TMM Lab 03: Assignment Modelling

RE.Wilson@bristol.ac.uk

10 October 2025

Starters: Parallel networks

These questions relate to the parallel network shown in Fig. 1, with n links. The simple case where $n = 2$ and the cost functions are $c_1(x_1) = x_1$ and $c_2(x_2) = 1$ recovers the Pigou example used in the prerecorded video. The idea here is to develop your basic understanding of traffic assignment mathematics before building up to more complex networks. Note that some of the questions here are best approached with by-hand calculation only, or by combinations of by-hand calculation followed by computer implementation.

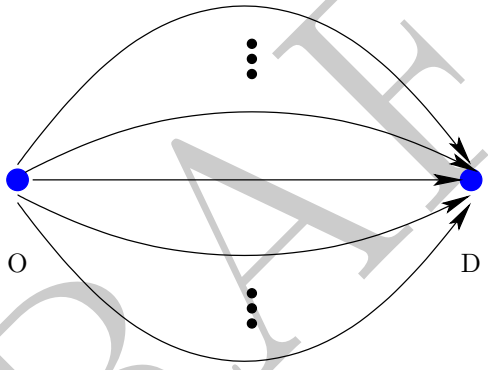

Figure 1: Parallel network with n links and a single OD pair with demand $d > 0$. Denote the link flows $x_1, x_2, \ldots, x_n \geq 0$ with costs $c_1(x_1), c_2(x_2), \ldots, c_n(x_n) \geq 0$.

1. By-hand calculation. Extend the Pigou example by computing the UE and SO flows for a network with two parallel links and cost functions $c_1(x_1) = 1/2 + x_1$ and $c_2(x_2) = 1 + x_2/2$. To do this, begin by choosing a special value of demand e.g., $d = 1$, so that we require $x_1 + x_2 = d$.

   (a) Compute the UE solution by the complementarity method — i.e., explore the 3 possible patterns of used/unused links and discover values of x_1, x_2 so that either both links are used ($x_1, x_2 > 0$) and their costs are equal, or, where one link is unused and has equal or higher cost than the other.

   (b) Formulate the total system cost as a quadratic in x_1, x_2. By writing $x_2 = d - x_1$, rewrite it as a quadratic in one variable x_1. Find the minimum (i.e., the SO assignment) noting that the case where both links are used $0 < x_1 < d$ and where one link is unused ($x_1 = 0$ or $x_1 = d$) require separate treatment.

   (c) Thus verify that the system cost of the SO assignment is less than or equal to the system cost of the UE assignment. Compute the Price of Anarchy.

(d) Re-do part (a) by instead minimising the Beckmann functional, following a similar technical approach to part (b) — thus confirm that this approach returns the same answer as the complementarity method.

(e) Re-do (a) and (b) for general demand d and plot the Price of Anarchy versus demand d. What is the maximum value?

2. Extend the previous question to consider more general cost functions $c_1(x_1) = a + x_1$ and $c_2(x_2) = 1 + bx_2$, where $a, b \geq 0$ are parameters. Note that a rescaling argument can be used to show that you may assume without loss of generality (WLOG) that $a, b \leq 1$.

   (a) Carefully write down the UE problem (in Beckmann form) and the SO problem — that is, give the quadratics to be minimised together with the constraints that must be satisfied.

   (b) Develop code that for fixed parameters a, b, and d, solves the minimisation problems from part (a). Clue: there are several approaches to this. The best is to use Matlab's in-built function `quadprog`. If you did Optimisation last year, this should be easy. However, you probably didn't, so we will help you out with this.

   (c) Develop code that for fixed a and b 'scans' over a suitable range of demand $d > 0$ and explores how the UE and SO assignments, and the Price of Anarchy, change as demand is varied. Study the solutions. Notice situations where e.g., the UE and SO solutions are the same, where they are different but costs are similar, and how links are used or unused at different demand values in each assignment.

   (d) For fixed a, b, plot the Price of Anarchy against demand d. Are there any principles as to how large d has to go to make sure you capture all the interesting behaviour?

   (e)* By hook or by crook, develop code to find the maximum PoA for any given a, b, and the demand d at which it occurs.

3.* Develop code that employs your answer to 2(e) and, either sweeps over $0 \leq a, b \leq 1$, or employs an optimisation routine, to find the maximum possible value of PoA for this set-up, thus verifying the classical result for affine cost functions.

We are now draining the dregs of parallel (increasing) cost functions — but here are some further possibilities for investigation.

4.* Explore parallel networks with n links. Maybe use affine cost functions with randomly chosen parameters. Set up and solve the quadratic minimisation problems. Enjoy. Explore Price of Anarchy.

5.** What about nonlinear cost functions?

Braess paradox

This section relates to the Braess network with nodes, links and routes as described in Fig. 2. Again — this is a mix of by-hand calculation and computation.

6. By-hand calculation. I think you can do all of this by solving UE with the complementarity method and it should be tractable by-hand. You can assume without proof that if the middle link is closed (and thus the network overall has a kind-of top-down symmetry), that the SO and UE assignments are the same and share traffic equally between the Northern and Southern routes.

