Stable-yet-Switchable (SyS) Attractor Networks

Subramoniam Perumal and Ali A. Minai

Abstract: **Recurrent neural networks functioning as associative memories are often studied and optimized for recall quality and capacity, with the focus primarily on the network's *stability*, i.e., convergence to stored attractors. However, the ability of networks to switch between attractors in a controlled way is also potentially a useful phenomenon. Networks that are stable under most conditions, but can be switched by specific stimuli may be used to model cognitive control and other time-varying cognitive phenomena. Such networks, which we term *stable-yet-switchable* (*SyS*) *networks*, are also of interest from the networks perspective, and the SyS properties of scale-free networks have been noted by researchers. In this paper, we consider networks with bimodal connectivity – a *core* of densely connected neurons and a larger *periphery* with sparser connectivity – and compare their SyS performance with random and scale-free recurrent neural networks. The results show that core-periphery networks have much better SyS performance than scale-free networks.**

I. Introduction

Recurrent neural networks with embedded attractors are widely used as models of associative memory [1], [2]. The attractors stored in the networks are meaningful patterns of neural activity (memories), which are recalled when the network is allowed to relax from initial states similar to these patterns, resulting in recall, pattern completion, noise reduction, etc. The primary concerns in these networks are convergence to the stored attractors from nearby initial states and the stability of these stored attractors. Relatively little attention has been focused on the *destabilization* of attractors. However, from a functional perspective, a network's ability to switch out of its current attractive state to another is of great significance. This is especially true in cases where attractor networks are used to model cognitive functions, many of which involve switching between functional states [3]–[6]. In such networks, each stable state represents a functional context. The network remains in the state corresponding to the appropriate context regardless of external stimuli, and switches to another stable state only when a stimulus or internally generated signal indicates a change of context. We term such behavior *stable-yet-switchable* (*SyS*).

Figure 1 shows a simple instance where SyS behavior can be useful. The processing network, P, responds to incoming stimuli in a context-specific way (i.e., the same stimulus can produce different responses in different contexts). The key point is that context information is only transiently available but must influence P over an arbitrarily long period until the next context switch, i.e., it must be latched [7]. This eliminates the use of approaches based on tapped delay lines or decaying memory [8]–[12]. In the system shown here, the Latching Network, L, performs this function. It recognizes a context based on external stimulus and the current state of P and switches to an attractor representing this context, thereby projecting a context-specific bias on P and influencing the flow of information through that system. Thereafter, the attractor in L remains stable until a new context is recognized, which triggers another switch. Thus, L must have SyS behavior: It should be able to monitor the stimulus and the state of P without switching in most cases, but switch reliably when these indicate a new context.

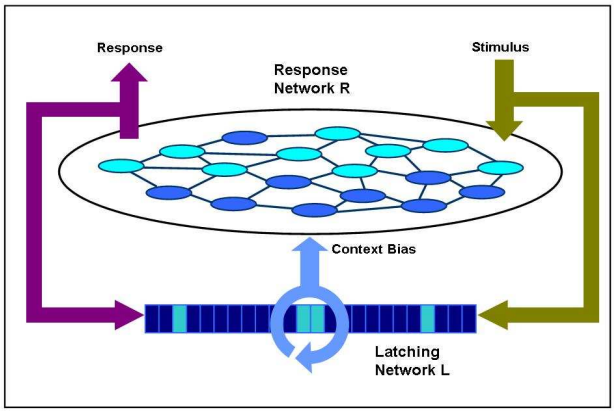

Fig. 1. A simple architecture for context-sensitive neural processing. The P network generates responses to incoming stimuli through complex recurrent dynamics, while the L network detects changes of context and latches a biasing signal for the duration of each context using attractors. The L network is SyS.

Previous work in our lab has shown that the approach shown in Figure 1 allows simple attractor networks using Hebbian learning to perform complex temporal tasks with context dependence [13]–[16], and may explain how context-dependent place representations arise in the rodent hippocampus [13], [17]. Subsequently, we have speculated that this may be a general mechanism underlying context-dependent functionality in the cortex [16], [18], [19], as well as the regulation of gene expression during development [20]. Such ideas are also implicit in models of behavioral control based on switching mediated by the basal ganglia

This work was supported by collaborative National Science Foundation Human and Social Dynamics Program grants to Ali Minai (BCS-0728413), which includes support from the Deputy Director of National Intelligence.

Subramoniam Perumal is with the Department of Computer Science, University of Cincinnati, Cincinati, OH 45221-0030, USA (email: subramoniam.p@gmail.com).

Ali A. Minai (corresponding author) is with the Department of Electrical & Computer Engineering, University of Cincinnati, Cincinati, OH 45221-0030, USA (phone: 513-556-4783; fax: 513-556-7326; e-mail: Ali.Minai@uc.edu).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

[21]–[26]. In the current paper, we focus on the SyS functionality *per se*, and propose a simple method for obtaining it in recurrent neural networks. Such networks could form important components of larger cognitive architectures. We only consider networks with bipolar neurons, but networks of real-valued neurons with the possibility of chaos are potentially also very interesting.

II. Inherent SyS Behavior

The finite-state machine formalism has generally been used to model controlled switching in neural networks [27], [28], [12], and has been proposed as the basis of cognitive control [27], [29], [30]. However, such models typically use error-driven learning algorithms, which require long learning periods and many iterations – both of which are biologically implausible. It is thus interesting to consider whether certain classes of simple recurrent networks are inherently capable of producing SyS behavior based on a simple prescription.

