Vertex Penalty Algorithm for the Traveling Salesman Problem

author
This Hypercard stack was prepared by:
Dennis L. Bricker,
Dept. of Industrial Engineering,
University of Iowa,
Iowa City, Iowa 52242
e-mail: dennis-bricker@uiowa.edu

Symmetric TSP

undirected network (N,A)

$$\text{Minimize} \sum_{(i,j)\in A} C_{ij}X_{ij}$$

$$\text{subject to} \sum_{(i,j)\in A} X_{ij} = 2 \quad \forall\, i\in N$$

$$X \in \mathcal{T}_1$$ = set of all spanning 1-trees of network

Lagrangian Relaxation

$$\text{Minimize} \sum_{(i,j)\in A} C_{ij}X_{ij} + \sum_{i\in N} \mathbf{u}_i \left(\sum_{(i,j)\in A} X_{ij} - 2 \right)$$

$$\text{subject to} \quad X \in \mathcal{T}_1$$

A multiplier is associated with each degree constraint, and the degree constraint is relaxed, with the objective "penalized" by violations....

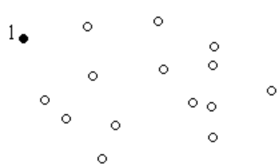
Lagrangian Relaxation

$$\Phi(\mathbf{u}) = \underset{X\in \mathcal{T}_1}{\text{Minimum}} \sum_{(i,j)\in A} \left(C_{ij} + \mathbf{u}_i + \mathbf{u}_j\right)X_{ij} - 2\sum_{i\in N} \mathbf{u}_i$$

For fixed u, this is a "minimum spanning 1-tree" problem!

Finding the Minimum Spanning 1-Tree

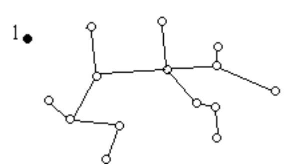
Select an arbitrary city, e.g., city #1



Further restrict the set of 1-trees to include only those for which node 1 has degree 2 and lies on a cycle!

Finding the Minimum Spanning 1-Tree

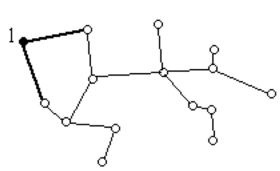
Select an arbitrary city, e.g., city #1



Find the minimum spanning tree of the set of cities excluding the selected city.

Finding the Minimum Spanning 1-Tree

Select an arbitrary city, e.g., city #1



Find the minimum spanning tree of the set of cities excluding the selected city.

Find the two nearest neighbors of the selected city, and add the two corresponding edges.

Lagrangian Dual

For each choice of vector u, the value of the Lagrangian relaxation $\Phi(\mathbf{u})$ provides us with a ***lower bound*** on the optimum TSP tour.

The Lagrangian Dual problem is to**...**

$$\underset{\mathbf{u}}{\text{Maximize}} \quad \Phi(\mathbf{u})$$

$$\Phi(u) = -2\sum_{i\in N} u_i + \underset{X\in \mathcal{T}_I}{\text{Minimum}} \sum_{(i,j)\in A} \left(C_{ij} + u_i + u_j\right)X_{ij}$$

For the purpose of finding the minimum spanning 1-tree, the length of edge (i,j) is

$$C_{ij} + u_i + u_j$$

i.e., the edge length is increased by the "penalties" of the 2 end vertices.

Suppose that we were to enumerate the (finitely many) 1-trees of the network:

$$\hat{X}^k \in \mathcal{T}_I \text{ , } k=1,2,\ldots K$$

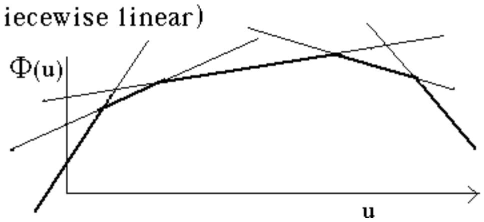
Then

$$\Phi(u) = \underset{1\le k\le K}{\text{Minimum}} \sum_{(i,j)\in A} \left(C_{ij} + u_i + u_j\right)\hat{X}^k_{ij} - 2\sum_{i\in N} u_i$$

i.e., Φ is the lower envelope of a finite set of linear functions of u (& is therefore concave piecewise linear)

$$\Phi(u) = \underset{1\le k\le K}{\text{Minimum}} \sum_{(i,j)\in A} \left(C_{ij} + u_i + u_j\right)\hat{X}^k_{ij} - 2\sum_{i\in N} u_i$$

i.e., Φ is the lower envelope of a finite set of linear functions of u (& is therefore concave piecewise linear)



If $\hat{X}^k$ is optimal in the evaluation of ,
then
$$\Phi(u) = \sum_{(i,j)\in A} \left(C_{ij} + u_i + u_j\right)\hat{X}^k_{ij} - 2\sum_{i\in N} u_i$$

and the vector γ

with
$$\gamma_i = \sum_{(i,j)\in A} \hat{X}^k_{ij} - 2$$

is a subgradient of Φ at u.

To adjust u, then, step in the direction γ.

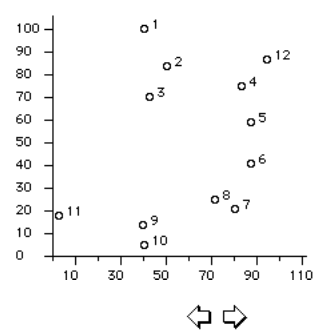
subgradient

$$\gamma_i = \sum_{(i,j)\in A} \hat{X}^k_{ij} - 2$$

That is, if the degree of node i exceeds 2 in the current minimum spanning 1-tree, then the "penalty" u_i should be increased, to discourage selection of edges incident to vertex i, while if the degree is less than 2 (i.e., 1), the "penalty" is decreased, to encourage the selection of another edge incident to vertex i.