   (a) Work through the lecture example carefully so that you understand it.

   (b) Will the network display the Braess paradox if we change the value of demand d?

   (c) Will the network still display the paradox if we change the link costs over so that c_1 and c_4 are constant and c_2 and c_3 are proportional to flow?

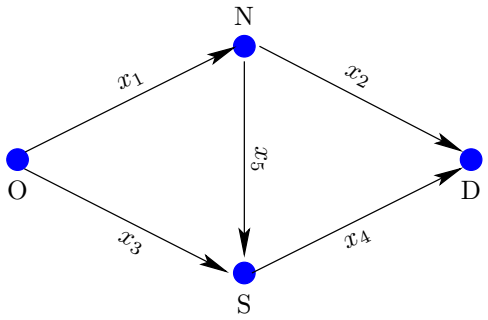

Figure 2: Braess network with labelling of nodes and link flows. The northern route is links 1,2. The southern routes is links 3,4. The mid-town route is links 1,5,4.

(d) Investigate what happens if the mid-town link is replaced by one of a constant cost $a > 0$, for various values of a.

7. By-hand calculation. Work through the UE and SO solutions of the Braess network for the parameters given in Sheffi p76 (p92 in the pdf). Here $c_1 = 10x_1$, $c_4 = 10x_4$, $c_2 = 50 + x_2$, $c_3 = 50 + x_3$ and $c_5 = 10 + x_5$, and demand $d = 6$ units. Hence demonstrate the Braess paradox. (Note Sheffi draws the network the other way up from me, with the mid-town link going up the page.)

8.* Consider the Braess network with $c_1(x_1) = a + x_1$, $c_4(x_4) = a + x_4$, $c_2(x_2) = 1 + bx_2$, $c_3(x_3) = 1 + bx_3$ and $c_5(x_5) = 0$. Derive conditions on the parameters a, b, and $d > 0$ for the Braess paradox to be displayed. If you really want to show off — do this by implementing the Beckmann method to solve for UE when the mid-town link is present, and explore with a numerical solver.

Things can only get bigger

So now we move on to bigger networks. Really the only way to address these is to use minimisation (either of the total system cost to find the SO assignment, or of the Beckmann functional to find the UE assignment).

9. Consider the 'right-pointing' scissors network discussed in the lecture videos, where there are two origins and one destination, and study the associated Matlab code `networkExampleLecture2021.m` which is set up to solve for assignments on this network using quadratic programs. Study and understand how the code works. In particular, confirm that the UE solution is the same irrespective of whether it is formed in route variables (with constraints that they add up to OD demands) or link variables (satisfying node conservation constraints). Observe that despite there being two OD pairs, this is an unrealistically simple 'converging flows' example which can be solved with link variables without keeping track of the different constituents of the flow on each link (what I call 'colouring' in lectures).

We are pretty much done with route variables for now. Although route variables avoid 'colouring' problems, the computation of all routes on any realistic network is enormous. However, to proceed in baby steps, for the time being we will restrict to one OD pair. So the formulation is: nasty network, links equipped with affine cost functions whose parameters are provided, one node identified as origin, one node identified as destination, demand provided, and problem formulated in terms of link variables, with node conservation constraints.

10. Study the provided exemplar code snippets in `BigNetworkExperiments.mlx` which set up such assignment problems on the square grid network introduced in last week's lab sheet. In particular:

(a) Note there is a general formulation you can adapt which will turn quite general problems into the relevant objective function and constraints for use with `quadprog`.

(b) Note the free-flow costs a_i have been set up equal to the length of the links (assuming the same speed on each). The parameters b_i represent the 'congestibility' of each link, something like the inverse of capacity. For simplicity, we could make them equal, although it might make solutions boring.

(c) Solve for the UE and SO solutions. Experiment with varying the demand. Write code to visualise the flow on each street and show how the network is filled out as the demand is increased.

(d)* For a given value of demand d we are going to study the relationship between link variables and route variables. Note the UE solution has been provided in terms of link variables x_i. For each link, assign a weight $a_i + b_i x_i$ which is the per-vehicle link cost that is actually realised in the UE assignment. Consider the network with these weights assigned to edges, and seek shortest paths. Discuss.

11. Grid networks?

12.** You'll notice that as demand is increased in the Q10 formulation, the links that connect directly to the origin and destination nodes become extremely heavily loaded. This is not realistic as these flows would exceed those links' finite capacities, the modelling of which we have neglected. Think about possible approaches where the origin (resp. destination) flow is spread out over a set of nodes. Note something like this might be needed if our demand is prescribed over an area (for example) at MSOA level.

Multiple OD pairs

13.* Study and play with the provided (?) Matlab code `megaProblem.mlx` for working with the grid network where every node is an origin and destination and thus in general flows need to be 'coloured' according to either their O or their D (not both). In particular, note the huge size of the variable vectors and constraint matrix, and observe how the latter is built up by stacked copies of the single OD problem.

14.** Beta skeletons, real-world maps, have a go?

Stochastic user equilibrium

As in previous years — I have decided to let you off and not include it in the core curriculum.