Ne'er the Twain Shall Meet?

Bridging the Network Optimisation Reality Gap

In principle it works like this ... A practitioner, working with a real-world problem, calls on the services of the theorist. The theorist takes the problem, abstracts and generalises it, finds a solution and hands back the finished model to the practitioner, to be applied as required. It is of course no secret that it often doesn't work like this! Science and technology in general and mathematics in particular have chequered histories. Among the great works, from the sub-atomic to the astronomical, are some less auspicious chapters. In these, bees do not fly, cricket balls do not swing and suspension bridges last forever - but then fall down in the first wind!

Some excellent, if worrying, examples of this are to be found in the field of communication network optimisation; a broad subject but one that may be summarised as the pursuit of the 'best' solution to various design/management issues in data communications, networking and internetworking. This is interesting because mathematics, through the nodes, edges, arcs and flows of graph theory, appears to model the locations, links and traffic of the Internet well. So why do the graph algorithms not work? Why is networking software sub-optimal? And why do those most closely involved often fail to see the problem? We try to answer these questions through two simple, but very real, networking problems. For balance, the first is concerned with building a new network; the second with making best use of an existing one.

In what follows, we use - very loosely and hopefully without insult - the term *theorist* to suggest a 'conventional' mathematician with pencil and paper. In contrast, a *practitioner* writes programs to design or control network equipment.

The 'Cheapest' Network

This may be the simplest - as in the easiest to state - networking problem there is. Given a number of locations, what is the best way to connect them together? Without further ado, we hand the problem to the theorist ...

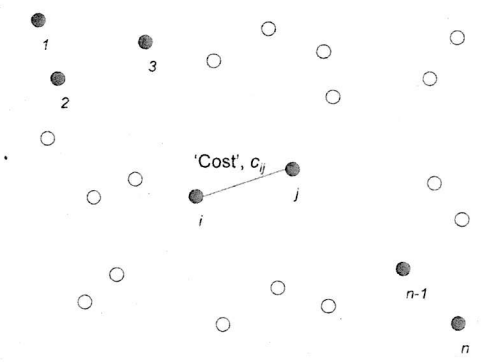

Figure 1. Nodes and costs

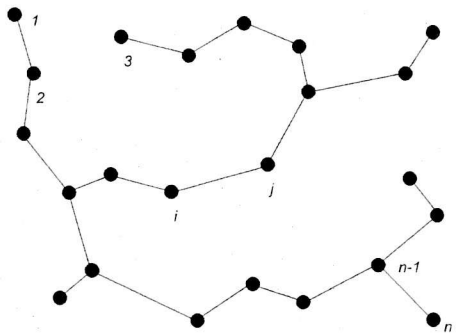

Figure 2. An MST solution

To optimise anything, we need an objective function. In this case, cost seems the likely candidate. Figure 1 lays out the model: n nodes with an $n \times n$ cost matrix $C = (c_{ij})$ where c_{ij} is the cost of connecting node i to node j. We could insist that costs be Euclidean $(c_{ij} = \left[(x_i - x_j)^2 + (y_i - y_j)^2\right]^{\frac{1}{2}}$ where (x_i, y_i) and (x_j, y_j) are the coordinates of i and j) or that the cost matrix be symmetric $(c_{ij} = c_{ji}$ for all $i, j)$ but neither is essential. Non-Euclidean costs allow local factors to be considered whilst an asymmetric cost matrix permits unbalanced loads so our model, at first sight, looks flexible enough to keep everyone happy. And in fact it gets better because our objective would now appear to be to find a tree T^* that minimises total cost $C = \sum_{(i,j) \in T^*} c_{ij}$ and some very old algorithms (Kruskal, 1956 and Prim, 1957 for example) will do this very efficiently. The result will be a minimal spanning tree solution something like Figure 2.

