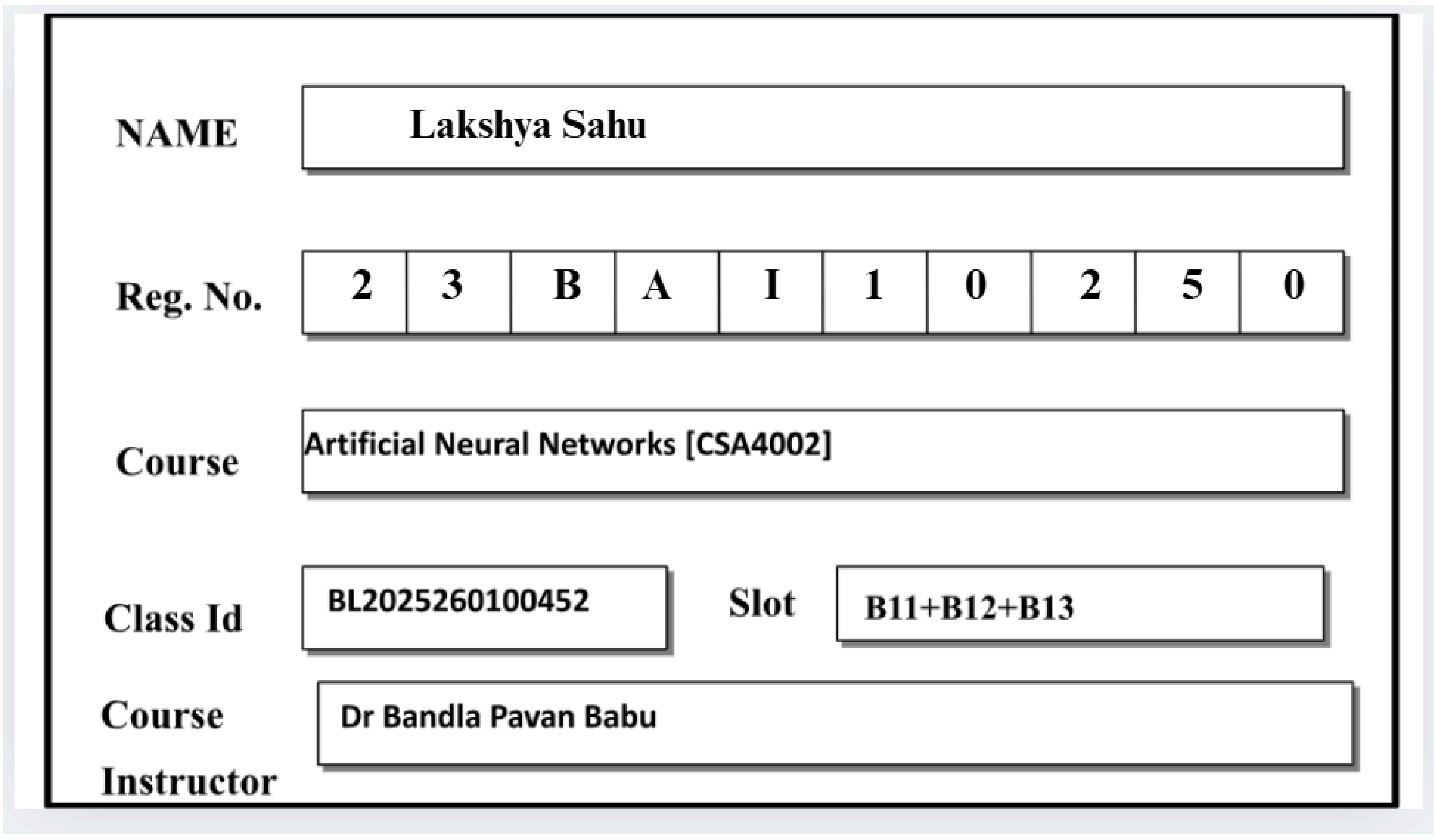
FALL SEM 2025-2026



NAME	Lakshya Sahu
Reg. No.	2 3 B A I 1 0 2 5 0
Course	Artificial Neural Networks [CSA4002]
Class Id	BL2025260100452
Slot	B11+B12+B13
Course Instructor	Dr Bandla Pavan Babu

PRACTICAL'S RECORD

VIT Bhopal
University Bhopal-Indore
Highway Kothrikalan,
Sehore
Madhya Pradesh – 466114

Reg.No: 23BAI10250

INDEX

Lab 1[a] :Parallel and distributed processing - I: Interactive activation and competition models

AIM: The objective of the IAC model is to illustrate the process of retrieving general and specific information from stored knowledge of specifics. Some of the features of the human memory that are illustrated with this model are:

1. Retrieval by key (name) and by context.
2. Retrieval with noisy clues.
3. Assignment of default values and spontaneous generalization.

APPARATUS: https://www.vlab.co.in/ , Laptop

THEORY: PDP Model Development

This model is illustrated by the Jets and Sharks database described in [McClelland, 1981] and also given in the following table.

Name	Gang	Age	Education	Marital Status	Occupation
Art	Jets	40s	Junior High	Single	Pusher
Al	Jets	30s	Junior High	Married	Burglar
Sam	Jets	20s	College	Single	Bookie
Clyde	Jets	40s	Junior High	Single	Bookie
Mike	Jets	30s	Junior High	Single	Bookie
Jim	Jets	20s	Junior High	Divorced	Burglar
Greg	Jets	20s	High School	Married	Pusher
John	Jets	20s	Junior High	Married	Burglar
Doug	Jets	30s	High School	Single	Bookie
Lance	Jets	20s	Junior High	Married	Burglar
George	Jets	20s	Junior High	Divorced	Burglar

Pete	Jets	20s	High School	Single	Bookie
Fred	Jets	20s	High School	Single	Pusher
Gene	Jets	20s	College	Single	Pusher
Ralph	Jets	30s	Junior High	Single	Pusher
Phil	Sharks	30s	College	Married	Pusher
Ike	Sharks	30s	Junior High	Single	Bookie
Nick	Sharks	30s	High School	Single	Pusher
Don	Sharks	30s	College	Married	Burglar
Ned	Sharks	30s	College	Married	Bookie
karl	Sharks	40s	High School	Married	Bookie
Ken	Sharks	20s	High School	Single	Burglar
Earl	Sharks	40s	High School	Married	Burglar
Rick	Sharks	30s	High School	Divorced	Burglar
Ol	Sharks	30s	College	Married	Pusher
Neal	Sharks	30s	High School	Single	Bookie
Dave	Sharks	30s	High School	Divorced	Pusher

An interactive activation and competition network consists of a collection of processing units organized into some number of competitive pools. There are excitatory connections among units in different pools and inhibitory connections among units within the same pool. The excitatory connections between pools are generally bidirectional, thereby making the processing interactive in the sense that processing in each pool both influences and is influenced by processing in other pools. Within a pool, the inhibitory connections are usually assumed to run from each unit in the pool to every other unit in the pool. This implements a kind of competition among the units such that the unit or units in the pool that receive the strongest activation tend to drive down the activation of the other units. The information about such data, if stored in a computer memory, can be accessed by name or by any other set of items which can serve as a key to a particular set of information. But this requires the method of access to be pre-programmed in the system. Moreover, certain characteristics of the data like the distribution of persons in different age groups or the 'common' characteristics among some persons, etc, can be obtained only by programming explicitly to derive the information embedded in the data. In other words, any information in the data has to be sought explicitly. On the other hand, human memory stores the information in the data in terms of patterns, and these patterns are useful to recall information even with partial clues. These features of the human memory can be demonstrated through representation in the form of a Parallel and Distributed Processing model, as shown in the figure below.

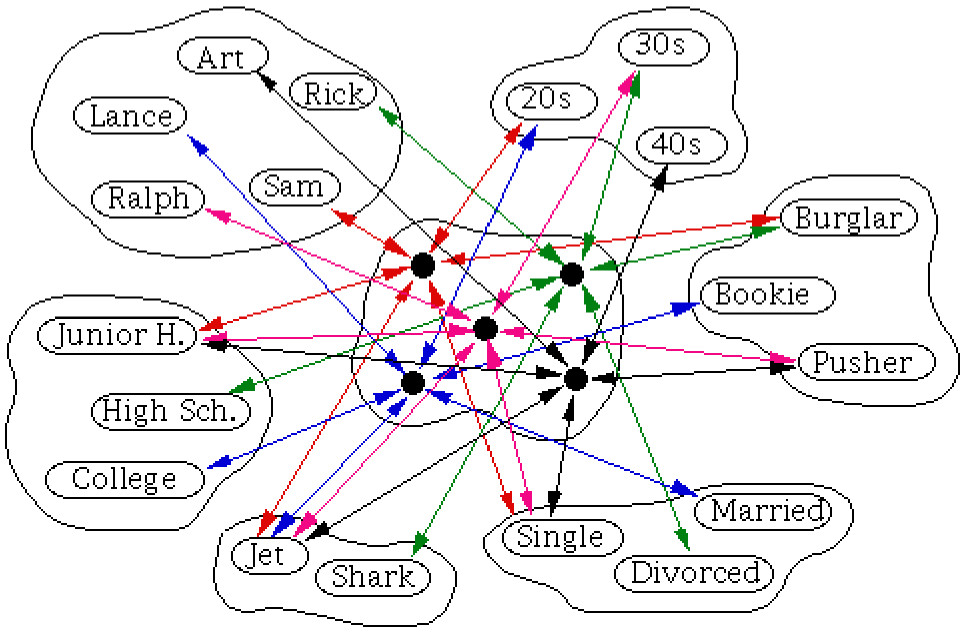


Figure 1: *Figure illustrating the units and connections among those units in different pools. Though here only some units are shown, we actually have 27 units in the 'Name' pool. Similarly there are 27 units in the 'Instance' pool, 2 in the 'Gang' pool and then 3 in the rest of the other 4 pools.*

In the figure, the units are organized into different pools, such as the 'Name' pool, 'Age' pool, etc. The number of units in each pool corresponds to different possibilities in that category, as for example, 27 units in the 'Name' pool and 3 units in the 'Age' pool, etc. There are as many pools as there are categories (6 in this case), plus one additional pool called the 'Instance' pool.

