A New Arc-Elimination Approach: Reducing the Complexity of Network Routing in Two Dimensions

Yuval Cohen
Department of Management & Economics
The Open University of Israel
16 Klausner Street, P.O. Box 39328, Tel-Aviv 61392, Israel
fax.: +972 - 3 - 646-0451 e.mail: yuvalco@oumail.openu.ac.il

Joseph Bukchin
Department of Industrial Engineering
Tel-Aviv University

Abstract

This paper presents a new technique to reduce the complexity of a network by eliminating most of the network's arcs. The reduction is done in a special way so that the reduced network could serve for any routing scheme where each node is visited only once. Examples of such routing schemes are: the Traveling Salesman Problem (TSP), the M Traveling Salesmen Problem (MTSP), the Shortest Path (SP), and the Vehicle Routing Problem (VRP). The reduction is done according to the user input of a worst-case deviation from the optimal routing solution. The main advantage of this approach is the ability to reduce the network complexity and still control the deviation from the minimal routing solution in terms of distance traveled. The other advantage is its general scope - the approach could be applied to many routing problems.
The approach is based on eliminating arcs as long as there is an acceptable alternative that cannot increase the distance between the two nodes by more then a certain value. Suprisingly, the arc elimination technique is not complex: the alternatives (of eliminated arcs) are never computed directly but are guaranteed by using spatial relationships. Thus, the algorithm complexity is polynomial with the number of nodes N (the complexity is $O(N^2*Log(N))$ see arc elimination procedure in section 4). Preliminary quantitative results show that as N grows the proposed technique reduces a mesh that includes N*(N-1) arcs into a network with c*N arcs, where c is a constant.

1. Related research

Lin and Karnighan [1] suggested, for each node, preserving the arcs to the five (5) nearest nodes, and eliminating all others. This heuristic rule is inadequate for large problems. For example, for a 532 city problem [2], one of the arcs of the optimal TSP route is the 22^{nd} nearest neighbor city for one of its end points. Another interesting approach is based on Delaunay Triangulation [3]. Delaunay Triangulation produces a reduced network by eliminating all the arcs that posses the following criterion: for each arc build a circle with a center in the middle of the arc and a circumference that passes through both nodes at the ends of the arc. If that circle does contain at least one other node the arc is eliminated. In [3] a second phase of routing on the reduced network follows. Routing in a reduced network based on geometric relaxation is investigated in [4]. Mata and Mitchell [5] suggested recursive dissection to transform the network into a reduced network. Using statistics results, the heuristic in [5] limits the number of arcs that can cross each dissection line by c*Log(N) (where c is constant and N is the number of nodes). For all the aforementioned techniques, there is no known bound on the deviation from optimality.

2. The deviation from optimality

We assume that the optimal route will visit each node only once. The new approach allows the user to specify the acceptable deviation from optimality, and according to that, the new algorithm (discussed below) eliminates arcs from the network. As the allowable deviation from optimality grows, the amount of eliminated arcs grows and the network complexity declines. The user can be somewhat perimissive with the allowable deviation since the actual deviation is expected to be normaly distributed around half of that allowable deviation.
In order to understand the technique there is a need to go through some geometrical considerations. A more elaborate work on all facets of this work is available in [6]. The next section presents these geometrical considerations.

3. Fundamental geometrical considerations

A preliminary explantion of geometrical considerations is a must for understanding the technique. For the discussion we shall first choose three nodes (X,Y, Z) in a lage network as follows: we start by chosing an arbitrary node and name it node X (see figure 1). Then we find the closest node to X and we call it Y.

Next we choose in the general direction of the line X-Y some nearby node, Z as depicted in Figure 1 below. The three nodes XYZ form a triangle with lines: X-Z=***a***, X-Y=***b***, and Y-Z=***c***. The added distance to a route Z-X by forcing a passage through Y (a route Z-Y-X) is the difference between the two paths. This is: ***(b+c)-a*** as depicted in Figure 1 below. We are now ready to add segment ***d*** that is the distance between node Y and the line ***a*** (line ***d*** originates in Y and is perpendicular to line ***a***).
The nodes Z, Y and the intersection of ***a*** and ***d*** form a right angle triangle, and so does nodes X, Y and the intersection of ***a*** and ***d***. From fundamental Geometric calculation (see appendix 2) we get: ***c-a1<d*** and ***b-a2<d***. Since by definition ***a1+a2=a*** we get ***(b+c)-a < 2d.***

