Network design for urban light transport

Problem presented by

Martin Lowson

Advanced Transport Systems

Problem statement

The Urban Light Transport (ULTra) project is concerned with the development of an on-demand transport system of driver-less taxis running on their own dedicated guideway network. The network consists of a number of stations together with a collection of one-way guideways linking them up. The Study Group was asked to develop a tool that when given the user demand, locations of the stations, the costs of building and the time taken to travel along all potential links, determines which links to include in the network. The objectives are to balance the benefit to users, in terms of convenience, with the construction cost, and to satisfy topological constraints determined by engineering considerations.

Study Group contributors

Z. Guo (Bao Steel)
R. Hoyle (University of Surrey)
D. Jefferson (Heriot-Watt University)
R. A. Leese (Smith Institute)
H. Mashhoudy (University of Coventry)
S. D. Noble (Brunel University)
S. Roper (University of Cambridge)
R. E. Wilson (University of Bristol)

Report prepared by

Rebecca Hoyle (University of Surrey)
Daniel Jefferson (Heriot-Watt University)
Robert Leese (Smith Institute)
Steve Noble (Brunel University)
Steve Roper (University of Cambridge)

1 Introduction

This report concerns the design of networks for personal public transport. It describes the problem as presented to the Study Group by Martin Lowson of Advanced Transport Systems, along with details of how we formulated our approach to the problem, and discussion of preliminary results.

The **U**rban **L**ight **Tra**nsport project (ULTra) is concerned with the development of an on-demand transport system of driver-less automatic taxis running on their own dedicated guideway network. Advanced Transport Systems will build the first such transport network in the world, in Cardiff city centre, within the next two years.

ULTra combines the advantages of mass public transportation (buses and trains) and personal private transportation (cars). Its network of stations and guideways provides personal public transportation when required, non-stop from any station on the network to any other station on the network.

All the network links provided by the guideways are one-way, in that vehicles can only pass along them in one direction. The orientation of a link cannot be changed. For there to be two-way travel directly between two stations requires the construction of two guideways oriented in opposite directions, thus incurring extra cost.

The broad question as posed to the Study Group was:

> Given a (time-independent) origin-destination demand matrix, with elements equal to the travel demand for journeys from each station to each other station, and given also the costs of building guideways between each pair of stations, what network should one build, taking account also of several specified design constraints?

We assume the following:

- The co-ordinates of the stations $P_1, \ldots, P_n$ are fixed in advance. Furthermore for all i and j the cost c_{ij} of building a link from P_i to P_j is prescribed, as is the corresponding user demand per unit time, d_{ij}.
- Junctions between guideways may only occur at stations and take one of three forms: two guideways merging into one, one guideway diverging into two or a combination of a merge and a diverge. Except at junctions, no guideways may cross, so consequently the network is planar.
- We do not consider the problem of capacity on the network.
- All vehicles travel at their constant top speed. Therefore the shortest travel time between the stations occurs along the path of shortest length.

The question of how to *dimension* the network must then be answered. The network is naturally modelled as a digraph, *i.e.* a set of vertices (representing stations) and directed edges, often called 'links' (representing guideways). We wish to know:

- Where should one include links to minimise the appropriately weighted sum of the total length of guideway built and the total lengths of journeys (weighted by user demand)?

- What algorithms are useful for finding the answer to this question for arbitrary d_{ij} and c_{ij}?

The outline of this report is as follows. In the next section we discuss our formulation of the problem in detail and show that it is NP-hard. We next describe methods for forming an initial feasible network and then how we apply the method of simulated annealing to improve upon the solution. Finally we discuss possibilities for further work based on our approach, and some results.

2 Defining the problem

In this section we describe the ULTra problem in detail. The problem is essentially to choose which links to put into a digraph in order to satisfy various constraints, imposed mainly by design considerations, and to minimise cost. We first describe the constraints on the links in the digraph. These fall into three categories, to do with topology, connectivity and planarity.

(i) The **topological constraints** restrict the set of links that may be present at a station. Firstly they require that if there are links into a station then there are also links out of the station and vice-versa. (This is necessary for otherwise vehicles arriving at the station would have no route out of the station and would be stuck there for ever.) Secondly they ensure that any junction is not too complicated and consists of either two guideways merging into one, a guideway diverging into two, or a combination of both. Thus the following topological configurations are possible at a station.

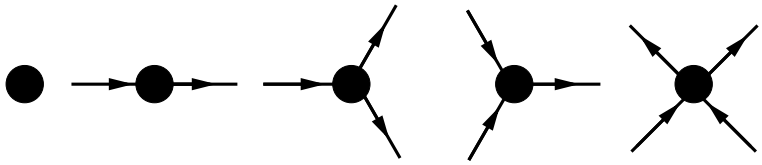

Figure 1: *Possible configurations of track at a junction.*

The solution to the problem has been implemented in MATLAB, and currently uses a slightly simpler version of these constraints in that any junction with in-degree two and out-degree two is allowed.

