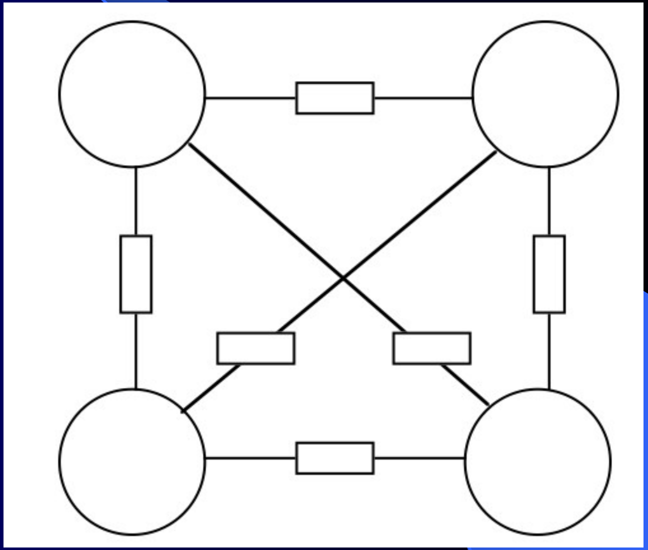
Relaxation and Hopfield Networks

- Totally connected recurrent relaxation networks
- Bidirectional weights (symmetric)
- Inputs/Outputs?
- Auto-Associator

  1001001

  0001101
- Nodes represent features of task
- Weights create an energy landscape
- When let go relaxes to closest energy minima
- Hopfield and *n*-body problems



Hopfield Network

- Basic node activation $V_j = g(H_j) = \frac{1}{2}(1 + \tanh(\sum_{i \neq j}^{n} T_{ij} V_i - I_j))$
  - *g*(*H*) can also be (-1,1) or the threshold function with -1,1 outputs
- A "happy" node

Hopfield Network

- Basic node activation $V_j = g(H_j) = \frac{1}{2}(1 + \tanh(\sum_{i \neq j}^{n} T_{ij} V_i - I_j))$
  - $g(H)$ can also be (-1,1) or the threshold function with -1,1 outputs
- A "happy" node
  - One whose activation and weights correlate with the activations of its neighbors

$$Hap_j = \sum_{i \neq j}^{n} T_{ij} V_i V_j$$

Hopfield Network

- Basic node activation $V_j = g(H_j) = \frac{1}{2}(1+\tanh(\sum_{i \neq j}^{n} T_{ij}V_i - I_j))$
  - $g(H)$ can also be (-1,1) or the threshold function with -1,1 outputs
- A "happy" node
  - One whose activation and weights correlate with the activations of its neighbors

$$Hap_j = \sum_{i \neq j}^{n} T_{ij}V_iV_j$$

- A "happy" network
  - One whose nodes are overall happy

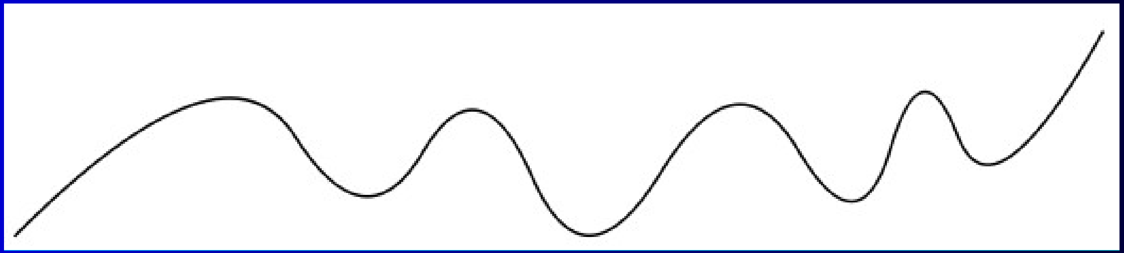
$$Hap_{net} = \sum_{i=1}^{n} Hap_i$$

Network Energy

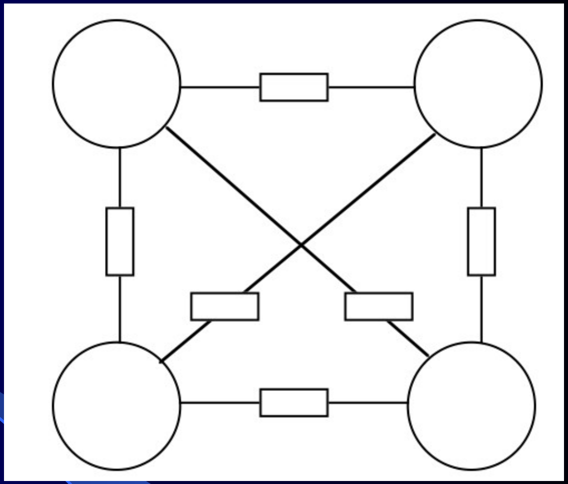
- Energy is just the negative of "happiness"

$$Energy = -\frac{1}{2}\sum_{ij}^{n} T_{ij}V_iV_j - \sum_{j=1}^{n} I_jV_j$$

- Appropriate activation correlations, given the weights, lead to low stable energy states



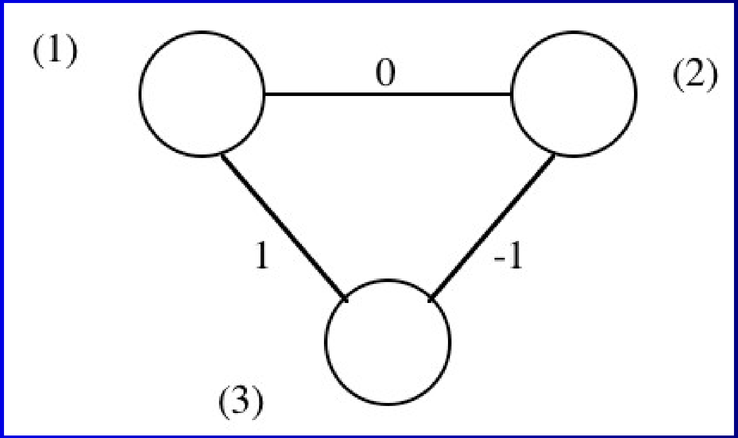
Node Update



- When do nodes update?
- Continuously in the "real" model
  - Slow in software, fast in hardware
- Typically implemented with "random" update
  - One node updates at a time but in random order
  - Synchronous or same order update can lead to oscillating states
- Processing – How do we initiate node activations
  - Random
  - Subset of nodes set to initial state, others random
  - Subset of nodes clamped, others set to random or initial state
- Will relax to nearest stable minima – gradient descent