PDP Model Description

Units within each pool are connected in an inhibitory manner, i.e., the output of each unit is fed with a negative weight to all other units in the same pool. The units in each pool are connected to the corresponding units in the instance pool in an excitatory manner, i.e., the connection weights are positive. For example, here, the 'Ralph' unit in the 'Name' pool, 'Jets' unit in the 'Gang' pool, 'in 30s' unit in the 'Age' pool, 'JH' unit in the 'Education' pool, 'pusher' unit in the 'Occupation' pool and 'Unmarried' unit in the 'Marital status' pool are all connected to the 'Ralph' unit in the 'instance' pool with positive weights. The units in the 'instance' pool are called "hidden" units, since by design, they are not accessible for any input or output. The units in all other pools are "visible" units. Only 5 of 27 units are shown in the 'Name' pool and the 'instance' pool for illustration. Also, the inhibitory connections within each pool are not shown in the figure.

There are a total of 68 units in the model. Each unit computes a weighted sum of the input values to the unit fed from other units, as well as from external inputs, if any. The weighted sum or the activation value is passed through a nonlinear output function, which gives as output the activation value itself, if the sum lies between some prespecified minimum and maximum values. Otherwise, the function gives either the minimum value at the lower limit or the maximum value at the upper limit. The state of the model is described as the collection of outputs of all the 68 units at any given instant of time. Starting from any state, the next state can be computed by selecting a unit at random and computing the weighted sum of its inputs first and then the output of the unit. Due to change in the value of the output of this unit, the model goes to a different state. Then another unit is selected at random, and the new state for that unit is determined. All the units are updated by selecting the units in a random sequence, to compute one cycle of activation dynamics. After several cycles, the model is guaranteed to reach a stable equilibrium state, when there will not be any further change in the state of the model.

For each set of external inputs, the model reaches a stable state eventually. From the stable state the stored data can be read out. For example, if we want the data about 'Ralph', the output (state) of the 'Ralph' unit in the 'Name' pool is set to maximum. Starting with some initial values of state on other units, the network states are computed for several cycles, until an equilibrium state is reached. At equilibrium, there will be one unit in each pool having a large positive value. Those units correspond to the data that belongs to 'Ralph'.

Features Demonstrated By The Model

The model demonstrates several features of the functioning of the biological neural network in human memory. Some of the features are: (a) Retrieving an individual's data from his name, (b) retrieval from a partial description, (c) graceful degradation, (d) default assignment, and (e) spontaneous generalization for novel inputs. The most important point is that the model stores the patterns embedded in the given data. Therefore from the model, one could get even such information for which the model was not explicitly designed. For example, in the feature of default assignment, the model gives possible good guesses about missing information, even though we do not know certain things about an individual. It evaluates the relative strengths of the attributes from the given data in a complex manner, which is difficult to describe explicitly. Thus, this model clearly brings out the distinction between computer memory and human memory for storage and retrieval of information. The model also brings out the features of content-addressable memories and associative memories for information retrieval. Note the distributed nature of the memory in this model, in the sense that the information is distributed in the weights throughout the network. Also note the parallel and distributed nature of the activation dynamics when the model is realized in hardware, and also when it is allowed to relax naturally changing from one state to another until an equilibrium state is reached from the given initial state and external input. Spontaneous generalization is the ability to provide a generalized picture of what is common to the memories that match the cues which are too vague to match a particular memory. If weights between i'th and j'th components is +ve, q = weight * activation excitation += q for all units.

if weights between i'th and j'th components is -ve,
q = weight * activation
inhibition += q for all units.

net input = (estr * external input) + (beta * excitation) + (gamma * inhibition)
delta excitation = (actmax - activation) * net input - decay * (activation - actrest)
activation += delta excitation
g Delta = g Delta + absolute(delta excitation)

delta inhibition = (activation - actmin) * net input - decay * (activation - actrest);
activation += delta inhibition;
g Delta = g Delta + absolute(delta inhibiton);

For more information and help regarding IAC, click here

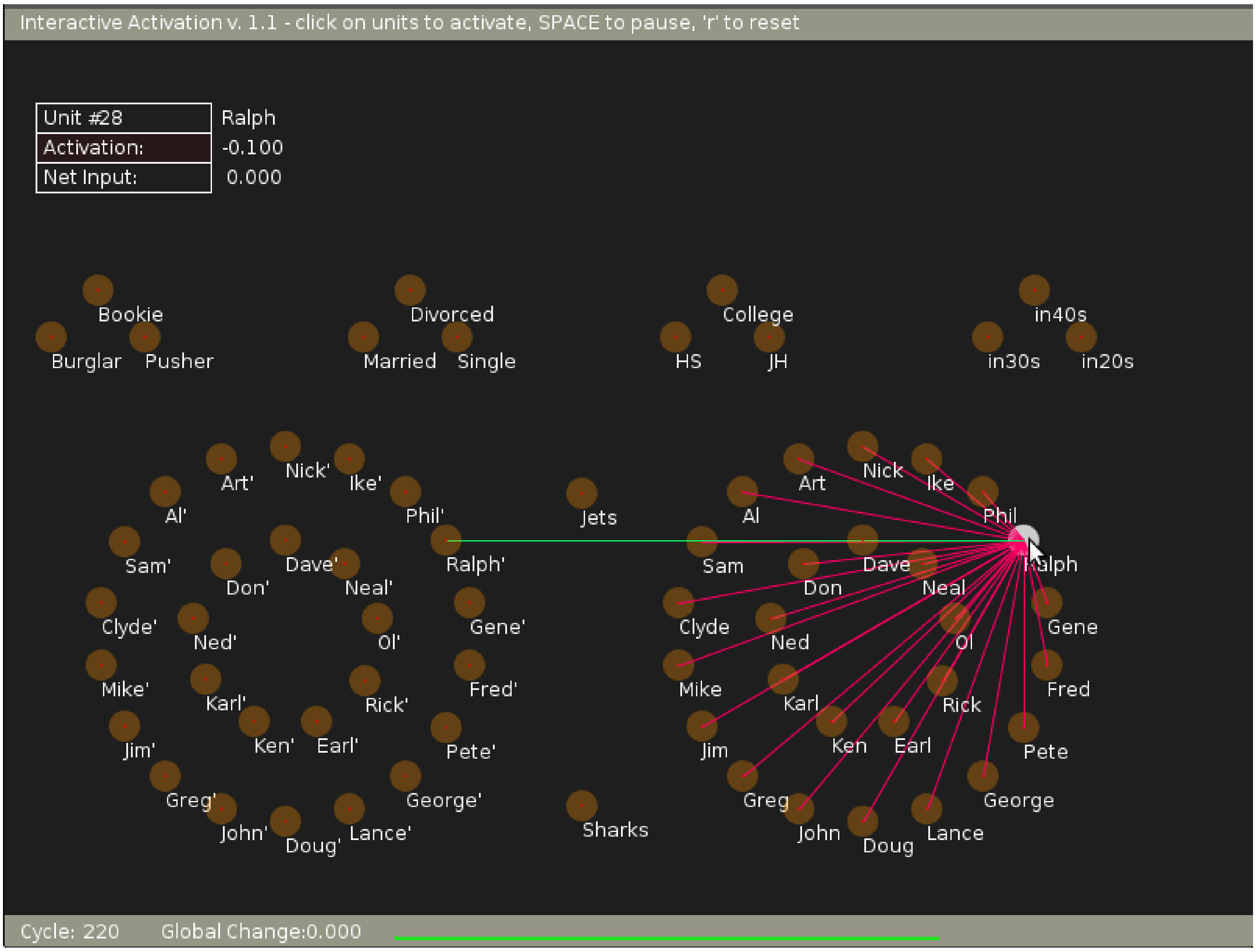
Illustration of Interactive Activation and Competition Model

There are a total of 68 units in this model. Each unit computes a weighted sum of the input values to the unit, fed from other units as well as external inputs, if any. The weighted sum or the activation value is passed through a non-linear output function, which gives as output, the activation value itself, if the sum lies between some prespecified minimum and maximum values. Otherwise, the function gives either the minimum value at the lower limit or the maximum value at the upper limit. The state of the model is described as output of all the 68 units at any given instance of time. Starting from any state, the next state can be computed by selecting a unit at random and computing the weighted sum of its inputs first, and then the output of the function.

Due to change in the value of the output of this unit, the model goes to a different state. Then another unit is selected at random and the new state for that unit is determined. All the units are updated by selecting all the units in a random sequence, to compute one cycle of activation dynamics. After several cycles, the model is guaranteed to reach equilibrium state, when there will not be any further change in the state of the model.