One class of networks where SyS behavior is well-known and well-studied are scale-free networks [31]. These networks are characterized by a power-law distribution of connectivity, i.e.,

$$p_k \sim k^{-\alpha} \qquad (1)$$

where p_k is the probability that a node has k connections and α is a parameter, typically between 2 and 3. In networks with independent random connections between nodes, the degree distribution is expected to be Poisson [32], which has an exponentially decaying tail, implying that most nodes have a similar "typical" connectivity. Almost all models of recurrent neural networks have followed this prescription, with a few recent exceptions [33]–[37]. The power-law connectivity distribution with its lowly decaying tail ("fat tail") implies that a significant number of nodes have exceptionally high connectivity, and serve as hub nodes, binding the network together. It has been shown that scale-free networks (also called power-law networks) are very robust to random disruptions of their nodes because of the strong integrity provided by the hubs, but are exceptionally sensitive to disruptions of the hubs [38]. Recently, Bar-Yam and Epstein [39] investigated the behavior of attractor networks with scale-free connectivity, and found that perturbations applied to the networks' hub nodes were far more disruptive than those applied to other nodes. In other words, they found SyS behavior, which is a particular case of the "robust-yet-fragile" characteristic seen in most complex systems [40].

The results of [39] suggest that scale-free networks can be used as the SyS block in the architecture of Figure 1. However, a natural question to ask is whether there is some architecture even better suited to provide SyS functionality. Based on logical considerations, we hypothesized that networks with a bimodal connectivity distribution represented such an architecture. These networks – termed *core-periphery* (*CP*) *networks* – have a relatively small subset of densely connected nodes, called the core, with the remaining nodes forming a sparsely connected periphery. Networks with bimodal degree distributions have been investigated for robustness, and found to be more robust than scale-free networks [41], [42]. In this paper, we report on a systematic comparison of random, scale-free and CP networks with respect to the SyS property, which is defined as follows:

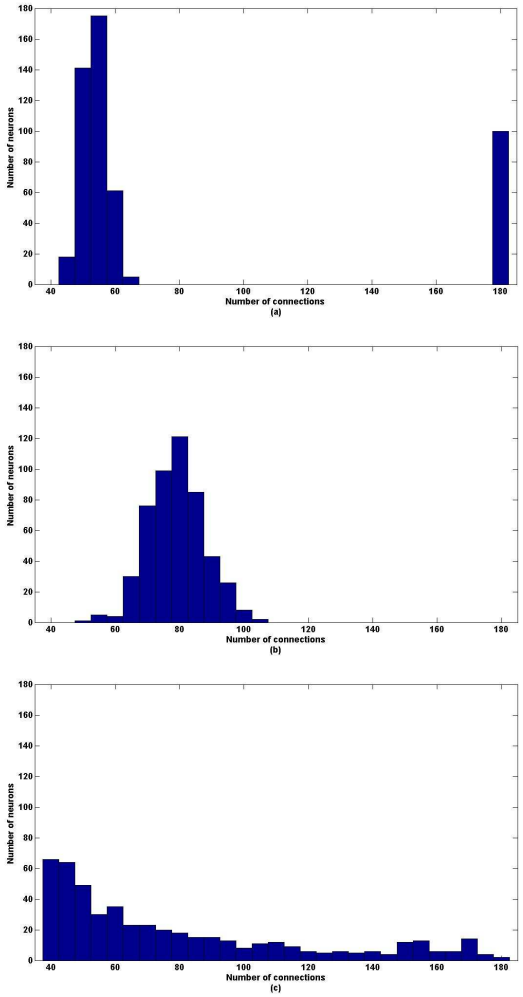

Fig. 2. Connectivity distributions in the three network models: (a) CP network; (b) Random network; (c) Scale-free network. All networks have approximately the same number of total connections.

Given a N-node recurrent network with M stored attractors, if the network has converged to any of its stored attractors:

1. ***The network should be robust***, i.e., almost all state perturbations up to size $m < N$ should leave the network in the same attractor.
2. ***The network should be switchable***, i.e., a few specific perturbations *of the same size* should switch it reliably to other stored attractors.

3. ***The switching should be targetable***, i.e., the same switching perturbation should always lead to the same attractor independent of network state.

Strictly speaking, condition 3 is not necessary for the SyS property, but is essential for it to be useful.

III. Models and Methods

A. The network model

The Core-Periphery (CP) network consists of a small core sub-network, and a much larger periphery sub-network. Each core neuron is connected to every other core neuron, and also to some periphery neurons. Each periphery neuron is very sparsely connected to other periphery neurons, and to a few core neurons. The connections between any two neurons *i* and *j* are symmetric i.e. $w_{ij} = w_{ji}$, where w_{ij} is the weight from neuron *j* to neuron *i*.

The result is a sparse network with a bimodal distribution of node degree. The hypothesis is that these networks would provide the needed stable-yet-switchable functionality: perturbations targeted at the highly-connected core neurons will cause switching while random perturbations that are spread across the whole network would not destabilize the current attractor. In this report, we use 500-neuron networks, with 100 neurons in the core and 400 in the periphery. The connectivity levels are: core-core: 100%; periphery-periphery: 8.5%; core-periphery: 20%.