(ii) To describe the **connectivity constraints**, the following terminology is useful. A digraph is *strongly connected* if for all pairs of distinct vertices v_i and v_j there is a directed path from v_i to v_j. A *strong component* is a maximal set of vertices forming a strongly connected subgraph. Note that the strong components form a partition of the vertices of the digraph. It is not necessary for the digraph corresponding to the ULTra network to be strongly connected, because it may be that some station would rarely be used and so it would not be cost effective to

build links to that station. Suppose however that it is possible to find two disjoint sets A_1 and A_2 of stations so that there are paths through the network beginning in A_1 and ending in A_2 but no paths beginning in A_2 and ending in A_1. Then as passengers make journeys from A_1 to A_2, vehicles will begin to accumulate within A_2 and never be able to get back to A_1. This situation is clearly undesirable and to prevent it we impose the constraint that there are no edges between distinct strong components of the digraph. An equivalent condition is to ensure that whenever there is a route from station P_i to P_j, there is also a route from P_j to P_i. The strong components can be found in time $O(n+m)$, where m is the number of links in the network, using an algorithm due to Tarjan [3]. For more information on connectivity in digraphs see [1].

(iii) Guideways that cross require the construction of either an additional junction or a bridge, both of which are costly and in the case of a bridge, unsightly, so these are forbidden. Consequently the digraph constructed must be **planar**. However checking this constraint is not currently implemented in the MATLAB code. Note that it is not too difficult to check whether the guideway linking stations v_1 and v_2 must cross the guideway linking stations v_3 and v_4. Write the equation of the line joining v_1 and v_2 as $\vec{a}_1 \cdot \vec{x} - b_1 = 0$ and the equation of the line joining v_3 and v_4 as $\vec{a}_2 \cdot \vec{x} - b_2 = 0$. Suppose the stations are at positions $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$, $\vec{x}_4$. Then the guideways cross if and only if both $(\vec{a}_1 \cdot \vec{x}_3 - b_1)(\vec{a}_1 \cdot \vec{x}_4 - b_1)$ and $(\vec{a}_2 \cdot \vec{x}_1 - b_2)(\vec{a}_2 \cdot \vec{x}_2 - b_2)$ are negative.

2.1 The objective function

In the ULTra problem, design decisions are based on the evaluation of an objective function that has two parts: a function C reflecting the infrastructure costs and a function B representing the benefit derived by users from the system.

(i) The **cost function** C is taken to be sum of the costs c_{ij} of building all the links in the digraph. If we let a_{ij} take the value one if the link from station to P_i to station P_j is included and zero otherwise then the digraph is determined by the adjacency matrix $A = [a_{ij}]$. The cost $C(D)$ of the digraph D with adjacency matrix A is given by

$$C(D) = \sum_{ij} a_{ij} c_{ij}.$$

The values of c_{ij} could either be prescribed individually or simply taken as (proportional to) the euclidean distance between the endpoints. However the latter approach might be overly simplistic since the presence of difficult terrain, lakes or rivers, for example, could severely increase the cost of building a guideway between two stations.

(ii) The **benefit function** B is designed to capture the benefit, or 'utility', that accrues to the users of the ULTra system. Such benefit is generally

called the consumer surplus in economics, and is often interpreted as the additional price that passengers would be willing to pay in order to receive the same service. We choose here to replace that willingness to pay with a measure based on journey times. Explicitly, it is assumed that the consumer surplus decreases linearly with journey time, to give a benefit function of the form

$$B(D) = \sum_{i \to j} d_{ij} \max\{a - bt_{ij}, 0\}.$$

The summation in $B(D)$ is over all pairs (i, j) with $i \neq j$ such that there is a route from station P_i to station P_j through D. Furthermore a and b are positive constants, and t_{ij} is the time taken to travel from P_i to P_j on the shortest route, given by $t_{ij} = d(i, j)/V$ where $d(i, j)$ is the distance travelled on the shortest path from P_i to P_j through D and V is the speed of the vehicle. The constant a represents the utility that passengers would derive from the ideal case of instantaneous transport from origin to destination, and b is the value that passengers attach to each unit of their own time (value that is lost as the journey time increases). In this model, there is a maximum journey time a/b, beyond which passengers derive no benefit.

So far, we have not mentioned the question of fares for travel on ULTra. The ULTra developers are anticipating that a fixed fare will be charged for each journey, regardless of distance, in which case a should be reduced by the amount of that fare. Alternatively one could keep the same value of a and change the interpretation of $B(D)$ to be the combined benefits of operator and passengers.

The cost and benefit functions are combined so that the overall objective function is to minimise their difference, that is to minimise

$$\sum_{i,j} a_{ij} c_{ij} - \sum_{i \to j} d_{ij} \max\{a - bt_{ij}, 0\}$$

subject to $a_{ij} \in \{0, 1\}$ and the digraph D satisfying the topological constraints, being plane and such that there are no edges between distinct strong components.

