A Neural Network Solution to the Generalized Orienteering Problem

by

Bruce Golden, University of Maryland
Qiwen Wang, Peking University
Xiaoyun Sun, Manugistics

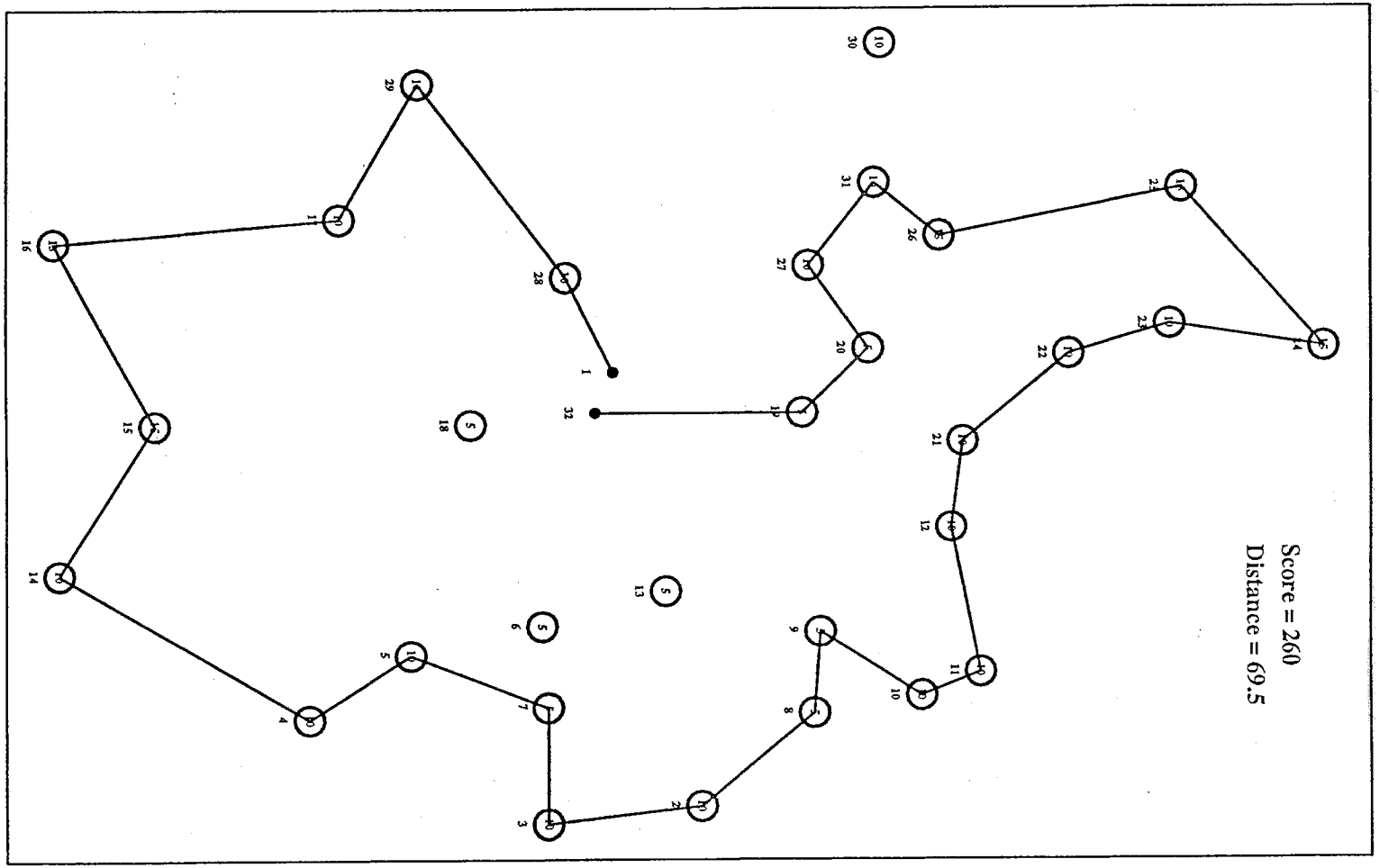
INFORMS
Dallas, October 1997

General Background

- Orienteering is a sport in which start and end nodes are specified along with other locations. These other locations have associated scores. Competitors seek to visit, in a fixed amount of time, a subset of these locations on the way from the start point to the end point in order to maximize total score.

- The sport is popular in Scandinavia and in the U.S. Military.

- In the OR literature, the orienteering problem (OP) denotes a class of NP-hard routing problems related to the TSP.


Score = 260
Distance = 69.5

- Key Question: Can a neural network approach compete with traditional OR heuristics developed over the past decade?

- Answer: Yes, see Wang et al., *Annals of Operations Research* (1995).

- In this paper, we define the (more difficult) generalized orienteering problem (GOP) and solve the GOP using neural networks

Definition of the GOP

- Start location is point 1
- End location is point N
- m independent goals
- Point i has score vector

$$S(i) = (S_1(i), S_2(i), \ldots, S_m(i))^T$$

- $S_g(i)$ is the score of point i w. r. t. goal g
- Examples of goals
  - natural beauty
  - historical significance
  - cultural and educational attractions
  - business opportunities

- The objective function is nonlinear and not continuously differentiable

$$Z = \sum_{g=1}^{m} W_g \left\{ \max_{i \in P} \left(S_g(i) \right) \right\}$$

- We approximate Z using Z′

$$Z' = \sum_{g=1}^{m} W_g \left[\left\{ \sum_{i \in P} \left[S_g(i) \right]^k \right\}^{1/k} \right]$$

- As k grows large, Z′ approaches Z
- The OP is a special case of the GOP (let k = m = 1)
- Next, we present a neural network model for solving the GOP.