The state of each neuron can either be 1 or -1, and all neurons are updated synchronously. This means that the network can sometimes converge to a period-2 cycle instead of a fixed points, but the results are qualitatively similar to those obtained for the asynchronous update case, which is more complicated to simulate. A threshold activation function is used to determine the activation state of neurons, and is given by:

$$f(x_i(t)) = \begin{cases} 1, & x_i(t) \geq 0 \\ -1, & x_i(t) < 0 \end{cases} \qquad (2)$$

where $x_i(t)$ is the net input to neuron *i* at time *t* given by:

$$x_i(t) = \sum_{j=1}^{N} W_{ij} f(x_j(t-1)) \qquad (3)$$

where $f(x_j(t-1))$ is the activation function of neuron *j* at time *t-1*, *N* is the number of neurons, W_{ij} is the weight from neuron *j* to neuron *i*.

B. Attractor patterns

Homogeneous, unbiased attractor patterns are used for all simulations. Each neuron has equal probability of being active i.e., having an activation state of 1, in each attractor, and the number of 1s and -1s in a pattern are equal. Since the core comprises the significant neurons responsible for switching, care is taken to see that each core neuron is active in approximately the same number of attractors to achieve parity of significance among core neurons. In the periphery, the neurons are chosen randomly for activation. Storing orthogonal attractor patterns reduces the effect of spurious attractors, so the patterns are made as orthogonal as possible.

C. Learning rule

A Hebbian learning rule is used to learn the attractor patterns by setting the weights between neurons. Mathematically the rule is given by:

$$W_{ij} = \frac{1}{N} \sum_{k=1}^{p} \delta_i^k \delta_j^k , \qquad (4)$$

where W_{ij} is the weight from neuron *j* to neuron *i*, *N* is the number of neurons, *p* is the number of stored patterns, and δ_i^k is the activation of neuron *i* for pattern *k*.

D. Network dynamics and output

With synchronous update, a network started from an initial condition could settle into one of four situations: 1) a stored attractor close to the initial state of the network; 2) a distant stored attractor; 3) a spurious attractor; or 4) a period-2 cyclic attractor. We allow 50 time steps for the network to converge after each perturbation, which defines a single *cycle* of the network..

Upon convergence, the final state of the network is determined by comparing the network output to each of the stored attractors using overlap metric given by:

$$\varphi(z,y) = \frac{1}{N}(z.y) \qquad (5)$$

where *z* is a stored pattern and *y* is the network's final state.

The overlap is a number between -1 and 1, with value 1 if both patterns are same; 0 if the number of similar bits is equal to the number of dissimilar bits; and -1 if all bits are dissimilar. It must be noted that since the stored attractors are approximately orthogonal to each other the overlap measure between any two stored attractors is approximately 0. Since small recall errors of a few bits are unavoidable with sparse-connectivity [1], an overlap of 0.94 or above is considered to indicate sufficient recall. This means that, for a network of size 500 neurons, an error of up to 15 bits is considered acceptable.

IV. Network Parameters

Functionally, the core represents the sensitive part of the network, while the periphery is the insensitive part. Perturbations targeted at the core have network-wide impact and cause switching, while those spread across the whole network primarily affect the periphery and do not have much impact overall, leaving the network stable.

Fig. 3 shows the stability of a 500 neuron (100 core, 400 periphery) network to perturbations of different sizes. The network has 10 stored attractors. Three cases are shown: a) Perturbation directed only at the core; b) Perturbation directed only at the periphery; and c) Perturbation spread uniformly over the entire network.

Figure 3(a) shows that perturbing slightly more than 50 neurons in the core (i.e., half the core) is enough to destabilize the attractor in some cases, and perturbing the entire core guarantees such destabilization. In contrast, Figure 3(c) shows that a perturbation of more than 175 neurons is needed to produce any destabilization for network-wide perturbations, and a 250-neuron perturbation is needed to guarantee switching. Figure 3(b) shows that the periphery by itself is even more stable. Thus, the CP network clearly shows the first two criteria for SyS behavior, as listed in Section II. The issue of targeting will be addressed below.

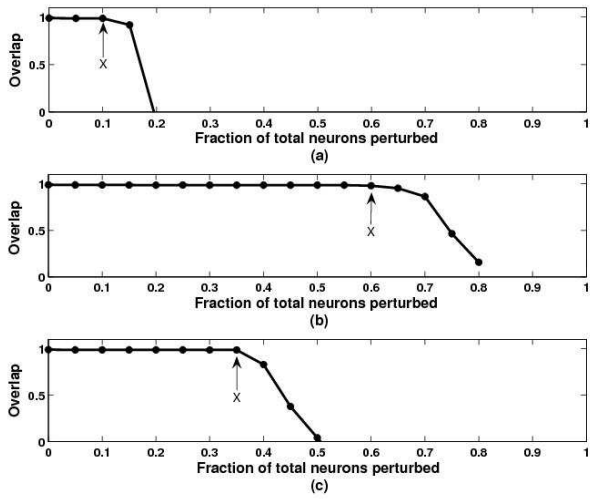

Fig. 3. Stable-yet-switchable behavior in CP networks. For each trial, the network is started in a stored attractor and a fraction of neurons is perturbed (flipped to complementary values). The figure shows the overlap of the network's final state with the initial attractor. The graphs are: (a) perturbation is applied only to the core; (b) perturbation is applied only to the periphery; (c) perturbation applied randomly across the network. X indicates the point where the network starts to destabilize. Each data point is averaged over 100 trials – ten with reach stored attractor as the initial state.

