
Using Energy-Minimizing Networks to Solve Constraint-Satisfaction Problems

Constraint-Satisfaction Problems

- **Minimum-energy seeking networks** (e.g. using annealing) can be used to find solutions to constraint-satisfaction problems (problems of finding optima, subject to constraints that can't be violated, rather than as computing a function).
- A number of other such problems have been studied in this context.

Traveling Salesperson Problem

- The problem is: given a set of n nodes (“cities”) with a specified minimum cost between each pair of nodes, find a permutation (“tour”) of the nodes that minimizes the summed costs between the nodes in the permutation sequence.
- The costs are symmetric, and the general problem does not require that there be any Euclidean relationship among the nodes.

Finding Solutions to the TSP using a Hopfield Net

- Global minimum is not necessarily found, although this might be doable with a Boltzmann style algorithm instead.
- The difficulty is encoding the instance of the TSP as a net:

minimal cost \square minimal energy

TSP Formulation

- Represent a given problem as a matrix:

- Cities correspond to rows.
- Positions on the tour correspond to columns.

● Example:

	1	2	3	4	5
A	0	0	1	0	0
B	1	0	0	0	0
C	0	0	0	0	1
D	0	1	0	0	0
E	0	0	0	1	0

means B occurs first on the tour, D occurs second, A third, E fourth, C fifth.

TSP Formulation

- Assume $\{0, 1\}$ values rather than $\{-1, 1\}$.
- The neurons correspond to entries in the matrix (n^2 neurons for n cities).
- Neurons in a row have inhibitory connections from other neurons in same row:
 - If one neuron is on, then others tend to be off, especially in minimum energy state.
- Similarly for neurons in the same column

TSP Formulation

- Need to favor tours that include *all* n cities, as opposed to just a subset of them.
- Need to represent costs between cities as neural weights:
 - Want to inhibit selection of adjacent cities in proportion to the cost between those cities.
 - Let X and Y be rows (cities) and i and j be columns (positions).

TSP Formulation

- Using the expression for energy in a Hopfield net $E = -\sum_{ij} w_{ij} y_i y_j$, the corresponding energy is computed to have the form

$$\begin{aligned}
 & A \sum_X \sum_i \sum_{j \neq i} y_{Xi} y_{Xj} \\
 & + B \sum_i \sum_X \sum_{Y \neq X} y_{Xi} y_{Yi} \\
 & + C \left(\sum_X \sum_i y_{Xi} - n \right)^2 \\
 & + D \sum_i \sum_X \sum_{Y \neq X} c_{XY} y_{Xi} (y_{Y,i+1} + y_{Y,i-1})
 \end{aligned}$$

- At energy minimum, only the last term, which represents the tour cost, is non-zero.
- We'll explain these terms one at a time.

$$A = \sum_X \sum_{i \neq j} y_{Xi} y_{Xj}$$

- The outer summation is over all cities X. The inner summations are over all pairs of distinct positions.
- There is a contribution of +1 if the same city occurs in more than one position in the tour.
- Therefore this term should ideally be 0.

$$B = \sum_i \sum_X \sum_{Y \neq X} y_{Xi} y_{Yi}$$

- The outer summation is over all positions in the tour. The inner summations are over all pairs of distinct cities in position i.
- There is a contribution of +1 if the same city occurs in more than one position in the tour.
- Therefore this term should ideally be 0.

$$C \left(\sum_x \sum_i y_{xi} - n \right)^2$$

- This term tries guarantees that all cities get used. If the summation is n , the term is 0. If it is less than n , the term will be positive.
- This term should ideally be 0.

$$D \sum_i \sum_x \sum_{Y \neq X} c_{XY} y_{xi} (y_{Y,i+1} + y_{Y,i-1})$$

- This term represents the cost of the tour. The outer sum is over all positions in the tour, the inner sums over all distinct pairs.
- c_{XY} represents the cost of going from X to Y . This term gets added provided X is at the i^{th} position in the tour, represented by $y_{xi} = 1$, **and** Y is either at the $(i+1)^{\text{th}}$ or $(i-1)^{\text{th}}$ position (it can't be at both, by the other constraints). $(i+1)$ and $(i-1)$ are computed mod n .

Weight Derivation

- In order to get the energy function to come out as specified, choose the weight from Xi to Yj as

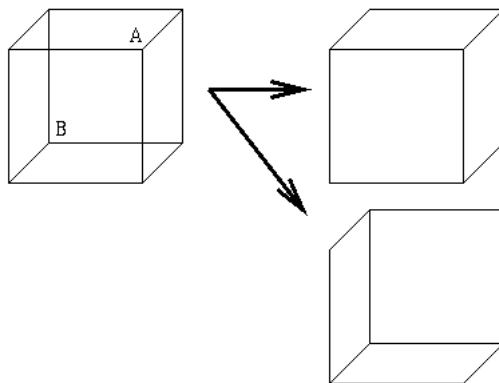
$$w_{XiYj} = -A\delta_{XY}(1 - \delta_j) - B\delta_j(1 - \delta_{XY}) - C - Dc_{XY}(\delta_{j,i+1} + \delta_{j,i-1})$$

for appropriate constants A, B, C, D.

- $\delta_{j,i}$ is the Kronecker delta (1 if $i = j$, 0 otherwise)

Necker Cube “Boltzmann” Demo

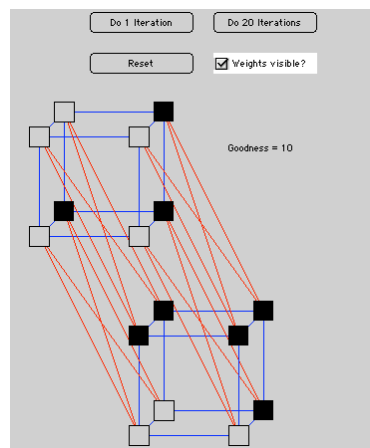
(<http://www.cs.cf.ac.uk/Dave/JAVA/boltzman/Necker.html>)



Constraints Represented

- Each vertex can have only one (of two possible) interpretation. Therefore there are negative weights connecting units representing different interpretations of the same vertex.
- The same interpretation cannot be given to more than one vertex - so units representing the same interpretation are connected with negative weights.
- Units from the same interpretation should be on together, so locally consistent units are connected with positive weights.

Necker Cube Demo



Neurons representing whether vertex is shown in **front** or **back**

Blue weights are positive, red are negative.