Computing and Optimizing Over All Fixed-Points of Discrete Systems on Large Networks

James R. Riehl[1*], Maxwell I. Zimmerman[2], Matthew F. Singh[1], Gregory R. Bowman[2], ShiNung Ching[1,3,4]

[1]Department of Electrical and Systems Engineering, Washington University in St. Louis, 1 Brookings Dr, St. Louis, MO 63130,

[2]Department of Biochemistry and Molecular Biophysics, Washington University in St. Louis, 660 S Euclid Ave, St. Louis, MO 63110

[3]Department of Biomedical Engineering, Washington University in St. Louis, 1 Brookings Dr, St. Louis, MO 63130

[4]Division of Biology and Biomedical Sciences, Washington University School of Medicine, 660 S Euclid Ave, St. Louis, MO 63110

*To whom correspondence should be addressed; E-mail: jrriehl@wustl.edu.

Abstract

Equilibria, or fixed points, play an important role in dynamical systems across various domains, yet finding them can be computationally challenging. Here, we show how to efficiently compute all equilibrium points of discrete-valued, discrete-time systems on sparse networks. Using graph partitioning, we recursively decompose the original problem into a set of smaller, simpler problems that are easy to compute, and whose solutions combine to yield the full equilibrium set. This makes it possible to find the fixed points of systems on arbitrarily large networks meeting certain criteria. This approach can also be used without computing the full equilibrium set, which may grow very large in some cases. For example, one can use this method to check the existence and total number of equilibria, or to find equilibria that are optimal with respect to a given cost function. We demonstrate the potential capabilities of this approach with examples in two scientific domains: computing the number of fixed points in brain networks and finding the minimal energy conformations of lattice-based protein folding models.

Introduction

A fundamental element in the study of physics and dynamical systems is the notion of fixed points or equilibria. In the absence of exogenous input or disturbance, a system will not deviate from a fixed point, and the dynamics in the neighborhood of that point determines its stability, i.e. whether it is an attractor, repeller, or other type of critical point. Indeed, the state of a system (for our purposes, defined on a network) evolves in time on a state space that is landmarked by equilibria. Knowledge of these equilibria can therefore help to predict long term evolution of system trajectories. It is often particularly useful to know the equilibria that are optimal with respect to some cost function, for example, to find ground states or minimum energy configurations of materials, protein sequences, or brain networks, or to determine the most cost efficient or beneficial states in social or economic networks.

We focus here on dynamical systems where the dependency structure of the components of the state is described by a graph, and each component can take one of a small number of values. These include simplified neural network models, models of decision or opinion dynamics on social and economic networks,

evolutionary games on networks, and chemical reaction and protein folding models. The notion of an equilibrium or fixed point has wide-ranging interpretations across these applications. In neural network models, fixed points represent invariant patterns of activation that can be associated with memories, cognitive task goals, or default states of awareness [1]. In protein folding models, the fixed points correspond to physically realizable protein structures [2]. Equilibria in social networks can be interpreted as states in which all individuals are satisfied with their respective actions, opinions, or votes [3]. A tool for computing the fixed points in these systems would thus be of great consequence across scientific domains.

Unfortunately, even for discrete valued systems, finding all fixed points is computationally infeasible [4, 5]. All is not lost however, if the network under consideration has certain advantageous properties. In particular, if the network topology can be repeatedly partitioned by cutting no more than some fixed number (k) of edges, we say it is *k-separable*, and the approach we put forth in this paper can find the solution with a computation time that is linear in the size of the network and exponential only in local properties. When this holds for sufficiently small values of k, the proposed method can find all equilibria for very large systems, but even when it does not hold, we can solve the problem on much larger networks than with the more naive methods that are typically used in practice. Accordingly, the main contribution of this work is a provably tractable method for searching the global equilibrium space of discrete-valued systems on k-separable networks, using a recursive procedure of graph partitioning and sparse set operations, augmented by efficient bookkeeping. We emphasize that the algorithm yields the *exact* set of global fixed points in a sparse representation, from which individual equilibria can be queried according to some desired criteria, for example, those that minimize an energy function. We also show how to quickly check the existence and total number of equilibria, and explain how broader characterizations such as fixed-point landscapes can be obtained without generating the entire equilibrium set.

To highlight the practical versatility of this method, we provide two case studies. In the first, we analyze the dynamics of functional brain networks extracted from the Human Connectome Project database. We show that such networks exhibit exponential growth in the number of fixed points as a function of interconnected brain regions, a prediction made in theoretical studies but here verified empirically for the first time. In the second case study, we address the decades-old challenge of finding low-energy configurations of proteins, toward facilitating a deeper understanding of protein evolution and design. We consider a prototypical protein lattice model and are able to efficiently find the global energy minima as well as perform an exact characterization of the ruggedness of the fixed-point landscape. Each of these case studies features a stochastic real world system for which a deterministic system is constructed and used as an analytical tool. Specifically, we define the deterministic system such that its fixed points contain the set of energy extrema of the corresponding stochastic system.

Discrete systems and their fixed points

We consider discrete-time and discrete-valued dynamical systems in the general form

$$\mathbf{x}(t+1) = f(\mathbf{x}(t)), \tag{1}$$

with state vector $\mathbf{x}(t) := [x_1(t), \ldots, x_n(t)]^\top$. Each component of the state $x_i(t)$ can take one of a finite set $\mathcal{S}_i$ of possible values. Define a directed graph $\mathbb{G} = (\mathcal{V}_0, \mathcal{E}_0)$ in which each node in $\mathcal{V}_0$ corresponds to a component of the state, and the edges $\mathcal{E}_0$ define dependencies in the system dynamics. Namely, an edge $(i, j) \in \mathcal{E}_0$ indicates that $x_i(t+1)$ may depend on $x_j(t)$. Conversely, the lack of an edge means there is no dependency. The dynamics are therefore separable into the components $x_i(t+1) = f_i(\mathbf{x}(t))$, in which f_i only depends on the states of node i and its neighbors in the network.

A fixed point or *equilibrium* $\mathbf{x}^*$ of the system is a state in which no node will change between time steps, i.e. $f(\mathbf{x}^*) = \mathbf{x}^*$. Let $\Omega(\mathcal{V}_0)$ denote the set of all such equilibria. Since $f(\mathbf{x})$ is in general nonlinear, computing $\Omega(\mathcal{V}_0)$ is a challenging task, and the main contribution of this work is a method for exploiting the network topology and local dynamics to compute a sparse representation of $\Omega(\mathcal{V}_0)$ in an efficient way.

Results

Our main result is an algorithm that computes and optimizes over the global equilibrium set of system (1). In this section, we describe how to do this using a combination of local dynamical analysis and recursive graph partitioning, discuss the computational complexity and related literature, and demonstrate the approach on practical applications in biochemistry and computational neuroscience.

Finding and optimizing over all fixed points

The approach begins by computing the set of *local equilibrium* (LEQ) states for each node in the network. For a given node i, we define the LEQ as the set of all system states $\mathbf{x}$ such that $x_i(t)$ will not change at time $t+1$ according to the local update rule $f_i(\mathbf{x}(t))$:

$$\mathcal{L}_i := \{\mathbf{x} \in \mathcal{S} : x_i = f_i(\mathbf{x})\}.$$

For sparsely connected networks in which the numbers of state values $|\mathcal{S}_i|$ is not too large, these sets are fast to compute and can be efficiently stored using the sparse representations described in **Methods**. An important observation is that the global equilibrium set is equal to the intersection of all LEQ sets. Unfortunately, computing all of these intersections may be too difficult for large networks. However, by using graph partitioning while retaining necessary information for each partitioned region, we can decompose the problem in a way that allows for exact construction of the full set of equilibrium points (see **Methods: Recursive partition-based intersections**). Specifically, by intersecting the LEQ sets within each partitioned subnetwork, we form a set of *regional equilibrium* (REQ) states. This effectively reduces the problem to finding

all compatible REQ states on the partitioned subnetworks, which can be achieved by intersecting neighboring LEQ sets if the subnetworks are small enough, or otherwise by performing an additional partition and proceeding recursively. The end result is a top-level REQ set along with a hierarchical data structure $\mathcal{T}$ that efficiently encodes all information necessary to generate the full equilibrium set $\Omega(\mathcal{V}_0)$.

What renders this algorithm tractable for k-separable networks is both the recursive decomposition and the fact that we need only a compact representation of the sets (including only the nodes on the partition boundaries) in order to construct higher level REQ sets and ultimately the global equilibrium set. The computational complexity thus depends primarily on the node degrees (number of connections to each node) and the number of edges that cross partition boundaries. Another key element in the approach is the construction of maps between the various stages of partitioning, which allow for quick access to the hierarchy of global, regional, and local equilibrium sets. Finally, if an energy or cost function is provided, the algorithm computes costs at each stage, which can be used to efficiently extract the optimal equilibrium states with respect to the given energy function. See **Methods** for a detailed presentation of this approach.

Computational complexity

Special cases of finding all fixed points of system (1) such as deciding on the existence of a Nash equilibrium for certain systems [5] and calculating the minimum-energy state of ferromagnetic spin models [4] are proven to be *NP-hard*. This generally means that the time required by any existing algorithm to compute a solution is exponential in the size of the problem (e.g., number of nodes in the network). For example, the naive approach of checking for fixed points among all possible configurations in a network in which nodes can take one of two states would require a time proportional to 2^n, and it is often not trivial to improve much on this brute force technique. When approximation is sufficient, methods such as simulated annealing [6, 7] and genetic algorithms [8, 9] can be effective in cleverly exploring the enormous state space, but these algorithms can generally only provide probabilistic guarantees on finding fixed points or global extrema. Otherwise, branch-and-bound algorithms [10] can reduce computation time by safely eliminating large regions of the total search space without compromising optimality. However, these methods often require significant effort in tailoring to a specific problem and their objective is typically to find a single global extremum. A recent approach to finding the global energy minimum on Ising lattices involves pairing combinatorial with convex optimization to converge upper and lower bounds on the solution, and was demonstrated on three-dimensional lattices containing up to 50 nodes, each of which can take two states [11]. In comparison, the method proposed here found the global minima and all fixed points on 3-dimensional lattices of 64 nodes, each of which can take three states (see **Case Study 2**), and is demonstrated in on other types of networks having hundreds and thousands of nodes (see Fig. 1).

Significant effort has also been put toward finding attractors in *Boolean networks*, in which nodes take one of two values (0 or 1) and local dynamics are based on logic rules. This work is most closely associated with genetic regulatory network models, where nodes represent genes that are either expressed (1) or not (0),

and edges indicate causal links between genes. Indeed, for the special case when each set $\mathcal{S}_i$ contains exactly two elements, system (1) is equivalent to a Boolean network. Several approaches have been proposed to find attractors in Boolean networks, including simulation [12], aggregation [13], binary decision diagrams [14], and satisfiability algorithms [15]. In particular, satisfiability rules paired with a partition of the Boolean network are used in [16] to find attractors in sequential and parallel implementations. We emphasize here three conceptual advancements over previous aggregation or partition-based techniques for finding attractors [13, 16]: (i) one level of partitioning may not be sufficient to make computation on large networks tractable – the extension to arbitrary levels of recursive partitioning yields what is to our knowledge the first provably tractable algorithm for computing fixed points on a particular class of networks, namely, those with local connectivity structure captured by the measure we call partition separability; (ii) computing the energy function at each stage of the decomposition allows for useful characterizations of the attractor landscape without explicitly computing the entire set of attractors, which may become prohibitively large; (iii) extending to more general discrete-valued networks makes it possible to analyze networks in which node states can take more than two values.

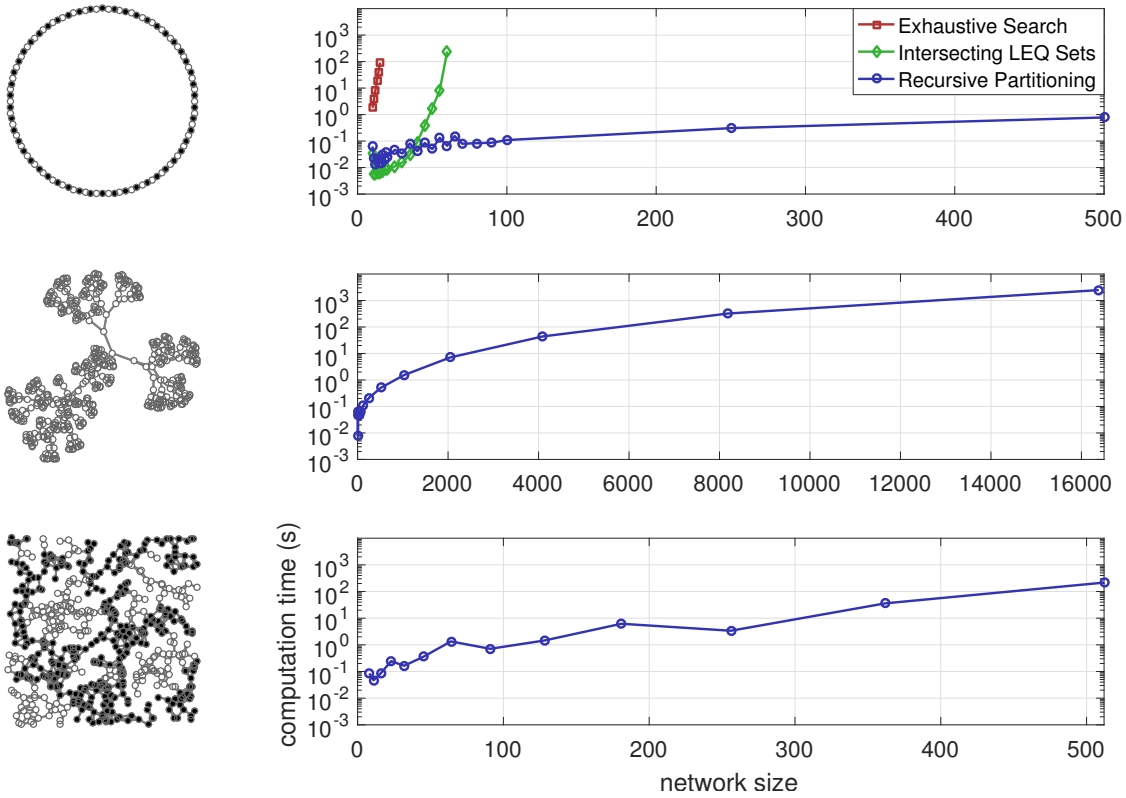

Figure 1: Time required to compute energy-extremal fixed points for three different network types: ring (top), tree (middle), and geometric random network (bottom). Node states are governed by linear threshold dynamics and energy is evaluated using a generalized Ising model (see **Supplementary Information (SI): Best-response and linear threshold dynamics**). Computation times for exhaustive search and intersecting all LEQ sets on ring networks are shown in the top panel. Black and white node colorings in the network diagrams respectively depict the states -1 and +1 in energy-extremal fixed points.