V. Comparison with Other Network Models

A. Comparison with Scale-free (SF) networks

Since SyS behavior has previously been investigated in scale-free networks [39], we compare it with that seen in the equivalent CP network. The number of connections for the CP network and SF network are kept approximately equal. Since the CP network core size is 100, the 100 most connected neurons of SF network are considered its core. The switching performances of Core-Periphery and Scale-free networks are compared in Figure 4 for random perturbations directed only at the core. It can be seen that even for perturbation of the entire core (0.2 on the x-axis) SF networks do not switch reliably, and larger perturbations are needed to guarantee switching. In contrast, CP networks switch reliably for perturbations of slightly less than the size of the core. These results are not unexpected because the core (defined as the 100 most connected neurons) accounts for many more connections in CP networks than in SF networks.

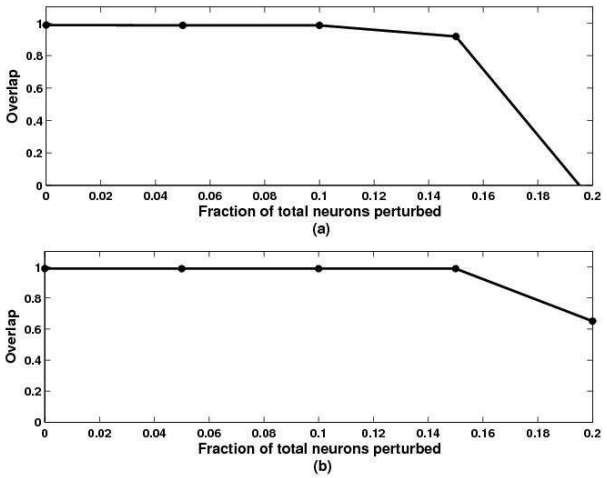

Fig. 4. Comparison of stability between CP and Scale-free networks for core-only perturbations. The *x-axis* represents the fraction of total neurons perturbed, and the *y-axis* the average overlap of final network state with the initial attractor. (a) CP network; (b) Scale-free network. Each data point is averaged over 100 trials.

Figure 5 shows more detailed results on what happens when core neurons in SF and CP networks are perturbed randomly. The networks are started in each of the 10 stored attractors and 1000 random core perturbations are applied in each case, giving a total of 10,000 trials. The state of the network is then checked after 50 iterations. Figure 5(a) shows the percentage of trials that settled in stored attractors other than the initial one. The first important observation is that this number is rather small for both networks, indicating that random perturbations – even focused in the core – do not usually cause switches to other stored attractors. This means that the perturbations needed for reliable targeted switching between stored attractors will require careful construction through learning or heuristic specification (see below). The second important observation is that CP networks are far more amenable to desirable switching than SF networks, which makes the former more promising as SyS networks. The primary reason for this disparity is shown by Figure 5(b): In the SF networks, most random core perturbations of size 85 or 90 (out of 100 core neurons) actually fail to destabilize the initial attractor. Larger perturbations do destabilize it, but much less so than for CP networks, where even a perturbation of size 80 causes switching 80% of the time, and in all cases for perturbations of size 95 or 100. This confirms the results shown in Figure 4, and again shows that CP networks are more switchable than SF networks. Figure 5(c) shows the percentage of trials where the networks switched to spurious attractors or to period 2 cycles (because of synchronous update). Here,

another interesting pattern is seen. While the CP network has a large number of such undesirable switches for all perturbation levels, the SF network has much fewer at the 85 and 90 neuron levels. However, this is only because the latter networks are not switching at all, and when they do switch in response to larger perturbations, the fraction of undesirable switches becomes similar to that for CP networks. Finally, Figure 5(d) shows the number of possible switches to stored attractors that are actually seen in the 10,000 trials. 9 such switches are possible from each initial attractors, giving a total of 90. The fraction of these that are actually seen is rather low for both networks, though far lower for SF networks. This means that the switches shown in Figure 5(a) are, in fact, not broadly spread out, but go only to a small fraction of stored attractors, further reinforcing the need for an efficient mechanism that can target all attractors from all others. Overall, the data in Figure 5 shows that, while both CP and SF networks show only a limited degree of desirable switching in response to random core perturbations, CP networks show much more promise for useful SyS behavior. The key, then, is to leverage this promise by finding *appropriate* perturbations for all desirable switches, and to minimize undesirable ones. The latter objective has to do with reducing spurious attractors, which we do not address in this paper (see [43], [44] for possible approaches). We do propose an approach for reliable targeting, as described next.

Targeted switching comparison: The perturbations used in the results discussed above were all random, and the focus was simply on considering stability vs. switching. However, it is also important that the switching be targetable, i.e., directed at specific target attractors. For this, it is important to develop a rule for choosing specific perturbations for particular target attractors. One naturally intuitive rule is to apply perturbations that make a fraction of the core resemble the corresponding bits in the target attractor. We call this *resemblance-based targeting*. This approach was tested for CP and SF networks and the results are shown in Table I. Each of the 10 attractors was used as the initial condition, and perturbations applied to switch it to each of the other 9 stored attractors – making a total of 90 trials per simulation. The data presented is averaged over 10 such simulations. The incorrect switches can be to stored or spurious attractors, or to 2-cycles due to synchronous updating.

From Table I it can be seen that for SF networks, 76.94% of trials remained stable in the initial attractor itself, with only 4.44% cases switching successfully. For CP networks, switching success with resemblance is 42.5%. The remaining trials mostly settled in spurious/cyclic attractors (46.95%) ,with a small percentage stable in the initial attractor itself (10.55%). While these results still do not indicate sufficiently reliable switching, they are a huge improvement over those seen for random core perturbations (Figure 5). Subsequent simulations with continuous switching rather than discrete trials show similar performance in that more natural situation.