Example

Random Symmetric TSP
(seed= 133398)

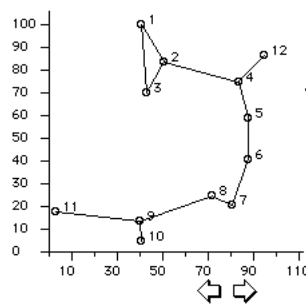


Finding Minimum Spanning 1-tree

- select an arbitrary node k
- find minimum spanning tree of the network with node k deleted
- add to the minimum spanning tree the 2 shortest edges incident to node

Iteration 1

Lower Bound: 260
Sum of excess degrees: 3



subgradient direction:

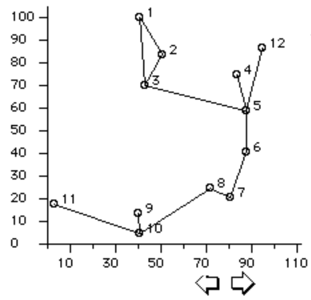
γ_i = +1 for i=2,4,9

γ_i = -1 for i=10,11,12

γ_i = 0 otherwise

Vertex penalty algorithm

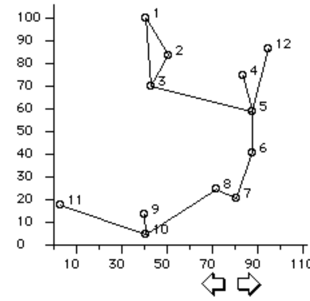
New minimum spanning 1-tree:



Note that the degrees of nodes 2, 4, and 9 were decreased because of the penalties...

Vertex penalty algorithm

Iteration 2 Lower Bound: 258.4666667
Sum of excess degrees: 4



subgradient direction:

γ_i = +2 for i = 5

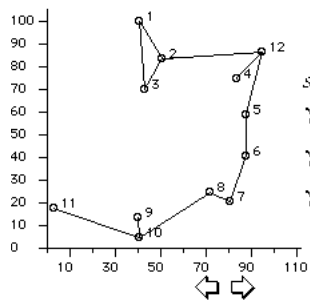
γ_i = +1 for i = 3,10

γ_i = -1 for i = 4,9,11,12

γ_i = 0 otherwise

Vertex penalty algorithm

Iteration 3 Lower Bound: 281.0711111
Sum of excess degrees: 3



subgradient direction:

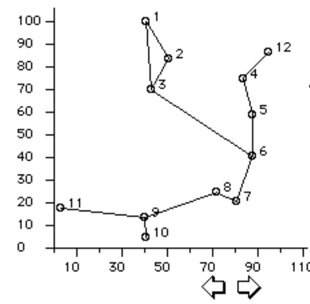
γ_i = +1 for i = 2,10,12

γ_i = -1 for i = 4,9,11

γ_i = 0 otherwise

Vertex penalty algorithm

Iteration 4 Lower Bound: 274.9712593
Sum of excess degrees: 3



subgradient direction:

γ_i = +1 for i = 3,6,9

γ_i = -1 for i = 10,11,12

γ_i = 0 otherwise

Vertex penalty algorithm

Iteration	Lower Bound	Excess Degree	at Nodes
1	260	3	2 4 9
2	258	4	3 5 10
3	281	3	2 10 12
4	275	3	3 6 9
5	286	3	3 8 10
6	292	4	2 4 7 9
7	302	1	2
8	307	2	3 5
9	313	2	3 4
10	313	2	8 12
11	316	3	3 7 11
12	316	3	2 10 12
13	316	2	3 4
14	318	1	6
15	318	1	8
16	317	3	2 7 12
17	318	3	3 7 11
18	319	2	4 10
19	320	2	2 12
20	319	2	3 5

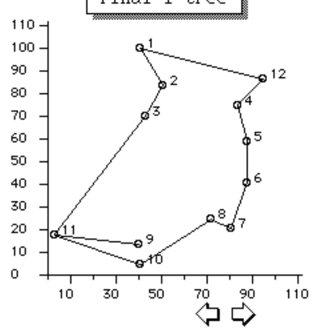
Vertex penalty algorithm

Iteration	Lower Bound	Excess Degree	at Nodes
21	320	1	11
22	319	3	2 10 12
23	320	3	3 6 10
24	320	2	8 10
25	320	1	5
26	320	1	4
27	321	1	8
28	321	3	2 7 12
29	321	1	6
30	321	1	11

Vertex penalty algorithm

***Failed to converge.
Greatest Lower Bound on tour length is 320.7602064

Final 1-tree



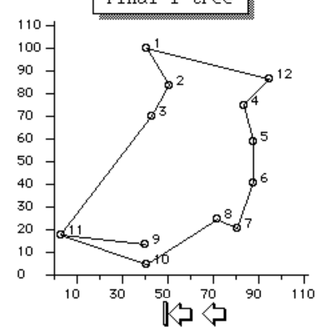
Final Penalties

i	P[i]
1	0.000
2	51.303
3	43.367
4	28.301
5	23.290
6	15.350
7	9.373
8	15.341
9	34.413
10	31.403
11	3.365
12	17.365

Vertex penalty algorithm

***Failed to converge.
Greatest Lower Bound on tour length is 320.7602064

Final 1-tree



The optimal tour length is 321, so the lower bound is quite "tight"!

Vertex penalty algorithm