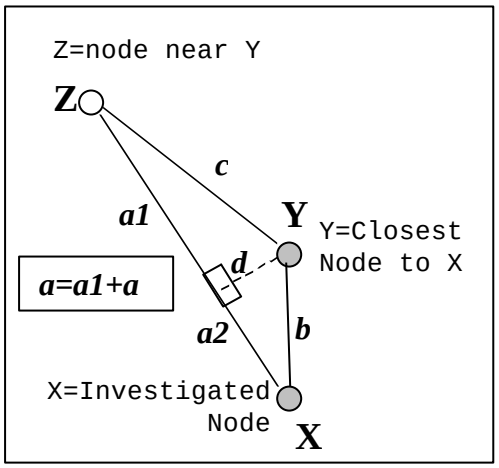


Figure 1.

The distance of going directly from Z to X is ***a***.
Going from Z to X to go through Y is ***b+c***.
Appendix 2 shows that ***b+c- a< 2d.***
Thus, we add less then distance *2d* by going from Z to X through Y.

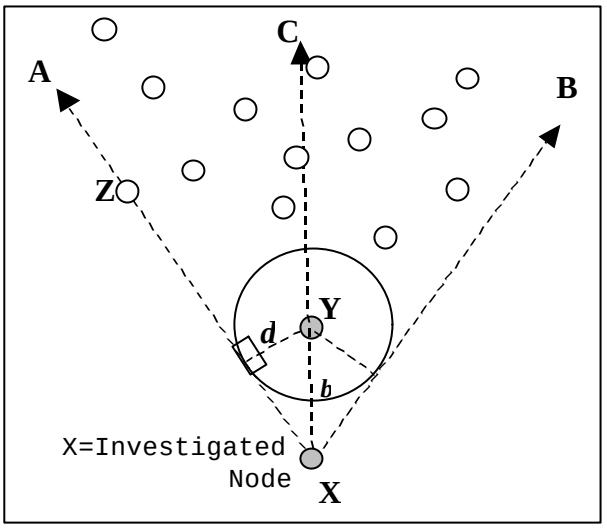


Figure 2.
All the nodes between rays A and B have the following feature in common: going from each node to X through Y adds less then a distance ***2d*** when compared with going straight.
Where ray A is an extension of line X-Z, and ray B is attained by extending the tangent to a circle with a center Y and a radius ***d***.

A ray is a line that emanates from a node and continues in its direction to infinity. Figure 2 depicts a ray A that extends the line from X to Z, a ray C that extends the line from X to Y, and a ray B. Using the same ***d*** we found for Z, and replacing Z by any node I between rays A and C, the rule: $(b_i+c_i)-a_i < 2d$ exists for all the nodes i between rays A and C (on ray C it becomes an equality). Utilizing symmetry, the same rule will be extended to the right hand side of node Y. For that purpose, we draw a circle with a radius ***d*** and a center point Y. Then, a tangent to that circle from node X is computed, and extended as ray B. As shown in figure 2, the upper bound ***2d***, for the added distance to/from X through Y, is not only the upper bound for Z but for all nodes between rays A and B. The original network is assumed to be a mesh. That is, with N nodes and N(N-1) arcs connecting each node to all other nodes. The following lemma is essential in controlling the deviation from the optimal rout.

Lemma 1: Assume the network is a mesh (N*(N-1) arcs). If we keep arc X-Y but cross out all other arcs emanating from X to the nodes between the two rays, A and B, the maximal deviation from an optimal route, that visits each node once, cannot exceed ***4d***.
Proof: For each node in such an optimal route, there must be one inbound and one outbound arc. By erasing the arcs as in lemma 1, the optimal route could be effected only if it contain, for X ,either an inbound arc, or an outbound arc that is in the area between rays A and B. For any node in that area (between A and B) reaching X through Y may cost up to ***2d***. The same is true for reaching the nodes between rays A and B from X: reaching them through Y may cost up to ***2d***. Thus, the worst case deviation from the optimal rout is ***2d*** on the approach, and ***2d*** on the departure. This gives ***2d+2d=4d.*** **Q.E.D.**

4. The proposed approach

The proposed approach allows the user to specify the acceptable deviation from an optimal route (for example, an optimal route of a TSP) that visits each node only once. The deviation is spread equally among all nodes. This means that if the specified deviation is *E*, and there are *N* nodes, approaching each node can deviate at most *E/N* from optimality.