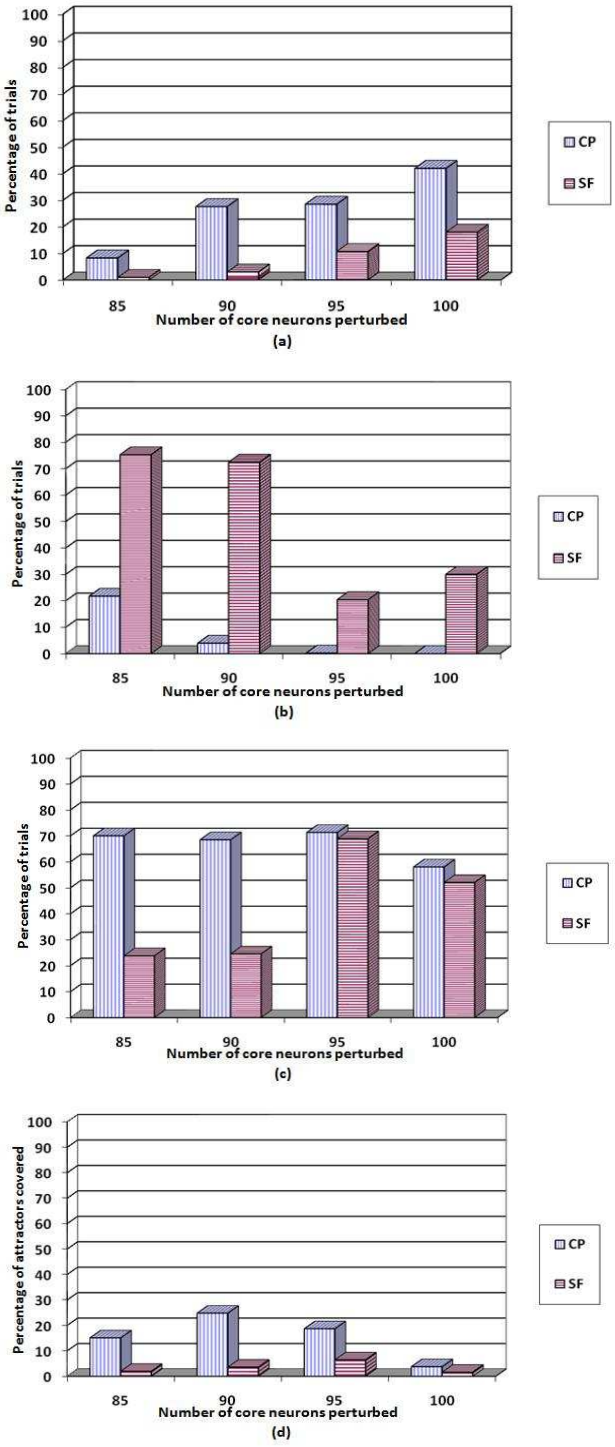

Fig. 5. Comparison of CP and SF networks. The data presented is average over 1000 trials for each of the 10 attractors with the simulation parameters obtained earlier. Subplots: (a) - percentage of trials settling in another stored attractor for varying perturbation from 85-100 core neurons; (b) - percentage of trials settling in initial stored attractor for varying perturbation from 85-100 core neurons; (c) - percentage of trials settling in spurious attractor/2-step cycle for varying perturbation from 85-100 core neurons; (d) - percentage of total attractors switches covered with varying random perturbations from 85-100 core neurons.

TABLE I
COMPARISON OF RESEMBLANCE-BASED TARGETING PERFORMANCE BETWEEN CP AND SF NETWORKS

Quantity measured	CP	SF
Percentage of trials with correct switches	42.5	4.44
Percentage of trials stable in initial attractor	10.55	76.94
Percentage of trials with incorrect switches	46.95	18.62

B. Comparison with Random networks

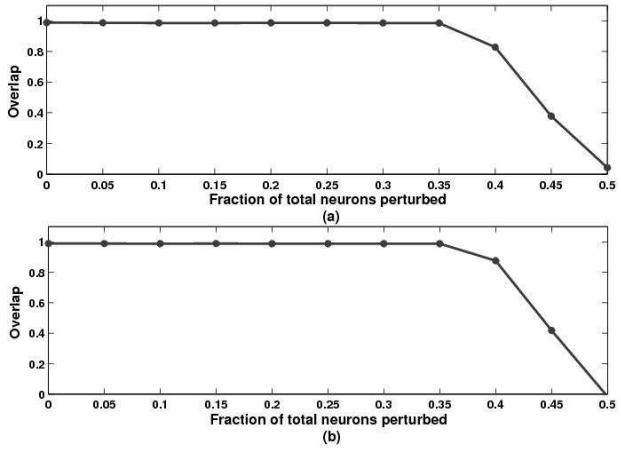

Fig. 6. Comparison between CP and Random networks for stability against network-wide random perturbations. The *X-axis* represents the fraction of total neurons perturbed, and the *y-axis* the average overlap of network outputs with the initial attractor. Graph (a) shows data for CP networks, and graph (b) for random networks.

Random networks are not really expected to show SyS behavior, but do provide a benchmark for stability in response to network-wide perturbations. For useful SyS performance, the stability of the CP network to such perturbations should be similar to that of random networks. This was tested by applying network-wide random perturbations to both CP and Random networks with approximately equal number of total connections. From Figure 6, it can be seen that both networks behave in a similar way as perturbation increases. The CP network destabilizes from the initial attractor approximately for the same perturbation as the Random network.