The key characteristic in determining how much advantage will be gained by using this approach is how easily the network can be decomposed into smaller parts. We define the *partition separability* of a network as the maximum number of edges that cross partition boundaries over all partitions used in the decomposition. For network topologies such that the partition separability is independent of network size, the computation time of the proposed algorithm is exponential only in local network properties, namely, the number of local connections or *degree* of the nodes. The top panel in Fig. 1 shows the substantial reduction in computational complexity when using the recursive partitioning approach compared to exhaustive search or intersecting all LEQ sets to compute all fixed points in networks governed by two-state best-response dynamics (See **SI: Best-response and linear threshold dynamics**). We note that the local intersection method is already a steep improvement over exhaustive search. While ring networks are particularly well-suited to this approach since they are 2-separable, perhaps even better suited are tree networks, which are 1-separable. Computation times for optimizing over the entire equilibrium space on tree networks are shown in the middle panel of Fig. 1.

Indeed, we can prove that the algorithm is tractable for k-separable networks. Let $\hat{d}$ denote the maximum degree and $\hat{s}$ the maximum number of local states $|\mathcal{S}_i|$ for any node $i \in \mathcal{V}_0$ in the network, while q denotes the number of partition groups at each level, and r is the maximum number of nodes in a group at the lowest level of partitioning. The following theorem provides the parameterized computational complexity of the proposed algorithm, namely, that it is linear in the size of the network and exponential only in the node degree and the partition separability of the network (proof in **Appendix**).

Theorem 1. *The complexity of computing the top-level REQ set $\Omega^p(\mathcal{V}_0)$ and the results tree $\mathcal{T}$ for a k-separable network is at most $O(\alpha n)$, where $\alpha := kq\hat{s}^{4kq} + r\hat{s}^{r\hat{d}}$.*

For other types of networks, the algorithm still faces the inevitable hardness constraint, but the complexity is still reduced to the degree and separability properties, allowing solutions on much larger networks than possible through more naive methods. For example, we can find all fixed points of these dynamics on geometric random networks, such as in the bottom panel of Fig. 1, provided they are sufficiently sparse and separable. However, some other standard graph topologies are less well-suited to this approach. For example, since Erdos-Renyi random graphs have no local connectivity bias, the partition separability will always grow with the size of the network, and while scale free networks might have some local connectivity, the large degrees of hub nodes can be prohibitive even for the calculation of local equilibrium states. Determining the partition separability of a given network is itself a computationally complex problem, but there exist fast graph partitioning algorithms, e.g. [17, 18], which can be used to approximate this quantity. See **SI: Computational complexity analysis** for a comparison of the separability of several different network types, as well as a detailed analysis and proof of the computational complexity.

Case Study 1: Finding the critical points in brain networks

A long-standing model of neural networks endows each neuron with a binary state representing whether it is active (firing) or inactive (not firing) [19]. Edges in such a model represent synaptic connections, which can be either excitatory, i.e. the firing of one neuron causes another to be more likely to fire, or inhibitory, i.e. the firing of one neuron causes another to be less likely to fire. Activation of a neuron is determined by a threshold on its presynaptic (incoming) activity.

In theoretical neuroscience, this class of model at the neuronal scale was initially used as a simplified descriptor of associative memories [1], where each fixed point in the system represents a complex but stationary pattern of neural activity corresponding to a particular memory, and, in the case of attractors, the region of attraction around the fixed point determines how easily the memory is recalled. Later work has examined stochastic variants of this model, closely related to the Ising model, to assess the role of correlations within neural populations and their effect on neural coding and information processing [20]. More recently, such models have also been used as the basis of theoretical studies to show how properties of neuronal networks might arise as a consequence of associative learning, wherein associations are encoded as equilibria of the underlying networks (27). The notion of energy via the Ising Hamiltonian is central to these analyses.

However, despite the relative simplicity of this model, many contemporary approaches rely on simulation studies and post-hoc statistical analyses [21]. Because the number of parameters in the model increases quickly with network size, such simulation-based approaches are inherently limited. Effort is thus being directed towards methods that can simplify the analyses of such models, especially as it pertains the model dynamics [22].

The overwhelming majority of functional network characterizations rely on static, graphical descriptions based on calculation of the pairwise correlation of the activity over a set of brain regions. Using dynamical systems models at such scales offers the potential for greater explanatory power relative to purely statistical descriptions [23, 24]. Such models span different spatiotemporal levels of description, from detailed biophysical models at neuronal scales, to mesoscopic, mean-field approximations at the level of brain regions. Often, these models incur a tradeoff between model complexity (including physiological interpretability/abstraction) and analytical tractability. The Ising framework provides a more abstract description of brain network activation, and has primarily been used as a means of analysis, especially in regards to establishing correlation between brain regions. However, the model itself does carry dynamical interpretability and recent efforts have highlighted its potential to generate hypotheses regarding the dynamical underpinnings of observed statistical outputs. For example, [25] uses the Ising framework to suggest how the architecture of a whole-brain scale network might impact its 'repertoire' of achievable states, as encoded through the network's equilibria. Such model-based analysis and hypothesis generation is in the spirit of what we highlight in our Case Study. Indeed, an emerging area of study in functional neuroimaging pertains to nonstationary fluctuations in network covariance [26]. At a dynamical level, such fluctuations are

fundamentally mediated by the attractor landscape of the system. However, given the high dimensionality, finding and characterizing all attractors is generally intractable. Here, as a first step to an attractor landscape characterization, we quantify the fixed points of 783 functional brain networks from the Human Connectome Project by ascribing linear threshold dynamics to the functional connectivity networks (see **Methods: Processing of data from Human Connectome Project**).

We observe in Fig. 2 that the number of fixed points increases exponentially with network size, with an exponent that varies for different resting state networks (subnetworks attributed to particular task specializations). In fact, the largest exponent was found in the frontoparietal network, suggestive of a large encoding capacity relative to networks such as a the ventral attention network, which possesses fewer stationary states. The frontoparietal network in particular is thought to be a key mediator of cognitive control [27, 28], though the extent to which its dynamics facilitate such is as yet opaque. Though our case study is an exploratory proof of concept, these results are intriguing as they suggest that this network may have a favorable dynamical landscape for encoding a diversity of cognitive states.

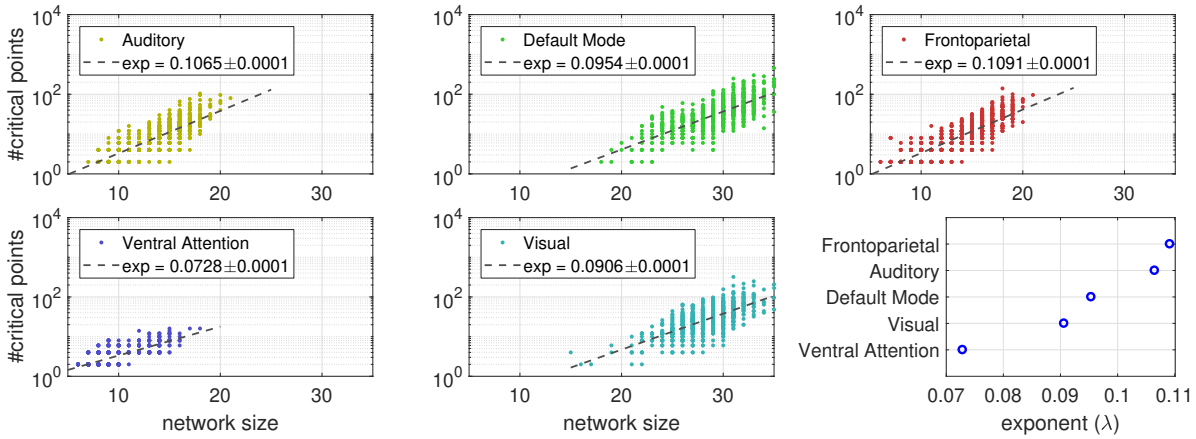

Figure 2: The number of critical points of discrete dynamical systems constructed from resting-state functional brain networks grows exponentially with network size, with an exponent that varies over the different resting state networks. The data comes from 783 participants in the Human Connectome Project (see **Methods**).

Case Study 2: Analysis of a coarse-grained lattice model for protein folding

The biological function of a protein is highly dependent on its structure, i.e. how the linear sequence of amino acids folds onto itself in 3D space. While this can occur in many different ways, it is well-established that the most probable structures are those that minimize free energy, which is a function of hydrophobic and other intramolecular forces. Although a real protein sequence is composed of 20 amino acids, a reduced-order model can provide valuable insight for protein analysis and design. In particular, we use a model with 3 types of amino acids or residues: those that have positive charge, those that have negative charge, and those in which hydrophobic forces dominate. We consider a 3D lattice structure in which the nodes take values $x_i \in \mathcal{S}_i := \{-1, 0, +1\}$, subject to the following energy function, which is a modification of the

Hydrophobic Polar model, first introduced in [29]:

$$\Phi(\mathbf{x}) := \sum_{i=1}^{n} \phi_i(x_i) = \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} x_i x_j + h_i(x_i),$$

where $\mathcal{N}_i := \{j \in \mathcal{V}_0 : (i,j) \in \mathcal{E}_0\}$ for each $i \in \mathcal{V}_0$ and

$$h_i(x_i) := \begin{cases} 1 & \text{if } x_i = 0 \text{ and } |\mathcal{N}_i| < d_{\max}, \\ b & \text{if } |x_i| = 1 \text{ and } |\mathcal{N}_i| = d_{\max}, \\ 0 & \text{otherwise,} \end{cases}$$

where $d_{\max}$ denotes the maximum node degree and b is a model parameter. We define the dynamics of the system such that at each time k, a random node updates to the state that minimizes the local energy $\phi_i(x_i)$. The point of this is not to directly model any physical dynamics, which are inherently stochastic, but rather to construct a deterministic system in which the set of fixed points is equivalent to the set of local minima of the energy function. We used the proposed algorithm to find all fixed points of this model for

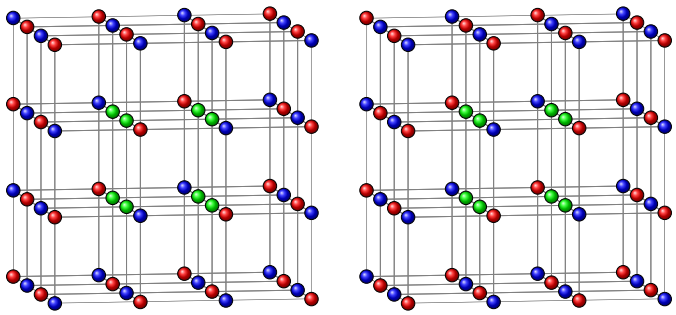

Figure 3: Optimal protein configurations on a $4 \times 4 \times 4$ lattice. Blue and red indicate positive and negative charges, respectively, and green corresponds to hydrophobic residues.

two different cases: a cubic lattice of dimension $4 \times 4 \times 4$, and a more irregular structure also having 64 nodes, with $b = 10$. Since each node can take three values, there are a total of 3^{64} possible configurations in both models. For the lattice model, we find that 587,636,902 are fixed points (a fraction 1.7×10^{-22} of the total), and of these, only two are global minima. For the irregular model, there are 60,594,240 fixed points (a fraction of 1.8×10^{-23}), 32 of which are global minima. Both optimal folding sequences for the regular lattice are depicted in Fig. 3, and the 32 global minima for the irregular model are shown in terms of eight independent components in Fig. 4. We can also plot the energy distributions of all fixed points for both models, as shown in Fig. 5. Perhaps counterintuitively, we observe a broader and more rugged fixed-point landscape in the cubic lattice than in the irregular lattice.

Note that cubic lattice structures are not k-separable for some constant value of k as the dimensions grow; rather, they are s^2-separable, where s denotes the dimension of one side. As a result, regular lattice structures are not ideally suited to this approach. The advantage of this method becomes more striking for structures with some irregularities, in which the network topology is more suitable for partitioning. Indeed,

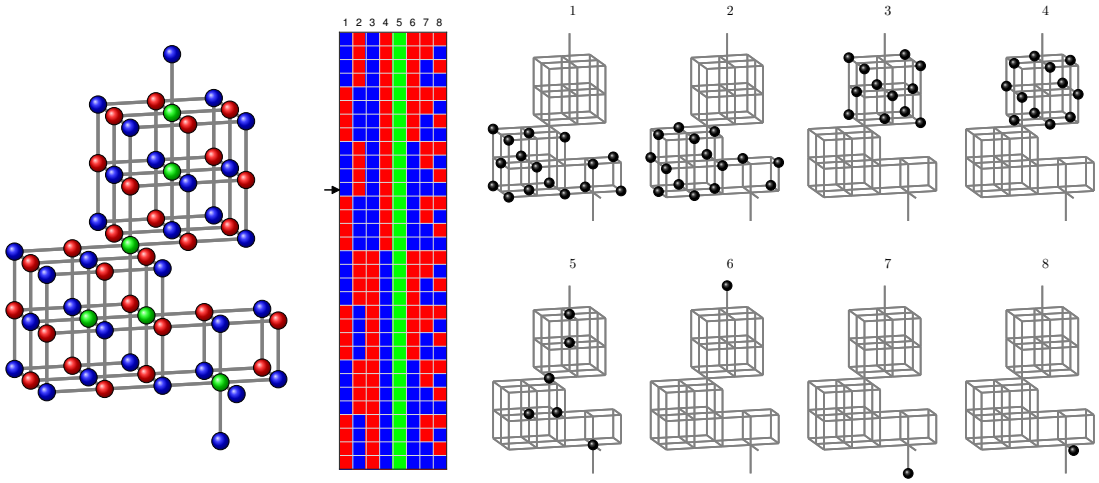

Figure 4: The 32 global energy minima in the irregular model, represented in the 32 rows of the color matrix, can be decomposed into 8 components, the nodes of which always take the same residue type, indicated by the color of the corresponding column entry. For example, the configuration shown on the left appears in row 12 of the color matrix, indicated by the arrow.

the energy distribution of the irregular lattice was computed in approximately $\frac{1}{30}$ the time needed for the cubic lattice.

Despite its simple appearance, the lattice model captures an essential challenge in protein folding and design: tackling sequence space. Even on this simplified model, the complexity of the problem is sufficient that an exhaustive search through sequence space is not possible. The standard practice for accessing sequence space relies on Monte Carlo searches, which fail to capture a vast majority of the fixed points for a given topology [30]. In the case of protein design, vast computational resources are spent sampling sequence space with the hopes of finding a sufficiently low local energy minimum. The proposed approach has the potential to significantly enhance this effort by providing a tractable means to map entire fixed-point landscapes, to within limits on protein size (see also **Discussion**).