For each set of external inputs, the model reaches a stable state eventually. From the stable state the stored data can be read out. For example, if we want the data about 'Ralph', the output (state) of the 'Ralph' unit in the 'Name' pool is set to maximum. Starting with some initial values of state on other units, the network states are computed for several cycles, until an equilibrium state is reached. At equilibrium, there will be one unit in each pool having a large positive value. Those units correspond to the data that belongs to 'Ralph'.

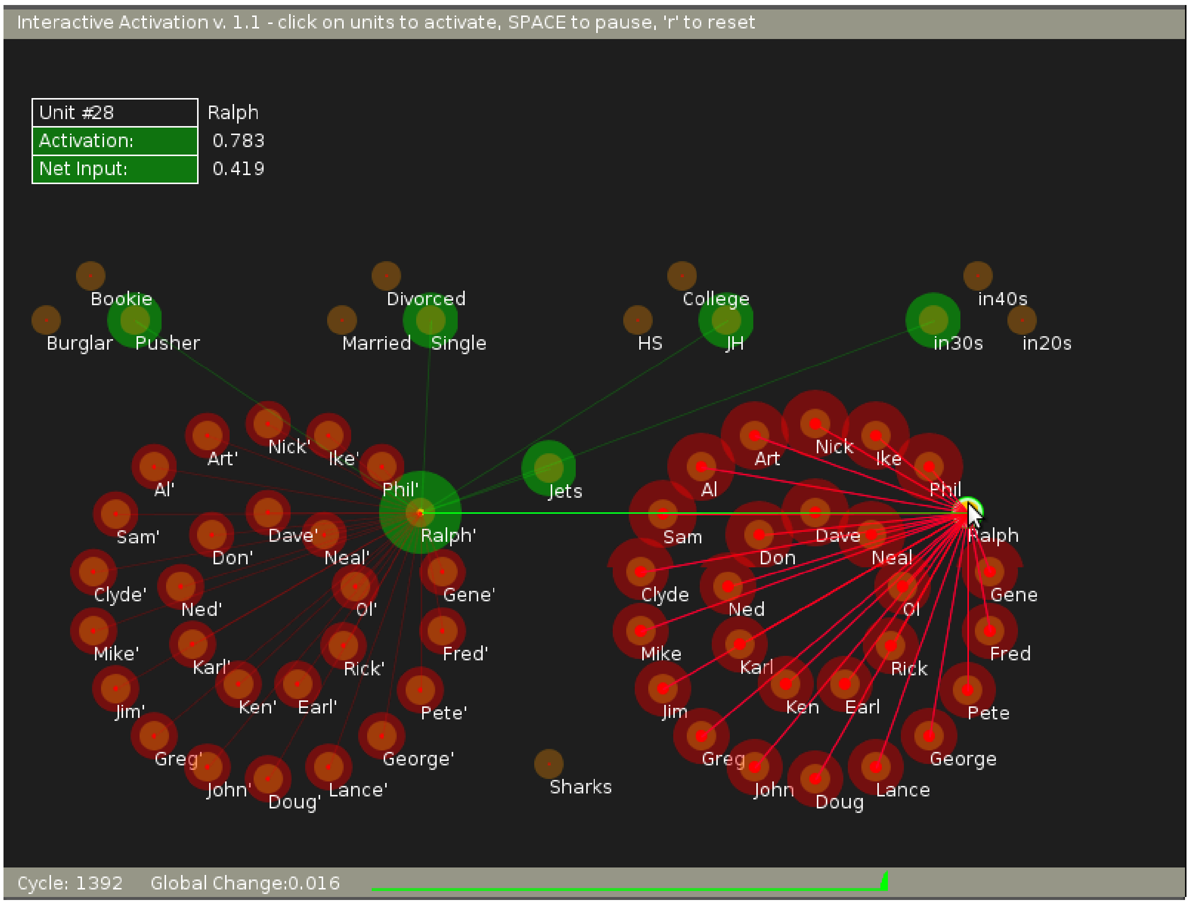
In the experiment here, in the initial state we can see, as illustrated by the following figure, how the 'Ralph' unit is connected in an inhibitory sense (illustrated by red connections) with the rest of the members of the 'Name' pool. Also it is connected by an excitatory connection to the unit 'Ralph' in the 'instance' pool (illustrated by a green connection).


Interactive Activation v. 1.1 - click on units to activate, SPACE to pause, 'r' to reset

Unit #28	Ralph
Activation:	-0.100
Net Input:	0.000

Cycle: 220 Global Change:0.000

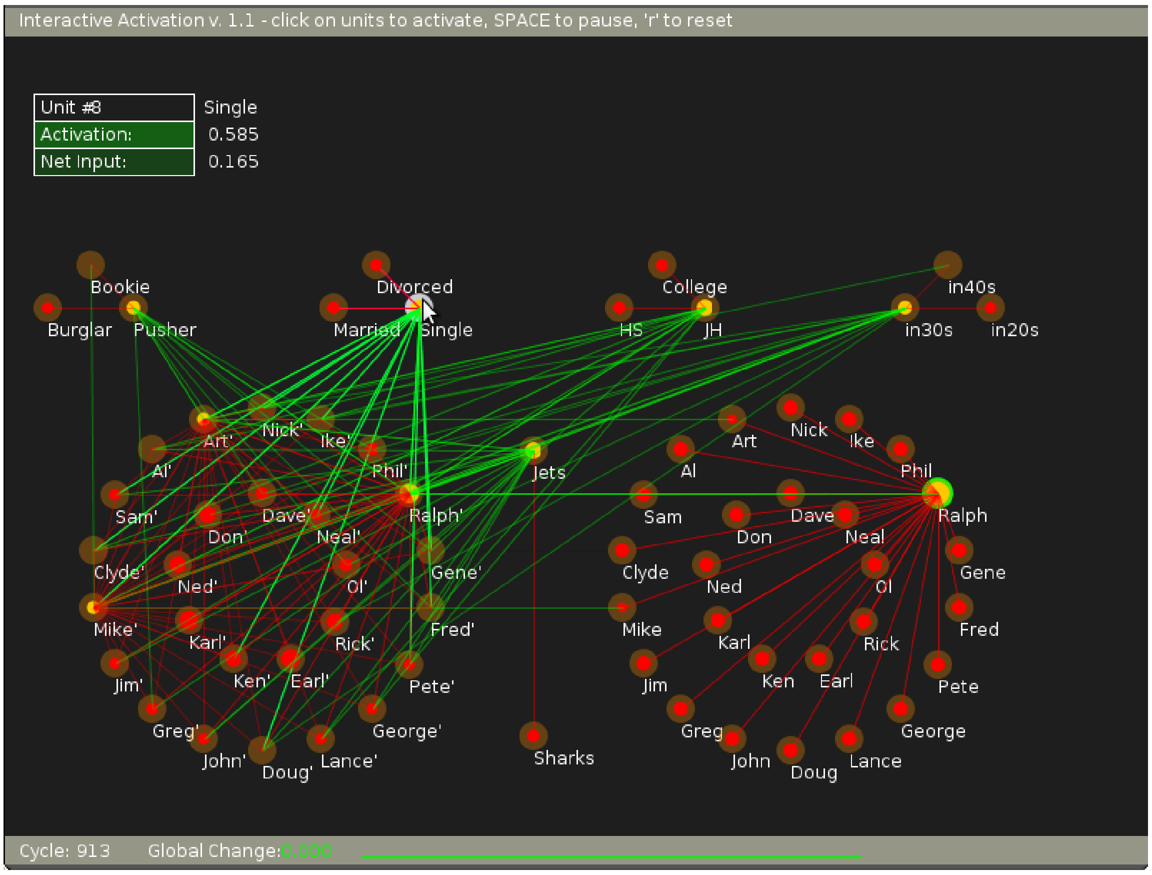
Now, when an external input is applied by changing the state of the 'Ralph' unit (as by clicking the mouse over 'Ralph' node in the 'Name' pool) we see that all the nodes settle down to a stable state after some time.


Interactive Activation v. 1.1 - click on units to activate, SPACE to pause, 'r' to reset

Unit #28	Ralph
Activation:	0.783
Net Input:	0.419

Cycle: 1392 Global Change:0.016

Finally when we try to read out the data related to the 'Ralph' unit from the model, we just need to visit every pool and find out which node has the highest positive value. That data belongs to the 'Ralph' unit.