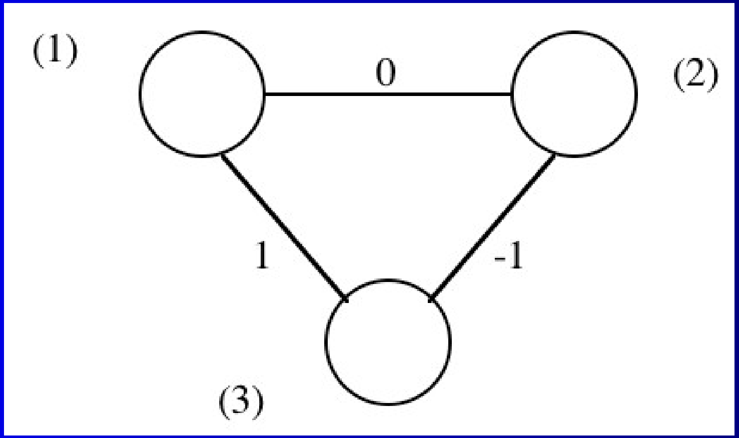
Relaxation

- What are the stable minima for the following Hopfield Network assuming bipolar states? Each unit is a threshold unit which outputs 1 if *net* > 0, else -1



Relaxation

- What are the stable minima for the following Hopfield Network assuming bipolar states? Each unit is a threshold unit which outputs 1 if *net* > 0, else -1



-1 -1 -1 => -1 1 -1 => -1 1 -1

-1 -1 1 =>

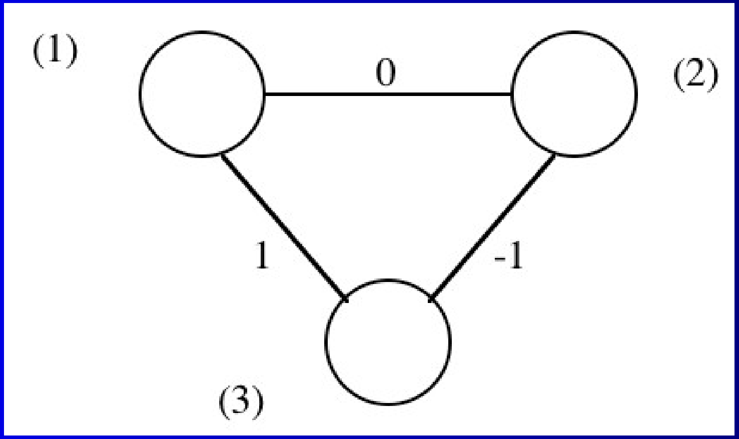
.

.

1 1 1 =>

Relaxation

- What are the stable minima for the following Hopfield Network assuming bipolar states? Each unit is a threshold unit which outputs 1 if *net* > 0, else -1



-1 -1 -1 => -1 1 -1 => -1 1 -1

-1 -1 1 =>

.

.

1 1 1 =>

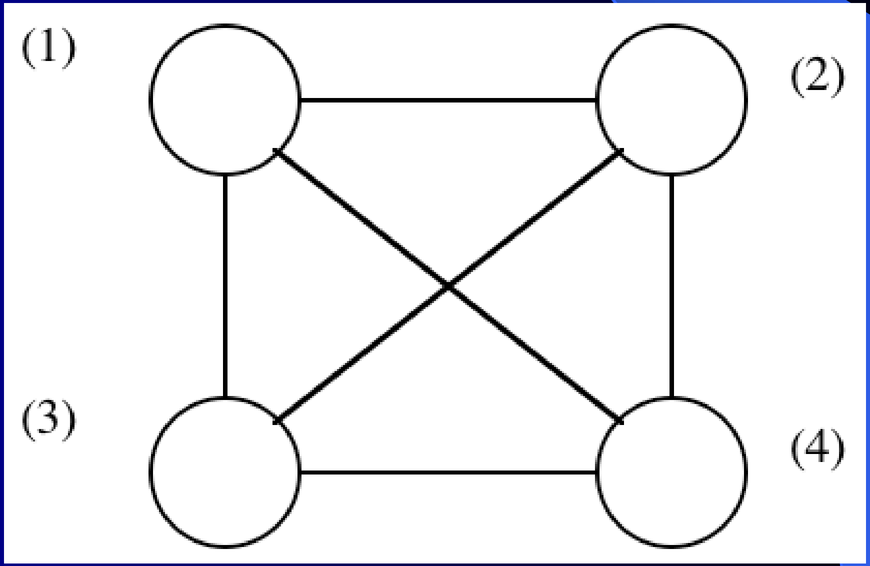
- Energy (-1,1,-1) = 0 - 1 - 1 = -2 (lowest possible for this net)

Hopfield as a CAM

- Hopfield can function as a content addressable memory (CAM) – aka Associative Memory
- For example, could memorize *m* faces, and then when shown a partial/noisy face, it would return the most similar face
- Network has nodes equal to the number of bits in the patterns
- Set the weights according to $$T_{ij} = \sum_{s=1}^{n} (2V_{i,s} - 1)(2V_{j,s} - 1)$$
- This just says that for each pattern, increment each weight between bits which have the same values, and decrement the weight between bits which have different values

Hopfield CAM examples

- What would the weights be set to for an associative memory Hopfield net which is programmed to remember the following patterns? Do you think the net would do well? (i.e. only have stable states are for the learned patterns)
- 1) 1 0 0 1
  0 1 1 0
- 2) 1 0 1 1
  0 1 1 1
  1 1 1 0
  0 0 0 1



Hopfield as an Associative Memory

- Number of stable storable patterns ≈ .15*n* for random bit patterns
- For larger percentage of *n*, many local minima arise
- There are more efficient ways to build CAMs
- Limited by weights just set as first order node correlations
- What would Hopfield weights be for the following?