In the discussion of Case Study 2: As previously remarked, finding the fixed points of a system does not directly provide any information about convergence or stability. However, when the dynamics are known to produce trajectories that are decreasing with respect to the energy function in the neighborhood of particular fixed points, then Lyapunov theory predicts that these fixed points will indeed be stable attractors.

Discussion

Scientific Impact

The ability to find and optimize over all fixed points in discrete systems on networks has the potential to assist and advance research across several scientific domains.

Finding critical points of generalized Ising models is a long-standing problem in theoretical physics [31] that has remained active over the years [32, 11]. Expanding the scale and types of structures for which we

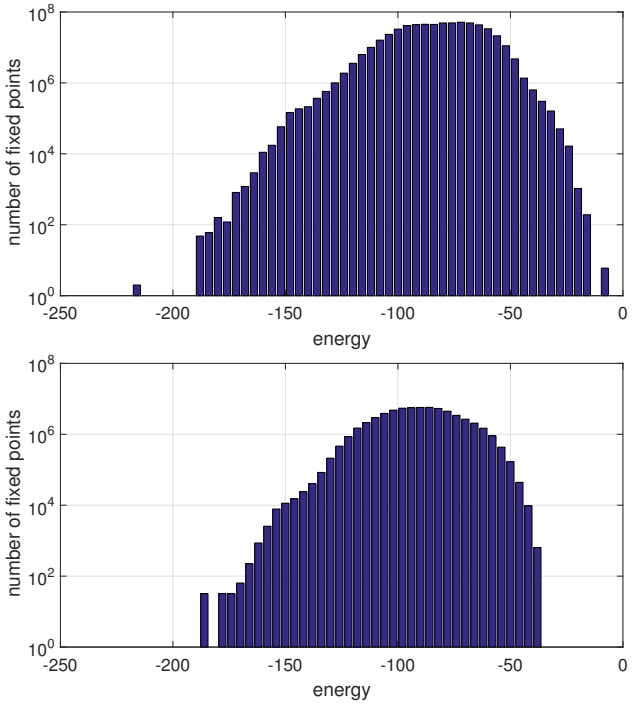

Figure 5: The distribution of energy minima in the cubic lattice (top) is both broader and more rugged than that of the irregular lattice (bottom).

can completely characterize the critical points has value in the study of magnetism, phase transitions, fluid dynamics, and other topics [33].

In biochemistry, finding the minimum energy protein sequence is a sought after task for which extraordinary computational resources and schemes have been devoted [34, 35]. Moreover, the ability to find all critical points can contribute to a greater understanding of the protein folding process, another target of large-scale computational projects [36]. Comparing the structure of proteins to the underlying fixed-point landscape can also give clues about the evolution of a protein sequence and provide greater insight into the impact of protein topology on evolutionary pathways.

Certain analyses in computational and theoretical neuroscience can be enabled through characterization of equilibria of (brain) network models. This is perhaps most prevalent in the study of memory encoding, where attractor networks have long been a popular model for the formation and recall of memoranda and other types of associations [37, 38]. More recently, notions of free energy minimization have been used as a theoretical schema within which to understand brain function at multiple spatial scales [39]. Testing such hypotheses has been difficult, due to the taxing nature of estimating energy landscapes from data. Perhaps as a result, the state of the art in brain network analyses, especially at whole-brain scales, has been largely limited to analyses of inter-region correlation and subsequent (static) graph theoretical analyses [40]. Nonetheless, key efforts in computational modeling and analysis are now underway to bridge architectural characterizations of brain networks to more faithful understanding of temporal dynamics over such networks [41, 42].

As shown here, the proposed methods can enable novel assessments of how brain network dynamics – via attractor and energy landscapes, which ultimately mediate the input to output lability of these networks [43] – are altered during cognition.

Limitations

The methods presented in this paper provide significant advances in the ability to computationally characterize the dynamics of complex networked systems with discrete state spaces. In the context of whole-brain modeling (Case Study 2), the discrete-state model may be informative on second-to-second time-scales, upon which populations can be thought of as being active or not. Here, however, the coarseness of the state space precludes analysis over finer time-scales, when brain regions undergo more nuanced patterns of activation. Findings such as those we presented in Case Study 2 should thus be interpreted at the appropriate spatial and temporal scale, and not as a characterization of dynamics over continuous state spaces. Because our proposed approach is not restricted to binary models, somewhat richer analyses could be performed by augmenting the Ising-framework with additional (discrete) states.

Thus, from a practical standpoint, the proposed method will perform best for systems that can be modeled with a few discrete states per unit and with relatively sparse connection structure. When either the number of discrete states or node degrees become large, as in the case of scale-free networks for example, then even the first step of computing the local equilibrium sets becomes intractable. For example, analysis of a full 20-amino-acid protein-folding model or a micro-scale model of brain activity, where neurons have thousands of synaptic connections, would currently lie outside the capabilities of the proposed algorithm as implemented on standard personal computers.

Methods

Sparse set notations and operations

It is convenient to represent a subset of the state space $\mathcal{S} := \prod_{i=1}^{n} \mathcal{S}_i$ as the set of global states in which some of the nodes take specified values while others are allowed to take any value. This can be expressed as a sparse vector or a set of $(index, value)$ pairs. For example, define $\mathbf{a} = \{\mathbf{x} \in \{-1, 1\}^5 : x_3 = 1, x_5 = -1\}$. We could alternatively denote this as $\mathbf{a} = \{(*, *, 1, *, -1) \in \mathcal{S}\}$ in which $*$ indicates that an entry may take any value. Although illustrative, this notation may become cumbersome for higher dimensional systems, so we introduce the following more compact sparse representation:

$$\mathbf{a} = \{(i_k, v_k)_{k=1}^{w}\}_{\mathcal{S}} := \{\mathbf{x} \in \mathcal{S} : x_{i_k} = v_k \ \forall k \in \{1, \ldots, w\}\},$$

where w is the number of nodes whose state is defined. According to this notation, the example above would be expressed as $\mathbf{a} = \{(3, 1), (5, -1)\}_{\mathcal{S}}$, where $\mathcal{S} = \{-1, 1\}^5$. We also use the notation $\mathbf{a}[i]$ to access the defined element $i \in \mathcal{V}_{\mathbf{a}}$, where $\mathcal{V}_{\mathbf{a}}$ denotes the set of all nodes assigned in $\mathbf{a}$. In the example, $\mathbf{a}[3] = 1$

and $\mathbf{a}[5] = -1$. We will achieve our objective primarily through the use of intersections and unions of these sparse representations of large subsets of the state space $\mathcal{S}$. The following example demonstrates these operations on two sparsely represented subsets $\mathbf{a}, \mathbf{b} \subset \{-1, 1\}^5$.

$$\begin{aligned}
\mathbf{a} :=& \{(3,1),(5,-1)\}_{\mathcal{S}} \\
\mathbf{b} :=& \{(2,1),(3,1),(4,1)\}_{\mathcal{S}} \\
\mathbf{a} \cap \mathbf{b} =& \{(2,1),(3,1),(4,1),(5,-1)\}_{\mathcal{S}} \\
\mathbf{a} \cup \mathbf{b} =& \{(3,1)\}_{\mathcal{S}}
\end{aligned}$$

We see here that computing the union of two sets is achieved by performing an intersection of the sparse representations and vice versa, which allows for efficient computation of these operations.

We sometimes need to represent ordered sets of these sparsely represented subsets. For example, define $\mathcal{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_\alpha)$ and $\mathcal{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_\beta)$ for some arbitrary integers α and β. Commonly used operations on these sets include the set of all pairwise intersections and the set of all pairwise unions, which we respectively denote by

$$\begin{aligned}
\mathcal{A} \Cap \mathcal{B} &:= \{\mathbf{a}_i \cap \mathbf{b}_j, \forall i \in \{1, \ldots, \alpha\}, \forall j \in \{1, \ldots, \beta\}\}, \\
\mathcal{A} \Cup \mathcal{B} &:= \{\mathbf{a}_i \cup \mathbf{b}_j, \forall i \in \{1, \ldots, \alpha\}, \forall j \in \{1, \ldots, \beta\}\}.
\end{aligned}$$

To denote the set of all m-way intersections and unions, we write

$$\bigCap_{i=1}^{m} \mathcal{A}_i := \mathcal{A}_1 \Cap \mathcal{A}_2 \Cap \cdots \Cap \mathcal{A}_m, \bigCup_{i=1}^{n} \mathcal{A}_i := \mathcal{A}_1 \Cup \mathcal{A}_2 \Cup \cdots \Cup \mathcal{A}_n.$$

It is straightforward to confirm that the commutative, associative, and distributive properties hold for these operations. It will also be convenient to relate whether or not elements of an ordered set $\mathcal{A}$ are subsets of elements of $\mathcal{B}$. For this purpose, we define the matrix of size $\alpha \times \beta$:

$$\Delta(\mathcal{A}, \mathcal{B}) := [\delta_{ij}(\mathcal{A}, \mathcal{B})]_{\alpha \times \beta}, \tag{2}$$

in which each entry is given by

$$\delta_{ij}(\mathcal{A}, \mathcal{B}) := \begin{cases} 1, & \text{if } \mathbf{a}_i \supseteq \mathbf{b}_j \\ 0, & \text{otherwise} \end{cases}. \tag{3}$$

In addition, let $\boldsymbol{\delta}_{i*}(\mathcal{A}, \mathcal{B}) := \{j : \delta_{ij}(\mathcal{A}, \mathcal{B}) = 1\}$ denote the index set of elements of $\mathcal{B}$ that are contained in $\mathbf{a}_i$, and let $\boldsymbol{\delta}_{*j}(\mathcal{A}, \mathcal{B}) := \{i : \delta_{ij}(\mathcal{A}, \mathcal{B}) = 1\}$ denote the index set of elements of $\mathcal{A}$ that contain $\mathbf{b}_j$.

Recursive partition-based intersections

The next step after computing the LEQ sets (as described in **Results: Finding and optimizing over all fixed points**) is to partition the network into several smaller subnetworks and compute the regional equilibria on

those subnetworks. The corresponding REQ sets can then be intersected using only the states of their boundary nodes to obtain a sparse representation of the global equilibrium set. At the end, we have all the information needed to quickly access any equilibrium without explicitly computing the whole set.

This method relies heavily on graph partitioning, which is itself a complex problem and an active research area. Although any valid graph partitioning algorithm can be used here, the choice may have a significant effect on performance. For example, an algorithm that tries to balance subgroup sizes while minimizing the number of edges that cross partition boundaries will generally lead to faster computation times than one that does not. It is also computationally advantageous for our purposes if the nodes on the partition boundaries have small LEQ sets. Optimally balanced graph partitioning is itself a computationally difficult problem. Fortunately, we do not require an optimal partition; rather, a reasonably good approximation will suffice, and recent research has shown the existence of algorithms that are tractable with respect to a parameterization of the requirements[44]. Indeed there are several approaches to choose from that are fast to compute relative to finding the fixed points. In our case studies and simulations, we used k-means clustering on the eigenvectors of the weighted adjacency matrix, with weights proportional to the size of the LEQ sets, which is a type of spectral graph partitioning algorithm [17].

Before proceeding, we introduce some definitions and notations related to partitioning the network. Let $\bar{\mathcal{V}} := \{\mathcal{V}_1, \ldots, \mathcal{V}_q\}$ denote a *partition* of the node set $\mathcal{V} \subseteq \mathcal{V}_0$ into q subsets, such that $\mathcal{V} = \bigcup_{k=1}^{q} \mathcal{V}_k$ and $\mathcal{V}_j \cap \mathcal{V}_k = \emptyset$ for all $j, k \in \{1, \ldots, q\}$, $j \neq k$. Further, we define the set of external boundary nodes ($\mathcal{V}_k^{\text{out}}$) as the set of nodes that are not in $\mathcal{V}_k$ but are adjacent (either to or from) a node in $\mathcal{V}_k$. In contrast, we define the internal boundary nodes ($\mathcal{V}_k^{\text{in}}$) as the set of nodes that are in $\mathcal{V}_k$ and are adjacent to a node that is not in $\mathcal{V}_k$. We also define the union of these two sets as the set of combined boundary nodes ($\mathcal{V}_k^{\text{bound}}$). Formally, these sets are defined as follows:

$$\mathcal{V}_k^{\text{out}} := \bigcup_{i \in \mathcal{V}_k} \mathcal{N}_i \backslash \mathcal{V}_k, \qquad \mathcal{V}_k^{\text{in}} := \bigcup_{i \in \mathcal{V}_k^{\text{out}}} \mathcal{N}_i \cap \mathcal{V}_k,$$
$$\mathcal{V}_k^{\text{bound}} := \mathcal{V}_k^{\text{in}} \cup \mathcal{V}_k^{\text{out}},$$

where $\mathcal{N}_i := \{j \in \mathcal{V} : (i,j) \in \mathcal{E} \text{ or } (j,i) \in \mathcal{E}\}$ denotes the set of all in- and out-neighbors of node i. We denote the set of interior nodes, having no out-of-group neighbors by

$$\check{\mathcal{V}}_k := \mathcal{V}_k \setminus \mathcal{V}_k^{\text{in}}.$$

Lastly, we respectively denote the union of all boundary nodes in a partition by

$$\mathcal{V}^{\text{part}} := \bigcup_{k=1}^{q} \mathcal{V}_k^{\text{bound}}. \tag{4}$$

By intersecting LEQ sets on a node set $\mathcal{V} \subseteq \mathcal{V}_0$, we form a set of *regional equilibrium* (REQ) states.

$$\Omega(\mathcal{V}) := \bigcap_{i=1}^{|\mathcal{V}|} \mathcal{L}_i. \tag{5}$$

Let r be a free parameter that defines the maximum size of a node set that we will not partition further, but rather compute the full equilibrium set using (5). The set $\mathcal{Z}(\mathcal{V})$ represents the full state space of the nodes in $\mathcal{V}$ and the empty set everywhere else:

$$\mathcal{Z}(\mathcal{V}) := \{\mathbf{x} \in \mathcal{S} : x_i = \emptyset,\ \forall i \notin \mathcal{V}\}. \tag{6}$$

We are now ready to introduce the core of the proposed method, which is the following recursive definition of a compact representation of the REQ sets $\Omega(\mathcal{V})$:

Equilibrium points on partition boundary nodes

$$\Omega^p(\mathcal{V}) := \begin{cases} \Omega(\mathcal{V}) & \text{if } |\mathcal{V}| \leq r \\ \bigcap_{k=1}^{q} \Omega^b(\mathcal{V}_k) & \text{if } |\mathcal{V}| > r \end{cases}, \tag{7}$$

$$\Omega^b(\mathcal{V}_k) := \Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k). \tag{8}$$

The set $\Omega^p(\mathcal{V})$ represents the set of all possible equilibrium values for nodes that lie on the boundary of the partition of $\bar{V}$. At the base level where $|\mathcal{V}| \leq r$, the problem is small enough to solve by intersecting all contained LEQ sets as in (5), and $\Omega^p(\mathcal{V})$ is equal to the full regional equilibrium set. Otherwise, we obtain $\Omega^p(\mathcal{V})$ by intersecting the sets $\Omega^b(\mathcal{V})$ for each partitioned subgroup. $\Omega^b(\mathcal{V}_k)$ is the set of all possible regional equilibrium values at the nodes that lie on the boundary of $\mathcal{V}_k$ with some other partitioned subnetwork V_j, $j \neq k$. This set is obtained by taking the union of the set $\Omega^p(\mathcal{V}_k)$ (defined recursively) with $\mathcal{Z}(\check{\mathcal{V}}_k)$. Taking the union of $\mathcal{Z}(\mathcal{V})$ with some sparsely represented set has the effect of decreasing the amount of information needed to represent the resulting set, since it will no longer contain any information about the nodes in $\mathcal{V}$. Notice that a graph partition $\bar{\mathcal{V}}$ is needed to compute the lower expression in (7).