Interactive Activation v. 1.1 - click on units to activate, SPACE to pause, 'r' to reset

Unit #8	Single
Activation:	0.585
Net Input:	0.165

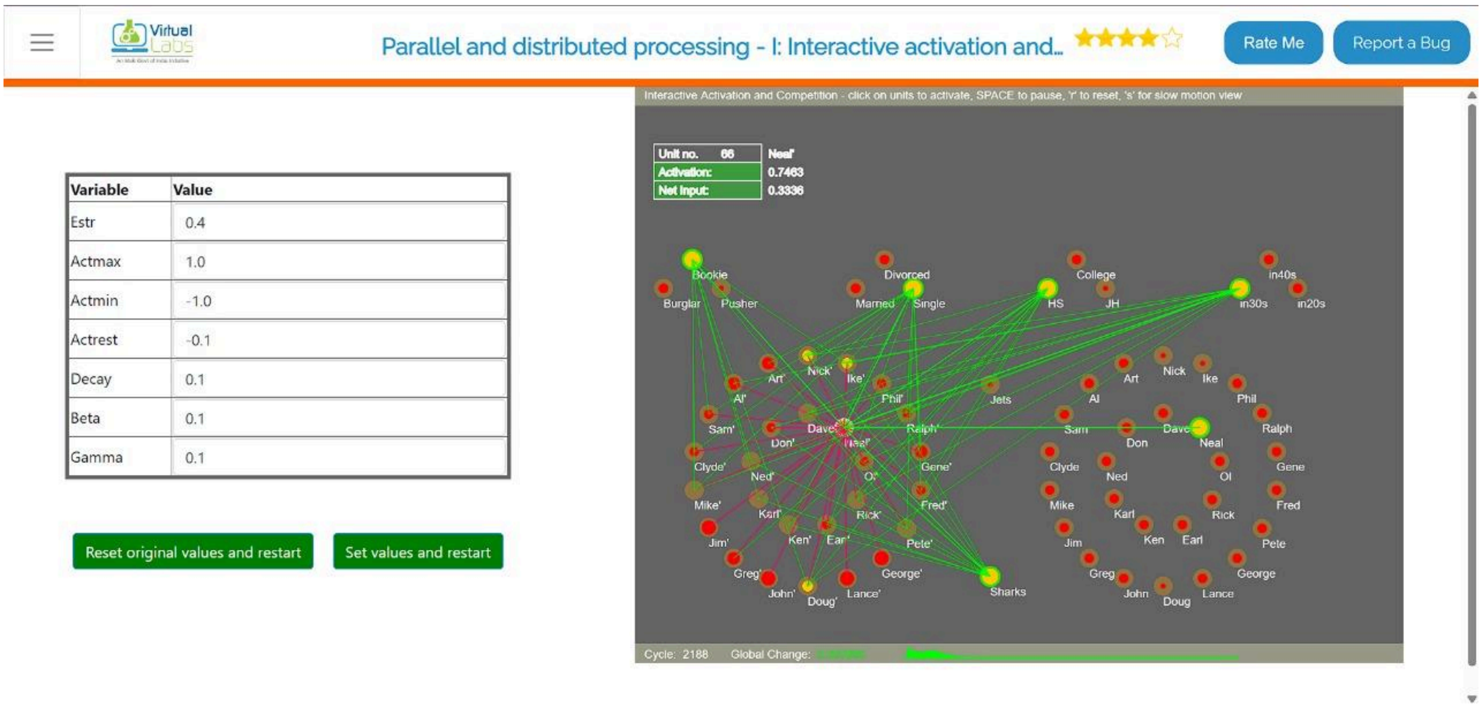
Cycle: 913 Global Change:0.000

For this experiment, we can provide external bias and hence activation to units of all the pools except the instance pool. The instance pool, which is also known as the hidden pool, is not accessible to the end user. Yet we can see the connections within and outside this pool by moving the mouse pointer to respective units.

PROCEDURE: Check the initial state of the model by moving the mouse over various units in different pools.

- Check the value for every unit and the inhibitory and excitatory connections it exhibits with units in same and different pools.
- Click on any unit in the 'Name' pool and let the model settle down (in a couple of seconds). The settling down can be observed through the global change scale in the bottom margin.
- Observe the connections between various units and pools (both green and red), popping up after the external input is provided.
- Check with all the pools, the units with highest values within them, and match with the table provided in the tutorial section whether it belongs to the same unit.
    i. Observe the bottom of the simulator to know which cycle number is going on and to see the g-delta plot and global change value.
    ii. Hover your mouse over any component to highlight the component and see its excitatory and inhibitory connections with other units. The activation and net input of that unit is displayed on top left.
    iii. Click on any unit (except in the instance pool) to give external input to that unit and activate it.
    iv. Observe how activation of that unit affects the other units activation and net inputs.
    v. Also observe how the global change spikes in the few cycles after giving external input to a unit.
    vi. Click on the same unit to take away the external input given to that unit.
    vii. Observe how the residual net input remains even after removing external input.
    viii. Press spacebar to pause the simulator in the current cycle. Press the spacebar again to resume cycles.
    ix. In a paused position, hover over units whose activation data you want. In a paused position, the cycle will be stopped and giving external input will be disallowed.
    x. Press r to reset the simulator with the same values just used in the simulator.
    xi. Press s to activate slow motion mode which performs cycles slower than actual speed to get a closer look at how the values of the units change. Press s again to resume real speed cycle change.
    xii. Change the values of the variables in the table (in range of (original value - 0.5) to (original value + 0.5)) and click on set values and restart to start a new set of cycles with changed values of variables.
    xiii. Click on Reset original values and restart to start a new set of cycles with original values of variables.

SIMULATION/MODEL:

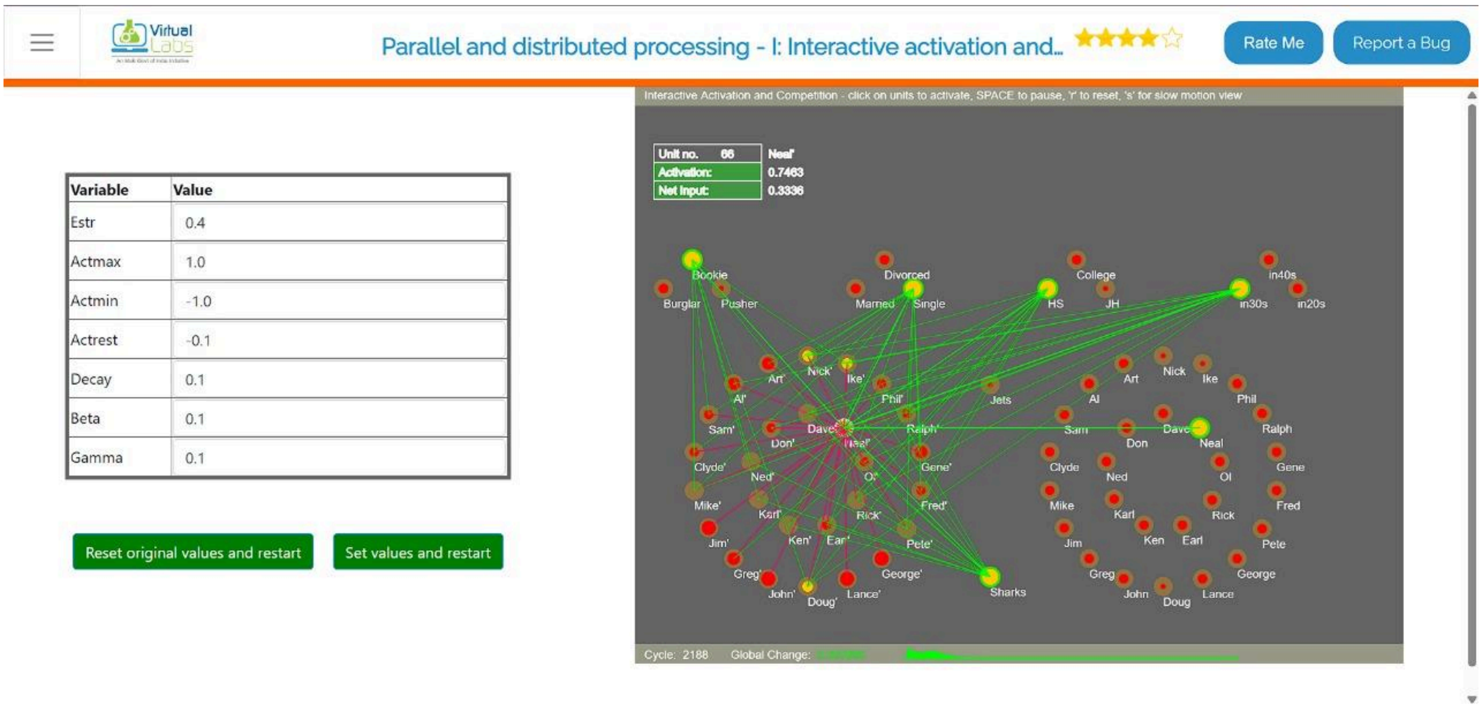


Parallel and distributed processing - I: Interactive activation and...

Interactive Activation and Competition - click on units to activate, SPACE to pause, 'r' to reset, 's' for slow motion view

Variable	Value
Estr	0.4
Actmax	1.0
Actmin	-1.0
Actrest	-0.1
Decay	0.1
Beta	0.1
Gamma	0.1

Unit no. 66	Neal'
Activation:	0.7463
Net Input:	0.3336


Parallel and distributed processing - I: Interactive activation and...

Interactive Activation and Competition - click on units to activate, SPACE to pause, 'r' to reset, 's' for slow motion view

Variable	Value
Estr	0.4
Actmax	1.0
Actmin	-1.0
Actrest	-0.1
Decay	0.1
Beta	0.1
Gamma	0.1

Unit no. 66	Neal'
Activation:	0.7463
Net Input:	0.3336

Results:
Speedup Observation:
Parallel processing (multi-threading) showed a significant reduction in execution time compared to sequential processing for matrix multiplication or prime number generation (depending on the task performed).

Example:
For a matrix size of 512x512, the sequential approach took X seconds, while the parallel approach with 4 threads took Y seconds, yielding a speedup of Z%.

