Transformations in the scale of behavior and the global optimization of constraints in adaptive networks

Richard A Watson, Rob Mills and C.L. Buckley

Adaptive Behavior 2011 19: 227 originally published online 2 August 2011
DOI: 10.1177/1059712311412797

The online version of this article can be found at:
http://adb.sagepub.com/content/19/4/227
Transformations in the scale of behavior and the global optimization of constraints in adaptive networks

Richard A Watson¹, Rob Mills¹ and C L Buckley²

Abstract
The natural energy minimization behavior of a dynamical system can be interpreted as a simple optimization process, finding a locally optimal resolution of problem constraints. In human problem solving, high-dimensional problems are often made much easier by inferring a low-dimensional model of the system in which search is more effective. But this is an approach that seems to require top-down domain knowledge—not one amenable to the spontaneous energy minimization behavior of a natural dynamical system. However, in this article we investigate the ability of distributed dynamical systems to improve their constraint resolution ability over time by self-organization. We use a “self-modeling” Hopfield network with a novel type of associative connection to illustrate how slowly changing relationships between system components can result in a transformation into a new system which is a low-dimensional caricature of the original system. The energy minimization behavior of this new system is significantly more effective at globally resolving the original system constraints. This model uses only very simple, and fully distributed, positive feedback mechanisms that are relevant to other “active linking” and adaptive networks. We discuss how this neural network model helps us to understand transformations and emergent collective behavior in various non-neural adaptive networks such as social, genetic and ecological networks.

Keywords
Hopfield networks, associative learning, dynamical systems, adaptive networks, constraint optimization, modularity, nearly decomposable systems, coarse-graining, self-organization, canalization

1 The computational abilities of dynamical systems
Physical dynamical systems with a large number of simple equivalent components have been shown to exhibit “emergent collective computational abilities” (Hopfield, 1982) such as implementing content-addressable memory or solving constraint satisfaction problems (Hopfield & Tank, 1985, 1986). In the latter, Hopfield and Tank equate the energy minimization behavior (Strogatz, 1994) of a dynamical system with an optimization process, that is, the system moves to configurations that better resolve the conflicting constraints between system variables. But actually, energy minimization in a simple dynamical system is equivalent to the simplest possible optimization algorithm, namely gradient descent (or incremental improvement), which in anything but the simplest of problems tends to find only locally optimal solutions. In human design-engineering and optimization, high-dimensional problems are often made much easier by inferring a low-dimensional model of the system (e.g., a
high-level representation that exploits modularity/problem decomposition), such that “local” search in this new or rescaled space is better able to find a globally optimal resolution of constraints. This is an approach that seems to require top-down domain knowledge and design intelligence, and does not appear to be amenable to the spontaneous energy minimization behavior of a natural dynamical system. However, can other types of dynamical systems, specifically self-organizing systems, perform more sophisticated forms of optimization? And conversely, can an optimization framework help us to better understand the behavior of natural self-organizing systems?

Our questions are motivated by consideration of self-organizing multiagent systems, such as species in an ecosystem or agents in a socioeconomic network, and their potential to exhibit emergent collective behaviors. In particular, we are interested in the possibility that such systems can spontaneously transform into a new system, operating at a higher level of organization (Maynard Smith & Szathmary, 1995; Simon, 1969), and that such a dynamical transformation may facilitate (or may even be equivalent to) a transition in the ability to resolve constraints between the system components. Our earlier work in this area, using abstract optimization algorithms loosely inspired by symbiogenesis (Mills, 2010; Mills & Watson, 2011b) suggested that simple associative rules could, in the right context, be sufficient to identify and exploit higher level problem structure; and individual-based evolutionary models of the formation of new units of selection (Watson, Palmius, Mills, Powers, & Penn, 2011; Watson, Palmius, Jackson, Mills, & Powers, 2011) showed that such associations could be favored by natural selection and thereby occur spontaneously under suitable conditions.

To formalize the formation of such relationships we can characterize multiagent ecosystems or socioeconomic network systems as “adaptive networks” (Gross & Sayama, 2009; Newman, 2003; Newman, Barabasi, & Watts, 2006); these exhibit the property that the structure of connections between agents affects changes to the agent behaviors and, vice versa, that the agent behaviors affect changes to the structure of connections between agents. The Hopfield network (Hopfield, 1982) easily accommodates such state/topology coadaptation and, at a very abstract level, provides a suitable system with which to explore how self-organization in adaptive networks alters their ability to resolve conflicting constraints between system components. Working with this analogy implied that the associative connections exploited by evolution and by our abstract algorithms might be directly analogous to associative learning (Hinton & Sejnowski, 1985; Hopfield, 1982) in the Hopfield network. Investigation of this potential homology has been fruitful (Davies, Watson, Mills, Buckley, & Noble, in press; Watson, Buckley, & Mills, 2009, 2011; Watson, Mills, & Buckley, in press), but we found that the usual interpretation of learned associations in the Hopfield network did not provide the same ability to resolve constraints as the associations that were exploited in the formation of new evolutionary units (or in our abstract algorithm). Specifically, the usual interpretation of learned associations in the Hopfield network provided an interesting improvement to what is essentially a gradient descent or incremental improvement method—but it did not really provide a scaling-up in the dynamical behavior of the system. However, in this article we find that a different interpretation of learned associations in the Hopfield network does provide a true rescaling of the dynamical behavior and constraint optimization ability. This article now provides a formal description of this difference (contrasting the usual type of associations, which we refer to as selective associations, with the new interpretation, which we call generative associations) and relates the findings more closely with self-organized dynamical systems and formal optimization methods.

For example, unlike simple gradient descent methods, state-of-the-art computational optimization techniques (e.g., Pelikan, 2002; Pelikan, Goldberg, & Lobo, 2002) have the option to use advanced model-building mechanisms that can identify and exploit high-level structure in a problem domain automatically (Mills, 2010; Mills & Watson, 2011b). In general terms, such methods seek to identify a low-dimensional search space within a high-dimensional problem domain by “memoizing” (Michie, 1968) partial solutions (Angeline & Pollack, 1992; Koza, 1994), or “chunking” (Rosenbloom & Newell, 1986) sequences of actions, to reuse them as macro-operations that rescale the search process. The resultant lower dimensional space can facilitate global constraint optimization far superior to local search methods (Mills & Watson, 2011b). But it has not previously been shown that it is possible for simple distributed mechanisms of self-organization, gradually changing the connections of an adaptive network, to cause it to effectively rescale its dynamics and hence move from local to global energy minimization in a similar manner.

Our aim is therefore to provide a concrete model of how an adaptive network can be transformed from one scale of dynamics to another via simple fully distributed self-organizing mechanisms. More narrowly, our objective is to devise a simple distributed mechanism that modifies the connections of a self-modeling (Watson, Buckley, et al., 2011) Hopfield network such that its dynamics change from local to global energy minimization. In particular, we aim to exploit implicit
modular, or nearly decomposable (Simon, 1969; Watson & Pollack, 2005), structure in the constraints of the system such that, whereas the original dynamics of the system search combinations of the original state variables, the transformed dynamics search combinations of higher level emergent variables.

Working within an adaptive network framework, we restrict attention to a self-organizing mechanism that changes the connections between components, that is, causes changes to weighted connections in the network. Specifically, we assume connections grow in strength in a manner that corresponds to correlations between state variables as per our prior work. This is a simple and fully distributed, positive feedback mechanism familiar in neural network research as associative or Hebbian learning (Ackley, Hinton, & Sejnowski, 1985; Hebb, 1949; Hinton & Sejnowski, 1983, 1985). We refer to this as a self-organizing mechanism rather than a learning mechanism because the patterns that are “learned” are created by the system’s own dynamics (rather than an external training regime). Although in this article we simply mandate that such changes to connections occur in our network, our recent work shows that Hebbian changes to connections are natural in many types of complex adaptive systems, and not just neural systems (Watson, Mills, et al., in press). Accordingly, we aim to demonstrate a mechanism that rescales the optimization ability of the Hopfield network and thereby to provide a formal model that helps us understand transformations in the scale of problem-solving behavior in other types of adaptive networks.