It is useful to store the results of this computation in the form of a tree graph $\mathcal{T} := (\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$, in which each node $V_\mathcal{V} \in \mathcal{V}_\mathcal{T}$ contains the partition $\bar{V}$, the set $\mathbf{\Omega}^p(\mathcal{V})$ of possible equilibrium states on the partition boundary nodes, and the sets $\mathbf{\Omega}^b(\mathcal{V}_k)$ and corresponding Δ matrices for each partitioned subnetwork:

$$V_\mathcal{V} := \big(\bar{\mathcal{V}}, \mathbf{\Omega}^p(\mathcal{V}), \big\{\mathbf{\Omega}^b(\mathcal{V}_k), \Delta\big(\mathbf{\Omega}^p(\mathcal{V}), \mathbf{\Omega}^b(\mathcal{V}_k)\big), \Delta\big(\mathbf{\Omega}^b(\mathcal{V}_k), \mathbf{\Omega}^p(\mathcal{V}_k)\big)\big\}_{k=1}^{q}\big)$$

for each partition $\bar{\mathcal{V}}$ performed in the course of the recursion. The edges $\mathcal{E}_\mathcal{T}$ in the tree simply connect each tree node $V_\mathcal{V}$ to the nodes corresponding to the partition of $\mathcal{V}$, i.e. $(V_\mathcal{V}, V_{\mathcal{V}_k}) \in \mathcal{E}_\mathcal{T}$ for each $k \in \{1, \ldots, q\}$. As a result, we obtain a compact representation of the global equilibrium set. Specifically, $\Omega^p(\mathcal{V}_0)$ is equivalent to the union of the global equilibrium set with all state configurations of nodes interior to the top level partition, as formalized in the following theorem (proof in Appendix).

Theorem 2. *For any partition $\bar{\mathcal{V}} = \{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ of an arbitrary node set $\mathcal{V}$ such that $|\mathcal{V}| > r$,*

$$\Omega^p(\mathcal{V}) = \Omega(\mathcal{V}) \Cup \mathcal{Z}(\mathcal{V}^{part}).$$

When paired with the information contained in $\mathcal{T}$, there is sufficient information in $\Omega^p(\mathcal{V})$ to efficiently compute the number of equilibria, the full equilibrium set, or only those equilibria that are optimal with respect to some cost function.

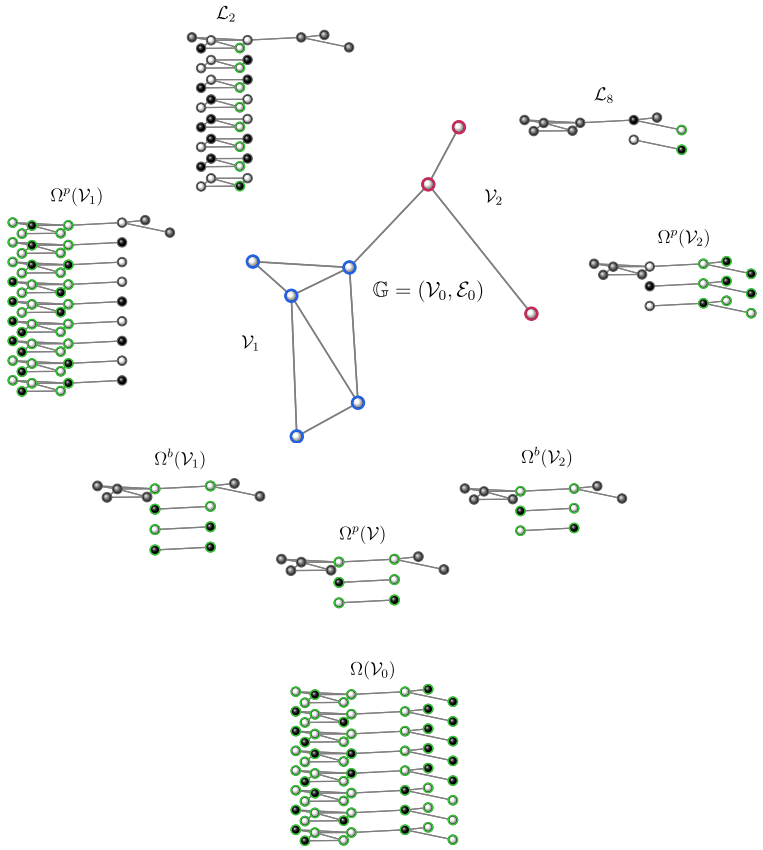

Figure 6: Example showing a network in which nodes can take one of two states marked by black or white (gray indicates nodes that can take any value), partitioned into two subnetworks $\mathcal{V}_1$ (blue) and $\mathcal{V}_2$ (red). We show two of the eight LEQ sets $\mathcal{L}_1$ and $\mathcal{L}_8$. Below them are the sets $\Omega^p(\mathcal{V}_1)$ and $\Omega^p(\mathcal{V}_2)$, generated form the intersection of their constituent LEQ sets. Further below are the compactly represented sets $\Omega^b(\mathcal{V}_1)$ and $\Omega^b(\mathcal{V}_2)$ on the two partition boundary nodes. Finally, we show the top-level set $\Omega^p(\mathcal{V})$, and the global equilibrium set $\Omega(\mathcal{V}_0)$ constructed from the information in $\mathcal{T}$.

To provide further intuition into the approach, we illustrate a small example in Figure 6. The network shown in the center is partioned into groups indicated by the colored node outlines. Best-response dynamics with an anticoordinating payoff matrix of $\left(\begin{smallmatrix} 0 & 3 \\ 1 & 0 \end{smallmatrix}\right)$ was used for all edges in the network (see **SI: Best response and linear threshold dynamics**). Two of the eight LEQ sets ($\mathcal{L}_1$ and $\mathcal{L}_8$) are shown towards the top, with the elements of the set in a stacked arrangement. The two PREQ sets ($\mathbf{\Omega}^p(\mathcal{V}_1)$ and $\mathbf{\Omega}^p(\mathcal{V}_2)$) corresponding to the intersections of the LEQ sets within their respective partitioned subgroups are shown at the far left and right. Compact representations of these sets on their boundary nodes are shown in the BREQ sets ($\mathbf{\Omega}^b(\mathcal{V}_1)$ and $\mathbf{\Omega}^b(\mathcal{V}_2)$) just underneath. Finally, the top level PREQ set $\mathbf{\Omega}^p(\mathcal{V})$ resulting from the intersection of the two BREQ sets and the expanded global equilibrium set $\Omega(\mathcal{V}_0)$ appear at the bottom.

Existence and number of equilibria

The number of equilibria can be computed using the same recursion structure as above and can occur either simultaneous to searching the equilibrium space or in post-processing. To do this, at the lowest level of partitioning (the top case of (10)), we simply count in (9) how many regional equilibrium states are included in each element of the PREQ set. These quantities are then passed to the higher level in the partition tree along with the requisite bookkeeping steps, ultimately resulting in the total number of equilibria at the top level, as specified in the following theorem (proof in the **Appendix**).

Theorem 3. *Given the tree $\mathcal{T}$ resulting from the computation* (7)-(6) *on system* (1)*, the total number of equilibria $|\boldsymbol{\Omega}(\mathcal{V}_0)|$ is given by*

$$n_{eq}(\Omega^p(\mathcal{V}_0)) := \sum_{j=1}^{|\Omega^p(\mathcal{V}_0)|} \check{\lambda}_j(\mathcal{V}_0), \tag{9}$$

where

$$\check{\lambda}_j(\mathcal{V}) := \begin{cases} 1, & \text{if } |\mathcal{V}| \leq r \\ \prod_{k=1}^{q} \lambda_j(\mathcal{V}_k), & \text{otherwise} \end{cases}, \tag{10}$$

$$\lambda_j(\mathcal{V}_k) := \begin{cases} 0, & \text{if } \boldsymbol{\delta}_{*j}\big(\boldsymbol{\Omega}^p(\mathcal{V}), \boldsymbol{\Omega}^b(\mathcal{V}_k)\big) = \emptyset \\ \gamma_i(\mathcal{V}_k),\ i \in \boldsymbol{\delta}_{*j}\big(\boldsymbol{\Omega}^p(\mathcal{V}), \boldsymbol{\Omega}^b(\mathcal{V}_k)\big), & \text{otherwise} \end{cases},$$

$$\gamma_i(\mathcal{V}_k) := \sum_{j=1}^{|\Omega^p(\mathcal{V}_k)|} \check{\lambda}_j(\mathcal{V}_k)\, \boldsymbol{\delta}_{ij}\big(\boldsymbol{\Omega}^b(\mathcal{V}_k), \boldsymbol{\Omega}^p(\mathcal{V}_k)\big).$$

Optimization on the global equilibrium set

Within the set of all fixed points, it is often useful to find the equilibria that optimize a given cost or energy function. An advantage of this approach is that the optimal equilibria can be quickly extracted from the results of the computation without generating the full equilibrium set, which can be very large in some cases.

Optimization using this approach works for the class of global cost functions arising from the linear superposition of possibly nonlinear local cost functions. Let $\Phi(\mathbf{x}) : \mathcal{S} \to \mathbb{R}$ denote a cost function on the state space $\mathcal{S}$, which is the summation of local node and edge costs:

$$\Phi(\mathbf{x}) := \sum_{i=1}^{n} h_i(x_i) + \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i} \phi_{ij}(x_i, x_j). \tag{11}$$

Suppose that we seek the set of equilibria that minimize this function:

$$\Omega^*(\mathcal{V}_0) := \{\mathbf{x} \in \Omega(\mathcal{V}_0) : \Phi(\mathbf{x}) \leq \Phi(\mathbf{y}),\ \forall \mathbf{y} \in \Omega(\mathcal{V}_0)\}. \tag{12}$$

It is also useful to define the cost of a particular local or regional equilibrium set, which we denote by $\boldsymbol{\omega}(\mathcal{V}) \in \Omega(\mathcal{V})$, where $\Omega(\mathcal{V}) := \{\boldsymbol{\omega}_1(\mathcal{V}), \ldots, \boldsymbol{\omega}_\alpha(\mathcal{V})\}$ and $\alpha := |\Omega(\mathcal{V})|$. Recalling that $\mathbf{x}[i]$ denotes the fixed

state of node i in the sparse representation set $\mathbf{x} \subseteq \mathcal{S}$, we note that $\boldsymbol{\omega}(\mathcal{V})[i]$ has the same interpretation for the equilibrium set $\boldsymbol{\omega}(\mathcal{V})$. We can now express the cost of $\boldsymbol{\omega}(\mathcal{V})$ as follows:

$$\boldsymbol{\Phi}(\boldsymbol{\omega}(\mathcal{V})) := \sum_{i \in \mathcal{V}} h_i(\boldsymbol{\omega}(\mathcal{V})[i]) + \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i \cap \mathcal{V}} \phi_{ij}(\boldsymbol{\omega}(\mathcal{V})[i], \boldsymbol{\omega}(\mathcal{V})[j]). \tag{13}$$

At the lowest level of the partition tree, the cost of each element $\boldsymbol{\omega}(\mathcal{V})$ is computed according to (13). Notice that while the costs associated with all nodes of $\mathcal{V}$ are included, only the costs of edges connecting pairs of nodes in $\mathcal{V}$ are included, while costs related to edges containing nodes not in $\mathcal{V}$ are excluded. These inter-region costs are and defined as follows:

$$\tilde{\boldsymbol{\Phi}}(\boldsymbol{\omega}(\mathcal{V})) := \sum_{k=1}^{q} \sum_{i \in \mathcal{V}_k^{\text{in}}} \sum_{j \in \mathcal{N}_i \cap \mathcal{V}_k^{\text{out}}} \phi_{ij}(\boldsymbol{\omega}(\mathcal{V})[i], \boldsymbol{\omega}(\mathcal{V})[j]). \tag{14}$$

Although the cost of a base-level equilibrium set $\boldsymbol{\omega}$ is unique and well-defined, the cost of a PREQ element $\boldsymbol{\omega}^p(\mathcal{V})$ or BREQ element $\boldsymbol{\omega}^b(\mathcal{V})$ may take multiple values, since they encapsulate the set of fixed points for all configurations of the interior nodes. Hence, we define the costs of these elements as ordered sets:

$$\begin{aligned} \boldsymbol{\phi}(\mathcal{V}) :=& (\{\phi : \exists i : \boldsymbol{\Phi}(\boldsymbol{\omega}_i^p(\mathcal{V})) = \phi\}, \rho), \\ =& (\{\phi : \exists i : \boldsymbol{\Phi}(\boldsymbol{\omega}_i^b(\mathcal{V})) = \phi\}, \rho). \end{aligned} \tag{15}$$

Note that the set of feasible cost values in the sets $\boldsymbol{\Omega}^p(\mathcal{V})$ and $\boldsymbol{\Omega}^b(\mathcal{V})$ are identical, since they include the same underlying REQ set $\Omega(\mathcal{V})$. Next, we define a matrix that maps the elements of a PREQ set to the number of ways each unique cost value is attainable in the respective node groups, in which we have labeled the rows and columns for ease of interpretation:

$$W^p(\mathcal{V}) := \begin{array}{c} \\ \boldsymbol{\phi}_1(\mathcal{V}) \\ \vdots \\ \boldsymbol{\phi}_\nu(\mathcal{V}) \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{\omega}_1^p(\mathcal{V}) & \cdots & \boldsymbol{\omega}_\mu^p(\mathcal{V}) \end{array} \\ \begin{bmatrix} w_{11}^p & \cdots & w_{1\mu}^p \\ \vdots & \cdots & \vdots \\ w_{\nu 1}^p & \cdots & w_{\nu\mu}^p \end{bmatrix} \end{array}, \qquad w_{ij}^p := \left| \{\mathbf{x} \in \boldsymbol{\omega}_j^p(\mathcal{V}) : \Phi(\mathbf{x}) = \boldsymbol{\phi}_i^p(\mathcal{V})\} \right|, \tag{16}$$

where $\mu := |\boldsymbol{\Omega}^p(\mathcal{V})|$, $\nu := |\boldsymbol{\phi}^p(\mathcal{V})|$, and the matrix $W^b(\mathcal{V})$ is defined similarly.