Fig. 7 compares the CP and Random networks for the percentage of total trials (1000 trials/attractor) that stay in the initial attractor for perturbation of 180 to 230 neurons, which is the range where destabilization begins to occur. The graph shows that Random networks are stable for a somewhat larger fraction of trials compared to CP networks, for same perturbation level, which is as expected. However, it is worth noting that this difference is small.

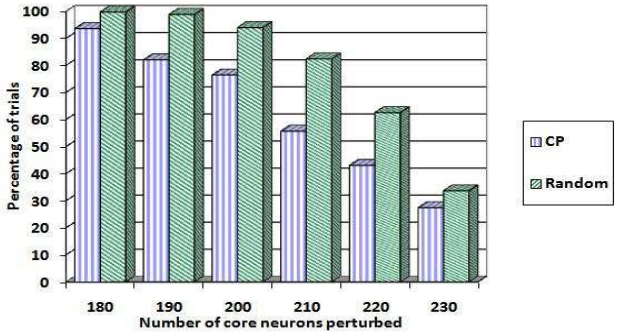

Fig. 7. Comparison of CP and Random networks showing the percentage of trials that remain stable in the initial attractor. The data presented is average over 1000 trials for each of the 10 attractors. As perturbation increases, both networks become less stable, with the CP networks slightly more so.

C. Capacity Analysis

A fully-connected Hopfield network with N neurons can store approximately $0.14N$ patterns [2]. However with sparse connectivity, the capacity of the network is expected to decrease. For this reason we stored only 10 patterns for our simulations with 500 neuron networks. Nevertheless, it is interesting to compare the capacities of the three types of networks. We do this through simulations, and the results are shown in Table II. It is clear that each of the 3 networks under study is able to store 10 attractors comfortably. However as the the number of stored patterns increases, all networks experience problems with recall. Random networks, with their homogeneous connectivity, are able to store the most patterns, followed by SF networks. CP networks show the least memory capacity. The reason for this is that the much larger periphery of the CP network (400 neurons), with its very sparse connectivity is unable to learn homogeneous patterns as easily as other two networks. As a future direction to increase capacity of CP networks, it would be interesting to study the network capacity for heterogeneous attractor patterns – where most activity (and thus learning) is concentrated in the core region.

TABLE II
CAPACITY COMPARISON BETWEEN RANDOM, CP AND SF NETWORKS

Number of attractors to be stored	Avg. actual number of attractors stored		
	Random	Core-Periphery	Scale-free
10	10	10	10
15	14.8	14.4	14.4
20	18.6	9.6	11.4
25	11.6	2.4	5.8
30	2.6	0.2	0.6

D. Continuous switching

All simulations used so far have used single trials where the network is initialized in an attractor, perturbed, and then observed until convergence (or for up to 50 steps). However, for the type of functionality described in Section I, an SyS network would operate within the flow of the larger system's dynamics, with no possibility of artificial resets. To evaluate the relative performance of the three network architectures in this situation, we simulated them under the following protocol. The network is initialized in a particular attractor and allowed to run without resetting. Every 20 steps (termed a *perturbation cycle*), it is perturbed by either a random, network-wide perturbation or by a targeted core perturbation resembling a stored attractor. The number of bits perturbed is set to 55 (though qualitatively similar results are obtained with as few as 35 bits). The expected (desired) behavior is that the random perturbations should not be able to switch the network while the targeted ones should.

Figure 8 shows the results for one network of each type, plotting the overlap of the network state with each of the 10 stored patterns over all the perturbation cycles. The time points where the targeted perturbations were applied are labeled at the top of each graph, with S_k indicating a targeting of attractor number k. For each 20-step cycle, only the last step is plotted so that the final responses to all perturbations can be seen in sequence (otherwise, a relaxation period is visible after each perturbation). As can be seen, the CP network switched properly in 8 out of 20 cases of targeted perturbation, with the remaining cases switching to spurious attractors. There were no switches to incorrect stored attractors. The random network never switched at all, since the perturbation used (80 bits) was insufficient to destabilize it. The SF network did switch in a few cases, but far less than the CP network. These results are consistent with those in Table I, and suggest that, while the CP network is far more suitable for SyS functionality than the SF model, further study is required to find truly reliable switching mechanisms. Informal experiments (not shown) suggest that cumulative rather than one-time perturbations may significantly increase reliability in all cases.

VI. Conclusion

As hypothesized, the CP network architecture showed the required SyS behavior. Its performance in switching was found to be somewhat better than SF networks. It was marginally less stable against network-wide perturbations than Random networks, but stable enough to provide good SyS functionality. With respect to targeting, CP networks responded much better to the resemblance-based approach than SF or random networks, but targeting accuracy was still about 44%. This shows that the CP architecture is a promising one, and clearly demonstrates the effect of bimodal connectivity compared to power-law connectivity on the "rosust-yet-fragile" aspect of the network.