A central focus of this article is the contrast between two types of associative connection. Specifically, characterizing the energy minimization behavior of the system as a “generate and test” (Winston, 1992) optimization process, we show that learned associations can straightforwardly modify either the generate function or the test (or “selection”) function of this process. We find that associations that alter the generate function are much more effective at enabling global minimization of energy. Such generative associations (in contrast to the conventional selective associations) have the effect of producing probabilistic, correlated, simultaneous state changes in multiple variables and correspond to emergent macro-variables that encapsulate subsolutions. We find that it is possible for a self-organized dynamical system with generative associations, using only simple distributed mechanisms, to transform into a new system operating at a higher level of organization, and we show that this transformed system minimizes total system energy more reliably and more completely than the original system. In effect this provides a novel distributed optimization algorithm. But more important for our current aims, this transformation is enacted through very simple, and fully distributed, positive feedback mechanisms that are relevant to other active-linking systems and adaptive networks (Gross & Sayama, 2009; Pacheco, Traulsen, & Nowak, 2006; Santos, Pacheco, & Lenaerts, 2006; Van Segbroeck, Santos, Pacheco, & Lenaerts, 2010). We therefore suggest that this neural network model helps us to understand transformations and emergent collective behavior in various non-neural adaptive networks such as social, genetic, and ecological networks; for example, the emergence of coalitions in social systems and of higher level adaptive units in the major transitions in evolution.

2 Methods

The Hopfield network (Hopfield, 1982) is an abstract model of neural architecture and a well-understood example of a simple dynamical system that has provided a vehicle for studying adaptive behavior (e.g., Goetz & Walters, 1997; Paine & Tani, 2005) and the computational properties of physical systems across many disciplines. Hopfield and Tank’s (1985, 1986) work makes an explicit bridge between the language of dynamical systems energy-minimization and the language of optimization or search which is very useful for formalizing the type of dynamical transformation we wish to study. Studying multilevel structure and multilevel optimization (Mills, 2010; Mills & Watson, 2011b) in the context of a neural network model has the advantage that the ability of distributed neural mechanisms to identify high-level features in a set of training samples (McClelland, Rumelhart, & PDP Research Group, 1986; O’Reilly & Munakata 2000), and the potential of recurrent neural networks to find solutions to constraint optimization problems (Hopfield & Tank, 1985; Tsirukis, Reklaitis, & Tenorio, 1989), are already well understood. Our recent work brings these two well-known uses together (Watson, Buckley, et al., 2009) in what we call a self-modeling system (Watson, Buckley, et al., 2011), a system that (implicitly) models its own behavior.

In overview, the model operates as follows. An energy function encodes an optimization problem which governs the initial energy minimization dynamics of the system (Hopfield & Tank, 1985). We implement an energy function that represents a constraint optimization problem by using a set of weights corresponding to constraints imposed by an external environment. The system is then repeatedly relaxed from many different random initial conditions, causing it to visit many different dynamical attractors in this energy function. If the transients of each relaxation are short compared with the duration of each relaxation this causes the system to spend most of its time at local minima in the energy function. Over the course of many
relaxations, Hebbian learning is used to modify a set of internal weights. This has the effect of enlarging the dynamical basin of attraction for the system’s current state configuration, and therefore tends to make the system more likely to visit state configurations that have already been visited, that is, simple positive feedback. But this will also enlarge the basin of attraction for other state configurations, even if they have not yet been visited, in proportion to their similarity to state configurations that have been visited. This associative generalization has the effect of increasing the likelihood that the system visits low energy attractors, including the globally minimal energy attractor, and increases the speed with which it is first found.

2.1 Discrete hopfield network with restarts, rHN

The state of a network consisting of \( N \) discrete states \( s_j = \pm 1 \) where \( i = 1, 2, \ldots, N \) can be written as \( S = (s_1, \ldots, s_N) \). The dynamics of the recurrent network used by Hopfield can be described by updates to individual states:

\[
\begin{align*}
    s_j(t+1) = \theta & \left[ \sum_j \omega_{ij} s_j(t) \right],
\end{align*}
\]

where \( \omega_{ij} \) are elements of the connection matrix \( \Omega \), and \( \theta \) is the Heaviside threshold function (taking values \(-1\) and \(+1\) for negative and non-negative arguments respectively). The Hopfield network is run by repeatedly choosing a unit, \( i \), uniformly at random and setting its state according to the above expression.

Hopfield (1982) showed that if the connection matrix is symmetric, \( \omega_{ij} = \omega_{ji} \), and under suitable constraint on the self-weights (here \( \omega_{ii} = 1 \)), all trajectories described by Equation 1 converge on point attractors which are minima of an energy or potential function given by:

\[
E_S = H(S, \Omega) \equiv - \sum_{ij} \omega_{ij} s_is_j
\]

Consequently one can describe the asymptotic behavior of such a network in terms of a process that minimizes this function. Accordingly, if the weights of a network are appropriately defined they can be used to define an energy function that represents an optimization problem, and running the network finds locally optimal solutions to the problem (Hopfield & Tank, 1985). The Boltzmann machine (Hinton & Sejnowski, 1985) is a discrete stochastic counterpart of the Hopfield network where a single state change is accepted probabilistically according to the change in energy it produces. We can describe such a dynamical process more generally via a probability of accepting a stochastic change to the system state:

\[
P[S(t+1) \leftarrow f(S(t))] = \sigma(T, \Delta E),
\]

where the variation operator \( f \) is, in the Boltzmann or discrete Hopfield cases, a “bit flip” operator defined as \( f(S) = (s_1, \ldots, -s_i, \ldots, s_N) \) where \( X \) is a uniform random variable on \([1, \ldots, N] \). \( \Delta E = E_S' - E_S \) is the change in energy implied by the new state \( S' = f(S) \), and \( \sigma(T, x) = \frac{1}{1 + \exp(-x/T)} \) is a sigmoid function of \( x \) where \( T \) is the temperature of the system; a parameter that indirectly controls the probability of an increase in energy. When the temperature is reduced gradually this describes a simulated annealing process (Kirkpatrick, Gelatt, & Vecchi, 1983). But for a deterministic system, as \( T = 0 \), we can simply write:

\[
P[S(t+1) \leftarrow f(S(t))] = \lambda(\Delta E).
\]

where \( \lambda \) is a threshold function taking values \( 0 \) and \( 1 \) for negative and non-negative arguments, respectively. Thus, the discrete Hopfield network and the Boltzmann machine with \( T = 0 \) are both equivalent to a bit-flip gradient descent algorithm. Although the continuous state version of the Hopfield network and the nonzero temperature Boltzmann machine are, in many circumstances, better optimizers than the bit-flip gradient descent algorithm, the discrete deterministic process is sufficient to illustrate the effect we wish to demonstrate in this article. This type of generate-and-test (or generate-and-select) energy descent process, that is, updating the network by accepting a stochastic change in state if and only if it decreases system energy, is conventional in stochastic local search or mutation-based evolutionary models (see Section 4), whereas the Hopfield equations of motion, which modify system states deterministically in the direction that minimizes energy, are conventional in neural networks and physical dynamical system models. However, these frameworks are interchangeable for the basic Hopfield network, and the generate-and-test framework naturally affords the incorporation of generative associations described below.

Since these deterministic dynamics will quickly find a local optimum in the energy function, it is useful to consider a “random restart” version of the Hopfield network, \( rHN \), where the state of the system takes a random state configuration, \( R = \{-1, 1\}^N \), every \( \tau \) time steps. More generally, we may suppose that the system is occasionally radically perturbed so as to cause it to move into the basin of a different attractor (this corresponds to many attempts at solving the problem, finding different locally optimal solutions). We refer to each
inter-reset duration of \( \tau \) time steps as a relaxation, and assess the minimum energy and average energy of states visited by the system over many relaxations. Note that the energy of configurations visited within a relaxation will be minimal at the end of that period, so it is often useful to monitor the energy of the system at that time. If each relaxation is long enough for the system to reach a local optimum, this will thus monitor the local optima visited by the system. We refer to this default type of restart Hopfield network (with no learning) as \( rHN-0 \).

Hopfield and Tank (1985) showed that the weights of a Hopfield network can be used to encode the constraints of a combinatorial optimization problem such that minima in the energy function correspond to local minima in the optimization problem. This provides a fully distributed problem-solving or optimization mechanism, but the solutions that this basic mechanism produces are only locally optimal and, in general, may be far from globally optimal.

### 2.2 Hebbian learning and self-modeling in the hopfield network

In the following experiments we modify the dynamics of the restart Hopfield network described above using self-modeling (Watson, Buckley, et al., 2011). This simply means that Hebbian learning (see below) is applied to the connections of the restart Hopfield network whilst it is running. Given that the duration of relaxations is long enough for the system to spend most of its time at local optima then in effect these locally optimal configurations become “training samples” for an associative memory. In this manner the system implicitly uses observations of its own optimization dynamics to identify problem structure and modify its dynamics, hence self-modeling. Clearly, if the system spent most of its time in arbitrary state configurations then this self-modeling could not yield any useful information. But given that the system spends most of its time at configurations that are low in energy, the model that is induced is not arbitrary even though it is modeling its own state dynamics. Importantly, a distribution of locally optimal configurations sampled by \( rHN-0 \) has the potential to exhibit correlations that can reveal problem structure. Thus the self-modeling protocol causes a network to implicitly self-generate a set of training samples from an energy function with the possibility of revealing the system’s constraint structure.