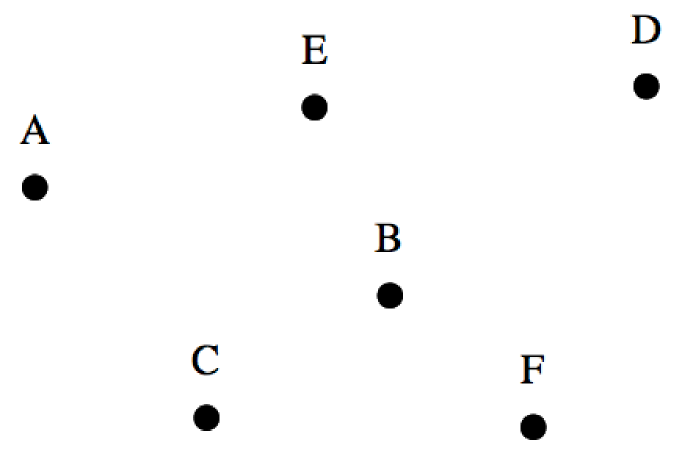
    0 0 1
    0 1 1
    1 0 1
    1 1 0

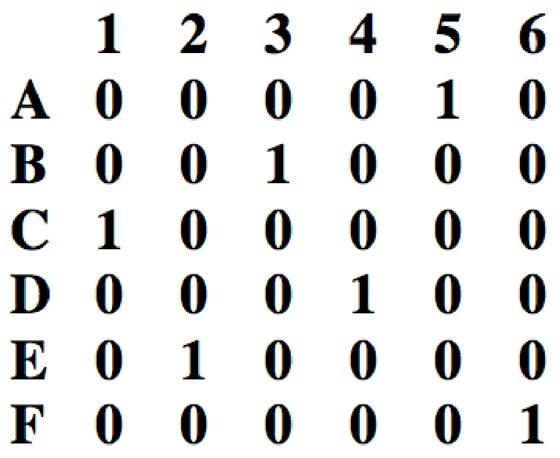
Execution Approach

- However, note a fundamentally different approach to execution than, for example, MLPs
- Rather than find a decision surface for classification, Hopfield returns the nearest memory/stable state
- Thus, appropriately adjusting and filling in the blanks of an initial state, such that it becomes an "appropriate" final state
- Real potential for Hopfield/relaxation was demonstrated when used as an optimization engine (rather than just a memory)
  - TSP – "Good" solutions in fast time