Unfortunately, give this solution back to the practitioner and they will laugh (or worse), and it is hardly difficult to see why. This is not a 'real' network. The MST solution is entirely impractical for a number of reasons, but two in particular scream out:

- Firstly, the network is hopelessly vulnerable. A single failure, of node i say or of the link (i,j) is catastrophic. There is no redundancy, no alternate routes, so one half of the network is cut off from the other.
- Secondly, some of the paths between nodes are ridiculously long, particularly between apparently near neighbours. Subscribers at nodes *2* and *3* for example are hardly likely to be impressed that data between them passes half way round the network.

OK, so the MST solution is not such a good one after all. The practitioner is more likely to be impressed with a network something along the lines of Figure 3. This is a good compromise. A more strongly-connected core network carries most of the traffic.

Although peripheral nodes may still lose contact through failure, the rest of the network is unaffected. If a core node or link fails there are alternate paths; and no paths are particularly long.

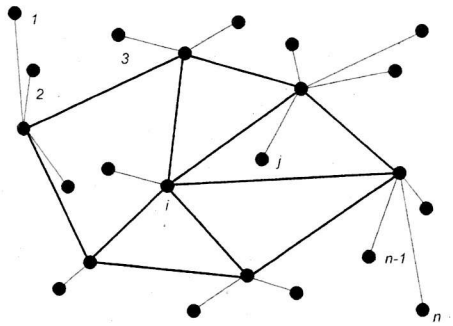

Figure 3. A more robust network

No need for the theorist to give up yet though. It is still no bad thing to be minimising cost but the solution must be constrained. We can seek to find an optimal set of core nodes, whilst constraining the structured solution. We can insist that each peripheral node connects to its nearest core node. We might constrain the path length between core (and hence peripheral) nodes or insist upon a minimum number of (node or link) independent paths between nodes but this, in practice, proves to be difficult (Garey & Johnson, 1979). Better in fact to specify a minimum degree for each core node. In fact, if necessary, peripheral nodes can connect to two or more core nodes for robustness (not shown in Figure 3). The old MST algorithms no longer work of course. In fact, it is now difficult to find perfectly optimal solutions at all for larger problems. However, a combination of greedy algorithms and local searches will usually come up with a tolerable solution. Heuristics may have replaced exact methods but we seem to still be doing a decent job.

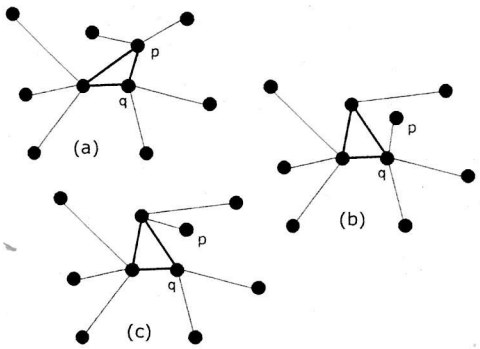

Figure 4. Costs depend on topology

Unfortunately the final blow comes when we take a closer look at these costs. Where does this matrix C come from? How do we determine the cost of connecting i to j? In fact the length of the link, Euclidean or otherwise, has very little to do, in practice, with its cost. Much more relevant is its capacity - the amount of traffic it can carry. A link must be able to carry not just the traffic between its own end-points but all traffic routed over that link. Figure 4, for example, shows three different connection configurations for the same node set. In 4(a) (p,q) is a core link and carries traffic between several node pairs. The 'same' peripheral link in 4(b) will cost less and in 4(c) there is no link at all.

The conclusion that follows is an uncomfortable one. To determine the cost of a link we need to know its capacity. But this can only be calculated if we know how much traffic it has to carry, which implies a knowledge of the topology of the network, which in turn is the solution we seek. We might well have a traffic matrix $T = (t_{ij})$ (where t_{ij} is the traffic originating at i, destined for j) - in fact it would be hard to design a network without it - but it hardly helps. T is a *requirement*, not a *plan*: we still have no idea where the traffic *goes*. To paraphrase: to specify the cost matrix C requires a knowledge of the solution network - *our input needs the output*. Hardly a well-formed optimisation problem!