The original connection matrix, \( O \), representing the problem, remains unaltered by this process but Hebbian learning is applied to all \( m_{ij} \), \( i \neq j \), in an internal connection matrix, \( M \), as below. Alternatively, two sets of weights are not required if \( H(S, \Omega) \) is replaced by a potential function (or “fitness function”) of arbitrary construction representing environmentally imposed problem constraints and \( M \) is the (single) set of internal weights representing the learned response to those constraints (see next section). The initial value of all \( m_{ij} \) is 0.

\[
m_{ij}(t + 1) = \gamma[m_{ij}(t) + \frac{\delta}{\tau}s_i(t)s_j(t)],
\]

where \( (\delta/\tau) \) is a learning rate and \( \gamma \) is a linear threshold function capping all learned weights at a magnitude of 1 (i.e., \( \gamma(x) = \max[-1, \min(1, x)] \)). If we assume that \( \tau >> t^* \), where \( t^* \) is the time for the system to reach a local optimum then the cumulative effect of learning over a relaxation will be approximately equal to a single application of learning at the end of each relaxation:

\[
m_{ij}(\tau + 1) = \gamma[m_{ij}(\tau) + \delta s_i(\tau)s_j(\tau)].
\]

This end-of-relaxation learning is computationally less expensive to implement so we use this in our experiments, but a continuous learning process may be more natural for models of some systems. We can see that \( M \) will become an associative memory model of (the local minima of) the original energy function (implicitly defined by \( \Omega \)).

### 2.3 Modifying dynamics with selective associations, \( rHN-S \)

This associative memory can then modify the dynamics of the system in two different ways depending on how these learned associations are interpreted. The first of these, which here we term selective associations, is to modify the energy function (Tsutsumi, 2003; Watson, Buckley, et al., 2011) such that stochastic changes to individual states are accepted if they decrease the value of a new energy function \( E' \) (Watson, Buckley, et al., 2011), that is,

\[
P[S_{(t+1)} \leftarrow f(S_{(t)})] = \lambda(\Delta E'),
\]

where \( E = H(S, \Omega) + H(S, M) = H(S, \Omega + M) \) (or more generally \( E' = Q(S) + H(S, M) \) where \( Q \) is any potential function on \( S \)); that is, the new energy function is simply the original energy function (Equation 2) plus the additional energy derived from the learned weights. This has the effect of changing the relative importance of different constraints in a manner that “warps” the energy surface (Tsutsumi, 2003). Because this is equivalent to modifying the original weights (that are supposed to represent the problem we are trying to solve) this is a somewhat strange thing to do to a Hopfield
network that is being used for optimization. However, the effect of this is interesting for three reasons:

1. Because the associative memory that is learned is trained at locally optimal solutions to the problem, and the effect of Hebbian learning is to increase the basin of attraction for the training samples, the effect of this learning is to increase the basin of attraction for the locally optimal configurations that are visited.

2. Because low-energy configurations tend to have larger basins of attraction (Coolen, 1991; Gardner, 1988; Kryzanovsky & Kryzanovsky, 2008), the lower energy attractors tend to be visited more often, and are therefore enlarged more often, than higher energy locally optimal solutions.

3. Because subpatterns that are common to many local optima tend to also be common to the global optimum, the generalization of patterns provided by associative memory (Fontanari, 1990; Jang, Kim, & Lee, 1992) has the potential to enlarge the basin of attraction for the globally optimal configuration even before it is visited for the first time (Watson, Buckley, et al., 2011; Watson, Mills, et al., in press).

Because there is no optimization benefit in recalling a solution only after it has been discovered by chance, this generalization ability is crucial for a memory process to become an optimization process.

Note that although the trajectory of the system through state space is modified by the new energy function, \( E \), as external observers of how the system has changed we will assess the quality of the solutions with respect to the original system constraints, that is, \( E \).

This model, \( rHN-S \), is a distributed means to increase the optimization ability of the Hopfield network (Watson, Buckley, et al., 2011). The learned associations in \( rHN-S \) modify the energy function, or the test (i.e., selection) part of a generate-and-test process—hence selective associations. We have investigated self-organization using selective associations in recent work (Davies et al., in press; Mills, 2010; Watson, Buckley, et al., 2009, 2011; Watson, Buckley, Mills, & Davies, 2010; Watson, Mills, et al., in press). But we find that this does not provide a true scaling-up in the dynamical behavior of the system. In this article we introduce a different type of association which we call generative associations.

2.4 Modifying dynamics with generative associations, \( rHN-G \)

Generative associations modify the generate part of the generate-and-test process, rather than the test part. Specifically, learned connections (probabilistically) produce correlated state changes in multiple state variables simultaneously (and the test or selection part of the process is unchanged; Figure 1).

In Figure 1b the selection criteria of variants is altered but the generation of variants is unchanged, whereas in Figure 1c the generation of variants is altered but the selection criteria of variants is unchanged. Another analogy that might be useful is that whereas \( rHN-S \) modifies the fitness values of a landscape, \( rHN-G \) can be understood as a process that modifies the neighborhood structure of a fitness landscape (Mills & Watson, 2011b).

Generative associations are implemented with a modified variation operator, \( g \) (replacing \( f \)). This operator interprets the learned weights, \( M \) (containing explicit knowledge of the problem structure), as defining probabilities of correlated state changes. To facilitate a unified definition of \( f \) and \( g \), we can redefine \( f \) in terms of \( g \) and a correlation matrix \( C = I \), were \( I \) is the identity matrix, then represents the state changes produced by \( f \) in the sense that single-variable state changes are allowed but no other variables are correlated with them, that is, under \( f \), each variable is (with respect to generating state changes) “correlated” only with itself (i.e., it has only self-connections). In contrast, under more general usage of \( g \), single state changes are augmented by correlations defined in \( M \) by using \( C = I + M \). Thus initially, when all \( m_{ij} = 0 \), \( g(S) = f(S) \), that is, both implement single-variable state changes at random. But later in the learning process, if some node \( i \) has strong learned connections to another node \( j \), then when a new state is assigned to node \( i \), an appropriate state is also assigned to node \( j \), with probability proportional to correlations observed in the past. Formally,

\[
g(S) = (s_1', \ldots, s_X', \ldots, s_N'),
\]

where \( X \) is a uniform random variable on \( \{1, \ldots, N\} \), and \( s' \) takes the new value \(-\theta(c_{\langle X,Y \rangle})\) if \( |c_{\langle X,Y \rangle}| > r \), where \( r \) is a random threshold (drawn once for each application of \( g \)) uniformly in the range \((a,1]\), and \( a \) is the mean magnitude of values in \( M \). Thus, because \( C = M + I \), the state of \( s_X \) is necessarily flipped (as was the case in \( f \)), but additionally all other states, \( s_{\not= X} \) will either be changed to agree with this new state when \( m_{XY} > 0 \), or disagree with the new state when \( m_{XY} < 0 \), with a probability that increases with the magnitude of \( m_{XY} \). The adjustment, \( a \), ensures that only learned correlations with above-average strength will cause additional state changes.

Thus, quite straightforwardly, this new operator recreates subpatterns in the state configuration according to the frequency with which they have been observed in
the past and does so using only the distributed network of pair-wise associations generated by normal Hebbian learning—but the mechanistic effect of these associations is interpreted differently. Of course, if the learning rate is too high this will have the effect of moving the system directly to a configuration (e.g., a local optimum) that has been visited before. But if the learning rate is appropriate, this mechanism offers the possibility of recalling commonly occurring partial solutions (because a given partial solution may occur in many different complete solutions) and importantly, creating new combinations of these subsolutions to explore the search space more efficiently. Thus correlated state changes that canalize commonly occurring partial solutions effect a rescaling of the dynamical behavior and effect a search process operating at a higher level of organization. But note that if $M$ has arbitrary entries, then the effect of $g$ is merely to produce random “macro-mutations” (Forrest & Mitchell, 1993) which are not sufficient to solve problems of the form we study here (Watson & Jansen, 2007).