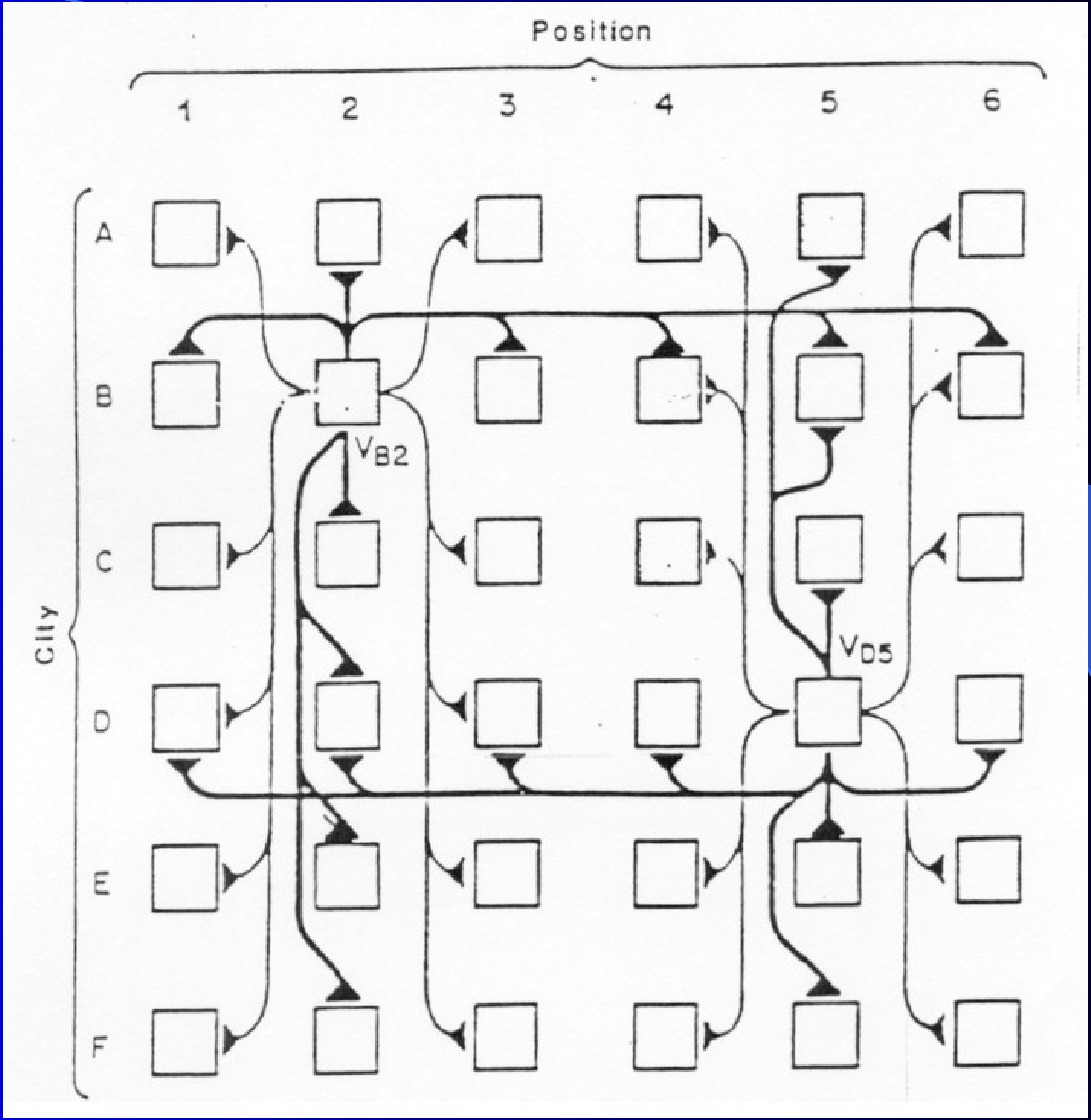
TSP Setup

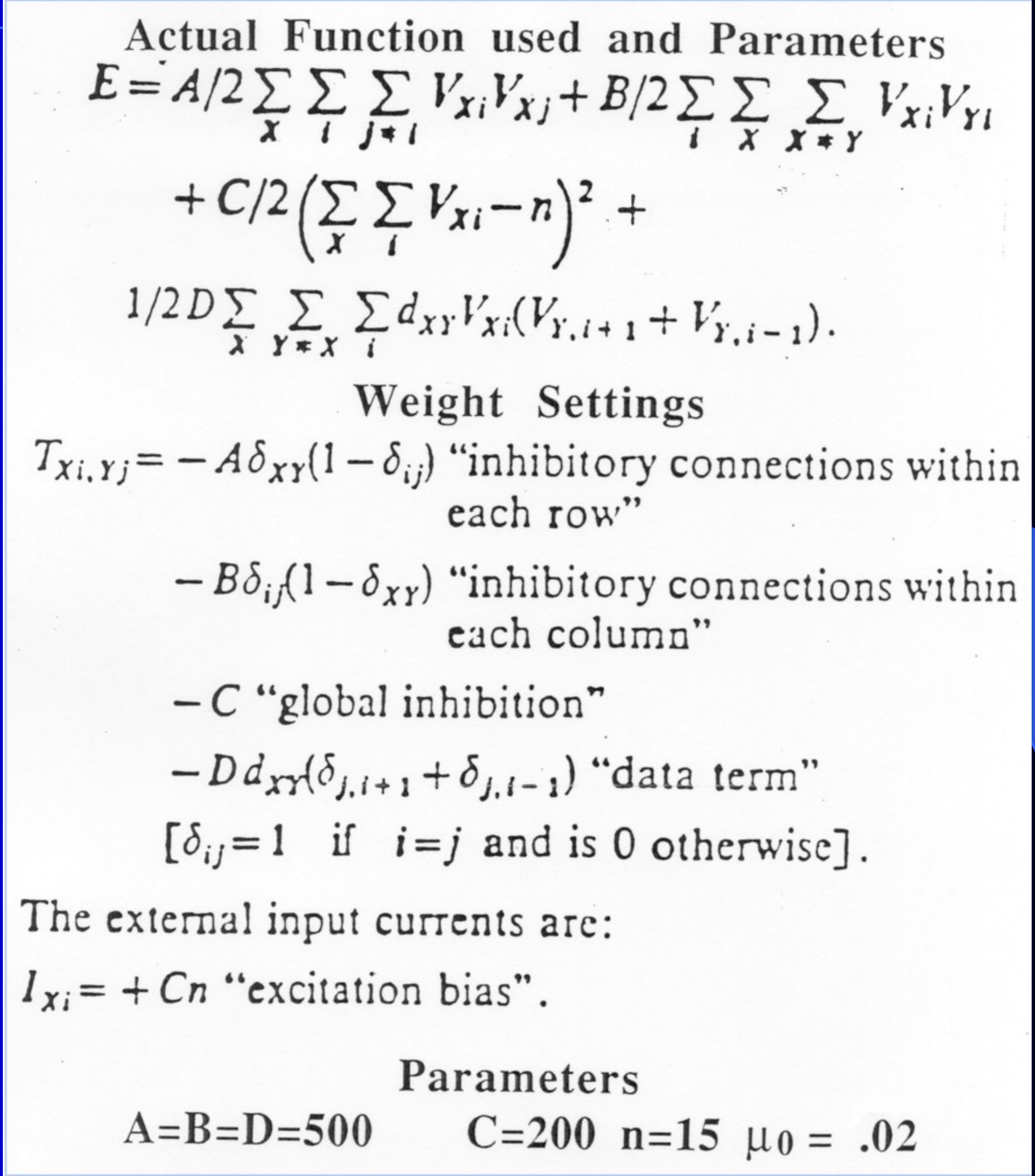
- We want shortest cycle with no repeat cites
- *n* cities require n^2 nodes
- 2^{n^2} possible binary final states for the network
- *n*! legal paths
- *n*!/2*n* distinct legal paths
- We want a legal path which is also short
- How do we set weights; energy equation or intuition?





	1	2	3	4	5	6
A	0	0	0	0	1	0
B	0	0	1	0	0	0
C	1	0	0	0	0	0
D	0	0	0	1	0	0
E	0	1	0	0	0	0
F	0	0	0	0	0	1


Position
1
2
3
4
5
6
City
A
B
C
D
E
F
V_{B2}
V_{D5}



Actual Function used and Parameters

$$E = A/2 \sum_X \sum_i \sum_{j \neq i} V_{Xi} V_{Xj} + B/2 \sum_i \sum_X \sum_{X \neq Y} V_{Xi} V_{Yi}$$

$$+ C/2 \left(\sum_X \sum_i V_{Xi} - n \right)^2 +$$

$$1/2 D \sum_X \sum_{Y \neq X} \sum_i d_{XY} V_{Xi} (V_{Y,i+1} + V_{Y,i-1}).$$

Weight Settings

$T_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij})$ "inhibitory connections within each row"

$-B\delta_{ij}(1-\delta_{XY})$ "inhibitory connections within each column"

$-C$ "global inhibition"

$-D d_{XY}(\delta_{j,i+1} + \delta_{j,i-1})$ "data term"