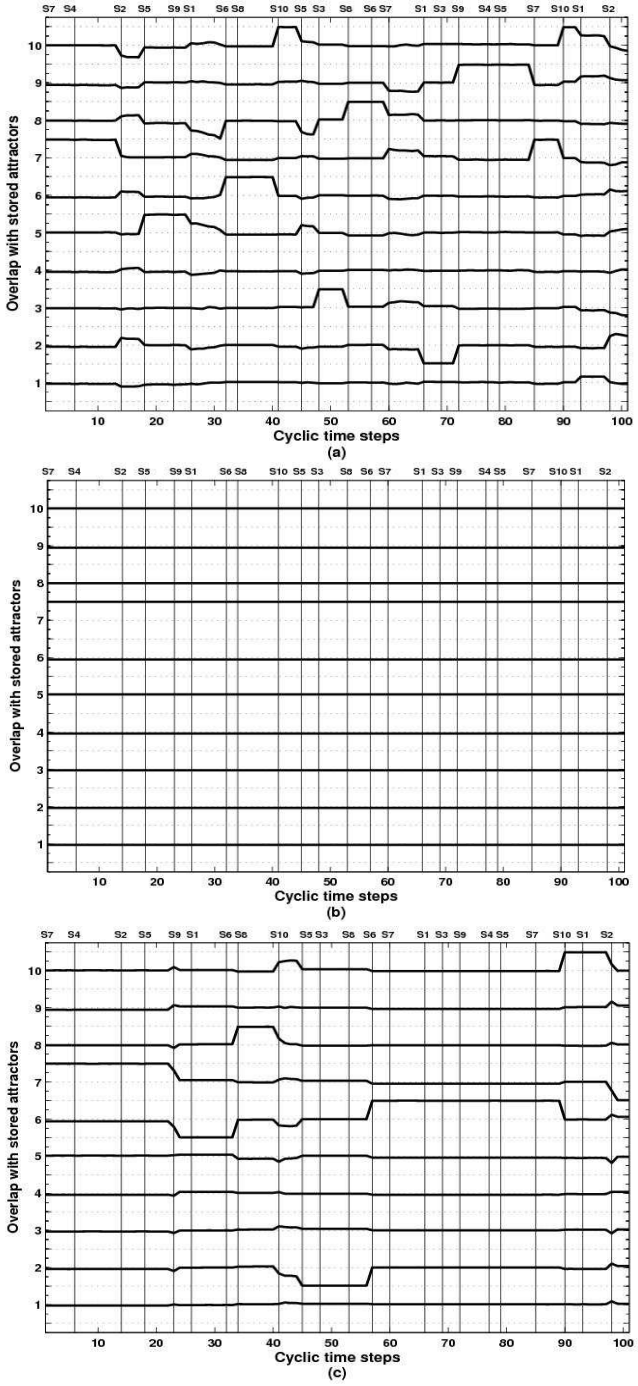

Fig. 8. Continuous-switching performance for equivalent CP, Random and SF networks. Only the last step of each perturbation cycle of 20 steps is plotted.

The CP network is limited in capacity compared to other two networks. It would be interesting to study the network behavior and capacity if heterogeneous binary attractor patterns are learned, with more activity concentrated in the core. Preliminary experiments indicate that this can improve performance dramatically.

The goal in this paper was to simply demonstrate the relative performance of CP, SF and Random networks. The CP networks simulated were not optimized with regard to core and periphery sizes or connectivities, and it may well be possible to obtain better performance.

Other interesting directions to explore are whether CP or SF SyS networks might be suitable as connectionist models of decision-making based on cumulative recurrent activation, and if modular networks with multiple cores might provide an efficient architecture for complex neural information processing.

ACKNOWLEDGMENT

This research builds on ideas developed in collaboration with Simona Doboli, whose contribution is gratefully acknowledged.

REFERENCES

[1] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, 1982.

[2] D.J. Amit, *Modeling Brain Function*. New York: Cambridge University Press, 1989.

[3] J. Fuster, *Cortex and Mind: Unifying Cognition*. New York: Oxford University Press, 2003.

[4] G.M. Edelman and G. Tononi, *A Universe of Consciousness: How Matter Becomes Imagination*, New York: Basic Books, 2000.

[5] G. Tononi, G.M. Edelman and O. Sporns, "Complexity and Coherency: integrating information in the brain," *Trends in Cog. Sci.*, vol. 2, pp. 474-484, 1998.

[6] H. Imamizu, T. Koroda, T. Yoshioka and M. Kawato. "Functional magnetic resonance imaging examination of two modular architectures for switching multiple internal models," *J. Neurosci*., vol. 24, pp. 1173-1181, 2004.

[7] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. on Neural Networks*, vol. 5, pp. 157-166, 1994.

[8] H. Sompolinsky and I. Kanter, "Temporal association in asymmetric neural networks,", *Phys. Rev. Lett*., vol. 57, pp. 2861-2864, 1986.

[9] D.L. Wang and M.A. Arbib, "Complex temporal sequence learning based on short-term memory", *Proc. of the IEEE*, vol. 78, pp. 1536-1543, 1990.

[10] M. Reiss and J.G. Taylor, "Storing temporal sequences," Neural Networks, vol. 4, pp. 773-787, 1991.

[11] T.M. Heskes and S. Gielen, "retrieval of pattern sequences at variable speeds in a neural network," Neural Networks, vol. 5, pp. 145-152, 1992.

[12] R. Sun and C.L. Giles (eds), *Sequence Learning: Paradigms, Algorithms, and Applications*. New York: Springer, 2001.

[13] A.A. Minai and P.J. Best, "Encoding spatial context: a hypothesis on the function of the dentate gyrus-hilus system," in *Proc. IJCNN*, Anchorage, AL, May 1998, pp 587-592.