As in $rHN-0$, a (single-variable or multivariable) state change produced by $g$ is tested against the original energy function, $E$, to see how it affects energy (a calculation based on the change in energy for each variable that changes state, rather than total energy, may be sufficient). If it reduces energy the new state configuration is retained, else it is rejected, that is:

$$P[S_{(t+1)} \leftarrow g(S_{(t)})] = \lambda(\Delta E).$$  \hfill (8)

Whereas we might think of the effect of selective associations as learning to respond to correlations in an input or stimulus, we can interpret the effect of generative associations as learning to produce correlations in an output or behavior. Generative associations, like selective associations, are consistent with Hebb’s fundamental conjecture (Hebb, 1949) that learning happens by modifying connections between neurons, and also consistent with Hebb’s rule that builds connections in proportion to past correlations. They also have the same effect as Hebbian learning in the sense that they make future system states more likely to recreate past correlations. But in conventional neural models (i.e., Equation 1) attention is placed on single state updates, decided on the basis of activation potentials that are fed into one neuron from many presynaptic neurons. This can only change the activation of one neuron at a time. In contrast, in this generative model, the focus is on the effect of an action potential that is fed out from one neuron into many postsynaptic neurons, producing correlated state changes in many neurons simultaneously (e.g., see “synchronous firing chains,” Abeles, 1982; and “in star” versus “out star” configurations, Grossberg, 1978). But clearly, our emphasis in this article is not on the physical implementation of the mechanisms involved but in the dynamical and computational properties of such associative connections.

More generally, we are interested in the dynamical and computational properties of a dynamical system undergoing self-organization via the formation of this type of association. For example, coadaptation between species that modifies the effective evolutionary unit, such as group formation and symbiogenesis (Mills & Watson, 2011a; Powers, Penn, & Watson, 2007; Watson, 2006; Watson & Pollack, 2003), is analogous to generative associations whereas coadaptation that modifies the “interaction coefficients” between species, that is, ordinary coevolution, merely affects the selective environment of the original evolutionary units (Lewis, 2009; Powers, 2010) and is analogous to selective associations. We will briefly discuss this and other non-neural systems later. But our narrow aim in this article is to show that although the effect of self-organization with generative associations corresponds to that of a model-building optimization algorithm, the “information” required for these mechanisms can be discovered by a simple distributed process and can be held implicitly in the distributed network of simple associative connections.

Figure 1. Different complementary ways to modify the trajectory of a dynamical system with respect to an energy surface. (a) Original dynamics: The current system configuration (black dot) is depicted in a local minimum on an energy surface (solid curve), single-state variants on either side (white dots) are higher in energy and hence the local trajectories of the system (arrows) return it to the original location. (b) Modified selection: the energy values of the neighboring positions when measured in the modified energy surface (dotted curve) enable the system to move to the right and thereby escape this local optimum in the original energy surface. (c) Modified generation: Here the system escapes the local optimum because the variants that are generated are not single-state changes but specific multistate changes, one of which is lower in energy.
2.5 Summary of models

Our underlying model is a Hopfield network with discrete state dynamics and multiple relaxations restarted from random initial conditions after each \( t \) time steps. Modifications to the default model using either selective or generative associations can be understood as a logically complementary pair of processes within the generate-and-test search framework. Specifically, the default generate-and-test procedure, \( rHN-0 \), generates a candidate state configuration from some probability distribution, for example, the set of single-bit-flips, then tests for an improvement (decrease in energy). \( rHN-S \) generates the same distribution of candidates as \( rHN-0 \) but uses a modified test—accepting decreases in the modified energy function. In contrast, \( rHN-G \) uses a modified generate procedure (macro-variations) but the original test function.

We can define all three algorithms using a general model denoted \( rHN(A,B) \) where \( A \) is a matrix defining the correlations used to generate variants (via the operator \( g \)), and \( B \) is a matrix defining the energy function used to assess the variant state configurations thus created (via the Hopfield equation, \( H(S,B) \), Equation 2). Specifically, \( rHN-0 = rHN(I, \Omega) \) where \( I \) is the identity matrix (producing single state changes only) and \( \Omega \) is the original energy function. Then \( rHN-S = rHN(I, \Omega + M) \) such that the energy function is augmented by \( M \), and conversely, \( rHN-G = rHN(I + M, \Omega) \) such that the correlations in the variations produced are governed by the learned matrix \( M \). This also suggests a general framework using \( M \) in both the generator and the test function, \( rHN(I + \alpha M, \Omega + (1-\alpha)M) \), creating an algorithm that can be parameterized from one to the other (i.e., \( \alpha = 0 \) defines \( rHN-S \) and \( \alpha = 1 \) defines \( rHN-G \)), but we have not investigated other values of \( \alpha \) in this article.

Note that when \( \delta = 0 \), both \( rHN-S \) and \( rHN-G \) are identical to the original nonadaptive Hopfield network, \( rHN-0 \). Similarly, the initial behavior of \( rHN-S \) and \( rHN-G \) before significant learning has taken place will also be qualitatively identical to \( rHN-0 \).

In some respects, \( rHN-S \) and \( rHN-G \) are very different mechanisms: the former uses standard bit-flip exploration but distorts the energy function and the latter enables macroexploration over the original energy function. However, they both use a matrix, \( M \), generated by the same fully distributed Hebbian learning mechanism, to exploit correlations in the state variables observed at locally optimal configurations (however, note that since \( M \) is learned from a self-referential examination of the system’s own dynamics, the values of \( M \) learned by \( rHN-S \) will diverge over time from those learned by \( rHN-G \) even though the distribution of locally optimal configurations is initially identical). Also, both algorithms use this data to make patterns of activation that have been observed in the past more likely to occur again in future dynamics (by either selecting those subpatterns or generating those subpatterns). Both therefore have the potential to exploit problem structure and in particular the ability of associative memory to generalize over a set of patterns (vital for finding novel patterns of low energy). However, clearly \( rHN-S \), being wedded to single-variable state changes, does not have the possibility of rescaling the search space into a higher level abstraction. Accordingly, we shall see that the way in which \( rHN-G \) uses learned correlations has capabilities that are not provided by \( rHN-S \). \( rHN-S \) is nonetheless a more obvious way of adding Hebbian learning to a Hopfield network than \( rHN-G \) and the comparison between the abilities of \( rHN-S \) and \( rHN-G \) is useful to verify that this more simplistic approach to “recalling past solutions” in a dynamical system with selective associations is not sufficient to solve problems that can be solved by rescaling the search space in a dynamical system with generative associations.

2.6 An idealized nearly decomposable constraint optimization problem

In an optimization framework, our aim is to provide a conceptual illustration of a generic capability to exploit problem structure using distributed mechanisms. This relies, of course, on the existence of structure in the problem domain/original system dynamics that can be exploited. Accordingly, rather than measure the impact of this mechanism in a specific application where the presence or absence of high-level structure is unknown, we investigate it using abstract cases where we can control the structure present in the system in a systematic manner.

Our test problems are represented using a multimodal energy function defined by different types of weight matrices, \( \Omega \), as below and the Hopfield equation (Equation 2)—this ensures that the problem class is a superposition of pair-wise constraints (Watson, Buckley, et al., 2011; Watson, Mills, et al., in press). We examine two different classes of system: random constraints (\( RC \)) and an explicitly modular constraint problem (\( MC \)) that facilitates supporting analysis and illustration. The modular problems are defined as a macroscale version of the random problems as discussed below.

Both these problems are equivalent to instances of MAX-SAT, graph-coloring and simple resource-allocation optimization problems (Watson, Buckley, et al., 2011) and are used here to illustrate the optimization capabilities and limitations of our model in a domain-independent manner. These investigations
illustrate the conceptual idea behind rHN-G and also help us to understand exactly the limitations of the model and, in particular, MC assists us in understanding the limitations of selective associations and the different affordances of generative associations.

The aim in defining MC is to construct an idealized exemplar—a system that can be solved easily by a process that can discover and exploit structure appropriately, but one that is, at the same time, pathologically difficult for a process that is unable to discover and exploit modularity. At the microscale MC has a neat and easily identifiable modular structure, but at the macroscale it has a random structure.

**RC. Random constraints:** A symmetric weight matrix of \( n \) variables where weights take values: \( \omega_{ij} = 1 \), if \( i = j \) else \( \omega_{ij} = \{-p, p\} \) with equal probability.

**MC. Modular constraints:** A symmetric modular connectivity weight matrix of size \( N = kn \) variables: \( \omega_{ij} = R \left( \left\{ \frac{j}{n}, \frac{i}{n} \right\} \right) \), where \( R \) is a matrix from the RC definition above with size \( n \). (For example, the main experiments use \( N = 200 \), \( n = 20 \), \( k = 10 \) to define 20 modules of 10 variables each).