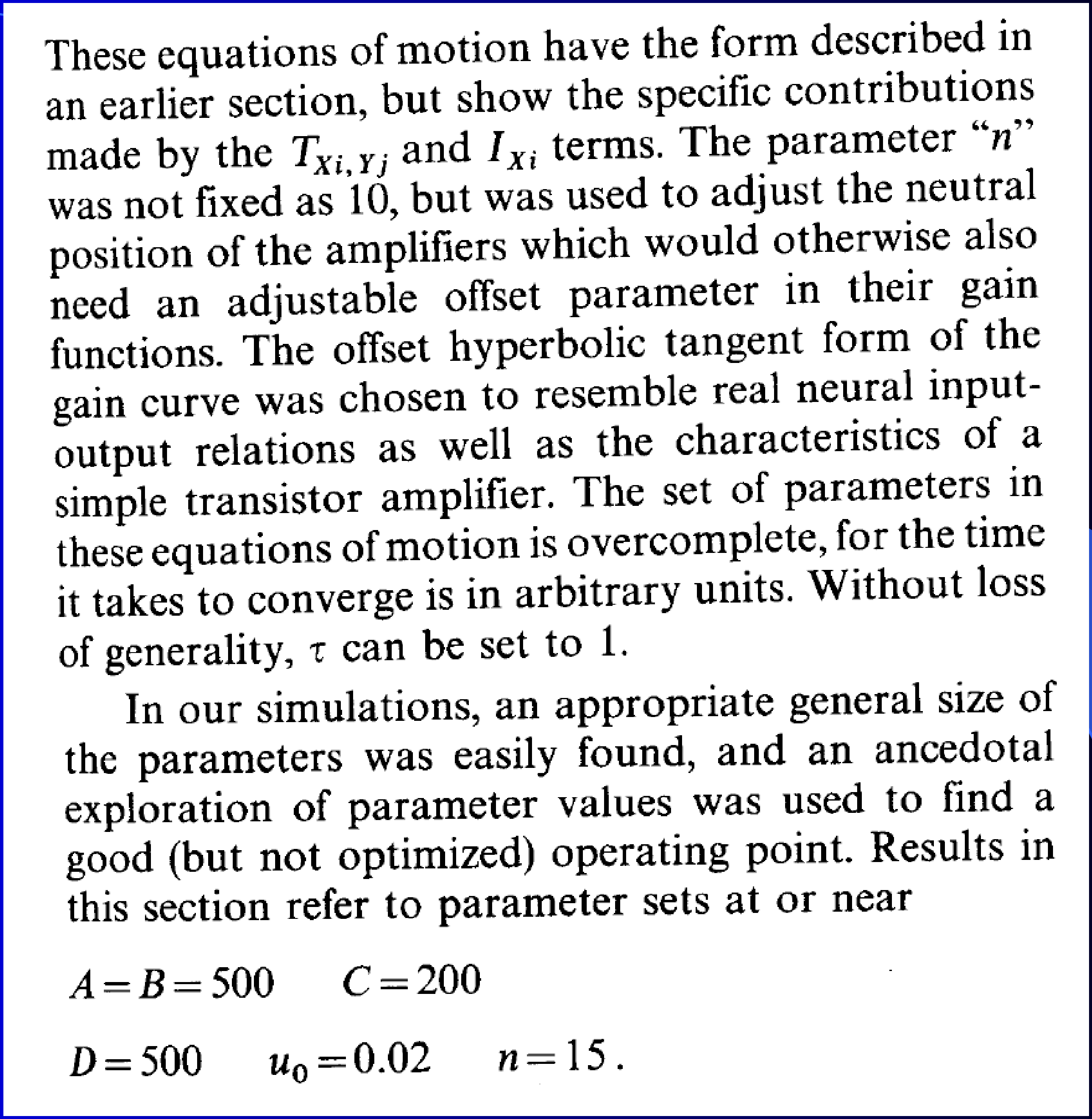
[$\delta_{ij} = 1$ if $i = j$ and is 0 otherwise].

The external input currents are:

$I_{Xi} = +Cn$ "excitation bias".

Parameters

A=B=D=500 C=200 n=15 $\mu_0 =$ .02



These equations of motion have the form described in an earlier section, but show the specific contributions made by the $T_{Xi,Yj}$ and I_{Xi} terms. The parameter "n" was not fixed as 10, but was used to adjust the neutral position of the amplifiers which would otherwise also need an adjustable offset parameter in their gain functions. The offset hyperbolic tangent form of the gain curve was chosen to resemble real neural input-output relations as well as the characteristics of a simple transistor amplifier. The set of parameters in these equations of motion is overcomplete, for the time it takes to converge is in arbitrary units. Without loss of generality, τ can be set to 1.

In our simulations, an appropriate general size of the parameters was easily found, and an ancedotal exploration of parameter values was used to find a good (but not optimized) operating point. Results in this section refer to parameter sets at or near

$$A = B = 500 \qquad C = 200$$

$$D = 500 \qquad u_0 = 0.02 \qquad n = 15\,.$$

- For *n* = 30 cities, there are 4.4×10^{30} distinct legal paths
- Typically finds one of the best 10^7
- Thus pruning 10^{23}
- How do we deal with local minima?