Clearly, computing the quantities (15)-(16) explicitly would require generating the entire set $\Omega(\mathcal{V})$. Fortunately, there is a more efficient way to compute these using the partition tree structure $\mathcal{T}$ and the cost values attained by lower level REQ sets. To help with this task, it will be convenient to define two sets of matrices. First, the matrix $\hat{W}(\mathcal{V}_k)$ maps the unique costs achievable by elements of the lower level BREQ sets $\boldsymbol{\Omega}^b(\mathcal{V}_k)$ to the current level PREQ elements $\boldsymbol{\omega}^p(\mathcal{V})$ with which the costs are compatible:

$$\hat{W}(\mathcal{V}_k) := \begin{array}{c} \\ \boldsymbol{\phi}_1(\mathcal{V}_k) \\ \vdots \\ \boldsymbol{\phi}_{\nu_k}(\mathcal{V}_k) \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{\omega}_1^p(\mathcal{V}) & \cdots & \boldsymbol{\omega}_\mu^p(\mathcal{V}) \end{array} \\ \begin{bmatrix} \hat{w}_{11} & \cdots & \hat{w}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{w}_{\nu_k 1} & \cdots & \hat{w}_{\nu_k \mu} \end{bmatrix} \end{array}, \quad \hat{w}_{ij} := \left| \{\mathbf{x} \in \boldsymbol{\Omega}^b(\mathcal{V}_k) \Cap \{\boldsymbol{\omega}_j^p(\mathcal{V}_k)\} : \Phi(\mathbf{x}) = \boldsymbol{\phi}_i(\mathcal{V}_k)\} \right|,$$

where $\nu_k := |\boldsymbol{\phi}(\mathcal{V}_k)|$. Second, the matrix $\hat{Z}(\mathcal{V}_k)$ contains the costs associated with each entry in $\hat{W}(\mathcal{V}_k)$, or ∞ if the entry corresponds to an infeasible pairing:

$$\hat{Z}(\mathcal{V}_k) := \begin{array}{c} \\ \boldsymbol{\phi}_1(\mathcal{V}_k) \\ \vdots \\ \boldsymbol{\phi}_{\nu_k}(\mathcal{V}_k) \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{\omega}_1^p(\mathcal{V}) & \cdots & \boldsymbol{\omega}_\mu^p(\mathcal{V}) \end{array} \\ \begin{bmatrix} \hat{z}_{11} & \cdots & \hat{z}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{z}_{\nu_k 1} & \cdots & \hat{z}_{\nu_k \mu} \end{bmatrix} \end{array}, \qquad \hat{z}_{ij} := \begin{cases} \boldsymbol{\phi}_i(\mathcal{V}_k) & \text{if } \hat{w}_{ij} > 0 \\ \infty & \text{if } \hat{w}_{ij} = 0 \end{cases}.$$

Next, we construct expanded matrices $\mathring{W}(\mathcal{V}_k)$ and $\mathring{Z}(\mathcal{V}_k)$ for each subgroup k. Let $\mathring{\boldsymbol{\phi}}(\mathcal{V})$ denote the sums of all configurations of the unique subgroup costs $\boldsymbol{\phi}(\mathcal{V}_k)$, of which the total number is $|\mathring{\boldsymbol{\phi}}(\mathcal{V})| = \prod_{k=1}^{q} |\boldsymbol{\phi}(\mathcal{V}_k)|$. Then the expanded matrix $\mathring{W}(\mathcal{V}_1)$ is given by:

$$\mathring{W}(\mathcal{V}_1) := \begin{array}{l} \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_1(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_1(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_1(\mathcal{V}) \\ \vdots \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_1(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_{\nu_q}(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\nu_q}(\mathcal{V}) \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_2(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_1(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\nu_q+1}(\mathcal{V}) \\ \vdots \\ \boldsymbol{\phi}_{\nu_1}(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_{\nu_{q-1}}(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_{\nu_q}(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\mathring{\nu}}(\mathcal{V}) \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{\omega}_1^p(\mathcal{V}) & \cdots & \boldsymbol{\omega}_\mu^p(\mathcal{V}) \end{array} \\ \begin{bmatrix} \hat{w}_{11} & \cdots & \hat{w}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{w}_{11} & \cdots & \hat{w}_{1\mu} \\ \hat{w}_{11} & \cdots & \hat{w}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{w}_{\nu_1 1} & \cdots & \hat{w}_{\nu_1 \mu} \end{bmatrix} \end{array},$$

where $\hat{w}_{ij}$ is equal to the ij^{th} entry of $\hat{W}(\mathcal{V}_1)$. The successive matrices $\mathring{W}(\mathcal{V}_k)$ are defined similarly, with $\hat{w}_{ij}$ appearing in the same row where $\boldsymbol{\phi}_i(\mathcal{V}_k)$ appears in the summations yielding the entries of $\mathring{\boldsymbol{\phi}}(\mathcal{V})$. Likewise, the expanded matrix $\mathring{Z}(\mathcal{V}_1)$ is given by:

$$\mathring{Z}(\mathcal{V}_1) := \begin{array}{l} \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_1(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_1(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_1(\mathcal{V}) \\ \vdots \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_1(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_{\nu_q}(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\nu_q}(\mathcal{V}) \\ \boldsymbol{\phi}_1(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_2(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_1(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\nu_q+1}(\mathcal{V}) \\ \vdots \\ \boldsymbol{\phi}_{\nu_1}(\mathcal{V}_1) + \cdots + \boldsymbol{\phi}_{\nu_{q-1}}(\mathcal{V}_{q-1}) + \boldsymbol{\phi}_{\nu_q}(\mathcal{V}_q) = \mathring{\boldsymbol{\phi}}_{\mathring{\nu}}(\mathcal{V}) \end{array} \begin{array}{c} \begin{array}{ccc} \boldsymbol{\omega}_1^p(\mathcal{V}) & \cdots & \boldsymbol{\omega}_\mu^p(\mathcal{V}) \end{array} \\ \begin{bmatrix} \hat{z}_{11} & \cdots & \hat{z}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{z}_{11} & \cdots & \hat{z}_{1\mu} \\ \hat{z}_{11} & \cdots & \hat{z}_{1\mu} \\ \vdots & \cdots & \vdots \\ \hat{z}_{\nu_1 1} & \cdots & \hat{z}_{\nu_1 \mu} \end{bmatrix} \end{array},$$

where $\hat{z}_{ij}$ is equal to the ij^{th} entry of $\hat{z}(\mathcal{V}_1)$. Next, let $\tilde{\boldsymbol{\phi}}(\mathcal{V}) := [\tilde{\boldsymbol{\Phi}}(\boldsymbol{\omega}_1^p(\mathcal{V})), \ldots, \tilde{\boldsymbol{\Phi}}(\boldsymbol{\omega}_{|\boldsymbol{\Omega}^p(\mathcal{V})|}^p(\mathcal{V}))]$ denote a row vector of inter-group costs. The combined expanded cost matrix $\mathring{Z}(\mathcal{V})$ and the number of instantiations of each cost $\mathring{W}(\mathcal{V})$ can be expressed as follows:

$$\begin{aligned} \mathring{W}(\mathcal{V}) :=& \mathring{W}(\mathcal{V}_1) \odot \mathring{W}(\mathcal{V}_2) \odot \cdots \odot \mathring{W}(\mathcal{V}_q), \\ \mathring{Z}(\mathcal{V}) :=& \sum_{k=1}^{q} \mathring{Z}(\mathcal{V}_k) + \tilde{\boldsymbol{\phi}}(\mathcal{V}) \otimes \mathbf{1}_{|\mathring{Z}_1(\mathcal{V}_1)|}, \end{aligned}$$

where $\odot$ denotes element-wise multiplication and $\otimes$ denotes the Kronecker product. Finally, the compact matrix $W^p(\mathcal{V})$ is constructed by extracting the unique cost values in $\mathring{Z}(\mathcal{V})$ and counting the total number of ways each cost is attained in $\mathring{W}(\mathcal{V})$. We again refer the reader to the Cost algorithm in **SI** for details on this procedure. The optimal equilibrium set $\Omega^*(\mathcal{V}_0)$ can then be obtained by following a similar procedure, as described in detail in the Expand algorithm (see **SI**). Note that by using the null cost function $\Phi(\mathbf{x}) = 0, \forall \mathbf{x} \in \mathcal{S}$, the set $\Omega^*(\mathcal{V}_0)$ is equivalent to the global equilibrium set $\Omega(\mathcal{V}_0)$ of (1).

References

[1] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[2] Roberto Santana, Pedro Larrañaga, and Jose A Lozano. Protein folding in simplified models with estimation of distribution algorithms. *IEEE transactions on Evolutionary Computation*, 12(4):418–438, 2008.

[3] P. Ramazi, J. Riehl, and M. Cao. Networks of conforming or nonconforming individuals tend to reach satisfactory decisions. *Proceedings of the National Academy of Sciences*, 113(46):12985–12990, 2016.

[4] Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[5] Jeremy Kun, Brian Powers, and Lev Reyzin. Anti-coordination games and stable graph colorings. In *International Symposium on Algorithmic Game Theory*, pages 122–133. Springer, 2013.

[6] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218, 2014.

[7] Sergei V Isakov, Ilia N Zintchenko, Troels F Rønnow, and Matthias Troyer. Optimised simulated annealing for ising spin glasses. *Computer Physics Communications*, 192:265–271, 2015.

[8] AZ Maksymowicz, JE Galletly, MS Magdon, and IL Maksymowicz. Genetic algorithm approach for ising model. *Journal of magnetism and magnetic materials*, 133(1-3):40–41, 1994.

[9] Volker Blum, Gus LW Hart, Michael J Walorski, and Alex Zunger. Using genetic algorithms to map first-principles results to model hamiltonians: Application to the generalized ising model for alloys. *Physical Review B*, 72(16):165113, 2005.

[10] Limor Drori and David Peleg. Faster exact solutions for some np-hard problems. *Theoretical Computer Science*, 287(2):473–499, 2002.

[11] Wenxuan Huang, Daniil A Kitchaev, Stephen T Dacek, Ziqin Rong, Alexander Urban, Shan Cao, Chuan Luo, and Gerbrand Ceder. Finding and proving the exact ground state of a generalized ising model by convex optimization and max-sat. *Physical Review B*, 94(13):134424, 2016.

[12] Alessandro Di Cara, Abhishek Garg, Giovanni De Micheli, Ioannis Xenarios, and Luis Mendoza. Dynamic simulation of regulatory networks using squad. *BMC bioinformatics*, 8(1):462, 2007.

[13] Yin Zhao, Jongrae Kim, and Maurizio Filippone. Aggregation algorithm towards large-scale boolean network analysis. *IEEE Transactions on Automatic Control*, 58(8):1976–1985, 2013.

[14] Desheng Zheng, Guowu Yang, Xiaoyu Li, Zhicai Wang, Feng Liu, and Lei He. An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks. *PloS one*, 8(4):e60593, 2013.

[15] Elena Dubrova and Maxim Teslenko. A sat-based algorithm for finding attractors in synchronous boolean networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(5):1393–1399, 2011.

[16] Wensheng Guo, Guowu Yang, Wei Wu, Lei He, and Mingyu Sun. A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks. *PloS one*, 9(4):e94258, 2014.

[17] Joao P Hespanha. An efficient matlab algorithm for graph partitioning. *University of California*, pages 1–8, 2004.

[18] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[19] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[20] Elad Schneidman, Michael J Berry II, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007, 2006.

[21] David B Kastner, Stephen A Baccus, and Tatyana O Sharpee. Critical and maximally informative encoding between neural populations in the retina. *Proceedings of the National Academy of Sciences*, page 201418092, 2015.

[22] Rubén Herzog, Maria-Jose Escobar, Rodrigo Cofre, Adrian G Palacios, and Bruno Cessac. Dimensionality reduction on spatio-temporal maximum entropy models of spiking networks. *bioRxiv*, page 278606, 2018.

[23] Michael Schirner, Anthony Randal McIntosh, Viktor Jirsa, Gustavo Deco, and Petra Ritter. Inferring multi-scale neural mechanisms with brain network modelling. *Elife*, 7:e28927, 2018.

[24] Joana Cabral, Morten L Kringelbach, and Gustavo Deco. Exploring the network dynamics underlying brain activity during rest. *Progress in neurobiology*, 114:102–131, 2014.

[25] Gustavo Deco, Mario Senden, and Viktor Jirsa. How anatomy shapes dynamics: a semi-analytical study of the brain at rest by a simple spin model. *Frontiers in computational neuroscience*, 6:68, 2012.

[26] Martin A Lindquist, Yuting Xu, Mary Beth Nebel, and Brain S Caffo. Evaluating dynamic bivariate correlations in resting-state fmri: a comparison study and a new approach. *NeuroImage*, 101:531–546, 2014.

[27] Michael W Cole, Tal Yarkoni, Grega Repovš, Alan Anticevic, and Todd S Braver. Global connectivity of prefrontal cortex predicts cognitive control and intelligence. *Journal of Neuroscience*, 32(26):8988–8999, 2012.

[28] Todd S Braver, Jeremy R Reynolds, and David I Donaldson. Neural mechanisms of transient and sustained cognitive control during task switching. *Neuron*, 39(4):713–726, 2003.

[29] Ken A Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[30] Po-Ssu Huang, Scott E Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320, 2016.

[31] Makoto Kaburagi and Junjiro Kanamori. A method of determining the ground state of the extended-range classical lattice gas model. *Progress of Theoretical Physics*, 54(1):30–44, 1975.

[32] Alexander K Hartmann. Ground states of two-dimensional ising spin glasses: fast algorithms, recent developments and a ferromagnet-spin glass mixture. *Journal of Statistical Physics*, 144(3):519, 2011.

[33] P Chandra, P Coleman, and AI Larkin. Ising transition in frustrated heisenberg models. *Physical review letters*, 64(1):88, 1990.

[34] Rhiju Das, Bin Qian, Srivatsan Raman, Robert Vernon, James Thompson, Philip Bradley, Sagar Khare, Michael D Tyka, Divya Bhat, Dylan Chivian, et al. Structure prediction for casp7 targets using extensive all-atom refinement with rosetta@ home. *Proteins: Structure, Function, and Bioinformatics*, 69(S8):118–128, 2007.