A small example RC matrix and corresponding MC matrix is given in Table 1. The MC matrix is simply a macroscale version of the RC matrix, but to exploit this structure successfully an algorithm will need to first identify which variables in MC belong together in a module and then search combinations of module sub-solutions as though each module were a single “emergent” variable.

The RC matrices create energy functions that tend to have many locally optimal configurations but for problems with \( n = 20 \) it is not particularly difficult to find low-energy configurations (i.e., significantly better than random configurations) using rHN-0 (i.e., a restart bit-flip hill-climber). A corresponding MC with \( k = 10 \), creating a problem with \( N = 200 \), is, in contrast, very difficult to optimize effectively with a local search process. Given that \( p \) is sufficiently small, the relatively strong and consistent intramodule weights on the block diagonal create local optima that correspond to each of the state configurations in the language, \( S = \{-1^k, 1^k\}^N \) (where \( n \) is the number of modules), and \( k \) is the size of each module (Figure 2). For example, in a MC system with \( k = 5 \) and \( n = 6 \) the state configuration \((1,1,1,1,1,1, \ldots, -1, -1, -1, -1, -1, -1, \ldots, -1, -1, -1, -1, 1, 1, 1, 1, 1)\), or \((1^5 - 1^5)^{1^5} \ldots - 1^5)\), is locally optimal. Note that such a configuration is equivalent to the state configuration \((1, -1, 1, 1, -1, 1)\) in the underlying R problem, and the set of locally optimal strings in MC correspond to arbitrary binary strings in this lower dimensional state space. Accordingly, solutions found by running rHN-0 on MC will (with respect to intermodule dependencies) be no better than random configurations in R. Moreover, the likelihood of escaping such locally optimal configurations in MC using random \( k \)-bit state changes would decrease exponentially with \( k \) even if one knew which bits needed changing (but not what values to change them to) and decreases exponentially in \( N \) if the module partitions are unknown (Watson & Jansen, 2007).

But, in the case where an adaptive system is able to identify the high-level problem structure in MC, and manipulate modules as if they were “emergent bits” of a lower dimensional macroscale problem, MC problems will become as easy as hill climbing in the underlying problem R. (Note that the ordering of rows and columns in MC may be randomly shuffled thus precluding any algorithm from exploiting adjacency information as a heuristic for identifying problem structure (Watson, 2006). Likewise, applying logical-XOR with a constant, randomly generated bit-string may be applied to the state configurations to preclude any assumption that all 1s or all -1s are special configurations.) In MC problems we will often measure only the energy contribution from intermodule constraints since the intramodule constraints are easily resolved and any resolution to them adds the same constant to total energy.

Energy-minimizing state-space trajectories in MC (as described by rHN-0) create an idealized nearly decomposable dynamical system (Simon, 1969; Watson, 2006; Watson & Pollack, 2005). The strong dependencies within modules in MC are quickly and easily resolved and the aggregate effect of one module on another is produced via many pair-wise connections that act in a coherent direction (governed by the corresponding entry in R). This effects an essentially low-dimensional interaction between any two modules (see also “R-matrix,” Wagner, Booth, & Bagheri-Chaichian, 1997). This utilizes our “modular interdependency” formalization (Watson, 2006; Watson & Pollack, 2005) of the nearly decomposable concept in an optimization framework. In practical terms MC can be loosely interpreted as a modular optimization problem such as a resource-allocation problem with modular constraints (Watson, Buckley, et al., 2011). It is worth emphasizing that if we were to approach the MC problem using a sensible problem representation that exploited the underlying \( n \)-dimensional search space defined by R, then it would be a fairly easy problem; however, our dynamical systems are not given such a representation, and without it, local search in the \( N \)-dimensional space cannot resolve the constraints of the underlying problem R. If the global optimum of R is unique then the basin of attraction for it under local search in MC may be as small as \( 2^{-n} \) of the configuration space, and to escape from any local optimum will
require a specific $k$-bit change to find a configuration of higher fitness.

### 3 Simulation results

Figure 3 shows some example trajectories of $rHN-0$ in a random (RC) problem. We see clearly that this problem has several different locally optimal configurations but it is not difficult to make significant reductions in energy (compared to the initial conditions) using $rHN-0$.

Figure 4 shows some example trajectories of $rHN-0$ in the modular problem corresponding to the random problem of Figure 3. We see that $rHN-0$ makes significant reductions in energy as we might expect. But, as predicted, $rHN-0$ is unable to resolve any of the intermodule dependencies in $MC$ and improvements shown in Figure 4a come solely from resolving the trivial-to-solve intramodule dependencies. This is revealed in Figure 4b where only the energy contributions from intermodule weights are plotted. These show almost no improvement on average because the intramodule weights are overpowering intermodule weights and causing the network to settle on whichever random configuration of $R$ was closest to the initial condition.

In Figure 5 we apply $rHN-G$ to the modular problem (reporting energies from the inter-module constraints only, as per Figure 4b). The initial trajectories are identical to those of $rHN-0$ (compare Figures 5a and 4b). The distribution of the end points of these trajectories, that is, locally optimal attractors, is shown in relaxations 1–1,000 of Figure 5b. When associative learning is activated (at relaxation 1,001 onwards) we see that within a very small number of relaxations $rHN-G$ has significantly improved its ability to resolve intermodule constraints and the distribution of attractor states is
very different from \( rHN-0 \). After associations have been learned, we see trajectories that fall in energy faster and to lower energy attractors (Figure 5c). By the end of the learning process, \( rHN-G \) is not only climbing out of the locally optimal configurations created by the intramodule weights, but it is consistently finding the best resolution of intermodule weights as well. This is because \( rHN-G \) is not (merely) equivalent to running \( rHN-0 \) on the underlying \( RC \) problem, but in fact it becomes equivalent to running itself, that is, \( rHN-G \), on the underlying \( RC \) problem—with the consequence of canalizing correlations in this reduced search space also.

Figure 5 already shows that generative associations significantly improve the ability of the system to globally optimize constraints; but is \( rHN-G \) succeeding in the manner we hypothesized? Is it transforming the scale of behavior/rescaling the search space to search in combinations of modules instead of combinations of the original problem variables? Because we know where the modules are in \( MC \) we can easily assess whether \( rHN-G \) is creating state changes that correspond to modules. Figure 6 examines the behavior of \( rHN-G \) on the modular problem in more detail. This shows that \( rHN-G \) learns the intramodule correlations quickly (Figure 6a) revealing the modular structure of the problem correctly (Figure 6b). Note that the individual squares in Figure 6b are blocks of 10 \( \times \) 10 variables, not individual pixels—this neatness results from learning intramodule correlations from locally optimal configurations. \( rHN-G \) exploits this learned structure to create correlated state changes that correspond to modules (Figure 6c–e). Thus \( rHN-G \) is effectively moving from one local minimum to

---

**Figure 3.** (a) Ten example trajectories of \( rHN-0 \) in a random (RC) problem (\( n = 20 \)). (b) The distribution of energies for initial (random) configurations in comparison to final (locally optimized) configurations (1,000 samples of each).

**Figure 4.** Ten example relaxations of \( rHN-0 \) on a modular problem corresponding to the random problem of Figure 3 (here, and henceforth except where noted, \( n = 20, k = 10, N = 200, p = 0.01, \tau = 2000 \)). (a) Reporting total energy, (b) reporting only the energy contributions from intermodule constraints. Values in (b) may increase because intermodule constraints are violated in favor of satisfying relatively strong intramodule constraints.
another, corresponding to the discrete points in Figure 2a, and hence behaving very like rHN-0 would behave in the underlying R problem.

Moreover, towards the end of this experiment, rHN-G is recalling combinations of multiple modules of very high quality (Figure 6f–i)—thus it not only learns the basic modularity of the problem in order to operate like rHN-0 at the macroscale, but additionally learns intermodule structure above the level of individual modules. This means that it is no longer using search in the space of module combinations but is simply recalling more or less complete state configurations—so the ability to exploit modularity, although vital to finding good solutions, is transient and becomes redundant (this is reminiscent of Hinton & Nowlan, 1987 and Mills & Watson, 2005 where the ability to find solutions becomes redundant when complete solutions can be remembered/inherited). There is, of course, no special feature of the system that prespecifies the scale of problem structure that might be discovered, so if higher level correlations exist, learned multimodule correlations that exploit them are to be expected (this indicates potential for exploiting hierarchical structure—see Lenaerts, Chu, & Watson, 2005 and Watson, 2006—or multiscale modularity not examined in this article).