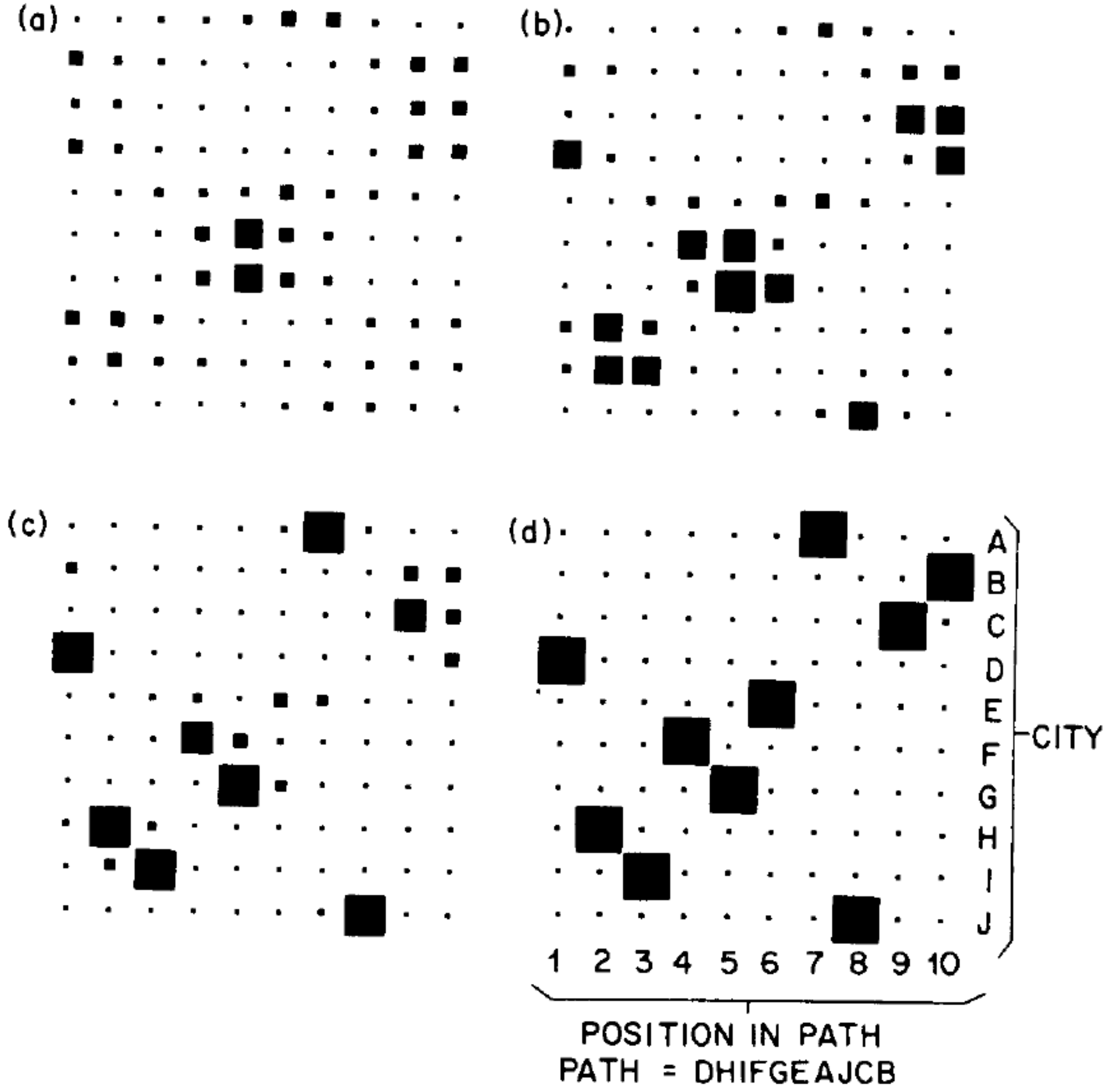


Fig. 2a–d. The convergence of the 10-city analog circuit to a tour. The linear dimension of each square is proportional to the value of V_{Xi}. **a**, **b**, **c** intermediate times, **d** the final state. The indices in **d** illustrate how the final state is decoded into a tour (solution of TSP)

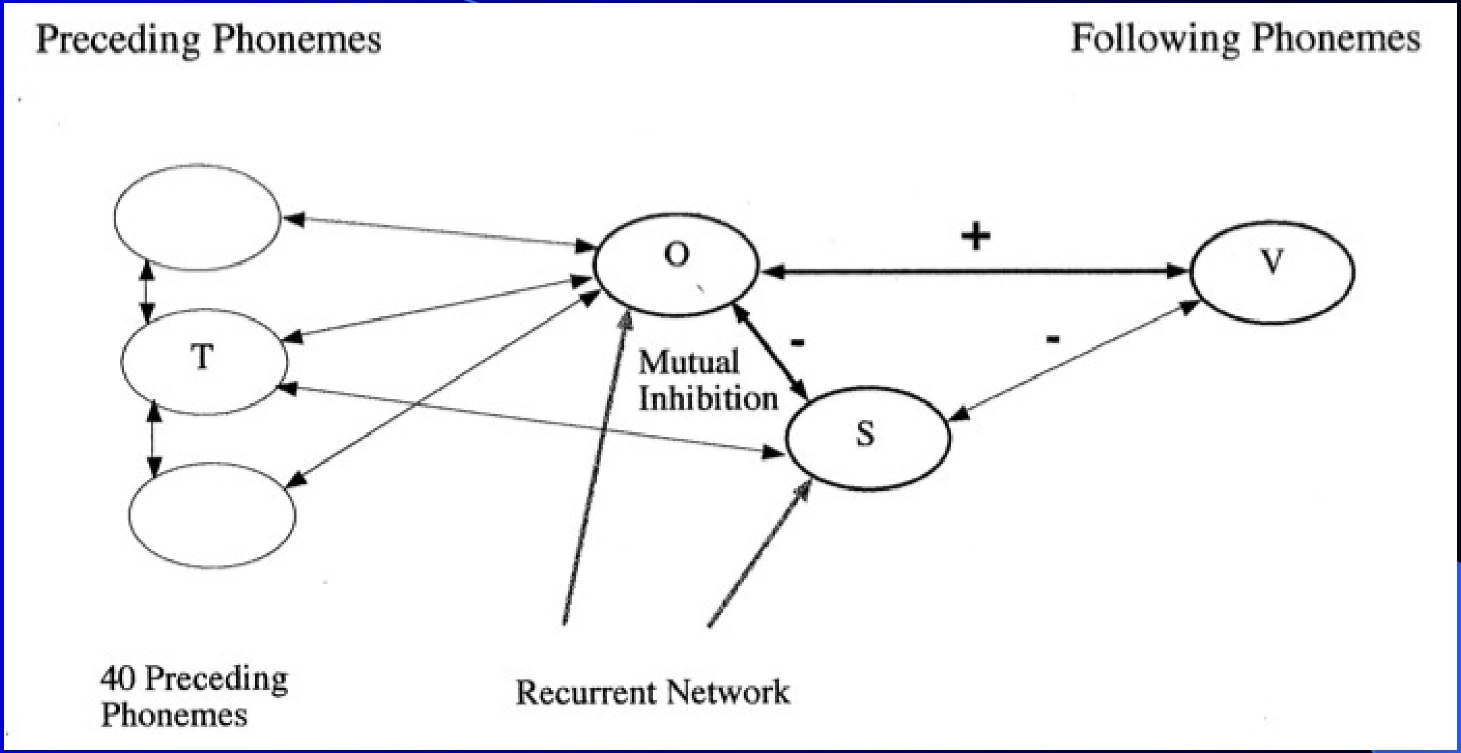
Hopfield Summary

- Highly Parallel, numerous hardware/optic implementations
- Significant biological plausabilities
- Not great as an associative memory (saturation issues), but for optimization there are no saturation issues – set weights to fit the problem
- Stability Issues – Local Minima
  - Original Hopfield net relaxes to a valid TSP about 20% of time
  - Our extensions to TSP (MCA and RR as part of Multcons): increase valid tours to 78%, decrease avg valid tour length by 64%
- Hopfield did not give a learning algorithm
  - We did BP style training, other efforts
- Basic Hopfield has no hidden nodes
  - Boltzmann, Multcons