[35] Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature Structural and Molecular Biology*, 18(10):1175, 2011.

[36] Adam L Beberg, Daniel L Ensign, Guha Jayachandran, Siraj Khaliq, and Vijay S Pande. Folding@ home: Lessons from eight years of volunteer distributed computing. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[37] Daniel J Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1992.

[38] Danke Zhang, Chi Zhang, and Armen Stepanyants. Robust associative learning is sufficient to explain structural and dynamical properties of local cortical circuits. *bioRxiv*, page 320432, 2018.

[39] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127, 2010.

[40] Gustavo Deco, Viktor K Jirsa, and Anthony R McIntosh. Emerging concepts for the dynamical organization of resting-state activity in the brain. *Nature Reviews Neuroscience*, 12(1):43, 2011.

[41] Andrea Avena-Koenigsberger, Bratislav Misic, and Olaf Sporns. Communication dynamics in complex brain networks. *Nature Reviews Neuroscience*, 19(1):17, 2018.

[42] Michael Breakspear. Dynamic models of large-scale brain activity. *Nature neuroscience*, 20(3):340, 2017.

[43] Nancy J Kopell, Howard J Gritton, Miles A Whittington, and Mark A Kramer. Beyond the connectome: the dynome. *Neuron*, 83(6):1319–1328, 2014.

[44] Daniel Lokshtanov and Dániel Marx. Clustering with local restrictions. *Information and Computation*, 222:278–292, 2013.

[45] Daniel S Marcus, Michael P Harms, Abraham Z Snyder, Mark Jenkinson, J Anthony Wilson, Matthew F Glasser, Deanna M Barch, Kevin A Archie, Gregory C Burgess, Mohana Ramaratnam, et al. Human connectome project informatics: quality control, database services, and data visualization. *Neuroimage*, 80:202–219, 2013.

[46] Jonathan D Power. A simple but useful way to assess fmri scan qualities. *Neuroimage*, 154:150–158, 2017.

[47] Matthew F Glasser, Stamatios N Sotiropoulos, J Anthony Wilson, Timothy S Coalson, Bruce Fischl, Jesper L Andersson, Junqian Xu, Saad Jbabdi, Matthew Webster, Jonathan R Polimeni, et al. The minimal preprocessing pipelines for the human connectome project. *Neuroimage*, 80:105–124, 2013.

[48] Jonathan D Power, Bradley L Schlaggar, and Steven E Petersen. Recent progress and outstanding issues in motion correction in resting state fmri. *Neuroimage*, 105:536–551, 2015.

[49] Ludovica Griffanti, Gholamreza Salimi-Khorshidi, Christian F Beckmann, Edward J Auerbach, Gwenaëlle Douaud, Claire E Sexton, Enikő Zsoldos, Klaus P Ebmeier, Nicola Filippini, Clare E Mackay, et al. Ica-based artefact removal and accelerated fmri acquisition for improved resting state network imaging. *Neuroimage*, 95:232–247, 2014.

[50] Evan M Gordon, Timothy O Laumann, Babatunde Adeyemo, Jeremy F Huckins, William M Kelley, and Steven E Petersen. Generation and evaluation of a cortical area parcellation from resting-state correlations. *Cerebral Cortex (New York, NY)*, 26(1):288, 2016.

[51] Jonathan D Power, Alexander L Cohen, Steven M Nelson, Gagan S Wig, Kelly Anne Barnes, Jessica A Church, Alecia C Vogel, Timothy O Laumann, Fran M Miezin, Bradley L Schlaggar, et al. Functional network organization of the human brain. *Neuron*, 72(4):665–678, 2011.

[52] Enzo Tagliazucchi, Dante R Chialvo, Michael Siniatchkin, Enrico Amico, Jean-Francois Brichant, Vincent Bonhomme, Quentin Noirhomme, Helmut Laufs, and Steven Laureys. Large-scale signatures of unconsciousness are consistent with a departure from critical dynamics. *Journal of The Royal Society Interface*, 13(114):20151027, 2016.

[53] Mark Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6):1420–1443, 1978.

[54] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[55] Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, 2005.

Appendices

Processing of data from Human Connectome Project

The data used consisted of resting state fMRI scans collected from 783 subjects during the Human Connectome Project (HCP). These subjects were part of the s900 release which contained a total of 818 subjects. However, 35 of these subjects which had been marked as having instabilities in the head coil were removed [45, 46]. Each subject underwent a total of four 15-minute scans divided among two separate scanning sessions. The two scans per session corresponded to a left-right and a right-left acquisition. Data was acquired at 3T with a temporal resolution (TR) of 720ms. Detailed information on HCP scan protocols are contained in [47].

The publicly available s900 data release includes the following processing procedure. Following the standard minimal preprocessing pipeline [47], the motion correction strategies recommended by Power and

colleagues were applied [48], with the exception of Frame Censoring (removing time points), which the authors acknowledge has proven controversial. The remaining motion correction procedure consists of the FSL ICA-FIX correction [49] and regressing out the twelve HCP motion parameters and global signal (cerebrospinal fluid, white matter, and grey matter). To remove signal drift, we applied a .009 Hz high-pass filter and detrended to remove respiratory artifact. Data were parcellated into the Gordon atlas [50] and functional connectivities were computed using the Pearson correlation between parcels for the combined data across scans. More details on the task specializations associated with the various resting-state networks can be found in [51].

To suppress weak correlations and to facilitate comparison between subjects that are independent of overall connectivity variations, we thresholded correlation coefficients to preserve 5% of all pairwise connections, following similar methods as in [52]. Fixed points were computed for best-response dynamics in which the payoff matrices between each pair of regions i and j is given by $J_{ij}I_2$, where J_{ij} denotes the Pearson correlation coefficient between respective regions and I_2 is the 2×2 identity matrix. These dynamics are guaranteed to converge to a local minimum of the Ising Hamiltonian:

$$\Phi(\mathbf{x}) := \sum_{i=1}^{n} \phi_i(\mathbf{x}), \quad \text{where} \quad \phi_i(\mathbf{x}) := -\sum_{j=1}^{n} J_{ij} x_i x_j.$$

Analysis of a coarse-grained lattice model for protein folding

For this case study, the recursive partitioning algorithm was augmented with two stages of trimming, one to eliminate infeasible entries in the LEQ sets, and a second to eliminate known infeasible entries in the intermediate REQ sets. These steps are possible due to the regularity of the lattice structure and the uniformity of the cost function across nodes, and are generally good practice when applicable because of the resulting computational savings.

Proof of Theorem 1

Proof. The overall computation involves three types of operations: local equilibrium checks, sparse set intersections, and computing the Δ mappings.

Computing the local equilibria: Finding the local equilibrium states at each node involves evaluating the local update rule for a maximum of $\hat{s}^{\hat{d}}$ total state configurations. Multiplying by the number of nodes n yields a computational complexity of $O(n\hat{s}^{\hat{d}})$ for this step.

Sparse set intersections: The sparse set intersections occur at two levels. At the base level, the intersection of a pair of local equilibrium sets involves checking a maximum of $(\hat{s}^{\hat{d}})^2$ overlapping state configurations. The result is then checked against a third local equilibrium set, and the process repeats until the maximum group size r is reached, yielding a total for each base-level partition group of

$$\hat{s}^{2\hat{d}} + 2\hat{s}^{3\hat{d}} + \cdots + (r-1)\hat{s}^{r\hat{d}} \leq r^2 \hat{s}^{r\hat{d}}.$$

Assuming maximum group sizes, this computation is performed approximately $\frac{n}{r}$ times, yielding an upper bound of $O(rn\hat{s}^{r\hat{d}})$ for the local intersections.

Intersections at all higher levels depend on the number of boundary nodes between neighboring partition groups, of which there are a maximum of $2k$. However, the states of all boundary nodes must be represented for each partition group. Since there up to q partition groups, all of which may be adjacent to each other, the maximum dimension in each group is $2kq$. Hence, each intersection involves checking up to $(\hat{s}^{2kq})^2$ overlapping state configurations, each of which may contain up to $2kq$ nodes, resulting in a complexity of $O(2kq\hat{s}^{4kq})$ for each partition that is performed. Since the total number of partitions is bounded above by n, we have a total complexity of $O(kqn\hat{s}^{4kq})$ for the higher level sparse set intersections. The computational complexity of all sparse set intersections is then no greater than $O(rn\hat{s}^{r\hat{d}} + kn\hat{s}^{4kq})$.

Computing the Δ matrices: The Δ matrices defined in (2) in the main article map regional equilibrium states between neighboring levels in the partition hierarchy. Since the maximum number of configurations on each level is $O(\hat{s}^{2kq})$, the number of entries in each Δ matrix is at most $O(\hat{s}^{4kq})$. Since each entry δ_{ij} requires at most $2kq$ computations (see (3) in the article), the computational complexity for constructing each Δ matrix is bounded by $O(kq\hat{s}^{4kq})$. Finally, since both the number of nodes and edges in the tree $\mathcal{T}$ are less than n, the number of Δ matrices to compute is at most $2n$, resulting in a complexity of $O(kqn\hat{s}^{4kq})$.

The overall computational complexity of computing $\mathcal{T}$ is therefore bounded by $O(kqn\hat{s}^{4kq}+rn\hat{s}^{r\hat{d}})$. □

Proof of Theorem 2

Proof. Since $|\mathcal{V}| > r$, according to (7) and (8),

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \left(\Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k)\right). \tag{17}$$

Pulling out the first term in this expression yields

$$\Omega^p(\mathcal{V}) = \left(\Omega^p(\mathcal{V}_1) \Cup \mathcal{Z}(\check{\mathcal{V}}_1)\right) \Cap \bigdoublecap_{k=2}^{q} \left(\Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k)\right)$$

which can be expanded to

$$\begin{aligned}\Omega^p(\mathcal{V}) = &\left(\Omega^p(\mathcal{V}_1) \Cap \bigdoublecap_{k=2}^{q} \left(\Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k)\right)\right) \\ &\Cup \left(\mathcal{Z}(\check{\mathcal{V}}_1) \Cap \bigdoublecap_{k=2}^{q} \left(\Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k)\right)\right).\end{aligned}$$

Due to (6) and the fact that the partition is disjoint, we know that $\mathcal{Z}(\check{\mathcal{V}}_1) \subseteq \Omega^p(\mathcal{V}_k)$ and that $\mathcal{Z}(\check{\mathcal{V}}_1) \Cap \mathcal{Z}(\check{\mathcal{V}}_k) = \emptyset$ for all $k \in \{1, \ldots, q\}$. It follows that

$$\Omega^p(\mathcal{V}) = \left(\Omega^p(\mathcal{V}_1) \Cap \bigdoublecap_{k=2}^{q} \left(\Omega^p(\mathcal{V}_k) \Cup \mathcal{Z}(\check{\mathcal{V}}_k)\right)\right) \Cup \mathcal{Z}(\check{\mathcal{V}}_1).$$

Noticing that the middle term is in the same form as (17), we can repeatedly apply the previous two steps to obtain

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \Omega^p(\mathcal{V}_k) \Cup \bigdoublecup_{k=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_k). \tag{18}$$

Next, suppose that $|\mathcal{V}_k| > r$ for each k and let $\bar{\mathcal{V}}_k$ denote a partition of each $\mathcal{V}_k$. We can then perform the exact same expansion of each $\Omega^p(\mathcal{V}_k)$ from (17) to (18), resulting in

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \left(\bigdoublecap_{j=1}^{q} \Omega^p(\mathcal{V}_{k,j}) \Cup \bigdoublecup_{j=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_{k,j}) \right) \Cup \bigdoublecup_{k=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_k).$$

Similarly as above, since $\mathcal{Z}(\check{\mathcal{V}}_{i,j}) \subseteq \Omega^p(\mathcal{V}_{k,j})$ and $\mathcal{Z}(\check{\mathcal{V}}_{i,j}) \Cap \mathcal{Z}(\check{\mathcal{V}}_{k,j}) = \emptyset$ for all $i, j, k \in \{1, \ldots, q\}$, $i \neq k$, we have

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \left(\bigdoublecap_{j=1}^{q} \Omega^p(\mathcal{V}_{k,j}) \right) \Cup \bigdoublecup_{j=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_{k,j}) \Cup \bigdoublecup_{k=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_k).$$

Since $\mathcal{Z}(\check{\mathcal{V}}_{k,j}) \subseteq \mathcal{Z}(\check{\mathcal{V}}_k)$ for each $j \in \{1, \ldots, q\}$, we can simplify as follows:

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \left(\bigdoublecap_{j=1}^{q} \Omega^p(\mathcal{V}_{k,j}) \right) \Cup \bigdoublecup_{k=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_k).$$

By repeatedly performing this expansion according to (8), the term on the left will eventually become the union of base-level regional equilibrium sets $\boldsymbol{\Omega}(\mathcal{V}_{k,j,\ldots})$, which is equal to $\boldsymbol{\Omega}(\mathcal{V})$. Furthermore, since $\bigcup_{k=1}^{q} \mathcal{Z}(\check{\mathcal{V}}_k) = \mathcal{Z}(\bigcup_{k=1}^{q} \check{\mathcal{V}}_k) = \mathcal{Z}(\mathcal{V}^{\text{part}})$, due to (4), we obtain

$$\Omega^p(\mathcal{V}) = \bigdoublecap_{k=1}^{q} \boldsymbol{\Omega}(\mathcal{V}_k) \Cup \mathcal{Z}(\mathcal{V}^{\text{part}}).$$

Finally, since

$$\bigdoublecap_{k=1}^{q} \boldsymbol{\Omega}(\mathcal{V}_k) = \bigdoublecap_{k=1}^{q} \bigdoublecap_{i=1}^{|\mathcal{V}_k|} \mathcal{L}_i = \Omega(\mathcal{V}),$$

we have the desired result $\Omega^p(\mathcal{V}) = \Omega(\mathcal{V}) \Cup \mathcal{Z}(\mathcal{V}^{\text{part}})$ and the proof is completed. □

Proof of Theorem 3

Proof. Suppose $|\mathcal{V}| \leq r$. Then

$$n_{\text{eq}}(\Omega^p(\mathcal{V})) = \sum_{j=1}^{|\Omega^p(\mathcal{V})|} 1 = |\Omega^p(\mathcal{V})| = |\Omega(\mathcal{V})|, \tag{19}$$

where we used the fact that $|\Omega^p(\mathcal{V})| = |\Omega(\mathcal{V})|$ from (7). Next, suppose $|\mathcal{V}| > r$. Then we have

$$n_{\text{eq}}(\Omega^p(\mathcal{V})) = \sum_{j=1}^{|\Omega^p(\mathcal{V})|} \prod_{k=1}^{q} \lambda_j(\mathcal{V}_k), \tag{20}$$

where $|\Omega^p(\mathcal{V})|$ is the number of unique configurations of partition boundary nodes in $\mathcal{V}$ that appear in the global equilibrium set, as proved in Theorem 2. Given one of these configurations, the regional equilibria in each group are independent from each other. Therefore, it suffices to show that $\lambda_j(\mathcal{V}_k)$ is equal to the number of REQ states associated with the set $\mathcal{V}_k$ that are contained in $\boldsymbol{\omega}_j^p(\mathcal{V})$, i.e. $\lambda_j(\mathcal{V}_k) = n_{\text{eq}}\left(\boldsymbol{\omega}_j^p(\mathcal{V}) \cap \Omega^p(\mathcal{V}_k)\right)$.