We can now substitute the worst-case deviation from optimality of ***2d*** (see section 3) in approaching each node, by *E*/*N*. We start by choosing a node X arbitrarily, and its nearest neighbor Y. We proceed by setting the rays A_{xy} and B_{xy} for node X in a way that ensures maximal deviation from optimality of *E*/*N* as discussed in section 3 above. After eliminating the arcs from X to the nodes between A_{xy} and B_{xy}, we choose the next nearest node from X that is not within rays A_{xy} and B_{xy} and repeat the procedure of building two new rays. We can also update the rays A_{xy} and B_{xy} and extend section A_{xy}-B_{xy} further as mentioned in the arc elimination procedure below and discussed in appendix 1. This procedure of choosing a nearest node that is not between any pair of rays and building two rays around it (and updating the existing rays) continues until all the nodes are properly contained between pairs of rays. Within each pair of rays, all arcs to node X are eliminated, except the arc from Y - the nearest node from X. Of course, we shall repeat this procedure eventually for all *N* nodes in the network, as presented in the "arc elimination procedure" below.

The arc elimination procedure is:

Set i=1, j=1.

1. Initialize: arbitrarily choose a node X_j (that had not been chosed already)and initiate a list L of all N nodes in the network except of X_j.
2. Find nearest remaining node to X_j from the list L and name it Y_i. Compute ***r*** = The distance from X_j to Y_i.
3. Draw a circle around the closest node (Y_i) with a radius $d_i=(E/N)/2$ or simply $d_i=E/2N$. If i=1 then go to 6.
4. Update the radiuses d_1 *to* d_{i-1} for all the previously chosen Y_i's ($Y_1,Y_2,...,Y_{i-1}$) by computing:
   b_j = the distance from X_i to Y_i.
   From step 2, ***r*** = The distance from X to Y_i
   Updated $d_j = b_j * \sin[\arccos(1-(E/N)^2/2b_jr+(E/N)/r-(E/N)/b_j)]$ (for all j=1,2..,i-1 see appendix 1.)
   For ***r*** within the range of: $\{ [(E/N)^2/2b-(E/N)] / (1-E/(Nb)) > r > b \}$ (see appendix 1.)
5. Rebuild all pairs of rays [update (A_1 and B_1),...,(A_i and B_i)] from X to the circumference of the new circles (with centers at Y_1 to Y_i and new radiuses d_1 to d_i), and extend them as pairs of infinite rays.
6. Eliminate the nodes between all the pairs of rays (A_i and B_i) from the list L.
7. If L is not empty, then i:=i+1, go to step 2. Else go to step 8.
8. Except the arcs from X_j to Y_k (for k =1,..,i), eliminate all arcs emanating from X_j to the nodes between the pairs of rays (A_1 and B_1),(A_2 and B_2),...,(A_i and B_i).
9. If j=N (all nodes were chosen as X_j) then end; Else j:=j+1 and go to step 1.

5. Conclusion

We presented here an effieicient technique for arc elimination in mesh networks. The arc elimination technique can eliminate most of the arcs in a large network and it leaves a reduced network as its product. The reduced network can still produce routes that are close to optimal. Dealing with a reduced number of arcs in a network has advantage for many routing techniques: branch and bound techniques, search techniques (e.g., genetic algorithms and simulated annealing), insertion techniques, and k-opt techniques.

References

[1] S. Lin and B. W. Karnighan, "An effective Heuristic Algorithm for the Traveling Salesman Problem", Operations Research, 21, 498-516, 1973.

[2] M. W. Padberg, & G. Rinaldi, "Optimization of a 532 city symmetric Traveling Salesman Problem by branch and cut", Operations Research Letters, 6, 1-7, 1987.

[3] Krasnogor, P. Moscato, and M.G. Norman, "A new hybrid heuristic for large geometric Traveling Salesman Problem based on Delaunay triangulation",Anales del XXVII Simposio Brasileiro de Pesquisa Operacional, Vitoria, Brazil, 6-8, Nov. 1995.