Figure 6 shows that generative associations enable the system to sample specific high-quality configurations that are distant in Hamming space from the current configuration without visiting the lower quality intermediate solutions (Figure 1c), and this gives the system the opportunity to escape from poor quality, locally optimal solutions that would otherwise trap the local search process. These operate by effectively modifying the neighborhood of the space such that particular correlated macrovariations occur with high probability. A search trajectory using these correlated changes may violate local energy minimization by skipping over intermediate configurations of higher energy.

The ability to exploit structure in this manner is not to be taken for granted despite the neat structure of the
Figure 6. Details of rHN-G on MC. (a) The ratio of learned intramodule to intermodule average weight magnitudes. This shows that for some period the correlations identified within each module are more than five times stronger than the correlations between variables in different modules. (b) The learned weight matrix, $M$, after 30 relaxations. This shows that the intramodule (block-diagonal) weights are strongly positive whilst the intermodule weights are lower in magnitude and vary in sign ($m_{ii}$ values are visualized as 0 to facilitate appropriate image scaling). Note that the individual squares apparent in this figure are not individual pixels but blocks of $10 \times 10$ pixels (compare with Figure 7a). (c) The frequency of state changes generated by this matrix that change more than one state variable simultaneously, and the frequency with which these improve the solution (i.e., lower energy). These are measured during relaxations 30–60, showing that at this stage “block-flips” of size $k = 10$ are common, but changes smaller than this do not occur (except changes of size 1). (d) Example state configurations over one relaxation (relaxation 60) visualizing example block-flips. (e) Example trajectories (relaxations 50–60, energies measured using only intermodule constraints). (f)–(i) as per b–e, for relaxations 230–260, showing that at this later stage in learning rHN-G is making more multistate changes and a greater proportion of them are likely to make multiblock state changes that update several modules simultaneously, that is, it is recalling useful combinations of modules, and creating larger instantaneous decreases in energy.
modules in the MC problem. Figure 7 is a control experiment using relaxations of length 200 time steps rather than 2,000 (measured over the same number of learning updates, relaxations 30–60, as per Figure 6b–e). This therefore reduces the separation of timescales between the use of associations (occurring every time step) and the modification of associations (occurring at the end of each relaxation). These relaxations are not long enough to ensure that a local optimum is reached, and therefore the associations learned are much less neat than those shown in Figure 6a. Although the learned weight matrix does still show evidence of the modules on the diagonal, the average size of multivariable state changes is not neatly peaked at multiples of k (compare Figure 7b with Figure 6c), and the multistate changes that do occur do not necessarily correspond to whole blocks (compare Figure 7c with Figure 6d). As a result, the ability to reduce energy (Figure 7e) is severely impaired.

In Figure 8 we compare rHN-G with the ability of rHN-S. We see that when p = 0.1 rHN-S is able to find good quality solutions. With intermodule dependencies of this strength, all configurations that resolve intramodule constraints are (still) locally optimal, but local optima where more intermodule constraints are resolved have larger basins of attraction than those where less intermodule constraints are resolved (this is true to some extent for all nonzero p). Thus some correlations between modules are observable in the initial dynamics of the system that rHN-S can learn and exploit. However, rHN-S can only make progress when intermodule weights are strengthened sufficiently to remove local optima (Watson, Buckley, et al., 2009, 2011), and when intermodule weights are initially weak this must be done slowly to avoid accidentally reinforcing spurious correlations. rHN-G, in contrast, is not limited by weak between-module correlations (see Supplementary online material); so long as it can learn the within-module correlations (which are easy to identify from locally optimal configurations) it will be able to flip/exchange one module-solution for another as required to search configurations at the macroscale. It can thereby reveal whether one module configuration
resolves the intermodule dependencies better than another. This difficulty of learning intermodule correlations with selective associations is exacerbated when intermodule weights are weaker (Figure 8c and d), but generative associations are relatively insensitive to this change since the intra-module correlations that need to be learned are still obvious (see also Figure S1). Once the intramodule correlations are encapsulated, the energy changes gained by resolving intermodule constraints are easily revealed even if small.

Note that the difference between rHN-S and rHN-G in their ability to resolve intermodule dependencies is greatest when intermodule weights are weakest, but in the limit, if intermodule weights are very weak then there is correspondingly little to be gained by resolving them. An examination of this trade-off in a simplified version of the MC problem (Mills, 2010; Watson, Buckley, et al., 2009; Supplementary on-line material) indicates that there is, nonetheless, a significant middle ground where intermodule dependencies are difficult to learn but make a significant contribution to system energy when resolved (Figure S1).

4 Discussion

The above results illustrate how a simple distributed associative mechanism can modify the structure and subsequent behavior of a dynamical system in a manner that rescales, or coarse-grains, the dynamics of the system to a higher level of organization. This affords search in a higher level abstraction of the original configuration space and has the effect of producing more reliable and more complete minimization of the system energy. rHN-G is a self-modeling dynamical system that rescales its own dynamics over time using simple distributed mechanisms that alter the relationships between its components using only local information (i.e., pair-wise correlations). Using learned associations in the generative sense (rather than the selective sense) is intrinsic to this result. This shows...
that, in a manner previously only seen in relatively sophisticated model-building optimization algorithms, distributed dynamical systems can be transformed by very simple distributed processes into systems that solve constraints between components more reliably and more completely.

In the following sections we briefly discuss some related optimization concepts before discussing various analogues of the underlying dynamical processes in other complex adaptive systems.

4.1 Optimization and computational complexity analysis

The optimization behavior of generative associations in $rHN-G$ depends on the presence of a problem structure that can be recognized and exploited. However, in related work (Mills, 2010), results on minimizing the energy of a two-dimensional Ising model demonstrate that an explicit modular structure, as per $MC$, is not required to observe the advantages of an algorithm that automatically identifies and exploits problem decomposition in this manner. In this case, correlations in locally optimal configurations reveal the lattice structure of the problem and learned associations enable a rescaling that effects changes in “domain-spins” rather than the spins of individual variables. This work also shows that whereas $rHN-S$ requires intermodule dependencies to be strong in order to learn and exploit them, an algorithm like $rHN-G$ can exploit modules even in the case where intermodule dependencies are obfuscated by relatively strong within-module constraints (see also, Supplementary on-line material).

In the results above we see that the relevant modules are discovered surprisingly quickly—within about 30 relaxations. The lower limit on this number is determined by the probability of observing within-module correlations without observing spurious between-module correlations. In fact, any single training sample will reinforce some association between any pair of modules, so learned associations must develop slowly enough for many observed configurations to average out spurious correlations. Analytic work (Mills, 2010) shows that this can be done in surprisingly few observations when local optima are used as the training samples. (In Mills, 2010, each observation is independent, whereas in $rHN-G$ even the first observation biases the second observation to some extent; however, the results here indicate that this is not a significant hindrance.)

Our model therefore illustrates how a distributed learning mechanism can be used to identify and exploit problem structure efficiently without a priori domain knowledge. One might reasonably question, however, exactly how the cost of inducing a model (that subsequently makes solving the problem easy) compares with the cost of “brute forcing” the solution without a model. In the Supplementary on-line material we show formally that it is possible in some problems to learn the model and solve the problem in time that is a polynomial function of the system size whereas solving the problem without model-building capabilities takes exponential time. This result is dependent on the presence of appropriate problem structure, of course (Wolpert & Macready, 1997), but there exist systems, namely modular ones, where the mechanism demonstrated can find low-energy configurations in polynomial time that cannot be found by a nonadaptive model in less than exponential time on average. The Supplementary material also shows a numerical study of the sensitivity of $rHN-0$, $rHN-S$ and a variant of $rHN-G$ (Mills, 2010) to the strength of intermodule connections, $p$. This study shows that there is a significant region of $p$ where the problem is difficult for $rHN-0$ and that both $rHN-S$ and $rHN-G$ offer an advantage in this region. But whereas $rHN-S$ finds low $p$ increasingly difficult (because intermodule dependencies are overpowered by intramodule dependencies), $rHN-G$ is almost insensitive to $p$.

