving rail links. The places and links are more important than the absolute impact figures, as the algorithm is approximate anyway and needs to be run through the exact version of the shortest paths calculation as a double check. Employment scenarios are exact solutions, though, while only the network scenarios are partially approximated.

While the analysis here focused on optimising for low carbon driving, other factors could be used, for example encouraging the creation of jobs in the north as part of a "levelling up" scenario. Environmental impact of building should also be taken into account. One thing which the analysis of nearly 100,000 computer generated scenarios did show is how difficult it is to make comparisons. Actual monetary cost, or the benefit to the economy would be an excellent metric, but would be dependent on a planning and spending review first. The amount of track added and cost per KM built is a usable metric, as is saved time added to the network, with each having various pros and cons depending on the scenario. No one single metric stands out from this work as being the best one to use.

The enormous number of scenarios produced using this methodology shows that there needs to be some kind of automatic way of labelling them. This would involve measuring benefits and costs of different kinds, along with a description of the geography of the scenario, which would be an interesting challenge for future work involving AI. The general idea is to sort and cluster the solutions using an AI embedding technique based on the gamestate.

Also for future work, is a comparison between the approximate shortest paths algorithm and a full run of the all pairs shortest path (APSP) using an exact method. This is simple to run against the data anyway as a verification process, but, even the best case of rail network modification, is likely to take five times as long to run[6]. The two methods do rely on different network graphs, but the methodology is simply to add a new rail link between zone centroids in the physical rail network graph and re-run APSP.

Finally, to sum up, the work here provides some initial evidence that the 'small changes' hypothesis works, but more data is required. It would be instructive to run smaller scenarios, for example 3 or 4 moves, which could be run faster. The question is whether the search space of scenarios is a hierarchy, where multiple smaller scenarios could be merged together to provide a 'super scenario' with more benefit? The additive nature of scenarios is not something that has been tested to date and this would drastically reduce the search space.

[6]The timing is based on a 2.5 min GPU optimised run using nvGraph. The road network would take over 2 hours for each of the 100,000 data points.

As comparisons with real world scenarios was a primary aim of this work, all the computer generated scenarios were sized accordingly and small scale scenarios were not on the agenda. The nature of the optimisation problem and its use in steering human planners through a computer driven optimisation of more focused designs is where this research is targeted.

8 Acknowledgements

DAFNI, Turing QUANT2 TU/ASG/R-SPEU-102, UCL CASA QUANT3?

References

[1] M. Batty and R. Milton. "A New Framework for Very Large-Scale Urban Modelling". In: *Urban Studies, 58(15)* (2021). Batty, M. and Yang, W. (Eds) (2022) A Digital Future for Planning, CASA, UCL, London, available at http://digital4planning.com/, pp. 3071–94. DOI: https://doi.org/10.1177/0042098020982252. URL: https://doi.org/10.1177/0042098020982252.

[2] G. Chaslot et al. *Monte-Carlo Tree Search: A New Framework for Game AI*. https://www.aaai.org/Papers/AIIDE/2008/AIIDE08-036.pdf.

[3] Camil Demetrescu and Giuseppe Italiano. "A New Approach to Dynamic All Pairs Shortest Paths". In: *J. ACM* 51 (Jan. 2004), pp. 968–992. DOI: 10.1145/780542.780567.

[4] HM Government. *High Speed 2 Integrated Rail Plan Whitepaper*. https://www.gov.uk/government/publications/integrated-rail-plan-for-the-north-and-the-midlands. [Online; accessed 20-April-2022]. 2021.

[5] A. Krizhevsky. *The Canadian Institute for Advanced Research: CIFAR 10 and CIFAR 100 Datasets*. https://www.cs.toronto.edu/~kriz/cifar.html.

[6] Stanford Vision Lab. *ImageNet*. https://www.image-net.org/.

[7] Y. LeCun, C. Cortes, and C. J. C. Burges. *Modified National Institute of Standards and Technology database*. http://yann.lecun.com/exdb/mnist/.

[8] Richard Milton. and Flora Roumpani. "Accelerating Urban Modelling Algorithms with Artificial Intelligence". In: *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management - GISTAM,* INSTICC. SciTePress, 2019, pp. 105–116. ISBN: 978-989-758-371-1. DOI: 10.5220/0007727201050116.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010. ISBN: 978-1292153964.

[10] Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. "Using Multi-level Graphs for Timetable Information in Railway Systems". In: Jan. 2002, pp. 43–59. ISBN: 978-3-540-43977-6. DOI: 10.1007/3-540-45643-0_4.

[11] Carl Steinitz. *A framework for geodesign : changing geography by design.* 1st ed. New York : Esri Press, 2012. ISBN: 9781589483330.