Neural Network Representation

1. Conceptually, we have a N by N matrix of neurons or nodes.

2. Each row corresponds to a particular point.

3. Each column corresponds to a particular position in the path from start to end.

4. V(i, j) is the state or activation level of the i, j node. At completion, V(i, j) is either 0 or 1.

5. For $j \leq N$, V(i, j) = 1 means point i is the j^{th} point visited in the path. We set V(1, 1) = V(N, N) = 1.

Input and Output in the Neural Network

- Input at neuron (i, j) = U(i, j)
- Output at neuron (i, j) = V(i, j) = $\dfrac{1}{1+e^{-U(i,j)}}$

The Energy Function

- The standard form of the Hopfield network includes a quadratic energy function and constant link weights.
- The energy function that we propose is a fourth order convex function. The weights are not constant, as they depend on the current state of the neural network.
- The energy function, denoted by E, is shown next. We seek to minimize E.

$$E = \frac{a}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{h\neq i}^{N} V(i,j)V(h,j) + \frac{b}{2}\left[\sum_{i=1}^{N}\sum_{j=1}^{N} V(i,j) - N\right]^2 + c\left[2 - V(1,1) - V(N,N)\right]$$

$$+ \frac{d}{2}H\left(\sum_{i=1}^{N}\sum_{h=1}^{N}\sum_{j=1}^{N-1} d(i,h)V(i,j)V(h,j+1) - D\right) - e\sum_{g=1}^{m} W_g\left\{\sum_{i=1}^{N} Q\left(\sum_{j=1}^{N} V(i,j)\right)\left[S_g(i)\right]^k\right\}^{1/k}$$

where

$$H(x) = \begin{cases} x^2 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$Q(x) = \begin{cases} x & \text{if } x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$

d(i, h) = travel cost from i to h

D = travel cost (time) limit

Logic of Energy Function

Term	Motivation
a	Penalize a column with more than one activated node
b	Ensure exactly N activated nodes—one per column
c	Start at point 1 and end at point N
d	Seek to ensure feasibility
e	Seek to maximize score

- Now, set c to zero and compute $\frac{\partial E}{\partial V(i, j)}$

Discrete-Time Dynamics

- We use asynchronous updating via the formula

$$\Delta U(i,j) = \frac{-\partial E}{\partial U(i,j)}\Delta t = \frac{-\partial E}{\partial V(i,j)} \times \frac{\partial V(i,j)}{\partial U(i,j)}\Delta t$$

$$= \frac{-\partial E}{\partial V(i,j)} \times V(i,j) \times (1 - V(i,j))\Delta t$$

- From the chain rule, $\frac{\partial E}{\partial t} = \frac{\partial E}{\partial V}\frac{\partial V}{\partial U}\frac{\partial U}{\partial t} < 0$. Energy decreases over time.

State Updating Algorithm

Step 0. Set parameter values for N1, N2, a, b, d, e, Δt, and δ. Let $r = 0$.

Step 1. Randomly initialize the state matrix to very small positive values. Let $r = r + 1$ and $i = 1$.

Step 2. Randomly choose one row or column and calculate $\Delta U(i, j)$ for each neuron in the row or column. Choose the neuron with the largest positive value and the neuron with the most negative value of $\Delta U(i, j)$. If they exist and exceed δ in absolute value, then update their states. Repeat this process until the state matrix reaches stability, i.e., until all $|\Delta U(i, j)| < \delta$.

Step 3. Select the largest state in each column. Set this state to 1 and set all other states in the column to 0. Construct the path corresponding to the state matrix. Apply the two-opt procedure to improve the path.

Step 4. If the length of the path does not exceed D, compare it with the best solution obtained so far. Store the best solution. If $i = N2$, go to Step 6.

Step 5. If the length of the path exceeds D, the path is infeasible. Increase d and decrease e. If the path is feasible, increase e and decreased d. Perturb the state matrix slightly. Let $i = i + 1$. Go to Step 2.

Step 6. If $r \leq N1$, go to Step 1. Otherwise, stop and display the best result.

Limited Computational Experience

Key Example

- **27 Chinese cities problem**
- **4 goals at each city**
- **Scores are scaled from 1 to 10**
- **Program runs on a PC 486/50 and requires approximately 3 minutes**
- **6 variants of the problem were solved**
- **The results look quite reasonable**

Running Time

- **We generated 15 problems, N ranged from 27 to 90**
- **Linear regression reveals that running time grows with N^2**

Conclusions and Future Work

- As a mathematical program, the GOP is a nonlinear, multi-objective integer program. This is an extremely hard class of problems to solve.
- Nonetheless, the modified, continuous Hopfield neural network approach (coupled with 2-opt) seems to work well.
- Can other nonlinear IP problems be solved similarly?
- How would more traditional heuristics perform on the GOP?
- Can we model the score as time-dependent?