4.2 Building blocks and model building

The idea of discovering and exploiting modules has been a “holy grail” for a subschool of evolutionary computation ever since the inception of the field (Goldberg, 1989; Holland, 2000). Initially, the idea was that sexual recombination or “crossover” would automatically identify good combinations of genes (“building blocks”) and combine them to find better, larger combinations of genes. However, there has been significant difficulty in illustrating that genetic algorithms, GAs, can achieve this (Forrest & Mitchell, 1993; Watson, 2006). Importantly, it is crucial to realize that, although a GA with crossover can, under some circumstances, identify and recombine building-blocks with provably superior time complexity to a noncrossover GA (Watson, 2006; Watson & Jansen, 2007; Watson, Weinreich, & Wakeley, 2011), this advantage relies intrinsically on the correspondence between epistatic dependencies and physical linkage, that is, the bits of a building block need to be next to each other on the genome. Thus the canonical GA cannot solve a problem like that which we address here, where adjacency information cannot be used (note that $rHN-G$’s ability to learn linkage information from observed correlations is entirely insensitive to the ordering of problem variables).

Accordingly, there has been considerable research effort directed at methods that discover and exploit building blocks without presupposing this “tight linkage,” as it is known. So called “linkage learning” GAs and “model-building” optimization algorithms include “estimation distribution algorithms” like the Bayesian...
optimization algorithm (Pelikan, 2002; Pelikan et al., 2002; Yu, Goldberg, Yassine, & Chen, 2003). rHN-G can be understood as a form of model-building optimization algorithm (see, in particular, bivariate models, e.g., Yu et al., 2003) but in addition to the differing conceptual motivations and distributed implementation there are also two important algorithmic distinctions (Iclanzan & Dumitrescu, 2007; Mills, 2010; Mills & Watson, 2011b): (a) Whereas previous methods have used induced models to bias the initial conditions of further search, we use the model to provide variation in a lower dimensional space. This is essential to scale up the search process properly (as per the original building-block intuition, Watson, 2006). (b) Whereas previous methods build a model from correlations observed in above-average-fitness configurations, we build the model from locally optimal configurations. This greatly amplifies the “signal” from which correlations can be learned and helps to separate meaningful correlations from spurious “noise” (compare Figures 6 and 7). A separation of timescales, using repeated local search from many different initial conditions whilst slowly learning problem structure, is essential to achieve this (Mills, 2010; Watson, Buckley, et al., 2011). These features (see also Iclanzan & Dumitrescu, 2007) are central to the behavior illustrated in this article and are developed formally in related work on multiscale search (Mills, 2010; Mills & Watson, 2011b). But crucially, in this article these behaviors are modeled in a distributed system without using centralized data structures or a centralized control architecture.

4.3 Implications for analogous systems

Cognitive problem solving Although we are using a neural network model, our main research questions are not motivated by cognitive systems. Nonetheless, distributed constraint optimization is relevant to cognitive problem solving (Ackley et al., 1985; Hinton & Sejnowski, 1983). Implications for problem solving of the self-modeling framework with Hebbian learning are discussed by Fernando, Goldstein, and Szathmáry (2010). These authors use Hebbian learning to modify the trajectory of a random-mutation hill-climbing process as per rHN-S.1 Fernando et al. suggest that the ability to (re)structure a local search process in this manner is relevant to solving “insight problems” that are otherwise problematic for local search processes. The additional ability (as demonstrated by rHN-G) to abstract-away or encapsulate low-level details and thereby reason about a system using higher level features or concepts, suggests aspects of cognitive problem solving that are also intuitive (Chase & Simon, 1973; Miller, 1956) and have been exploited in classical (symbolic) AI (Rosenbloom, & Newell, 1986), but not previously shown in a distributed neural architecture. Crucially, such processes enable improvements to a proposed solution/neural activation pattern that cannot be explained by incremental improvement in the original space of problem variables/neural activations because they involve (individually inferior) changes to multiple variables that are beneficial only in combination.

Multiagent systems/games on networks The default dynamics defined by the Hopfield equation are analogous to those of a multiagent system where agents play pair-wise games on a network (Davies et al., in press; Pacheco et al., 2006; Van Segbroeck et al., 2010). Each connection in the network represents a game between two agents. The pay-offs for those games, defined by $Q$, constitute a mixture of coordination games ($\omega_{ij} > 0$) and anti-coordination games ($\omega_{ij} < 0$) (Davies et al., in press). Each agent (node) adopts the strategy (state) that maximizes their pay-off given the conflicting constraints of the multiple games in which they participate. The symmetric nature of connections ($\omega_{ij} = \omega_{ji}$) ensures that each interaction is a potential game, and the fact that the utility of an agent is a sum over many potential games ensures that the total system can be described as a potential game also (Chapman, 2009; Chapman, Rogers, Jennings, & Leslie, in press), therefore exhibiting only fixed-point attractors/Nash equilibria. Solving the constraint optimization problem defined by $Q$ is then equivalent to the problem of maximizing total welfare in the corresponding multiagent network (Chapman, 2009), and local optima found by rHN-0 correspond to Nash equilibria where no single agent can improve their utility by changing strategy unilaterally.

Our recent work shows that if agents gradually change their “perception” of the utility function with another player, so as to prefer the situations they currently find themselves in, then these preferences modify subsequent agent interactions according to Hebbian principles (Watson, Mills, et al., in press), causing the system as a whole to improve its ability to find configurations of high total welfare in the same manner as rHN-S (Davies et al., in press). This is closely related to a simple form of reinforcement learning where agents play the strategy that maximizes utility given the expected strategies adopted by other agents based on past behavior (Leslie & Collins, 2006; Watson, Mills, et al., in press; Davies et al., in press). But given rapid relaxations and slowly changing associations (Watson, Buckley, et al., 2011), a simple habituation of observed correlations at equilibria achieves the same outcome as a preferential reinforcement of good correlations (Davies et al., in press). Generative associations in such a system would be analogous to the formation of coalitions where agents use a correlating mechanism.
(i.e., the random variable \( X \), in Equation 7) to enact simultaneous strategy changes and share the utility gained (i.e., adopt strategy changes if they increase group utility) (Chapman, 2009; Chapman et al., in press; Hart & Mas-Colell, 2000; Leslie & Collins, 2006; Young, 2001).

**Ecological networks** Our recent work has investigated a Lotka–Volterra model of an ecosystem of multiple species with evolvable interspecies fitness dependencies analogous to selective associations. This work shows that natural selection at the species level to alter these fitness dependencies results in associative learning at the system level (Lewis, 2009). An abstract model of evolved symbiotic relationships that effect changes in the unit of selection (Mills & Watson, 2011a; Watson, Palmius, Mills, et al., 2011; Watson, Palmius, Jackson, et al., 2011; see also Powers, Mills, Penn, & Watson, 2009; Powers, Penn, & Watson, 2007; Powers, 2010 for discussion of general concepts) illustrates \( rHN\)-style adaptation in an evolving ecosystem. This latter work provides a model for the major transitions in evolution where “entities that were capable of independent replication before the transition can replicate only as part of a larger whole after the transition (p. 6)” (Maynard Smith & Szathmary, 1995) (Watson, 2006; Watson & Pollack, 2003). In that work we endow agents with the ability to affect the probability of “co-dispersal” with other species and we observe that the associations they evolve create new units of selection that canalize naturally co-occurring species in the same manner that \( rHN\)-G creates “emergent macro-variables” that canalize partial solutions. In the current neural network model we are better able to formalize the adaptive capabilities of this process and understand its relationship to associative learning.

**Gene expression networks** At a completely different biological scale, the same principles apply to the evolution of connections in a gene expression network (Watson et al., 2010). That is, if a gene expression network, controlling the mapping between genotype and phenotype, is evolved on a given fitness landscape it will tend to increase the robustness of, or “canalize” (Riegler, 2008; Siegal & Bergman, 2002; Wagner et al., 1997), phenotypes that are locally optimal. Evolved changes to the connections in this network therefore build up an associative memory of past selective environments that enables superior ability to maximize fitness. We are not yet sure whether this result is analogous to \( rHN-S \) or \( rHN-G \)—both seem to be possibilities.

Since canalization reduces the effective degrees of freedom of a system, it might seem counterintuitive that it has the effect of increasing adaptability or evolvability in this manner. But canalization and evolvability are really two sides of the same coin (Draghi, Parsons, Wagner, & Plotkin, 2010; Kirchner & Gerhart, 1998; Watson, et al., 2010). That is, removing some degrees of freedom whilst retaining others can enhance the ability of the system to find solutions by focusing variation in useful regions of configuration space. These ideas are related to evolved exploration distributions (Izquierdo & Fernando, 2008; Jones, Arnold, & Bürger, 2007; Toussaint & von Seelen, 2007) and “facilitated variation” where an evolved “memory” of past selective environments and generalization to new selective environments has also been suggested (Parter, Kashlan, & Alon, 2008).