As a small example, consider 3 stations, arranged in an equilateral triangle. The cost of all the links in either direction is then the same and we can call it λ. Then

$$c_{ij} = \lambda \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}. \tag{1}$$

We assume the demand is symmetric, *i.e.* as many people want to go from P_i to P_j during the day as from P_j to P_i during the day. Then

$$d_{ij} = \begin{pmatrix} 0 & d_1 & d_2 \\ d_1 & 0 & d_3 \\ d_2 & d_3 & 0 \end{pmatrix}. \tag{2}$$

There are 64 possible digraphs for this three station arrangement. However, symmetry can reduce the problem. Suppose the demands have rotational symmetry, *i.e.*

$$d_{ij} = d \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}. \tag{3}$$

Now we expect the solution to have rotational symmtery, so we have the possible solutions shown in Figure 2.

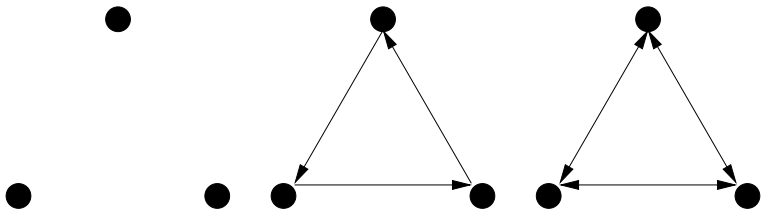

Figure 2: *Possible solutions to the symmetric three-station example.*

In the first possibility, there are no guideways built at all. We derive no benefit from this digraph and incur no cost either. In the second possibility, there exsists a route from everywhere to everywhere, with

$$t_{ij} = \begin{pmatrix} 0 & 1 & 2 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}. \tag{4}$$

The cost is 3λ and the benefit is $3d\,(2a - 3b)$ for this second instance. In the third instance there exists a route from everywhere to everywhere and

$$t_{ij} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}. \tag{5}$$

The cost is 6λ and the benefit is $6d\,(a - b)$.

We can rescale the variables λ, d, a and b, by writing $\alpha = ad/\lambda$, $\beta = bd/\lambda$, and dividing the objective function throughout by 3λ, to give objective function values

$$O_1 = 0 \tag{6}$$

$$O_2 = 1 - 2\alpha\left(1 - \frac{3\beta}{2}\right) \tag{7}$$

$$O_3 = 2 - 2\alpha\,(1 - \beta) \tag{8}$$

for the three possibilities, respectively. This allows an interesting comparison to be made. Depending on the values of α and β, each of the three digraphs may be the optimum in the sense of minimizing the objective function.

For example, take $\beta = \frac{2}{3}$; then for $0 < \alpha < \frac{3}{2}$ the best solution is not to build any links at all. This makes sense as α is proportional to demand and we expect that, given a cost, unless the demand is high enough, it is not worth building a network. More interestingly, suppose the demand is fixed so that $\alpha = 3$, say; then

$$O_1 = 0 \tag{9}$$
$$O_2 = -5 + 9\beta \tag{10}$$
$$O_3 = -4 + 6\beta. \tag{11}$$

As β increases from 0 we have the optimum being to build a one-way system, then switching to build a two-way system and finally not to build at all. Here, the constant β is a measure of how consumers value their time. If consumers are prepared to wait, then one can safely build a one-way system. As the value of their time increases one must satisfy them by building a two-way system, and finally travel-time becomes so costly that it is not worth building at all.

2.2 NP-hardness of the problem

It is not difficult to see that, as stated, the ULTra problem is NP-hard, implying that calculating the exact solution is almost certainly computationally intractable in general. This fact has led us to adopt a heuristic approach to the problem, which we discuss in the next two sections. For background information on computational complexity, see [2].

To see that the ULTra problem is NP-hard, we give a Turing reduction from the Symmetric Travelling Salesman Problem (STSP). An instance of the STSP consists of a set $\{v_1, \ldots, v_n\}$ of points together with positive integers r_{ij} for each i, j with $i \neq j$ and $1 \leq i, j \leq n$, satisfying $r_{ij} = r_{ji}$. The objective is to find a permutation π_0 achieving the minimum over all permutations π of

$$\sum_{i=1}^{n-1} r_{\pi(i)\pi(i+1)} + r_{\pi(n)\pi(1)}.$$

The problem is that faced by a travelling salesman who must visit n cities precisely once and return to the starting location in the shortest possible time.

Given an instance of the STSP, let $R = \sum_{ij} r_{ij} + 1$ and construct an instance of the ULTra problem with n stations $v_1, \ldots, v_n$, $a = 2Rn^2$, $b = 0$, $d_{ij} = 1$ for all i and j with $i \neq j$, and $c_{ij} = r_{ij} + 2R$. (This construction can easily be carried out in polynomial time.)

We will assume that $n > 1$. First note that a solution to the ULTra problem corresponding to a strongly connected digraph has cost at most $-2Rn^3(n-1)+2Rn(n-1)+R$, whereas a solution that does not correspond to a strongly connected digraph has cost at least $-2R(n(n-1)-1)n^2$. Comparing the two, we see that an optimal solution to this instance of the ULTra problem must be strongly connected.