So, with our theoretical model in ruins, what do we do? In truth it is usually left to the practitioner. Something has to work somehow. The following is typical. If we can only determine the cost of a known network then let's start from there. Put *all* nodes in the core network and directly connect every pair. Knowing (or maybe even guessing) T, it is easy to calculate (or approximate) the traffic on each link, its cost and therefore the cost of the complete solution. Of course, it will be a very expensive solution but that makes it easy to find a better one. Consider, in turn, dropping each node from the core network, decide where the affected traffic goes (it will be re-routed via other nodes/links, possibly making good use of spare capacity), recalculate the cost, and choose the drop that maximises the saving. At an appropriate point, and subject to constraints, consider dropping links from the core network as well. Stop when there do not appear to be any further drops that will improve the cost.

And this 'double-drop' algorithm works. At least, that is, it finds a solution. It will not be optimal of course. It will be some local cost minimum, possibly considerably above the true optimum cost, which may well come from a completely different topology, and the worst of it is that we have no way of knowing exactly how bad it is. But it works nonetheless. We are hardly in a position to criticise the practitioner for such a clumsy effort - something is better than nothing. The theorist has failed. A design, any design, is needed now - work starts next week!

'Optimal' Routing

In contrast, we consider our second example initially from the point of view of the practitioner, albeit one willing to borrow an idea or two from the theorist in the first place. We take, as our starting point a network whose physical topology is already established - although possibly not *100%* reliable. Figure 5 is a good example with which to work, concentrating on the core rather than the peripheral network. The thick/thin, solid/dotted lines simply give a general impression that some links may be larger (have a greater capacity) or more reliable than others.

We also appear to be giving each link a cost! This may appear contradictory, considering the conclusions of the previous example. But this is different, surely? The network topology is now known, the links are in place: of course we know how much each one costs!

And yet, in fact, it is not as simple as that; not if we are now to consider how the traffic is to be directed, *routed*, over the network. Routing is a key process within the Internet - one in which little, if anything can be assumed. Even if the network has been designed with the routing implicit in the double-drop process in the first place (and it probably hasn't - it has more likely just 'evolved'), traffic characteristics change. They change in both the short and long term. Also, things go wrong: links and

nodes fail. Any routing process in a large network must be flexible, *dynamic*. In this environment, the notion of cost needs to be treated carefully. The initial installation cost of each link is hardly likely to be relevant now.

What we are really after is a measure of how efficient a routing strategy is (so that we can try to improve it). The cost of a link is therefore its tendency to restrict this efficiency. In a simple sense, it may be a fixed measure of the link's capacity but it may have a dynamic interpretation - such as instantaneous load, delay or failure. Whatever the practitioner decides the essential metric to be, that is what the routing should seek to minimise. In fact, this opens up an entirely new line of enquiry. *What* exactly should we be trying to measure, maximise or minimise? Throughput? Delay? Customer satisfaction? Bank balance? We simply do not have the room to pursue this particular question here but it turns out to be a fascinating study: a start is made in Grout et al., 2004.

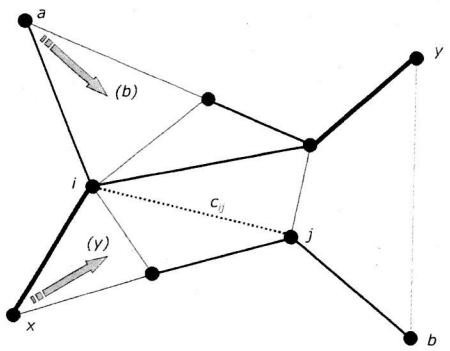

Figure 5. A routing problem

To make any progress at all, we will have to narrow the field - take an example. A typical routing protocol in widespread use, *Open Shortest Path First* (*OSPF*) (Moy, 1998) uses link capacity as its default measure. In fact it calculates link cost as $c = 10^8 / b$ where b is its speed, or *bandwidth*. (Bandwidth is an analogue term unfortunately mutated by the practitioner to mean the speed in bits per second of a digital link.) Thus a *100Mbps* LAN link has cost *1*, a *2Mbps* serial connection, *50*, etc. Minimising cost means finding routes using links with higher bandwidth - which makes sense, surely?