Scalability:
Performance improved with an increase in threads (up to a threshold), after which overhead (context switching, resource contention) degraded efficiency.

Load Balancing:
Dynamic scheduling (e.g., chunk partitioning) demonstrated better workload distribution compared to static scheduling.

Conclusion:
Parallel processing using multi-threading effectively reduces computation time for CPU-intensive tasks by leveraging concurrency.
The speedup is limited by factors like Amdahl's Law (inherent sequential portions), thread overhead, and hardware constraints (CPU cores).
Proper synchronization (e.g., mutexes for shared resources) is critical to avoid race conditions and ensure correctness.

Applications:
Scientific Computing: Parallel matrix operations accelerate simulations (e.g., climate modeling).

Big Data Processing: Distributed frameworks like Hadoop/Spark use similar principles for large-scale data analysis.

Real-Time Systems: Multi-threading is used in gaming, financial trading, and robotics for concurrent task execution.

Lab 1[b] :Parallel and distributed processing - II: Constraint satisfaction neural network models

AIM: The objective of Parallel and Distributed Processing (PDP) Models is to demonstrate some of the features of the biological neural networks. Constraint Satisfaction (CS) Model illustrates how we attempt to build concepts or arrive at conclusions based on some limited, partial, and sometimes partially erroneous knowledge. Following are the goals of the experiment:

- To illustrate how a large number of weak constraints together will evolve into a definitive conclusion.
- To illustrate how our concepts of various types of rooms can be captured by the CS PDP model from samples of description of these rooms.

APPARATUS: https://www.vlab.co.in/ , Laptop

THEORY:
Example of recognition of handwritten characters
In an apparently simple task of recognition of handwritten characters, it is difficult to articulate precisely what we capture as features in the patterns of several samples of each character. But when we are presented with a new sample of a handwritten character, most of the time we have no difficulty in recognizing it correctly. It is likely that from the samples of a handwritten character we may have captured a large amount of weak evidence of features in our memory, so that with a new sample as input, the memory relaxes to a state that satisfies as many constraints as possible to the maximum extent.

Constraint satisfaction model

The above idea of constraint satisfaction can be captured in a PDP model consisting of several units and connections among the units. In this model the units represent hypotheses and the connections represent the knowledge in the form of constraints between any two hypotheses. It is obvious that the knowledge cannot be precise and hence the representation of the knowledge in the form of constraints may not also be precise. So the solution being sought is to simultaneously satisfy as many constraints as possible. Note that the constraints usually are weak constraints, and hence all of them need not be satisfied fully as in the normal constrained optimization problems. The degree of satisfaction is evaluated using a goodness-of-fit function, defined in terms of the output values of the units as well as the weights on the connections between units. The model we have is already trained and has weights assigned to it. It is these weights that aid in making the original hinton diagram. We can further train the network by providing our input on which descriptor best fits which room and update the weights accordingly. These new weights are what aid in the making of the new Hinton Diagram.

The clamping of descriptors is like an external input given to the network. When we do so and test the network, after a few cycles, we see the descriptors belonging to the room type of the descriptor that was clamped to be lit up whereas others are not. Example of capturing the concept of the type of room from descriptors

The constraint satisfaction PDP model is illustrated here with an example of how our concepts of various types of rooms can be captured by the model from samples of description of these rooms. Let us assume that we collect data from subjects about their understanding of the following five types of rooms: Living room, kitchen, bedroom, office and bathroom. In order to elicit information from the subjects, 40 descriptors are provided to them, in terms of which each subject can be asked to give his/her view of the above room types. The descriptors are shown in the following figure

Ceiling	Walls	Doors	Windows	Very-large
Large	Medium	Small	Very-small	Desk
Telephone	bed	Typewriter	Book shelf	Carpet
Books	Desk-chair	Clock	Picture	Floor-lamp
Sofa	Easy-chair	Coffee-cup	Ashtray	Fireplace
Drapes	Stove	Coffeepot	Refrigerator	Toaster
Cup board	Sink	Dresser	Television	Bathtub
Toilet	Scale	Wen	Computer	Clothes-hanger

Figure 1: *Table showing the descriptors used in the example.*

Each subject can be asked to mark which of these descriptors are valid for each room type. From the data collected from a number of subjects, the weights between units are computed. Here the units represent the descriptors. The output of a unit is binary, indicating whether the description is present or not. The connection weights between units are derived from the co-occurrence patterns of the descriptors in the responses of the subjects for all the room types. One method of deriving the weights is as follows:

$$w_{ij} = -\log\left(\frac{P(x_i = 0 \text{ and } x_j = 1)\cdot P(x_i = 1 \text{ and } x_j = 0)}{P(x_i = 0 \text{ and } x_j = 0)\cdot P(x_i = 1 \text{ and } x_j = 1)}\right)$$

where (w_{ij}) represents the symmetric weight connecting the unit (j) and unit (i). The numerator represents the product of probabilities that the hypotheses of the units i and j are competing with each other, i.e., one unit has the value (x_i) = 1 and the other has the value (x_j) = 0. The denominator represents the product of probabilities that the hypotheses of units i and j support each other. Thus if the evidence is greater for supporting hypotheses, then the weight (w_{ij}) will be positive, otherwise it is negative. Note that the probabilities are replaced by the relative frequencies in the data. In addition, each unit can have a bias reflecting the prior information about the hypothesis the unit represents. In this case the bias is given by b_i = -log [P(x_i=0) / P(x_i=1)]

There can be direct evidence for a hypothesis through an external input. The corresponding input unit could be clamped indicating that the hypothesis is either always 'ON' or always 'OFF'. Other types of external input could be a graded one indicating weak evidence.

A constraint satisfaction model can be displayed by means of a Hinton diagram as shown in the following figure. Each larger square in the figure represents a unit. There are 40 units corresponding to 40 descriptors. Within the square of each unit a replica of all the 40 units are displayed as dots, each dot representing the unit in its relative position in the diagram. Around each dot, a white square indicates a positive weight connecting the unit representing the dot and the unit enclosing the dot. Thus, for example, the white square on the second dot in the unit for 'ceiling' indicates that the 'walls' unit is connected to the ceiling unit with a positive weight. The size of the white square indicates the strength of the positive connection. Likewise in the last unit corresponding to 'oven', the small dark square around the last but one dot indicates that the units 'oven' and 'computer' are connected with a negative weight. There are many units which have no connections at all.

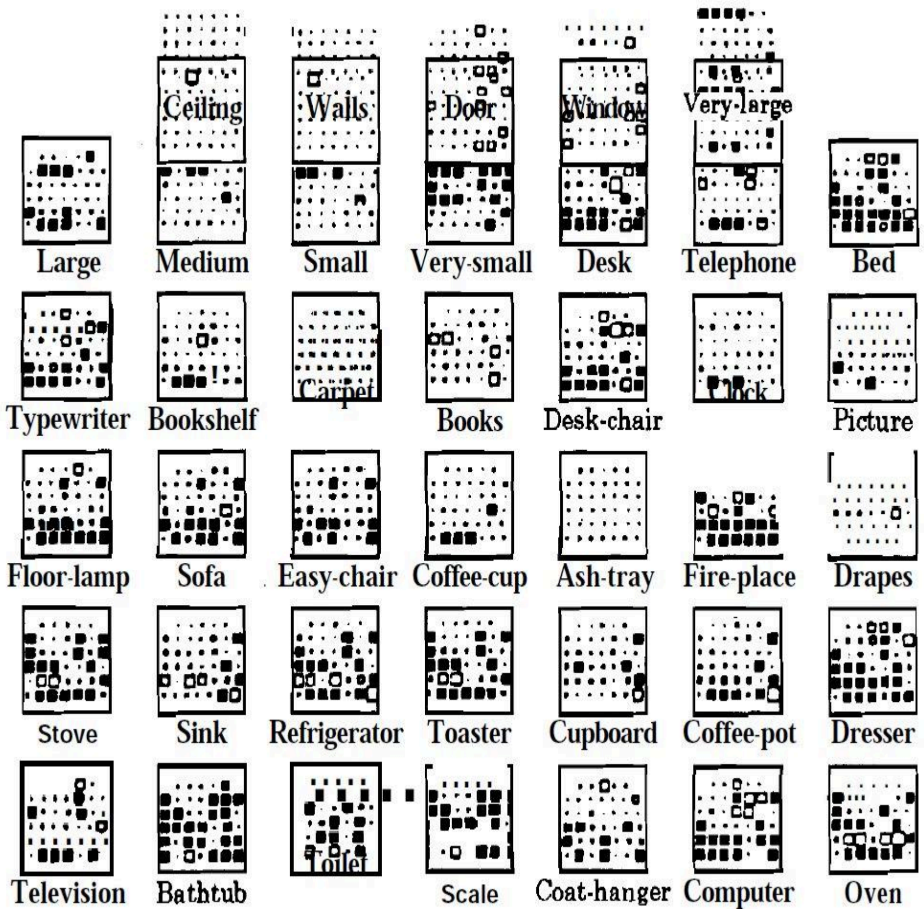


Figure 2: *Hinton diagram for the 'rooms' example.*

The model is allowed to relax by computing the next state for each unit selected at random, computing the sum of its weighted inputs and thresholding the weighted sum using a hard-limiting output function. For a given external evidence, say like 'oven' and 'ceiling' in the following figure, the state of the network after each cycle is shown in the figure. After 17 cycles the model settles down to an equilibrium state closest to the given external evidence, and the state description gives a description of the concept of the room satisfying the external evidence, namely 'kitchen', in this case. Thus the PDP model clearly demonstrates the concepts of rooms captured by the weak constraints derived from the data given by the subjects. The model captures the concepts of the five room types at the equilibrium states corresponding to the description that best fits each room type. A goodness-of-fit function (g) is defined for each state ((x_1,x_2, ...,x_N)) , where (x_i) = 1 or 0, as

g = Σ_{i,j=1}^{N} w_ij x_i x_j + Σ_{i=1}^{N} e_i x_i + Σ_{i=1}^{N} b_i x_i

$$g = \sum_{i,j=1}^{N} w_{ij} x_i x_j + \sum_{i=1}^{N} e_i x_i + \sum_{i=1}^{N} b_i x_i$$

where (e_i) is the external input given as output of the (i^{th}) unit and (b_i) is the bias of the unit (i). At each of the equilibrium states the goodness-of-fit function is maximum. The model not only captures the concepts of the room types, but it also gives an idea of their relative separation in the 40 dimensional space.