[4] D. Naddef, & G. Rinaldi, "The graphical relaxation: A new framework for the symmetric Travelling Salesman polytope", Math. Programming, 58, 53-88, 1992.

[5] C.S. Mata and J. Mitchell, "Aproximation algorithms for Geometric tour and network problems", Proceedings of the 11th ACM Symposium on Computational Geometry, 360-369, 1995.

[6] Y. Cohen, "Arc elimination in a Euclidian mesh for complexity reduction of network routing problems", Working paper 2002-6-1, The Open University of Israel, Management and Economics Dept, 2002.

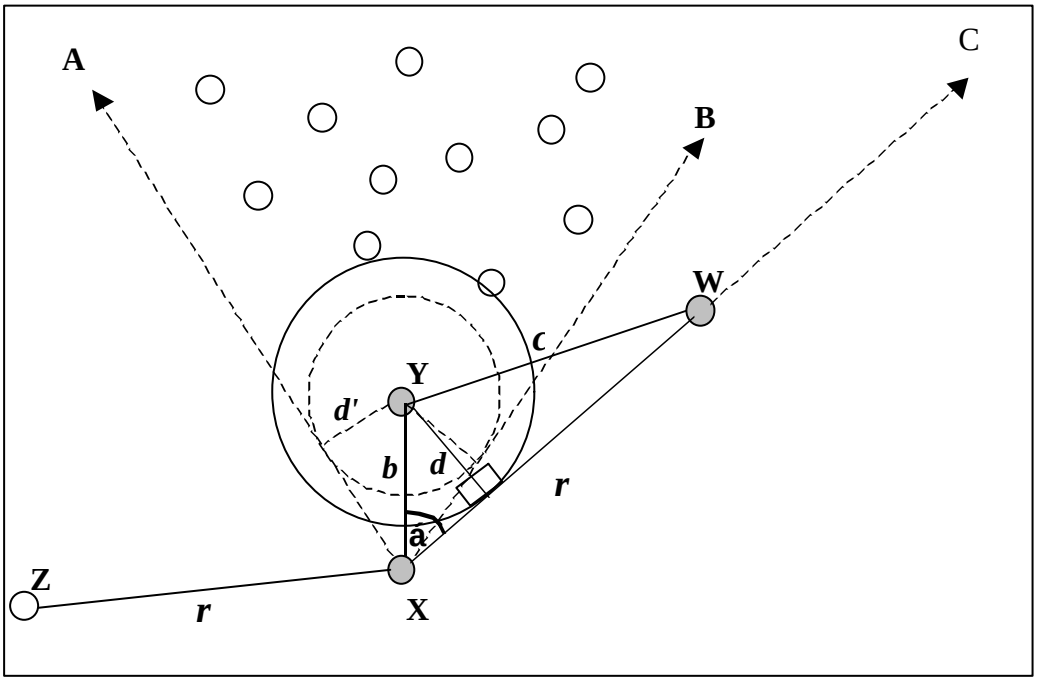
Appendix -1 : Updating the radius for computing a new pair of rays

In this appendix we derive the formula for the radius d_i used in the elimination procedure of section 1.2. Assume Z is the newest, closest node that was found out of section A-B. By finding the next closest node to X (ignoring nodes in section A-B) we gather more information. We know that there is no node in the vicinity of A-B that is closer to X than the the new closest node Z. We call this new distance ***r.*** This means that within a radius ***r*** of X there are no other nodes (except in section A-B). This information enables us to extend section A-B further. We utilze an imaginative node "W" that is located a distance ***r*** from node X and has a path Z-Y-X with length of ***r***+E/N (from which the distance W-Y could be computed as r+E/N-b).

If we extend the line X-W in the figure below, all points on ray C (emenating from W continuing the direction of X-W) and between rays C and B, have smaller deviations (see proof in [6]). For the sake of simplification, the second iteration is studied and the formulas do not have indexes. In the algorithm, however, most variables carry indexes, and care must be exercized in applying them.