Well, yes. No argument so far. The problem comes with the implementation. Both OSPF and a newer protocol, *Intermediate System to Intermediate System* (*IS-IS*) (Gredler and Goralski, 2004), use Dijkstra's shortest path algorithm or *DSPA* (Dijkstra, 1959) to find these minimum routes. Actually the practitioner is inclined to be a little smug about this: DPSA is both optimal and efficient and comes straight from the theorist's textbook. So it looks like the job's done.

But up jumps the theorist! DSPA is optimal, yes, but it only finds the shortest (least cost) path from one node to another - *in isolation*. If there were only these two nodes in the network, with no other traffic around, it would work wonderfully - but that is not the case. *All* node pairs are (potentially) sending traffic to each other. The independent-pairwise nature of DSPA cannot take into account how routes *compete* for use of these links.

We need to be clear what the problem is here because of course there *are* algorithms that will calculate multiple paths in a network – that isn't the issue. Consider the routes from nodes a to b and x to y using (or not using) the link (i,j) in Figure 5. Either route using the link will reasonably accrue a cost c_{ij} but if *both* use it then, surely, this cost itself has to be reconsidered? Again, using capacity as our example of a cost metric, the cost c_{ij} will be based on the ability of the link to provide a certain level of throughput for a stream of traffic. If it has to deal with two such streams then clearly this ability is reduced; we could argue - possibly simplistically - that it is halved. However we measure it, the cost of the link will increase. In OSPF terms, it may be doubled.

Again, in case this seems too obvious, consider Figure 5. If DSPA, working on the route from a to b, includes the link (i,j) then this adds c_{ij} to its cost. However, if another route shares the link then c_{ij} is doubled so another c_{ij} is added *to the same route*. But the same is also true of the second route. The contribution of the link to the cost of each route has doubled and to the network routing as a whole increased by a factor of four. The real implication is that, had we known this when we started, we might have chosen different - and better - individual routes. DSPA, working independently for each route, has let us down. And, of course, the problem increases in complexity when we consider *all* routes competing for *all* links. Individually 'optimised' routes do not produce optimal routes for the good of the network as a whole. Unfortunately, OSPF and IS-IS are about as sophisticated as current Internet routing protocols get. There may be a few better ones on the drawing board but they have yet to leave it!

Actually, as an aside, it is a complete nonsense to assume that costs are always *added* in this way anyway. Fair enough, if we are considering something like network delay then adding delays through individual links should give the overall delay across the network. But *capacity* measures the link's ability to carry traffic (rather than discard it). It identifies potential bottlenecks. Would not a *minimum* or *maximum* be better than a *summation*? How about cost calculated as probability of failure? This needs a *product* surely? Obvious says the theorist? Try telling the practitioner using the existing protocols!

Well anyway, taking a conscious decision *not* to turn off into such treacherous ground, we shall turn a blind eye to the DSPA/OSPF/IS-IS shortcomings of the previous paragraph and deal with the multiple route problem. How can we derive (or even approximate) the best overall routing when the interaction of these routes interferes with the very costs by which we calculate them? Now, when you put it like that, it starts to sound familiar. Is this not similar to the example from the first section? We dealt with that (at least the practitioner did) using the double-drop method, using what the theorist called an initial solution and local search. Can we do the same here?

Sounds reasonable. What would be our starting point? Well, Dijkstra of course. Start with the routes found by applying DSPA individually but then look to see the effect of shared links. Now recalculate the costs for the affected links (with as much or as little sophistication as you like) and find the 'true' cost of the routing for the entire network. Now apply the local search - little tweaks or *perturbations* to this solution. Consider varying each of the routes, in turn, and in their own various ways. Recalculate costs as before, for the affected links and for the network, and implement the change that maximises the improvement. Stop when there are no more improvements. No, it will not be optimal - but it will be a lot better! Well done. What a shame it won't work!