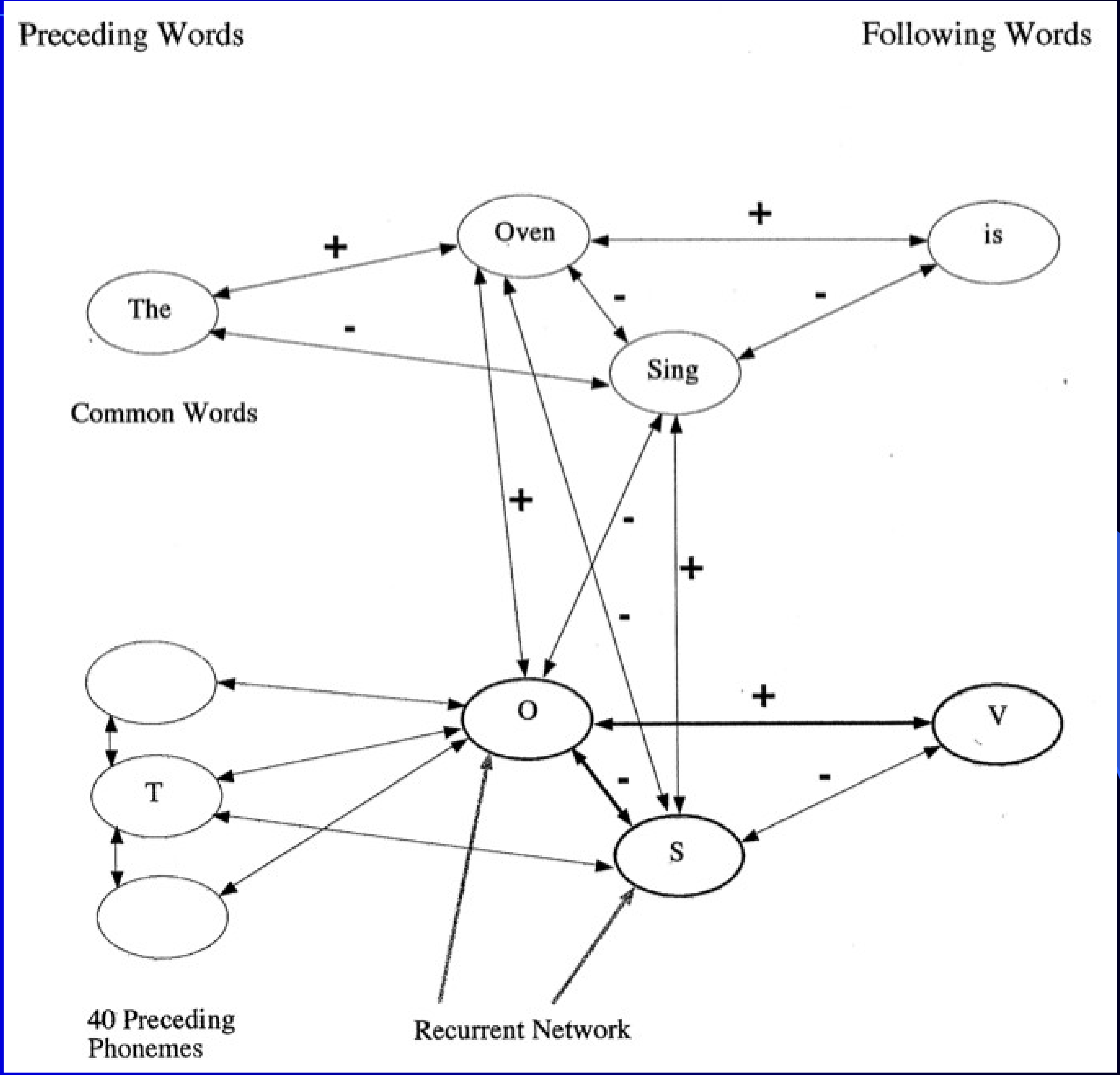
Multcons

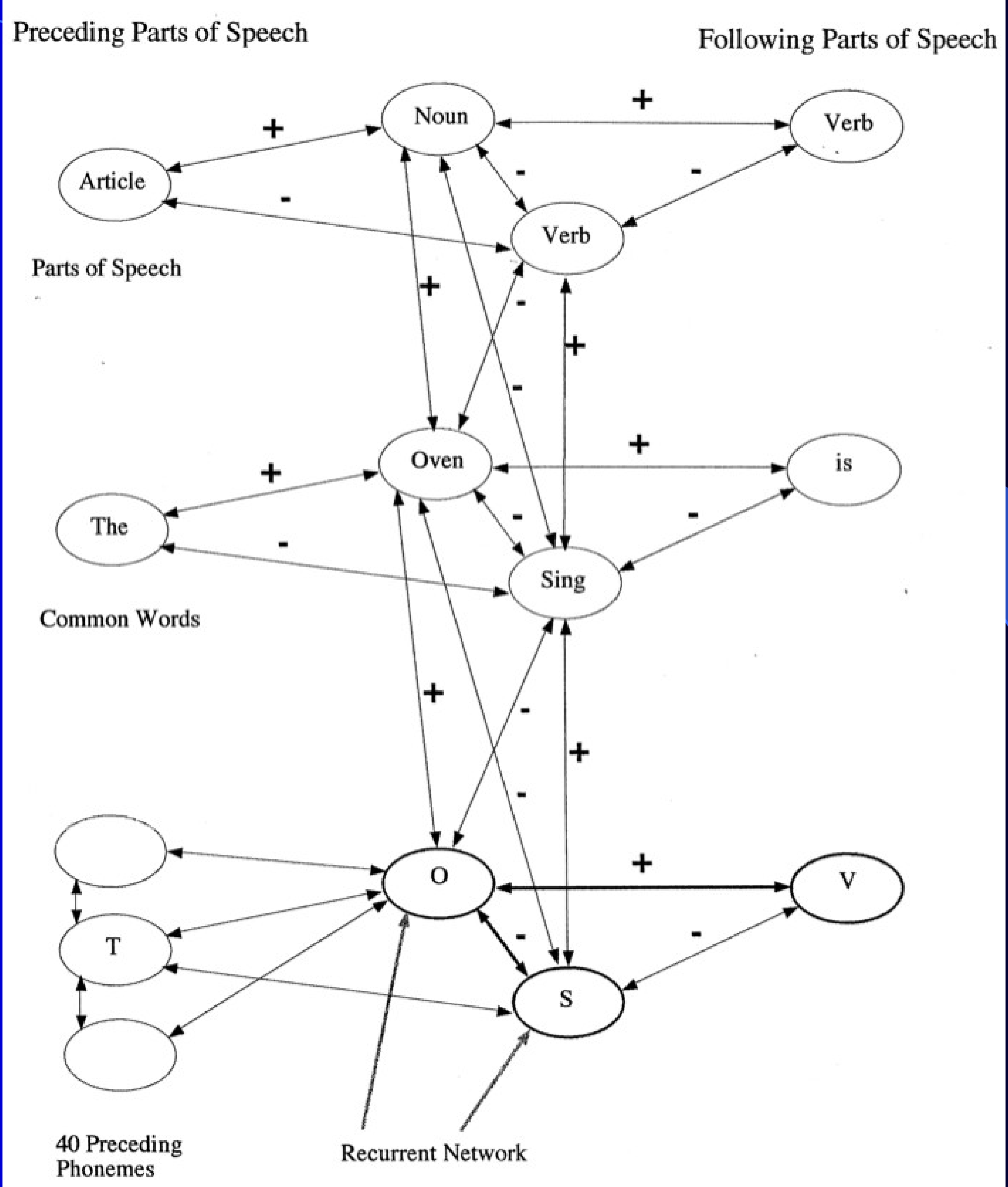
- Multi-layer temporal constraint satisfaction networks
- Speech Recognition, Free Market Economics
- Multi-layer relaxation network
- Weights are asymmetric (don't observe cycles)
- Weights are learned – Backpropagation style learning with potential hidden nodes between nodes and layers