Define:

E - Acceptable deviation (in distance) from the distance of the optimal route
N - Number of nodes in the network
X - An arbitrarily chosen node
Y - The closest node to X.
Z - The closest node to X after the node elimination step (see step 6 in section 1.2)
r - The distance from X to Z
W - An imaginery node built at distance r from X at the vicinity of a ray.
b - The distance from X to Y
c - The distance from Y to
d' - The old radius
d - The new radius



By definition ***E, N***, and ***b*** are given for a given network. For a certain iteration in the elimination procedure of section 1.2 ***r*** is also known. So we need to update the rays (A_1 and B_1) to (A_{i-1} and B_{i-1}).

Consider the second iteration (for simplicity). We have two relationships (depicted in the figure below).

1) $E/N= (b+c) - r$ (***r*** = distance W-X. Going X-Y-W consumes **all** the available deviation =E/N).

So: $c= E/N+r-b$

$c^2=(E/N)^2+2r(E/N)-2b(E/N)+r^2-2rb+b^2$

2) $c^2=[b*\sin(á)]^2+[r-b*\cos(á)]^2$ (Pithagorean calculations).

$c^2= b^2-2rb*\cos(á)+r^2$

Equating 1 and 2 we have:

$(E/N)^2+2r(E/N)-2b(E/N)+r^2-2rb+b^2 = b^2-2rb*cos(á)+r^2$

Reducing both side by b^2 and dividing both sides by $-2rb$ **we have:**

$Cos\ (á) = 1-(E/N)^2/2br +(E/N)/r-(E/N)/b$

➔ $á = ArcCos[1-(E/N)^2/2br+(E/N)/r-(E/N)/b]$

And from trigonometry: $d = b* Sin\ (á)$

Note that for a given N and E, $á$ is a function of r and b_i. Moreover, if [(E/2N)<b_i] as r increases $Cos\ (á)$ decreases and $á$ increases. If [(E/2N)>b_i] as r increases $Cos\ (á)$ increases and $á$ decreases.
Conclusion:

The new radius $d = b* Sin\ (á) = b*$ **Sin** $[ArcCos\ (1-(E/N)^2/2br +(E/N)/r-(E/N)/b)]$

In this formula, r must be in the range of: $\{[(E/N)^2/2b-(E/N)] / (1-E/(Nb)) > r > b\}$

Proof (for the range of *R*):
1. b is the distance from X to Y and is shorter than r which is the distance from X to Z.
2. The maximal radius d for finding tangents for rays is b. When $d=b$ the arcs in the entire hemisphere (180^0) could be eliminated.

In that case: $d = b$ ➔ $1=Sin\ [ArcCos\ (1-(E/N)^2/2br +(E/N)/r-(E/N)/b)]$
➔ $ð/2 = ArcCos\ (1-(E/N)^2/2br +(E/N)/r-(E/N)/b)$ ➔ $1-(E/N)^2/2br +(E/N)/r-(E/N)/b) = 0$
➔ $r= (E/N)^2/2b-(E/N) +(rE/N)/b)$ ➔ $r-(rE/N)/b) =(E/N)^2/2b-(E/N)$
➔ $r*(1-E/(Nb)) = (E/N)^2/2b-(E/N)$
(r = [(E/N)2/2b-(E/N)] / (1-E/(Nb))

Appendix 2: Proof that $c-a1<d$

Since c is the hypoteneous of a right angle triangle we know that $c> a_1$ and therefore:

1. $c-a_1>0$

Thus, the following inequalities (that need a proof) are equivalent:

2. $c-a_1<d \rightarrow (c-a_1)^2<d^2 \rightarrow c^2-2ca_1+a_1^2<d^2$

Since all the inequalities in 2 are equivalent, proving one of them is enough.

From Pythagoras we know that

3. $c^2-(a_1)^2 = d^2$

Thus: $d^2 = \underbrace{c^2-(a_1)^2}_{d^2} = \underbrace{c^2-2ca_1+a_1^2}_{(c-a_1)^2} \underbrace{-2a_1^2 +2ca_1}_{2a_1(c- a_1)}$

From 1. we know that $c-a_1>0$ and therefore: $2a_1(c- a_1)>0$

Thus: $d^2=(c-a_1)^2$ + positive element
Taking the positive element out ($2a_1(c- a_1)$) we get:

$d^2>(c-a_1)^2$

Taking the square root:
We get: $d>(c-a_1)$
Q.E.D.