[14] S. Doboli and A.A. Minai, "Latent attractor selection for variable length episodic context stimuli with distractors," in *Proc. IJCNN*, Portland, OR, July 2003, pp. 1643-1648.

[15] S. Doboli and A.A. Minai, "Using latent attractors to discern temporal order," in *Proc. IJCNN*, Budapest, Hungary, July 2004.

[16] S. Doboli and A.A. Minai, "Latent attractors: A general paradigm for context-dependent neural computation," in *Trends in Neural Computation*, K. Chen and L. Wang, Eds. New York: Springer Verlag, pp 135-169.

[17] S. Doboli, A.A. Minai, and P.J. Best, "Latent attractors: a model for context-dependent place representations in the hippocampus," *Neural Computation*, vol. 12, pp.1003-1037, 2000

[18] S. Doboli, A.A. Minai and V.R. Brown, "Adaptive dynamic modularity in a connectionist model of context-dependent idea generation," in *Proc. IJCNN*, Orlando, FL, August 2007.

[19] L.R. Iyer, A.A. Minai, S. Doboli and V.R. Brown, "Modularity and self-organized functional architectures in the brain," *Proc. 7th Int. Conf. on Complex Sys.*, Boston, MA, October 2007.

[20] A. Ghanem and A.A. Minai. "A Modular gene regulatory network model of ontogenesis," in *Proc. 7th Int. Conf. on Complex Sys.*, Boston, MA, October 2007.

[21] J.C. Houk and S.P. Wise, "Distributed modular architectures linking basal ganglia, cerebellum, and cerebral cortex: their role in planning and controlling action," *Cerebral Cortex*, vol. 5, pp. 95-110, 1995

[22] J.C. Houk, "Agents of the mind," *Biol. Cybern*., vol. 92, pp. 427-437, 2005.

[23] A.M. Graybiel, "Building action repertoires: memory and learning functions of the basal ganglia," *Curr. Opinion in Neurobiology*, vol. 5, 733-741, 1995.

[24] A.M. Graybiel, "The basal ganglia and chunking of action repertoires," *Neurobiology of Learning and Memory*, vol. 70, 119-136, 1998.

[25] T.E. Hazy, M.J. Frank and R.C. O'Reilly " Banishing the homunculus: making working memory work," *Neuroscience*, vol. 139, 105-118, 2006.

[26] R.C. O'Reilly and M.J. Frank, "Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia," *Neural Computation*, vol. 18, 283-328, 2006.

[27] J.L. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, pp. 179-211.

[28] A. Cleeremans, D. Servan-Schreiber and J. McClelland, "Finite-state automata and simple recurrent networks," Neural Computation, vol. 1, pp. 372-381, 1989.

[29] B. Ans, Y. Coiton, J.C. Gilhodes and J.-L. Velay, "A neural network model for temporal sequence learning and motor programming,", Neural Networks, vol. 7, pp. 1461-1476, 1994.

[30] M. Botvinick and D.C. Plaut, "Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action," Psych. Rev., vol. 111, pp. 395-429, 2004.

[31] A.-L. Barabasi and R. Albert, "The emergence of scaling in random networks," *Science*, vol. 286, pp. 509-512, 1999.

[32] B. Bollobas, *Random Graphs*, London: Academic Press, 1985.

[33] J.W. Bohland and A.A. Minai, "Ef£cient associative memory using small-world architecture," *Neurocompuing*, vol. 38-40, pp. 489-496, 2001.

[34] O. Sporns and G. Tononi, "Classes of network connectivity and dynamics," *Complexity*, vol. 7, pp. 28-38, 2002.

[35] O. Sporns, D.R. Chialvo, M. Kaiser and C.C. Hilgetag, "Organization, development and function of complex brain networks," *Trends in Cog. Sci*., vol. 8, pp. 418-425, 2004.

[36] N. Davey, S.P. Hunt and R.G. Adams, "High capacity recurrent associative memories," *Neurocomputing*, vol. 62, pp. 459-491, 2004.

[37] O. Sporns, G. Tononi and G.M. Edelman, "Theoretical neuroanatomy: relating anatomical and functional connectivity in graphs and cortical connection matrices," *Cerebral Cortex*, vol. 10, pp. 127-141, 2000.

[38] R. Albert, H. Jeong and Al.-L. Barabasi, "Error and attack tolerance of complex networks," *Nature*, vol. 406, pp. 378-382, 2000.

[39] Y. Bar-Yam and I. R. Epstein, "Response of complex networks to stimuli," *Proc. Natl. Acad. Sci. USA*, vol. 101, pp. 4341-4345, 2004.

[40] J.M. Carlson and J. Doyle, "Highly optimized tolerance: robustness and design in complex systems," *Phys. Rev. Lett*. Vol. 84, pp. 2529-2532, 2000.

[41] G. Paul, T. Tanizawa, S. Havlin and H.E. Stanley, "Optimization of robustness of complex networks," *Eur. Phys. Journal B*, vol. 38, pp. 197-191, 2004.

[42] A.X.C.N. Valente, A. Sarkar and H.A. Stone, "Two-peak and three-peak optimal complex networks," *Phys. Rev. Lett*., vol. 92, p. 118702, 2004.

[43] G. Athithan and C. Dasgupta, "On the problem of spurious patterns in neural associative models," *IEEE Trans. on Neural Networks*, vol. 8, pp.1483-1491, 1997.

[44] A. V. Robins and S. J. R. Mccallum, "A robust method for distinguishing between learned and spurious attractors," *Neural Networks*, vol. 17, pp. 313-326, 2004.