We can check for compatibility between each entry of $\Omega^p(\mathcal{V})$ and the constituent BREQ states $\Omega^b(\mathcal{V}_k)$ using $\boldsymbol{\delta}_{*j}\big(\Omega^p(\mathcal{V}), \Omega^b(\mathcal{V}_k)\big)$. If there exists j such that this set is empty for some k, then $\lambda_j(\mathcal{V}_k) = 0$ and there are no equilibria. Otherwise, $\boldsymbol{\delta}_{*j}\big(\Omega^p(\mathcal{V}), \Omega^b(\mathcal{V}_k)\big)$ is a singleton since $\Omega^b(\mathcal{V}_k)$ is defined on a subset of the nodes defined in $\Omega^p(\mathcal{V})$. One can then check that the quantity $\gamma_i(\mathcal{V}_k)$ is precisely equivalent to $n_{\text{eq}}\left(\boldsymbol{\omega}_j^p(\mathcal{V}) \Cap \Omega^p(\mathcal{V}_k)\right)$, which is then equal to $|\boldsymbol{\Omega}\left(\boldsymbol{\omega}_j^p(\mathcal{V}) \Cap \Omega^p(\mathcal{V}_k)\right)|$ if $|\mathcal{V}_k| \leq r$, and otherwise the number of compatible lower-level equilibria, and the proof proceeds by induction.

□

Best response and linear threshold dynamics

A common type of network dynamics involves nodes that optimize local cost or *utility* functions. For our purposes, these dynamics are particularly useful in constructing systems that will converge to fixed points and local minima of a function. Let $u_i(x_i, \mathbf{x}_{-i})$ denote the utility to node i when the states of all nodes other than i are collected in the vector $\mathbf{x}_{-i}$. A simple best-response update is given by

$$x_i(t+1) \in \mathcal{B}_i(t), \tag{21}$$

where the set of best responses is

$$\mathcal{B}_i := \left\{ x \in \mathcal{S}_i \,\middle|\, u_i(x, \mathbf{x}_{-i}) \geq u_i(y, \mathbf{x}_{-i}) \quad \forall\, y \in \mathcal{S}_i \right\}.$$

In particular, suppose that each node can take one of two possible states, i.e. $\mathcal{S}_i = \{+1, -1\}$ for all $i \in \mathcal{V}$, and the utility of each node can be expressed as the sum of outcomes of 2×2 matrix games defined on each outgoing edge from node i:

$$u_i(x_i, \mathbf{x}_{-i}) := \sum_{j \in \mathcal{N}_i} \pi^{i,j}_{x_i, x_j},$$

where the *payoff matrix* $\pi^{i,j}$ defines the outcome of a game in which node i chooses the row and node j chooses a column:

$$\pi^{i,j} := \begin{array}{c} \\ + \\ - \end{array} \begin{array}{c} \begin{array}{cc} + & - \end{array} \\ \begin{pmatrix} a_{ij} & b_{ij} \\ c_{ij} & d_{ij} \end{pmatrix} \end{array}, \qquad a_i, b_i, c_i, d_i \in \mathbb{R}.$$

The utility can then be expressed as $u_i(x_i, \mathbf{x}_{-i}) =$

$$\begin{cases} \frac{1}{2}\sum_{j\in\mathcal{N}_i} a_{ij}(1+x_j) + b_{ij}(1-x_j), & \text{if } x_i = +1 \\ \frac{1}{2}\sum_{j\in\mathcal{N}_i} c_{ij}(1+x_j) + d_{ij}(1-x_j), & \text{if } x_i = -1 \end{cases}.$$

The best-response update (21) now becomes

$$x_i(t+1) = \text{sign}\Big(\sum_{j\in\mathcal{N}_i}(\delta_{ij}^+ - \delta_{ij}^-) + x_j(t)(\delta_{ij}^+ + \delta_{ij}^-)\Big),$$

where $\delta_{ij}^+ = a_{ij} - c_{ij}$ and $\delta_{ij}^- = d_{ij} - b_{ij}$. Further simplification reveals a type of linear threshold model, in which a weighted sum of neighbor states is compared to a local threshold value τ_i:

$$x_i(t+1) = \text{sign}\Big(\sum_{j\in\mathcal{N}_i} w_{ij}x_j(t) + \tau_i\Big),$$

where $w_{ij} := \delta_{ij}^+ + \delta_{ij}^-$ and $\tau_i := \sum_{j\in\mathcal{N}_i}(\delta_{ij}^+ - \delta_{ij}^-)$. Threshold models are prominent in various research fields, from computational neuroscience [1] to sociology [53]. A remarkable property of these models in addition to their ability to describe wide-ranging physical phenomena is that, when nodes update asynchronously, the networks will converge to a fixed point if any of the following conditions hold:

1. all weights (payoff matrices) are symmetric, i.e. $w_{ij} = w_{ji}$ ($\pi^{ij} = \pi^{ji}$), for all $i, j \in \mathcal{V}$ [1];
2. all neighbors are treated equally, i.e. $w_{ij} = w_{ik}$ ($\pi^{ij} = \pi^{ik}$) for each $i \in \mathcal{V}$ and all $j, k \in \mathcal{N}_i$, and all nodes are *coordinating*, i.e.$\delta_{ij}^+ > 0$ and $\delta_{ij}^- > 0$ [3];
3. all neighbors are treated equally and all nodes are *anticoordinating*, i.e. $\delta_{ij}^+ < 0$ and $\delta_{ij}^- < 0$ [3].

Graph partitioning and separability analysis

Although it is computationally complex to determine the partition separability of networks in general, there are several graph partitioning algorithms that try to minimize the number of edges between adjacent partition groups while producing groups of approximately equal size [17, 44]. To determine which types of networks are best suited to the proposed approach, we recursively applied the algorithm in [17] on three standard network topologies: geometric random networks (constructed as in **Computational complexity simulations**), small-world networks (constructed using the Watts-Strogatz method [54] with four nearest-neighbor connections and rewire probability $\frac{1}{3}$), and scale-free networks (constructed using preferential attachment with a minimum degree of three [55]). Fig. 7 shows the ratio of the maximum number of edges crossing partition boundaries to the total number of edges in 100 random networks for each of these types. We observe that geometric random networks and small world networks tend to be separable along a smaller portion of edges than scale-free networks, and are therefore better suited to the methods proposed in this article.

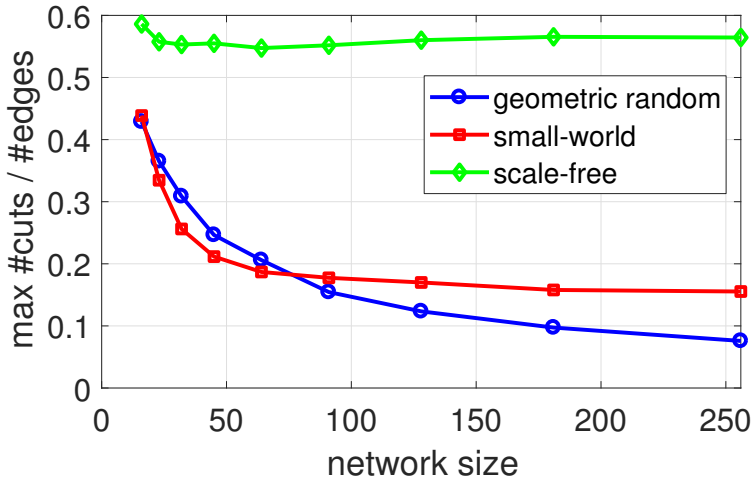

Figure 7: Maximum number of edges crossing partition boundary for 100 randomly generated networks of each type (geometric random, small-world, scale-free).

Computational complexity simulations

The computational complexity study shown in Fig. 1 in the main article consisted of three different network topologies (ring networks, tree networks, and geometric random networks), governed by the best-response threshold dynamics described above. Energy was computed using (11) with $\phi_{ij}(x_i, x_j) := x_i x_j$ and $h_i(x_i) := 0$ for each node $i \in \mathcal{V}$.

The ring networks were undirected and anticoordinating payoff matrices of $\left(\begin{smallmatrix} 0 & 3 \\ 1 & 0 \end{smallmatrix}\right)$ were used for all edges.

Undirected tree networks used in Fig. 1 were constructed by starting from a root node and randomly adding between 0 and 4 branches with uniform probability (0.2), and repeating the process at each of the leaf nodes for up to a maximum number of generations, which was increased from 1 to 11. Coordinating payoff matrices of $\left(\begin{smallmatrix} 5 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ were used for all edges.

Geometric random networks were constructed with sizes $n := \lceil 2^\kappa \rceil$ for each $\kappa \in \{3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9\}$ by fixing a mean degree $\bar{d} = 5$ and then setting the connection radius for each size n to $R := \sqrt{\frac{1+\bar{d}}{\pi n}}$. After uniformly randomly distributing n nodes in the unit square, edges were added between every pair of nodes within a distance R of each other. Any disconnected networks were discarded. Coordinating payoff matrices of $\left(\begin{smallmatrix} 5 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ were used for all edges.

Algorithm pseudocode

To augment the mathematical framework provided in the previous sections, we provide here a step by step implementation of the main algorithm in pseudocode. Due to their length, we provide the algorithms for computing the cost and expanding the optimal or full equilibrium set in the **Appendix**.

Algorithm: Main($\mathcal{V}_0, \mathcal{L}$)

$\boldsymbol{\Omega}^p(\mathcal{V}_0), \mathcal{T} :=$ FindEQ($\mathcal{V}_0, \mathcal{L}$) // compute top-level PREQ set $\boldsymbol{\Omega}^p(\mathcal{V}_0)$ and results tree $\mathcal{T}$
$\boldsymbol{\phi}^p(\mathcal{V}_0), W^p(\mathcal{V}_0) :=$ Cost($\mathcal{V}_0, \Phi, \mathcal{T}$) // compute the unique cost values $\boldsymbol{\phi}^p$ and the number of ways they are attained W^p
Choose $\phi^* \in \boldsymbol{\phi}^p(\mathcal{V}_0)$ // ϕ^* can be min, max, or other cost value
$\mathcal{I}^p := \{j : W^p_{ij}(\mathcal{V}_0) > 0 \text{ and } \boldsymbol{\phi}^p_i(\mathcal{V}_0) = \phi^*\}$ // index set of elements of $\boldsymbol{\Omega}^p(\mathcal{V}_0)$ that can attain the cost ϕ^*
$\Omega^*(\mathcal{V}_0) :=$ Expand($\phi^*, \mathcal{I}^p, V_{\mathcal{V}_0}$) // compute set of all fixed points whose cost is ϕ^*

Algorithm: FindEQ($\mathcal{V}, \mathcal{L}$)

if $|\mathcal{V}| > r$ **then**
  $\bar{\mathcal{V}} = \{\mathcal{V}_1, \ldots, \mathcal{V}_q\} :=$ Partition($\mathcal{V}, q$) // using, e.g. [17]
  foreach $k \in \{1, \ldots, q\}$ **do**
    $\boldsymbol{\Omega}^p(\mathcal{V}_k), \mathcal{T} :=$ FindEQ($\mathcal{V}_k, \mathcal{L}$) // $\boldsymbol{\Omega}^p(\mathcal{V}_k)$ is defined on the node set $\mathcal{V}_k^{\text{part}}$
    $\boldsymbol{\Omega}^b(\mathcal{V}_k) := \boldsymbol{\Omega}^p(\mathcal{V}_k) \Cup \mathcal{Z}(\mathcal{V}_k)$ // $\boldsymbol{\Omega}^b(\mathcal{V}_k)$ is defined on the node set $\mathcal{V}_k^{\text{bound}}$
    Compute $\Delta\big(\boldsymbol{\Omega}^p(\mathcal{V}_k), \boldsymbol{\Omega}^b(\mathcal{V}_k)\big)$
    Compute $\Delta\big(\boldsymbol{\Omega}^b(\mathcal{V}_k), \boldsymbol{\Omega}^p(\mathcal{V})\big)$
  end
  $\boldsymbol{\Omega}^p(\mathcal{V}) := \bigcap_{k=1}^q \boldsymbol{\Omega}^b(\mathcal{V}_k)$ // PREQ set is intersection of all subgroup BREQ sets
  $V_{\mathcal{V}} := \big(\bar{\mathcal{V}}, \boldsymbol{\Omega}^p(\mathcal{V}), \{\boldsymbol{\Omega}^b(\mathcal{V}_k), \Delta\big(\boldsymbol{\Omega}^b(\mathcal{V}_k), \boldsymbol{\Omega}^p(\mathcal{V})\big), \Delta\big(\boldsymbol{\Omega}^p(\mathcal{V}_k), \boldsymbol{\Omega}^b(\mathcal{V}_k)\big)\}_{k=1}^q\big)$ // define results node
  $\mathcal{V}_{\mathcal{T}} := \mathcal{V}_{\mathcal{T}} \cup \{V_{\mathcal{V}}\}$ // add results node to $\mathcal{T}$
  $\mathcal{E}_{\mathcal{T}} := \mathcal{E}_{\mathcal{T}} \cup \{(V_{\mathcal{V}}, V_{\mathcal{V}_1}), \ldots, (V_{\mathcal{V}}, V_{\mathcal{V}_q})\}$ // add a new edge for each subgroup to $\mathcal{T}$
else
  $\boldsymbol{\Omega}^p(\mathcal{V}) := \bigcap_{i=1}^{|\mathcal{V}|} \mathcal{L}_i$ // base-level PREQ set is intersection of all LEQ sets
  $V_{\mathcal{V}} := \big(\bar{\mathcal{V}}, \boldsymbol{\Omega}^p(\mathcal{V}), \emptyset\big)$ // base-level results node
  $\mathcal{V}_{\mathcal{T}} := \mathcal{V}_{\mathcal{T}} \cup \{V_{\mathcal{V}}\}$ // add results node to $\mathcal{T}$ (no new edges since there was no partition)
end
return $\boldsymbol{\Omega}^p(\mathcal{V})$,$\mathcal{T}$ // recall that $\mathcal{T} = (\mathcal{V}_{\mathcal{T}}, \mathcal{E}_T)$

Cost and Expansion Algorithms

Here we provide pseudocode implementations of the algorithms for computing the costs and expanding the optimal or full equilibrium sets.