Secondly note that a strongly connected digraph with n edges has cost at most $-2Rn^3(n-1)+2Rn+R$, whereas a strongly connected digraph with more than n edges has cost at least $-2Rn^3(n-1)+2R(n+1)$. Since a strongly connected digraph with n

vertices must have at least n edges an optimal solution must have n edges and correspond to a TSP tour. The cost of the optimal solution will be $-2Rn^3(n-1)+2Rn+C$, where C is the cost of the shortest TSP tour. Hence the ULTra problem is NP-hard.

3 The starting configuration

The first stage in our algorithm produces an initial solution. This is a set of edges which forms a feasible solution of reasonable cost. Note that a simple way to do this is to form a TSP tour using one of the many construction heuristics, such as farthest insertion. The method we use is a little more complicated but is essentially a greedy algorithm applied in a careful way to ensure that the topological, planarity and connectivity constraints are all observed.

Since links between stations that are a long way apart are likely to be costly, we begin by restricting the set of possible links to those which join neighbouring, or close to neighbouring stations. This means that the initial solution will contain only relatively short links. We do this using a Delauney triangulation (a construction that MATLAB can produce using an integral function), which may be thought of as follows. Given a set of points, $V=\{v_1,\ldots,v_n\}$, define the half-plane H_{ij} to be the set of points for which the distance from v_i is at most the distance from v_j. Let

$$V_i=\bigcap_{j\neq i} H_{ij}.$$

Thus V_i is the Voronoi region around v_i. Now construct a triangulation on V by adding an edge between v_i and v_j if V_i and V_j intersect in a line segment. This gives the Delauney triangulation of V. In our initial solution we allow links between stations that are of graph distance at most two in the Delauney triangulation (where every edge has distance one) so let E^2 denote the ordered pairs of stations satisfying this condition. Furthermore let H denote the graph with vertex set V and links E^2. Since the Delauney triangulation, viewed as an undirected graph, is 2-connected and plane, by Robbins' Theorem [1] it is possible to orient a subset of the edges to give a feasible solution to the ULTra problem.

The main part of the procedure to construct an initial solution is as follows. We choose a vertex v of V and form a sequence $u_1=v$, $u_2,\ldots$ of vertices from V. From u_j we choose the next vertex u_k from amongst the out-neighbours of u_j in H in order to minimise $c_{jk}+w(k-1-t_k)$, where w is a penalty function preventing us from revisiting a recently visited vertex and $t_k=0$ if u_k is unvisited but is equal to j if $u_k=u_j$. A sensible choice for the penalty function is $w(x)=\delta x^{-\gamma}$ for suitably chosen δ and γ.

Since the graph is finite, eventually we reach a stage where $u_j=u_k$ for some $j<k$. When this happens, the links $(u_j,u_{j+1}),\ldots,(u_{k-1},u_j)$ are inserted into the initial solution.

We now delete from H any links that are not compatible with the links already present in the initial solution. More precisely we delete any link which crosses a link in the initial solution and any link which cannot be added without violating the topology constraints, *i.e.* causing a vertex to have in or out-degree three, or causing a vertex to have in and out-degrees both equal to two in a way that cannot be created as a combination of a merge and diverge.

Now let $U = \{u_j, u_{j+1}, \ldots, u_{k-1}\}$ and replace the vertices of U in H by a single vertex v_0. We add a link (v_0, v_i) if there is a link (u, v_i) for some $u \in U$. The cost of this link is given by the minimum cost of all the links from vertices in U to v_i. Links (v_i, v_0) are added similarly. We now start to repeat the procedure above choosing $u'_1 = v_0$ as our starting point and picking an out-neighbour u'_2 as before. We continue to construct a sequence of vertices in the same way as before but with one proviso, which affects the cost of a link closing the cycle by returning to v_0 and ensures that such a link is compatible with the first link chosen. Suppose that the minimum cost over links in H from vertices in U to u'_2 was achieved by the link (u, u'_2). The cost of adding a link (v, v_0) is given by the minimum cost over all links in H from v to vertices of U that are compatible with (u, u'_2).

The process is repeated until H becomes a single vertex. Once this has happened we have a strongly connected digraph that can be used as an initial solution.

4 Combinatorial optimization

The problem of deciding on a good network design is an example of a *combinatorial optimization* problem. This wide class of problems can be be phrased in a general way by specifying a set $\mathcal{R}$ of configurations, a subset of feasible configurations $\mathcal{S}$ and an objective function $O : \mathcal{R} \longrightarrow \mathbb{R}$. If we define O in such a way that a lower value corresponds to a better solution, the problem is then to find s^* such that $O_{\text{opt}} = O(s^*) = \min_{s \in \mathcal{S}} O(s)$. There are various computational approaches available, depending on the precise nature of the problem. Some problems are solvable in polynomial time, meaning that there are efficient algorithms that find the minimum possible value of O, in time that increases only polynomially with the size of the problem. The ULTra problem, in contrast, is NP-hard and more akin to the travelling salesman problem, where no such algorithms are known. Calculating the global optimum would take a time that grows exponentially with the size of the problem, and so instead we use approximations or heuristics that give near-optimum solutions in reasonable times.