- All nodes/aspects interact before relaxing to a final utterance



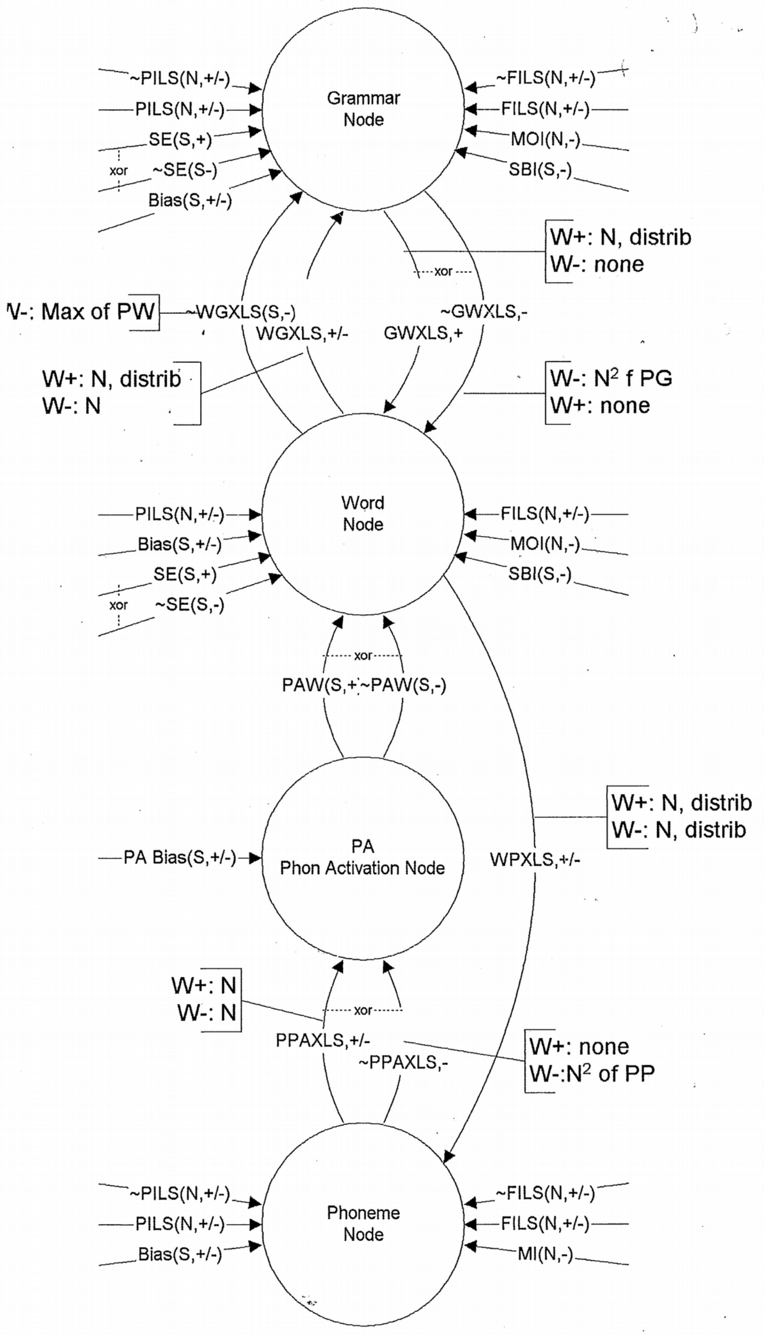
- Speech utterance first processed by an MLP to give initial phoneme estimates at different sample times
- Then network allowed to start relaxing


Preceding Words
Following Words
Oven
is
The
Sing
Common Words
O
V
T
S
40 Preceding Phonemes
Recurrent Network
+
-


Preceding Parts of Speech
Following Parts of Speech
Noun
Verb
Article
Verb
Parts of Speech
Oven
is
The
Sing
Common Words
O
V
T
S
40 Preceding Phonemes
Recurrent Network
+
-

Multcons Notes

- There are many nodes in the network. However, in software unlikely nodes (low activations) are quickly pruned allowing high efficiency
- Weights are learned with perceptron style updates or Backpropagation with hidden nodes between layers to learn higher order combinations
  - Train on an many example utterances. For each utterance, if a node does not relax to the proper target, then adjust its incoming weights to make that node wore likely to hit the target next time (using perceptron or BP learning rules with appropriate learning rates).
    - BP learning update rule is slightly changed due to MCA vs Sigmoid activation function



FILS - Follows Intralayer Synapse. Weight based on the probability of Y given that X follows.
~FILS - Not Follows ILS. Weight based on the probability of Y given that X does not follow.
PILS - Preceeds Intralayer Synapse. Weight based on the probability of Y given that X preceeds.
~PILS - Not Preceeds ILS. Weight based on the probability of Y given that X does not preceed.
SE - Sequential Excitation. Weight based on a sequence of active nodes or properly bounded by silence.
SBI - Silence boundary inhibition. Inhibition due to a silent area found in the middle of a word.
MOI - Mutual Overlap Inhibition. Inhibitory weight between overlapping nodes based on activations and amount of overlap.
MI - Mutual Inhibition. Inhibitory weight based on excitation of other phonemes.
Bias - Offset value for node.
WGXLS - Word-to-Grammar eXtra Layer Synapse. Weight based on probability of grammar node Y given that word node X is active.
GWXLS - Grammar-to-Word eXtra Layer Synapse. Weight based on probability of word node Y given that grammar node X is active.
~GWXLS - Not Grammar-to-Word eXtra Layer Synapse. Probability of word node Y given that grammar node X does is not active.
PAW - Phoneme Activation (PA)-to-Word. Weight based on the probability of word node Y given that the combined activation of word Y's phonemes is positive.
~PAW - Not PA-to-Word. Weight based on the probability of word node Y given that the combined activation of the word Y's phonemes is negative.
PA Bias - Bias for PA node
PPAXLS - Phon-to-PA Node eXtra Layer Synapse. Weight based on the presence of phoneme X in word Y
~PPAXLS - Not Phon-to-PA node eXtra Layer Synapse. Weight based on the absence of phoneme X in word Y
WPXLS - Word-to-Phon eXtra Layer Synapse. Weight base of probability of Phoneme Y given word X.