Algorithm: Cost$(\mathcal{V}, \Phi, \mathcal{T})$

if $|\mathcal{V}| > r$ **then**
  foreach $k \in \{1, \ldots, q\}$ **do**
    $\boldsymbol{\phi}^p(\mathcal{V}_k), W^p(\mathcal{V}_k) := \text{Cost}(\mathcal{V}_k, \Phi, \mathcal{T})$ // recursive call to this function on subgroup $\mathcal{V}_k$
    // w^b_{ij} will be the number of ways cost $\boldsymbol{\phi}^p_i$ is attained by BREQ element $\boldsymbol{\omega}^b_j(\mathcal{V}_k)$
    $W^b(\mathcal{V}_k) := [w^b_{ij}]_{|\boldsymbol{\phi}(\mathcal{V}_k)|, |\boldsymbol{\Omega}^b(\mathcal{V}_k)|} : w^b_{ij} = 0, \forall i, j$ // $[x_{ij}]_{m,n}$ denotes an $m \times n$ matrix whose ij^{th} entry is x_{ij}
    foreach $i \in \{1, \ldots, |\boldsymbol{\phi}(\mathcal{V}_k)|\}$ **do**
      foreach $\ell \in \{1, \ldots, |\boldsymbol{\Omega}^b(\mathcal{V}_k)|\}$ **do**
        Find $j : j \in \boldsymbol{\delta}_{i*}(\boldsymbol{\Omega}^p(\mathcal{V}_k), \boldsymbol{\Omega}^b(\mathcal{V}_k))$ // $\boldsymbol{\delta}_{i*}(\boldsymbol{\Omega}^p(\mathcal{V}_k), \boldsymbol{\Omega}^b(\mathcal{V}_k))$ is a singleton set
        $w^b_{ij} := w^b_{ij} + w^p_{i\ell}$ // add the number of ways $\boldsymbol{\phi}(\mathcal{V}_k)$ is attained by corresponding PREQ element
      end
    end
    $\hat{W}(\mathcal{V}_k) := [\hat{w}_{ij}]_{|\boldsymbol{\phi}(\mathcal{V}_k)|, |\boldsymbol{\Omega}^p(\mathcal{V})|} : \hat{w}_{ij} = 0, \forall i, j$ // $\hat{w}_{ij}$ is the number of ways $\boldsymbol{\phi}_i(\mathcal{V}_k)$ is attained by $\boldsymbol{\omega}^p_j(\mathcal{V})$
    $\hat{Z}(\mathcal{V}_k) := [\hat{z}_{ij}]_{|\boldsymbol{\phi}_k|, |\boldsymbol{\Omega}^p(\mathcal{V})|} : \hat{z}_{ij} = \infty, \forall i, j$ // $\hat{z}_{ij}$ will be the cost $\boldsymbol{\phi}_i(\mathcal{V}_k)$
    foreach $i \in \{1, \ldots, |\boldsymbol{\phi}(\mathcal{V}_k)|\}$ **do**
      foreach $\ell \in \{1, \ldots, |\boldsymbol{\Omega}^b(\mathcal{V}_k)|\}$ **do**
        foreach $j \in \boldsymbol{\delta}_{\ell*}(\boldsymbol{\Omega}^b(\mathcal{V}_k), \boldsymbol{\Omega}^p(\mathcal{V}))$ **do**
          $\hat{w}_{ij} := \hat{w}_{ij} + w^b_{ij}$ // add the number of ways $\boldsymbol{\phi}_i(\mathcal{V}_k)$ is attained by corresponding BREQ element
          $\hat{z}_{ij} := \boldsymbol{\phi}_i(\mathcal{V}_k)$
        end
      end
    end
  end
  // now construct expanded matrices $\mathring{W}(\mathcal{V})$ and $\mathring{Z}(\mathcal{V})$ whose rows correspond to all compatible configurations of the subgroup BREQ sets $\boldsymbol{\Omega}^b(\mathcal{V}_k)$ for $k \in \{1, \ldots, q\}$
  $\mathring{W}(\mathcal{V}) := \hat{W}(\mathcal{V}_1)$ // $\mathring{w}_{ij}$ is the number of ways $\boldsymbol{\omega}^p_i(\mathcal{V})$ can be instantiated by the j^{th} configuration of subgroups
  foreach $k \in \{2, \ldots, q\}$ **do**
    // $\otimes$ denotes the Kronecker product and $\odot$ denotes element-wise multiplication
    $\mathring{W}(\mathcal{V}) := (\mathring{W}(\mathcal{V}) \otimes \mathbf{1}_{|\hat{W}_1(\mathcal{V}_k)|}) \odot (\mathbf{1}_{|\hat{W}_1(\mathcal{V})|} \otimes \hat{W}(\mathcal{V}_k))$ // $|M_1|$ denotes the number of rows in M
  end
  foreach $k \in \{1, \ldots, q\}$ **do**
    $\mathring{Z}(\mathcal{V}_k) := \begin{cases} \hat{Z}(\mathcal{V}_1) & \text{if } k = 1 \\ \mathbf{1}_{|\hat{Z}(\mathcal{V}_1)|} & \text{if } k > 1 \end{cases}$ // the ij^{th} entry of $\mathring{Z}(\mathcal{V}_k)$ is $\boldsymbol{\phi}_i(\mathcal{V}_k)$ for the j^{th} subgroup configuration
    foreach $\ell \in \{2, \ldots, q\}$ **do**
      if $k = \ell$ **then**
        $\mathring{Z}(\mathcal{V}_k) := \mathring{Z}(\mathcal{V}_k) \otimes \hat{Z}(\mathcal{V}_\ell)$
      else
        $\mathring{Z}(\mathcal{V}_k) := \mathring{Z}(\mathcal{V}_k) \otimes \mathbf{1}_{|\boldsymbol{\phi}(\mathcal{V}_\ell)|}$ // $\mathbf{1}_m$ denotes the vector containing m ones
      end
    end
  end
  $\tilde{\boldsymbol{\phi}}(\mathcal{V}) := [\tilde{\boldsymbol{\Phi}}(\boldsymbol{\omega}^p_1(\mathcal{V})), \ldots, \tilde{\boldsymbol{\Phi}}(\boldsymbol{\omega}^p_{|\boldsymbol{\Omega}^p(\mathcal{V})|}(\mathcal{V}))]$ // row vector of inter-group costs
  $\mathring{Z}(\mathcal{V}) := \sum_{k=1}^{q} \mathring{Z}(\mathcal{V}_k) + \tilde{\boldsymbol{\phi}}(\mathcal{V}) \otimes \mathbf{1}_{|\mathring{Z}_1(\mathcal{V}_1)|}$ // total costs for each configuration
  $\boldsymbol{\phi}(\mathcal{V}) := \{\phi < \infty : [\exists i, j : \mathring{z}_{ij} = \phi]\}$ // $\mathring{z}_{ij} = \infty$ means that $\boldsymbol{\phi}_i(\mathcal{V})$ cannot be attained by the j^{th} configuration
  $W^p(\mathcal{V}) := [w^p_{ij}]_{|\boldsymbol{\phi}(\mathcal{V})|, |\boldsymbol{\Omega}^p(\mathcal{V})|} : w^p_{ij} = 0, \forall i, j$ // initialize to matrix of all zeros
  foreach $\mathring{z}_{ij} \in \mathring{Z}(\mathcal{V}) : \mathring{z}_{ij} < \infty$ **do**
    Find $\ell : \boldsymbol{\phi}_\ell(\mathcal{V}) = \mathring{z}_{ij}$
    $w^p_{i\ell} := \mathring{w}_{ij}$
  end
else
  $\boldsymbol{\phi}(\mathcal{V}) := (\{\phi : \exists i : \boldsymbol{\Phi}(\boldsymbol{\omega}^p_i(\mathcal{V})) = \phi\}, \rho)$ // compute costs of all elements of $\boldsymbol{\Omega}^p(\mathcal{V})$
  foreach $i \in \{1, \ldots, |\boldsymbol{\phi}(\mathcal{V})|\}$ **do**
    $w^p_{ij} := \left|\{\mathbf{x} \in \boldsymbol{\omega}^p_i(\mathcal{V}) : \Phi(\mathbf{x}) = \boldsymbol{\phi}_j(\mathcal{V})\}\right|$ // record how many times each cost value is attained
  end
end
return $\boldsymbol{\phi}(\mathcal{V}), W^p(\mathcal{V})$

Algorithm: Expand$(\phi^*, \mathcal{I}^p, V_{\mathcal{V}})$

$(\bar{\mathcal{V}}, \mathbf{\Omega}^p(\mathcal{V}), \mathcal{D}(\mathcal{V})) := V_{\mathcal{V}}$ // expand results node
if $\mathcal{D}(\mathcal{V}) = \emptyset$ **then**
  return $\mathbf{\Omega}^p(\mathcal{V})$
end
$\mathcal{D}(\mathcal{V}) := \{\mathbf{\Omega}^b(\mathcal{V}_k), \Delta\big(\mathbf{\Omega}^b(\mathcal{V}_k), \mathbf{\Omega}^p(\mathcal{V})\big), \Delta\big(\mathbf{\Omega}^p(\mathcal{V}_k), \mathbf{\Omega}^b(\mathcal{V}_k)\big)\}_{k=1}^{q}$ // expand subgroup data
foreach $i \in \mathcal{I}^p$ **do**
  foreach $k \in \{1, \ldots, q\}$ **do**
    $\mathcal{I}^b := \bigcup_{j \in \mathcal{I}^p} \boldsymbol{\delta}_{j*}(\mathbf{\Omega}^p(\mathcal{V}), \mathbf{\Omega}^b(\mathcal{V}_k))\}$ // get subgroup BREQ indices from PREQ indices
    $\hat{\mathcal{I}}(\boldsymbol{\omega}_i^p(\mathcal{V}_k)) := \bigcup_{j \in \mathcal{I}^b} \boldsymbol{\delta}_{j*}(\mathbf{\Omega}^b(\mathcal{V}_k), \mathbf{\Omega}^p(\mathcal{V}_k))$ // get subgroup PREQ indices from subgroup BREQ indices
    $\hat{Z}(\mathcal{V}_k) := \boldsymbol{\phi}(\mathcal{V}_k)$ // $\boldsymbol{\phi}(\mathcal{V}_k)$ was computed in Cost function
  end
  foreach $k \in \{1, \ldots, q\}$ **do**
    if $k = 1$ **then**
      $\mathring{Z}(\mathcal{V}_k) := \hat{Z}(\mathcal{V}_1)$ // $\mathring{Z}(\mathcal{V}_k)$ will be an expanded $\hat{Z}(\mathcal{V}_k)$ to allow for all subgroup configurations
    else
      $\mathring{Z}(\mathcal{V}_k) := \mathbf{1}_{|\hat{Z}(\mathcal{V}_1)|}$
    end
    foreach $\ell \in \{2, \ldots, k\}$ **do**
      if $k = \ell$ **then**
        $\mathring{Z}(\mathcal{V}_k) := \mathring{Z}(\mathcal{V}_k) \otimes \hat{Z}(\mathcal{V}_\ell)$
      else
        $\mathring{Z} := \mathring{Z}(\mathcal{V}_k) \otimes \mathbf{1}_{|\hat{Z}(\mathcal{V}_\ell)|}$
      end
    end
  end
  $\tilde{\boldsymbol{\phi}}(\mathcal{V}) := [\tilde{\mathbf{\Phi}}(\boldsymbol{\omega}_1^p(\mathcal{V})), \ldots, \tilde{\mathbf{\Phi}}(\boldsymbol{\omega}_{|\mathbf{\Omega}^p(\mathcal{V})|}^p(\mathcal{V}))]$ // row vector of inter-group costs
  $\mathring{Z}(\mathcal{V}) := \sum_{k=1}^{q} \mathring{Z}(\mathcal{V}_k) + \tilde{\boldsymbol{\phi}}(\mathcal{V}) \otimes \mathbf{1}_{|\mathring{Z}_1(\mathcal{V}_1)|}$ // $\mathring{Z}_j(\mathcal{V})$ contains the total cost for each configuration of $\boldsymbol{\omega}_j^p(\mathcal{V})$
  foreach $k \in \{1, \ldots, q\}$ **do**
    $Z^*(\boldsymbol{\omega}_i^p(\mathcal{V}_k)) := \{\mathring{z}_{j\ell} \in \mathring{Z}(\mathcal{V}_k) : \mathring{z}_{j\ell} = \phi^*\}$ // subgroup costs for each configuration that can attain ϕ^*
  end
end
$\Omega^*(\mathcal{V}) := \emptyset$
foreach $i \in \mathcal{I}^p$ **do**
  foreach $j \in \{1, \ldots, |Z^*(\boldsymbol{\omega}_i^p(\mathcal{V}_1))|\}$ **do**
    $\hat{\Omega}(\mathcal{V}) := \emptyset$ // initialize the set
    $\hat{\phi}_j := Z_j^*(\boldsymbol{\omega}_i^p(\mathcal{V}_k))$
    foreach $k \in \{1, \ldots, q\}$ **do**
      // find subgroup indices that are contained in the current PREQ element and can attain the cost $\hat{\phi}_j$
      $\mathcal{I}^* = \{\ell \in \hat{\mathcal{I}}(\boldsymbol{\omega}_i^p(\mathcal{V}_k)) : w_{\ell i}^p(\mathcal{V}_k) > 0 \text{ and } \hat{\phi}_j \in \boldsymbol{\phi}^p(\mathcal{V}_k)\}$
      if $\mathcal{I}^* = \emptyset$ **then**
        $\hat{\Omega}(\mathcal{V}) := \emptyset$
        break // this particular subgroup configuration will not attain $\hat{\phi}_j$
      else
        $\hat{\Omega}(\mathcal{V}_k) := \text{Expand}(\hat{\phi}_j, \mathcal{I}^*, V_{\mathcal{V}_k})$ // recursively expand the lower level groups
      end
      if $\hat{\Omega}(\mathcal{V}) = \emptyset$ **then**
        $\hat{\Omega}(\mathcal{V}) := \Omega^*(\mathcal{V}_k)$
      else
        $\hat{\Omega}(\mathcal{V}) := \hat{\Omega}(\mathcal{V}) \Cap \hat{\Omega}(\mathcal{V}_k)$
      end
    end
    $\Omega^*(\mathcal{V}) := \Omega^*(\mathcal{V}) \cup \hat{\Omega}(\mathcal{V})$ // add these optimal (regional) equilibrium sets to the result set
  end
end