4.1 Informal discussion

The issues involved in seeking the global minimum can be summarised with the aid of a diagram and an analogy with mountaineering. Consider the mountain shown below with some adventurers at the top, who wish to get to the bottom.

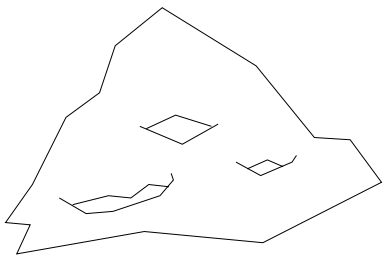

Figure 3: *A mountain with local minima.*

First, suppose that visibility is poor, so that the adventurers can see only the area immediately around their current location. They also have little time to decide in which direction to go. Their strategy is to set off from the summit, look around randomly and make a move in the first direction that goes down the mountain. They may be lucky and choose a path straight down the mountain, but they may also encounter one of the holes, *i.e.* a *local minimum*, where their strategy tells them to stay put, since no downward route is available. They have no idea how far away they are from the base of the mountain. The outcome depends very much on where the starting point is. General deficiencies of such *iterative improvement* algorithms are:

- When a minimum is reached there is no indication of proximity to the global minimum.
- The solution depends heavily on the starting position.
- It is difficult to tell how long the algorithm will take to terminate.

Possible improvements are to extend the range of investigation before deciding on the next direction in which to move, to try several different starting positions, to keep information about previous descents, or to have the option of backtracking to previous positions.

4.2 Simulated annealing

A general and widely applicable advance over iterative improvement algorithms is to use *simulated annealing*. The simple approach of iterative improvement is supplemented with some random steps that may increase the value of the objective function. The frequency at which we make these random moves depends on a *temperature*, which decreases over time according to a *cooling scheme*. This is the type of algorithm used to approach the ULTra problem.

The name 'simulated annealing' is borrowed from the physical process in which the temperature of a solid is raised so that it becomes liquid. The temperature is then reduced, slowly enough so that the system is in thermal equilibrium. Done properly, the solid arranges itself into a regular lattice, which is a lowest-energy state, corresponding to the global minimum of the objective function in combinatorial optimization. If the temperature is reduced too rapidly, the solid freezes with defects, corresponding to a local minimum.

In our notation, the algorithm generally works in the following way:

- We begin at some temperature T and with some feasible configuration s.
- For each temperature we generate new configurations. How many we generate for each temperature depends on our *cooling schedule*.
- Suppose we have a candidate configuration t, chosen randomly from the neighbourhood of s (the neighbourhood of s is the set of all configurations reachable by applying one of the allowed local moves to s). Then t is accepted with probability 1 if $\Delta O_{st} = O(t) - O(s) \leq 0$

and with probability $\exp(-\Delta O_{st}/kT)$ if $\Delta O_{st} > 0$, where k is some constant. This is called the *Metropolis criterion*.

- After a number of iterations, the temperature is reduced, again according to our *cooling schedule* until some cut-off is reached at which we terminate the algorithm.

For different choices of cooling schedule we can produce various bounds on the probability that the global minimum is obtained and the expected time taken to find a near-optimal solution. Further details can be found in [4].

5 Application to the ULTra problem

In the ULTra problem, the set $\mathcal{R}$ is the set of all possible digraphs given the various station locations. The set $\mathcal{S}$ is the set of all digraphs which satisfy the constraints of topology, connectivity and planarity and the objective function O, which, as described elsewhere, depends on the user demand, the shortest path travel times between the stations and how costly the links are to build. Dependence on the travel times along the shortest paths means that removing single links from the digraph may radically alter the current value of the objective function. Moving around the neighbourhood of the current configuration s may produce variation in O on the scale of $O(s)$.

5.1 Changing individual links

Before discussing the way the algorithm finds feasible moves, we need to define a temperature function that gives the probability of accepting a bad move (*i.e.* to an increased value of the objective function) at any particular step[1]. The function chosen at the Study Group was

$$T(k) = c_1(1 - \tanh(c_2 k)).$$

Here k is the number of successful moves so far. The constant c_1 is between zero and one and is equal to the temperature at the start. The constant c_2 is greater than zero and controls the rate the temperature decays. The precise nature of the function is not important. Any function of the same general form would serve the same purpose.

We also need to define a cut-off point at which the algorithm should terminate. This is taken to be the point at which the value of the objective function falls below a certain threshold, or the number of iterations of the algorithm exceeds a certain number.

The starting point for the algorithm is an initial guess, constructed according to Section 3, which satisfies the topological constraints. The objective function is calculated for this initial guess and the steps of the algorithm are as follows:

(i) Randomly choose a link, *i.e.* randomly choose a pair of stations.

[1]The notion of temperature used here differs from the notion previously described in Section 4.2. There the probability of accepting a bad move, rather than being equal to the temperature, is a function of the number of iterations, the change in the objective function and a temperature which decreases every so often according to a cooling schedule. During the Study Group week a simpler approach to cooling was taken.