FORMULAE USED

if the descriptor is clamped, it is given activation 1, otherwise 0. nextState[i] of a descriptor is the sum of the products of activation[j] with weights[i][j] if nextState[i] is greater than the threshold, it is given value 1, otherwise 0. if nextState[i] is 1, activation is set to 1 irrespective of its initial value. activation of each descriptor is calculated for 16 cycles one after another and displayed.

For further information, refer to the references given in the references section for this experiment.

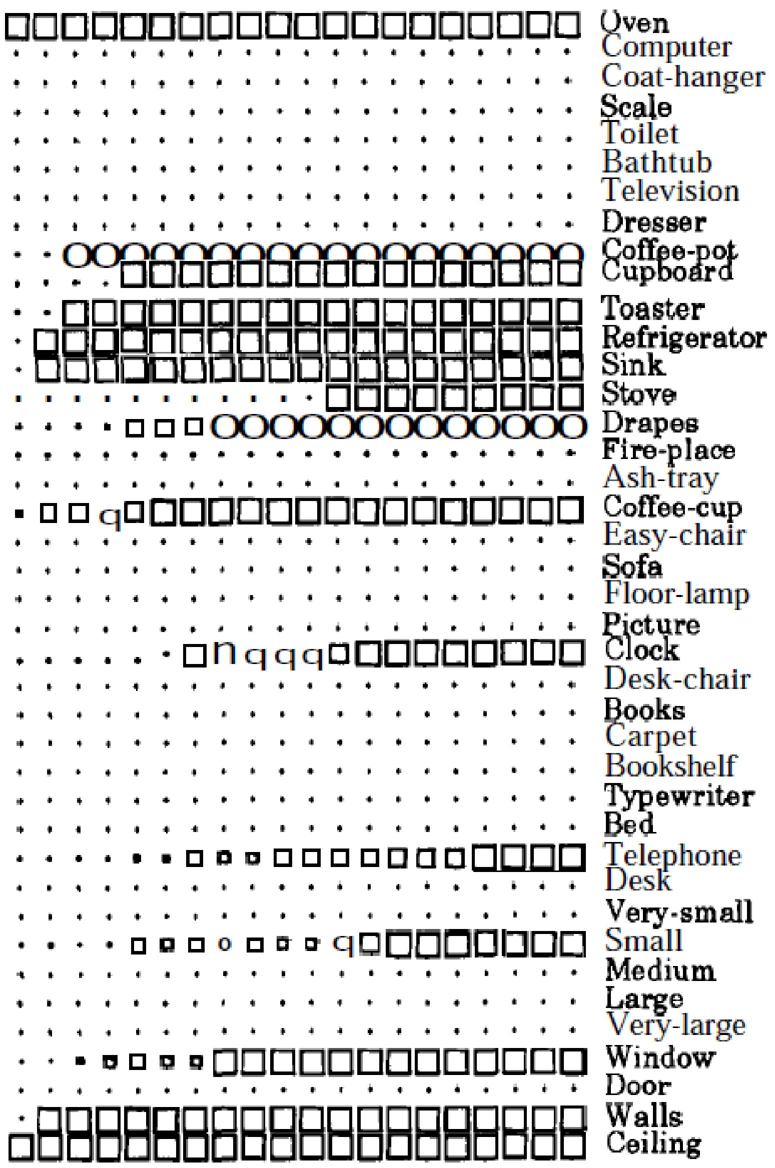
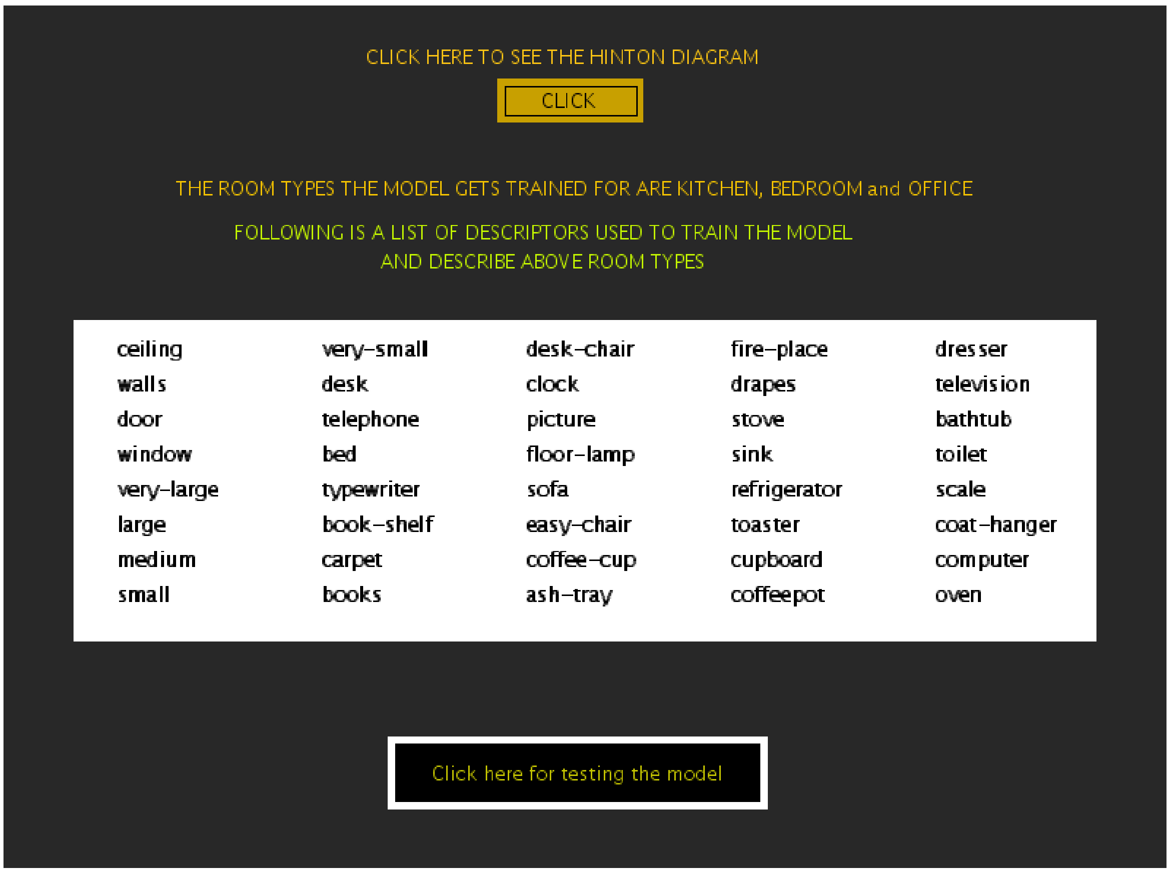


Figure 3: *The state of the CS model after each cycle, starting with an initial state where the units 'ceiling' and 'oven' are clamped.*

Illustration of Constraint Satisfaction Model

In this experiment, we use the PDP model called the Constraint Satisfaction Model, to illustrate how we as humans, attempt building of concepts as well as to arrive at conclusions given weak evidence. We start with a set of 40 descriptors used to characterize different room types. These descriptors can also be referred to as hypotheses . Also present with the model is the relationship between these descriptors, acting as constraints, which can either be weak or strong. So eventually given a set of such hypotheses and weak or strong constraints between these, we arrive at some knowledge towards a particular room type in this example. The experiment first shows all the 40 descriptors involved in capturing knowledge for a given room type.


CLICK HERE TO SEE THE HINTON DIAGRAM

CLICK

THE ROOM TYPES THE MODEL GETS TRAINED FOR ARE KITCHEN, BEDROOM and OFFICE

FOLLOWING IS A LIST OF DESCRIPTORS USED TO TRAIN THE MODEL
AND DESCRIBE ABOVE ROOM TYPES

ceiling	very-small	desk-chair	fire-place	dresser
walls	desk	clock	drapes	television
door	telephone	picture	stove	bathtub
window	bed	floor-lamp	sink	toilet
very-large	typewriter	sofa	refrigerator	scale
large	book-shelf	easy-chair	toaster	coat-hanger
medium	carpet	coffee-cup	cupboard	computer
small	books	ash-tray	coffeepot	oven

Click here for testing the model

Figure 1: *Experiment window showing descriptors and other buttons.*

The constraint satisfaction model which is illustrated in this experiment is already trained. We can view the model using Hinton Diagram by clicking on the "CLICK" button on top. A new panel showing the Hinton Diagram for all the descriptors appears where 40 hypotheses have their corresponding squares with constraints between them and other hypotheses being displayed within the larger square. Moving the mouse over the squares presents their zoomed picture at the bottom of the page.

