Combinatorial optimization using Hopfield Net - Intuitions

1 Sorting

1.1 Equations

- $E_1 = \frac{A}{2}\left[\sum_i(\sum_\alpha x_{i,\alpha} - 1)^2 + \sum_\alpha(\sum_i x_{i,\alpha} - 1)^2\right]$
  Same as TSP

- $E_2 = B\sum_i\sum_\alpha x_{i\alpha} * S_i * \alpha$ (where S_i is the value at ith position in the input array)
  Summation over "$value * position$" should be minimized (this will give reverse sorting)

1.2 Weights and Thresholds

- $w_{ij} = -A$
  Weight values for the neurons in the same row/column. If some value S_i is set into some position j, S_i can not go to any other position and no other value can be set in position j, so inhibit corresponding neurons. Other weight values will be 0

- $\theta = 2A - S_iB\alpha$
  This depends on values of A and B. If the value S_i has already been set to some other place or if some other value has been set to position α, look for the product $S_i\alpha$ and depending upon the values of A and B, make some decision.

Following problems are from Graph Theory. While solving them, we add the threshold value to find the net input of a neuron. We subtract the term with threshold in the engergy equation.
For all problems, we are assuming values of A and B to be 1

2 Vertex Cover

2.1 Equations

- $E_1 = \sum_i v_i$
  Minimize the no. of vertices in the solution.

- $E_2 = \sum_i \sum_j d_{ij} \overline{v_i \vee v_j}$
  Cover each edge.
  If $d_{ij} = 1$ i.e. v_i and v_j are adjacent, then at least one of them must be there in the solution (to cover the edge). If none of them is in the solution, $\overline{v_i \vee v_j}$ will be 1 and the whole term will evaluate to 1, increasing the energy. HN will try to avoid this in order to reach the stable state.

2.2 Weights and Thresholds

- $w_{ij} = -2d_{ij}$
  If some vertex v is in the solution, it reduces the requirement for its neighbours to be in the solution.

- $\theta_{ij} = 2\sum_i d_{ij} - 1$
  If all edges of a particular vertex v has been covered by the neighbours, v is not required in the solution. In this case, $2\sum_i d_{ij}$ will remove the effect of the input from the neighbours and -1 will not allow the neuron to fire. If at least one edge is not being covered i.e. at least one neighbour is at 0 output, total input from neighrous will be less than $2\sum_i d_{ij}$ and the neuron for v will fire.

3 Independent Set

3.1 Equations

- $E_1 = \sum_i \overline{v}_i$
  Maximize the no. of vertices in the solution.
  Minimize the no. of neurons not fired.

- $E_2 = \left[\sum_i \sum_j e_{ij}(v_i \wedge v_j)\right]$
  There should not be an edge between the vertices in the IS.
  If there is an edge and both v_i and v_j are 1 in the solution, this term will add something to the energy. Again, HN will try to avoid this situation

3.2 Weights and Thresholds

- w_{ij}=-e_{ij}
  Every fired neighbour will add -1 to the net input of this neuron.

- θ_i=1
  Don't let this neuron fire if any of the neighbours has fired.

4 Clique

4.1 Equations

- $E_1 = \sum_i \overline{v}_i$
  Maximize the no. of vertices in the solution.
  Minimize the no. of neurons not fired.

- $E_2 = -\left[\sum_i \sum_j (e_{ij} - 1)(v_i \wedge v_j)\right]$
  Confirm that there is an edge between every pair of vertices.
  If there is no edge between v_i and v_j and still both of them are in the solution, this will add something to energy. HN will avoid this.

4.2 Weights and Thresholds

- w_{ij}=-$e_{ij} - 1$
  Every not-fired neighbour will add -1 to the net input of the selected neuron.

- $\theta = 1$
  Let this neuron fire only if the net input is 0 i.e. every neighbour has fired.

5 Matching

Same as Independent Set.
If edges i and j have some common vertex, $d_{ij} = 1$ else $d_{ij} = 0$.
e_i - whether the edge is in the solution or not.