(ii) Randomly choose between the following two options:

(a) Add the chosen link if is not already present or remove it if it is.

(b) If the chosen link is present, randomly chose between either changing its beginning station or its end station.

(iii) If the chosen move violates the topological constraints, then reject it and go back to step (i).

(iv) Calculate the objective function for the network modified by the chosen move. If the objective function has decreased, accept the move. If the objective function has increased, accept the move with probability T (where T is the temperature). Otherwise reject the move and go back to step (i). If the move is accepted, continue.

(v) Stop if the objective function has decreased below a specified threshold, or a specified maximum number of iterations has been reached. Otherwise continue.

(vi) Update the temperature T and return to step (i).

The details of the implementation of this algorithm are given in the flowchart in Figure 4.

5.2 Larger possible moves: ears

Potentially it is possible to improve the above algorithm by allowing it to make more radical moves (which still satisfy the topological constraints) than the rather piecemeal moves described above. One way to do this is to allow moves that involve the addition or removal of whole chains of links, rather than individual links alone, in the following manner. The idea here is to form an ear. A link is inserted which branches off the existing network to an isolated station, and then successive links are added to form a chain linking up currently isolated stations, eventually connecting back up with the existing network.

Step (i) of the above algorithm selects a random link (a, b). We test whether this link is the potential start of an ear, which will be the case if the link is not currently present, its presence would not violate the topological constraints at a and if b is currently an isolated station. If these conditions are satisfied, the ear-generating algorithm proceeds as follows:

(i) Choose the nearest station to b other than a (and b itself). Here 'nearness' could be defined in terms of euclidean distance or cost. The euclidean definition perhaps helps to maintain planarity. Call this station b^*.

(ii) If b^* is an isolated station insert the link (b, b^*) and repeat from (i) with $a = b$ and $b = b^*$. Otherwise continue.

(iii) If b^* is not isolated and the presence of the link (b, b^*) does not violate the topological constraints, insert it. This completes the formation of the 'ear', and the ear-generating algorithm terminates. The effect of this new route on the objective function is then calculated and the