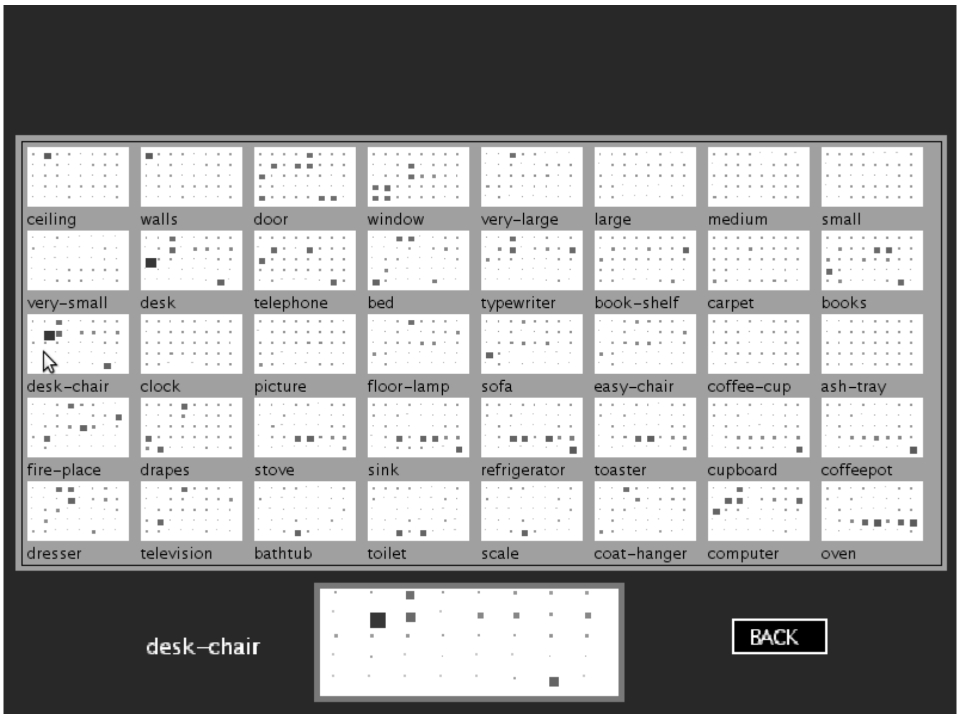
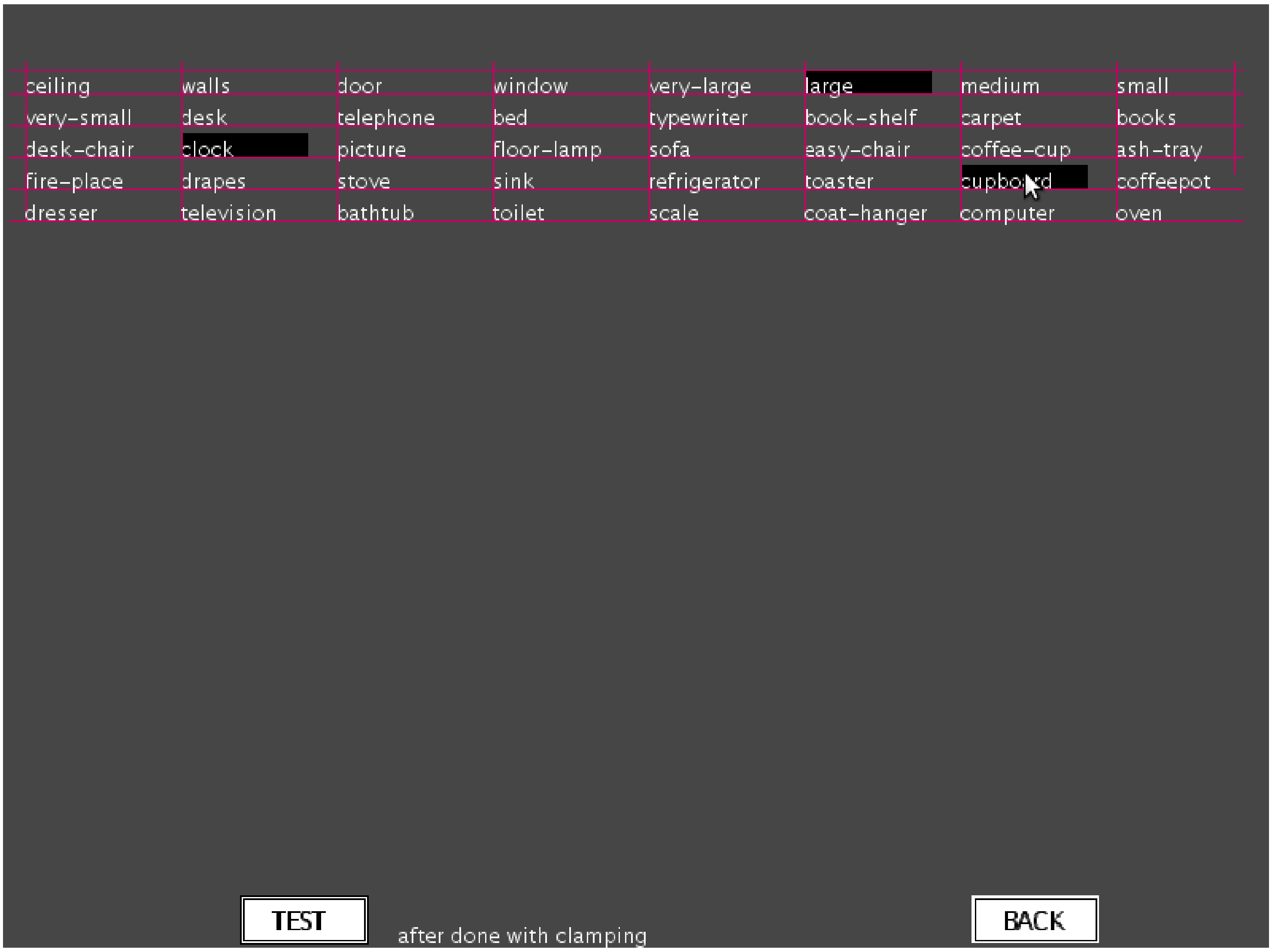


Figure 2: *Experiment panel showing the Hinton diagram for the descriptors.*

In the above diagram we notice that each large square represents a unit. There are 40 units corresponding to 40 descriptors. Within the square of each unit a replica of all the 40 units are displayed as smaller squares or dots, each dot representing the unit in its relative position in the diagram. Around each dot, weak constraints are depicted by smaller squares in lighter gray shade. Similarly strong constraints are represented by darker gray or black color with a square of larger size. There are many units which have no connections at all.

The constraint satisfaction model is used to illustrate the ability of human beings to build concepts or arrive at conclusions based on partial and sometimes erroneous knowledge. The key idea of the model is that a large number of weak constraints when put together, evolve into a definitive conclusion. To test the model, some units can be clamped or say provided a constant external bias. As shown in the figure below, we can click on the units and have these clamped.



ceiling	walls	door	window	very-large	large	medium	small
very-small	desk	telephone	bed	typewriter	book-shelf	carpet	books
desk-chair	clock	picture	floor-lamp	sofa	easy-chair	coffee-cup	ash-tray
fire-place	drapes	stove	sink	refrigerator	toaster	cupboard	coffeepot
dresser	television	bathtub	toilet	scale	coat-hanger	computer	oven

Figure 3: *Testing of the model by clamping some units.*

After clamping the model is allowed to relax by computing the next state for each unit selected at random. The state computation is done by computing the sum of the weighted inputs and thresholding the weighted sum using a hard-limiting output function. After we click the test network button, we find that after running through some iterations the network settles to a state where no further change in the state of descriptors is visible. And then the state description gives the description of the concept of the room satisfying the external evidence or clamping