**General dynamical systems** The notion that self-organization can transform a high-dimensional (disordered) system into a low-dimensional (ordered) system is familiar in many domains. Often we are interested in characterizing the emergent low-dimensional variables (see “order parameters,” Haken, 1983) to provide a description that represents this order, and in using these to characterize a transformation in the system dynamics to a higher level of organization (Fuller, 1975). Clearly, the behavior of \( rHN\)-G with learned associations on an \( N\)-dimensional \( MC \) problem has a low-dimensional description—the learned associations reduce the effective degrees of freedom of the system. And it arrived at this configuration of associations by self-organization—by the application of simple local rules that change the relationships between system components in a manner determined by its own dynamics.

However, we are not merely interested in the fact that the system has a low-dimensional description, but more specifically, we are interested in the similarities and differences between this new low-dimensional behavior and the original behavior of the system. That is, after some learning (Figure 6b–e) \( rHN\)-G behaves like \( rHN-0 \) on the underlying \( n(=N/k)\)-dimensional \( RC \) problem. To some degree, this is similar to a coarse-grained description of the behavior of its former self, that is, of \( rHN\)-G without learned associations (i.e., \( rHN-0 \)) on the full \( N\)-dimensional \( MC \) problem. Thus, the emergent variables in \( rHN\)-G reflect the modularity/correlations in its initial dynamics. But \( rHN\)-G with learned associations is not exactly a low-dimensional description of its former self—if it were, it would reach the same attractors as \( rHN-0 \). But \( rHN\)-G does not find the same distribution of attractors that \( rHN-0 \) finds on the full \( N\)-dimensional \( MC \) problem; it finds the distribution of attractors that \( rHN-0 \) finds on the underlying \( n\)-dimensional \( RC \) problem. Accordingly, \( rHN\)-G is not just transforming into any low-dimensional system, nor, more specifically, into a low-dimensional model of itself—but, in fact, it is transformed into a new system, one with improved ability to minimize energy. It does so by creating shortcuts in the
neighboring structure of state-configuration space; enabling movements in a lower dimensional space that skips over intermediate configurations of higher energy.

We can make some sense of this idea of a dynamical system that spontaneously improves by taking a “metadynamical system” perspective (Bourgine & Varela, 1992). Because the association variables are treated separately from the state variables we can treat them as parameters of the state dynamics even though they are in actuality (slow changing) variables. In the joint space of state variables and associations together we simply observe a dynamical system doing what it does naturally—there is no sense in which the system is improving its ability to minimize energy. But if we regard the associations as parameters of the state dynamics, then we can characterize the changes in the associations by how they transform the dynamics of the state variables. In particular, we can assess whether this transformation improves the ability of the state dynamics to minimize energy, as we have shown. This enables us to characterize the system behavior as “problem solving” rather than just self-organizing, and it enables us to say that the system is transformed into a different system (with respect to the behavior of its state dynamics), even though at the same time it is merely the same system in a new configuration (when described in terms of states and associations together). Likewise we can characterize how the evolution of robustness effects an increase in evolvability (an improvement in ability to evolve), even though at the same time it is merely the (normal) evolution of a given system (the genotype and the gene regulation network together) that produced this result (Watson et al., 2010). This metadynamical system perspective is useful in other, non-network, models too—for example, the evolution of social traits with the concurrent evolution individual traits that alter the contextual parameters of that social behavior (Powers, Penn, & Watson, 2011). But in an adaptive network, the manner in which changes to connections can alter state dynamics is much richer than that which is possible in non-network-based metadynamical systems and specifically has potential to exploit principles of associational memory.

5 Conclusions

This article uses the Hopfield network as a model system to extend the emergent collective computational abilities (Hopfield, 1982) of dynamical systems. This provides a concrete model of how a self-organizing dynamical system or adaptive network can transition from one scale of dynamics to another over time as a result of the changing relationships between components. The processes involved use only very simple, and fully distributed, positive-feedback mechanisms, which are relevant to other active-linking systems and adaptive networks, beyond neural network models (Watson, Mills, et al., in press). We have been investigating the effect of selective associations in various adaptive networks such as social (Davies et al., in press), genetic (Watson et al., 2010), and ecological networks (Lewis, 2009). However, the generative associations studied in this article offer a transformation in the dynamical behavior of a system, and the ability of the system to minimize energy, that is not observed with selective associations. Specifically, whereas selective associations reduce the system’s ability to minimize energy, that is not observed with selective associations. Specifically, whereas selective associations reduce degrees of freedom by guiding state trajectories in particular directions, generative associations “fold-away” redundant degrees of freedom, effectively encapsulating partial configurations/partial solutions, and skipping over energy barriers/fitness valleys rather than trying to smooth them out. Computationally, rHN-G thus provides a distributed optimization algorithm with similar capabilities to our formal optimization algorithms that explicitly operate over several scales of organization (Mills, 2010; Mills & Watson, 2011b).

This suggests a neurally plausible algorithm with potential importance for cognitive problem solving, but because the model with generative associations is still fully distributed it is also relevant to non-neural adaptive networks. For example, the behavior exhibited in the evolution of new units of selection (Watson, Palmius, Mills, et al., 2011) is closely analogous to the behavior of generative associations in the Hopfield model shown here. In fact, it seems that the difference between selective and generative associations is closely analogous to the difference between ordinary coevolution and symbiosis that creates new units of selection, respectively (Watson, Palmius, Jackson, et al., 2011). Together this suggests the exciting idea that we can view the organization of symbiotic relationships in an ecosystem as a distributed optimization process greater than the sum of the parts, that is, over and above the evolution of the individual species.

In conclusion, we find that it is possible for a dynamical system to be modified by generative associations into a new system, operating at a higher level of organization, which minimizes total system energy more reliably and more completely. The Hopfield model with generative associations offers an algorithmic framework for understanding transformations and emergent collective behavior in various (non-neural as well as neural) domains, for example, the emergence of coalitions in social systems and of higher level adaptive units in the major transitions in evolution.

Note

1. Although these authors attribute the improved optimization ability to natural selection (Fernando & Szathmáry, 2010), no replication is necessary to implement this
algorithm, as shown here and in (Watson, Buckley, et al., 2009, 2011), and the associative network employed in (Fernando et al., 2010) is not evolved but learned via Hebb’s rule from locally optimal solutions as per rHN-S.

Acknowledgments
Thanks to Simon Powers, Alex Penn, Archie Chapman, David Iclanzan, Jason Noble, Seth Bullock, Jordan Pollack, and Chrisantha Fernando for helpful discussion.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References
Davies, A. P., Watson, R. A., Mills, R., Buckley, C. L., & Noble, J. (in press). If you can’t be with the one you love, love the one you’re with: How individual habituation of agent interactions improves global utility. Artificial Life.


**About the Authors**

**Richard Watson** was awarded his Ph.D. degree in computer science from Brandeis University (USA) in 2002. His background in computational models of evolution was complemented by a research fellowship at Harvard University in the Department of Organismic and Evolutionary Biology. Since 2004 he has been a senior lecturer in the Natural Systems group, part of the School of Electronics and Computer Science at the University of Southampton (UK). In 2006 he was a recipient of the IEEE international award “Ten to Watch in Artificial Intelligence” as part of the IEEE’s celebration of 50 years of AI. He is presently Vice President of the International Society for Artificial Life. His publications span topics including artificial life, robotics, evolutionary modeling, population genetics, evolutionary computation, and computational biology; and he is author of *Compositional Évolution: The Impact of Sex, Symbiosis, and Modularity on the Gradualist Framework of Evolution* with MIT Press. His current research focuses on understanding the algorithmic principles of natural processes that create adaptive biocomplexity.
Rob Mills is a postdoctoral research fellow in the Natural Systems group, part of the School of Electronics and Computer Science at the University of Southampton. His research interests, bridging the domains of evolutionary computing and evolutionary biology, are focused around understanding the mechanisms that lead to the modular organization of interactions in evolving systems, as well as the adaptive consequences arising from such organization. Other interests include the evolution of evolvability, innovation, and multiscale processes; and the interaction between learning and evolution. Address: School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK. E-mail: rob.mills@soton.ac.uk

Christopher L. Buckley is a research fellow in the Centre for Computational Neuroscience and Robotics at the University of Sussex. His research involves using the tools and techniques of non-linear dynamical system theory to understand the functions of the nervous system. His current focus is on invertebrate sensory (moth antennal lobe) and motor (snail feeding CPG) circuits. He also has an interest in nonlinear dynamical phenomena in biological systems more generally. Address: School of Informatics, University of Sussex, Sussex House, Brighton, BN1 9RH, UK. E-mail: C.L.Buckley@sussex.ac.uk