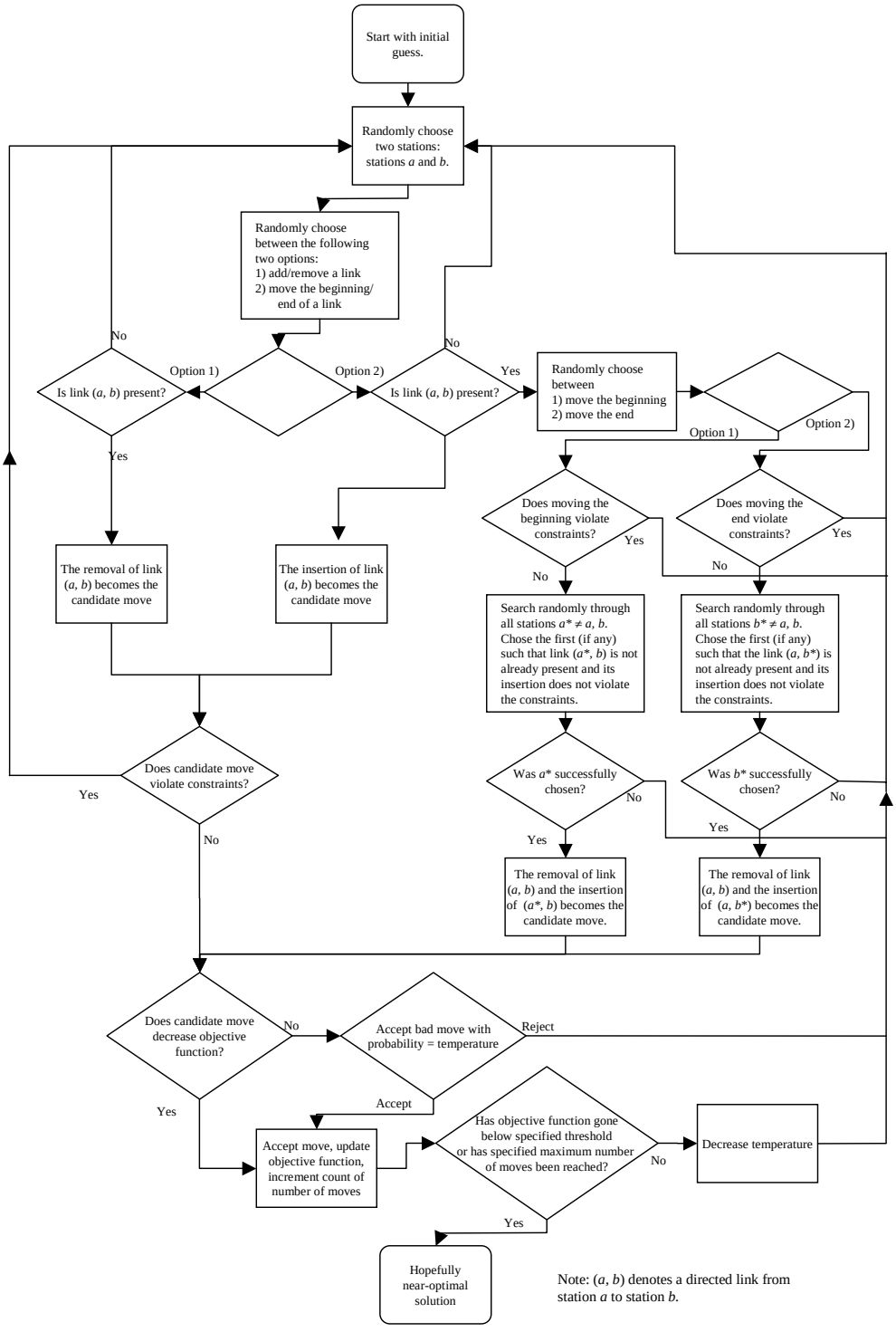

Figure 4: *Flowchart for the simulated annealing algorithm.*

simulated annealing algorithm decides whether to accept it or reject it (*i.e.* abandon the whole new route). Otherwise continue.

(iv) If the presence of (b, b^*) violates the topological constraints the whole new route is abandoned and the simulated annealing algorithm tries another move.

Clearly, ears can be added in the opposite direction, *i.e.* starting at the end of the ear, in the same way. A similar idea can be used to delete existing ears.

5.3 Other factors

In the computer algorithm as implemented in the time available to the Study Group, the issue of planarity (the avoidance of crossing links) has not been addressed. For the most part, the algorithm seems to avoid nonplanar solutions automatically, probably because such solutions are relatively costly (in terms of the objective function).

Another issue that has been avoided here is that of congestion. The above algorithms do not involve any check on the possibility that the usage of certain links may become unfeasibly high for particular networks. This could be addressed by imposing additional constraints on moves. For example, we could require that along any particular route there cannot be two successive merges with other routes without a divergence of routes in between. Checking this type of additional topological constraint is straightforward in principle, but might be complicated in practice. An alternative way to deal with congestion would be to estimate the number of vehicles per unit time passing along any particular link (using an approach similar to that used in the calculation of benefit in the objective function). This estimate of usage could then be incorporated as an element of cost in the objective function.

A further issue is that of robustness of the network. If one of the links were to be unusable for a period of time, how would the network respond and would the resulting, albeit temporary, network have the desired topological properties and give convenient journeys to customers?

6 Some results

6.1 A small example

We illustrate the algorithm developed in this report by applying it to two examples. The algorithm is rich and merits a deeper discussion than is possible here. The results of running it on more examples would likewise merit further investigation.

Consider the demand matrix

$$d_{ij} = \begin{pmatrix} 0 & 15 & 1 \\ 15 & 0 & 5 \\ 1 & 5 & 0 \end{pmatrix}. \tag{12}$$

The three stations are located at the vertices of an equilateral triangle, with the same travel times between all pairs of vertices. The constants a and b in the benefit function are chosen so that $a = 1$, with b equal to either $\frac{1}{10}$, $\frac{1}{3}$ or $\frac{1}{2}$.

The starting digraph for each run is the top digraph in Figure 5. For each different value of b, the algorithm produces different solutions. Recalling the discussion of the objective function in Section 2.1, we expect that for small b the solution to be a one-way loop through all the stations. As we increase b, we expect more complicated solutions to appear and finally for large b we expect that not building anything will be the best solution.

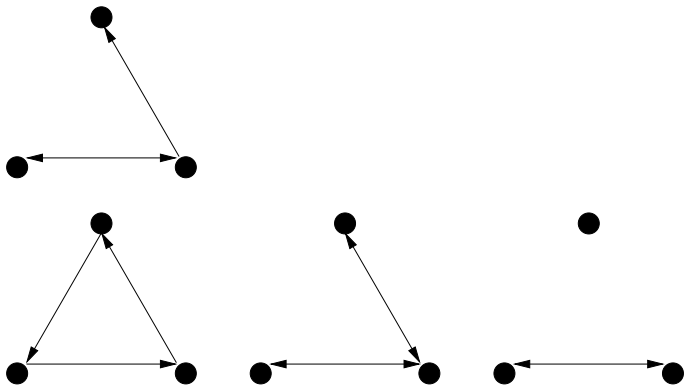

Figure 5: *Start position and solutions for* $b = \frac{1}{10}$, $b = \frac{1}{3}$ *and* $b = \frac{1}{2}$.

The bottom row of digraphs in Figure 5 are the final configurations for $b = 1/10$, $b = 1/3$ and $b = 1/2$, respectively. We see that as the customers value their time more and more, we have to build two-way guideways in order to minimise the cost incurred from making long journeys. Eventually a point is reached where it only becomes feasible to build the link with most demand. At b equal to approximately $\frac{3}{4}$ the best solution is not to build at all.