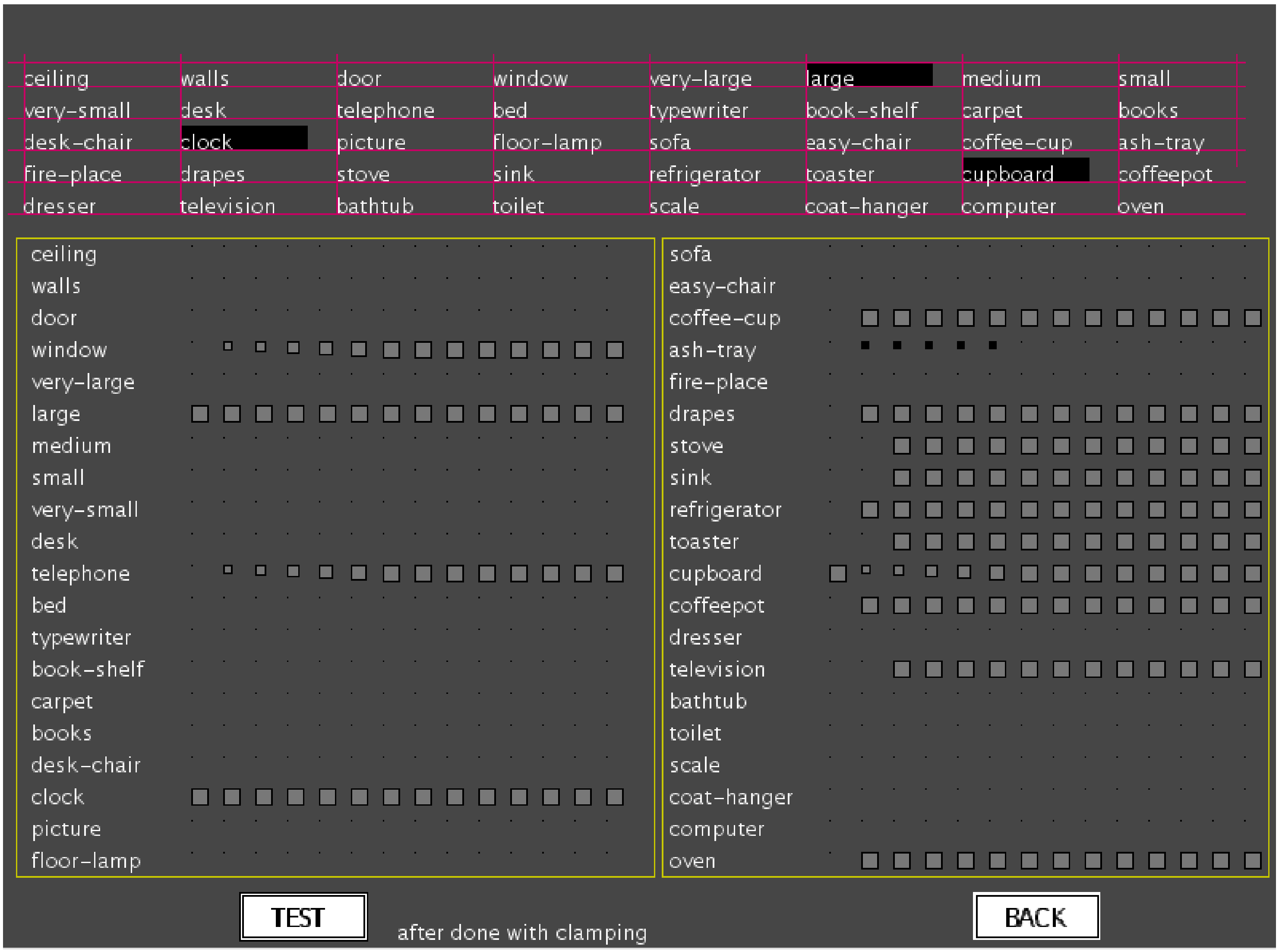


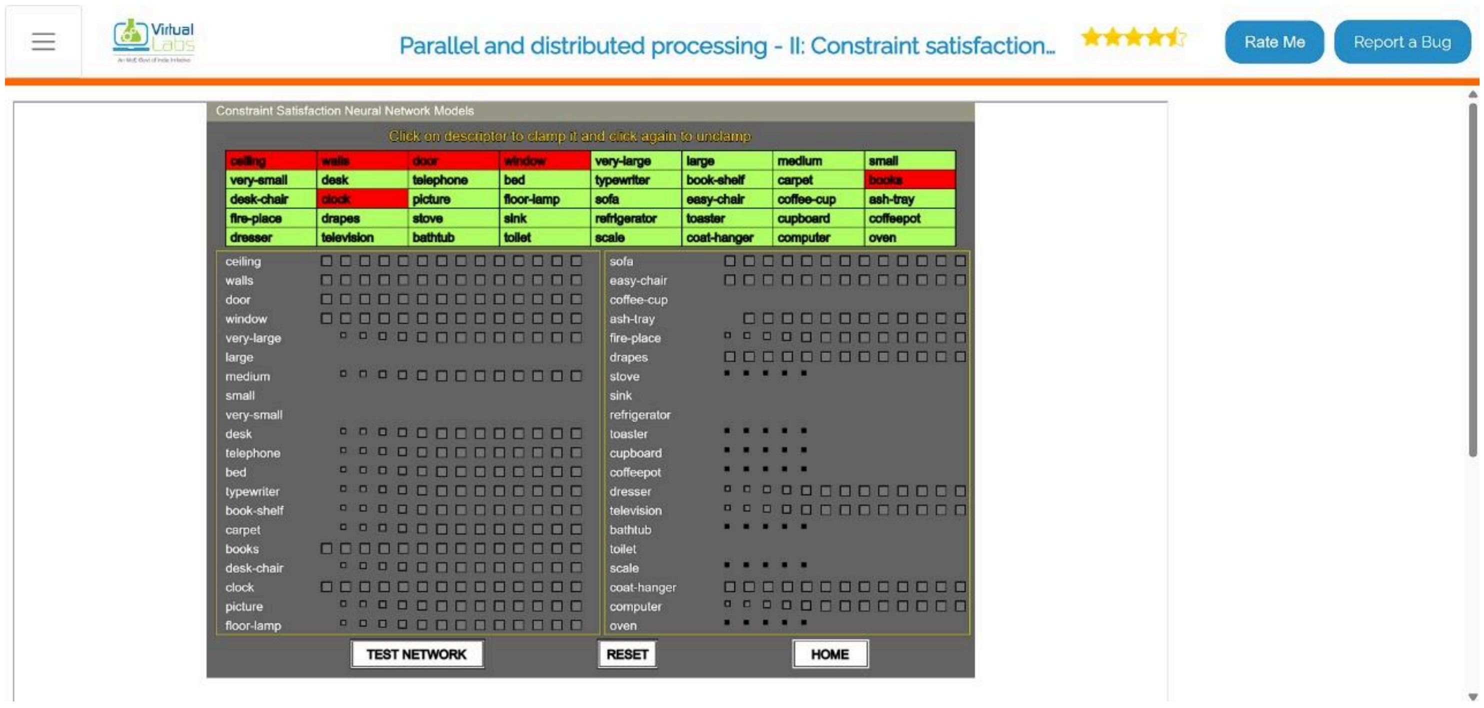
Figure 4: *Figure showing the states of descriptors after reaching equilibrium.*

PROCEDURE:

1. Click on the "click" button below "click here to see Hinton Diagrams".
2. Click on "Click here to see the original hinton diagram with preset weights" to load the hinton diagram for the already trained network with current weights.
3. Hover your mouse over any of the rectangles for units to see its hinton diagram in zoomed up version towards the bottom of the canvas.
4. Click on the "back" button to go back to the menu for more choices.
5. Click on the "Click here to further train the model".
6. Click on any room choice to select descriptors for that room.
7. Click on any descriptors you wish to attribute to the room choice that you made.
8. After making at least one selection of descriptors for each room type, click on "train model and show Hinton Diagram".
9. Hover your mouse over any of the rectangles for units to see its new hinton diagram in zoomed up version towards the bottom of the canvas.
10. Click on the "back" button to go back to the page to select the room and descriptors for that room.
11. Click on the "reset" button to reset descriptors and room choices.
12. Click on the "back" button to go back to the menu for more choices.
13. Click on the "home" button to go back to the main menu.
14. Click on the "click here for clamping descriptors"
15. Click on at least one descriptor to clamp it.

16. Click on the descriptor again to unclamp it.
17. Click on "test network" to run 16 cycles of the network.
18. Click on a descriptor to clamp it and simultaneously see the change in the network.
19. Click on the "reset" button to reset the clamping.
20. Click on "Home" button to go back to the main menu

SIMULATION/MODEL:



ceiling	walls	door	window	very-large	large	medium	small
very-small	desk	telephone	bed	typewriter	book-shelf	carpet	books
desk-chair	clock	picture	floor-lamp	sofa	easy-chair	coffee-cup	ash-tray
fire-place	drapes	stove	sink	refrigerator	toaster	cupboard	coffeepot
dresser	television	bathtub	toilet	scale	coat-hanger	computer	oven

Results:
Distributed Processing Efficiency:
MPI-based parallelization (e.g., for matrix multiplication or Monte Carlo simulations) showed faster execution compared to single-node processing as the problem size scaled.

Example: For a large dataset (e.g., 10^6 data points), the sequential approach took X seconds, while distributed processing across 4 nodes took Y seconds, achieving a speedup of Z%.

Communication Overhead:
Increasing the number of nodes reduced computation time but introduced communication latency (e.g., during MPI_Send/MPI_Recv operations).
Optimal performance was observed at N nodes, beyond which overhead outweighed benefits.

Fault Tolerance:
Checkpointing (if implemented) ensured task resumption during node failures, though at a minor performance cost.

Conclusion:
Distributed computing (using MPI) significantly accelerates large-scale computations by dividing workloads across multiple machines.

Key Challenges:
Synchronization: Non-uniform delays due to network latency.
Scalability: Performance plateaus as communication overhead dominates.

Proper workload partitioning (e.g., block-cyclic distribution for matrices) and minimal data exchange are critical for efficiency.

Applications:
High-Performance Computing (HPC):
Weather forecasting, fluid dynamics simulations (e.g., using MPI in supercomputers).

Distributed Machine Learning:
Training models on large datasets (e.g., TensorFlow/PyTorch with Horovod).

Blockchain Networks:
Consensus algorithms (e.g., PoW/PoS) rely on distributed processing.

Edge Computing:
Decentralized processing for IoT devices (e.g., smart cities).