The stumbling block this time is not *how* to calculate costs, but *where*! The process we have described takes a global view of the network. (We might well ask how it could be otherwise.) If a

network-optimal routing is to be applied then surely it has to be determined *centrally*, in its entirety, then passed to individual network devices to be implemented. Unfortunately, that is not how Internet routing protocols work. Routing is a *distributed* process – it runs independently on each router. Although routers exchange routing *information* to learn the current shape and state of the network, they do not share routing *intent*. There is no interchange that would allow the global link loads to be determined under a *future* routing plan. Hence the effect on link attributes and costs can not be determined as suggested above and the necessary global view of a routing strategy cannot be taken. Distributed routing protocols share past and present information but not their view of the future. Perhaps they should but they don't! Is there anything that can be done about it?

Well, there are some suggestions – *Ant Colony Optimisation* (*ACO*) (Johnson & Perez, 2005) for example appears to have promise. ACO mimics the way individual ants share information to cooperate to find the best strategy for the good of the colony – in finding food for example. In fact, with the newer ideas like this, we see for the first time the theorist and the practitioner beginning to work together – *understanding* and *sharing* a common view of the problem and the requirements of a successful solution. The sad fact is that initiatives like this have emerged before and come to nothing. We are still some way away from seeing ACO routing implemented on a production Internet router.

A Conclusion of Sorts

There is certainly no intention to take sides here. Hopefully the examples given balance the arguments - or maybe apportion blame equally? The theorists are doing good mathematics and the practitioners are solving real problems so no-one is exactly at fault. What is true is that there is a gulf, a reality gap, between the two that probably should be closed, or at least reduced. But, how is this to be done?

Well, to an extent, the process has already begun - and has been going for some time. Of course not all graph algorithms are useless in the real world; if it were not for Dijkstra's algorithm, network routing would be very crude indeed. Some networking solutions are extremely well designed and even elegant - the principles of Internet multicasting, and public-key cryptography (Forouzan, 2003), for example.

And yet, in other areas, the misunderstanding is still there - and needs to be addressed. The theorists are still over-simplifying certain problems. It might not be obvious that an MST solution suits *any* real-world application, let alone a networking one. The practitioners are still using words like 'optimal' when they really mean 'a bit better'; Internet routing is still some way from *optimal*. At the heart of it all is probably a lack of communication. At the very least, we need to get the theorist and practitioner together; better still, to consider why, after all this time, they still misunderstand each other. Better again, start some dialogue and look for solutions. There are so many *good* algorithms out there and so many problems crying out for them. In fact, MST algorithms *are* useful in some real world applications - eliminating loops from bridged networks for example (Norton, 2001). ACO may provide a coherent routing strategy from distributed routing processes - but the ideas have to be implemented. What we need is for the solutions to *match* the problems - and *this* is the *real* challenge. Perhaps an IMA conference on 'Practical Optimisation'?❑

Acknowledgement

Thanks to Mike Headon for proof-reading this piece.

REFERENCES

Dijkstra, E.W. (1959) "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, Vol. 1, pp269-271.

Forouzan, B.A. (2003) *Data Communications and Networking*, McGraw-Hill.

Garey, M.R. & Johnson, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman.

Gredler, H & Goralski, W. (2004) *The Complete IS-IS Routing Protocol*, Springer-Verlag UK.

Grout, V., Houlden, N., Davies, J., McGinn, J. & Cunningham, S. (2004) "A Unified Framework for optimal Routing", *in System Integration for an Integrated Europe (ed. Ehleman, J.)*, Preciosa.

Johnson, C.M. & Perez, E. (2005) "An Ant Colony Optimization Algorithm for Dynamic, Multi-Objective Network Routing", *Proceedings of the International Conference on Internet Technologies and Applications (ITA 05)*, 7th-9th September 2005, Wrexham, UK.

Kruskal, J.B. (1956) "On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem", *Proceedings of the American Mathematical Society*, Vol. 7, pp48-50.

Moy, J.T. (1998) *OSPF: Anatomy of a Routing Protocol*, Addison-Wesley.

Norton, M. (2001) "Understanding Spanning Tree Protocol: The Fundamental Bridging Algorithm, http://www.oreillynet.com/pub/a/network/2001/03/30/net_2nd_lang.html

Prim, R.C. (1957) "Shortest Connection Networks and Some Generalizations", *Bell Systems Technical Journal*, Vol. 36, pp1389-1401.