6.2 Bristol example

The Study Group had available data for a potential deployment of ULTra in Bristol, in the form of a symmetric demand matrix, based on an analysis of peak hour car trips between 16 different zones of the city. In the analysis of this data, a station was attached to each zone and, in the absence of additional information, we took the cost of building a link to be proportional to the euclidean distance between the stations. We can generate a range of solutions, depending on the choice of parameters in the objective function. Two possibilities are shown in Figures 7 and 8, both derived from the starting configuration of Figure 6. The spatial coordinates of the stations have been rescaled to lie inside a unit square.

The objective function in Figure 7 weights the user benefits more heavily, relative to construction costs, than in Figure 8, and this is reflected in the inclusion of some long, expensive links. However, both examples consist mainly of unidirectional loops, rather than pairs of parallel guideways in opposite directions. Further increases in the importance of the user benefit would be expected to lead to the appearance of more parallel guideways and fewer loops. Note also that Figure 7 exhibits some violations of

the planarity constraints, since these have been omitted from the initial implementation in MATLAB.

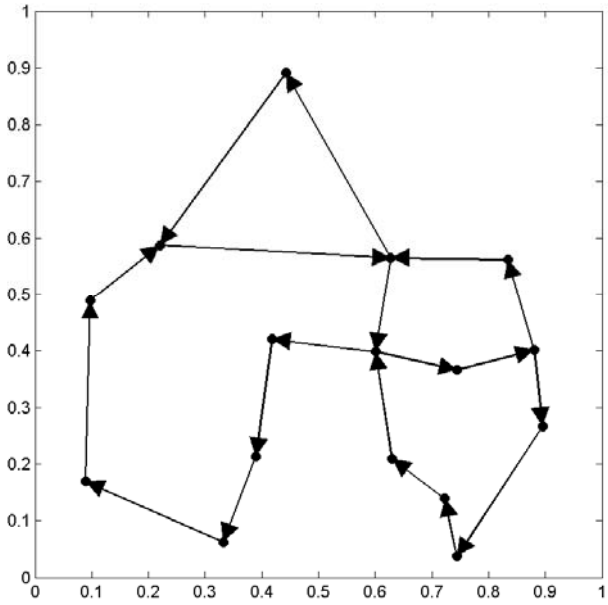

Figure 6: *Starting position for Bristol data.*

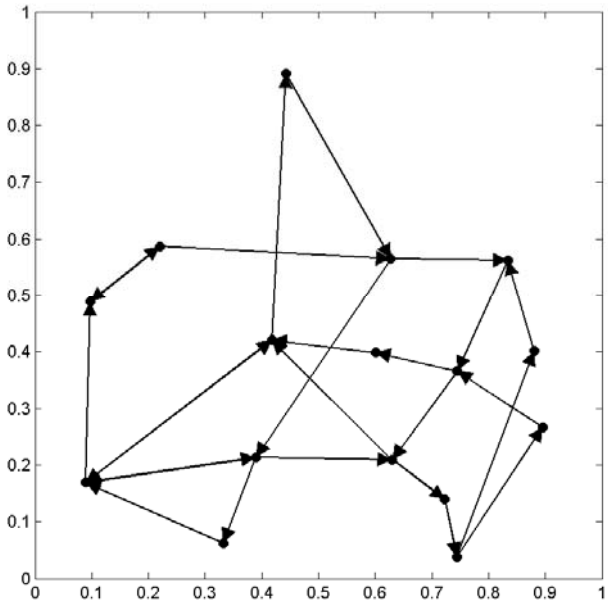

Figure 7: *Final network configuration for Bristol data, with the objective function equal to* $B(D) - C(D)$.

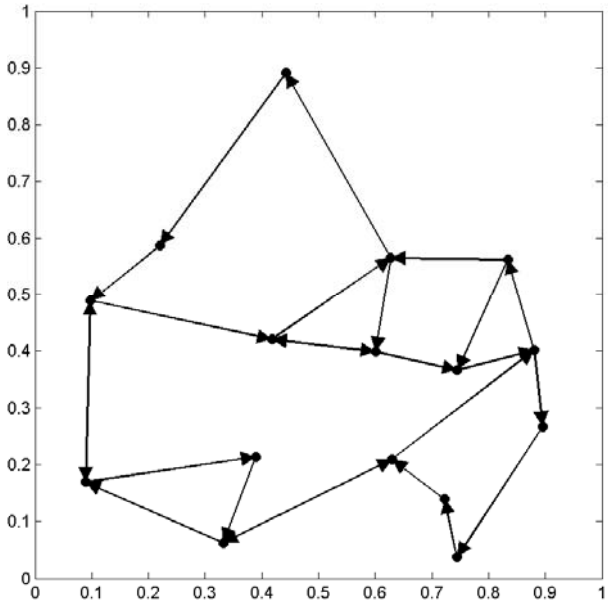

Figure 8: *Final network configuration for Bristol data, with the objective function equal to* $B(D) - 5C(D)$.

References

[1] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications.* Springer, 2001.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[3] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[4] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company, 1987.