Maximizing the Eigenvalue-Gap and Promoting Sparsity of Doubly Stochastic Matrices with PSO

Panos K. Syriopoulos(✉), Nektarios G. Kalampalikis, and Michael N. Vrahatis

Computational Intelligence Laboratory, Department of Mathematics, University of Patras, 26110 Patras, Greece
{p.syriopoulos,nkalamp,vrahatis}@math.upatras.gr

Abstract. The eigenvalue-gap of doubly stochastic matrices with sparsity constraints is maximized using the unified particle swarm optimizer. This is possible through the use of an iterative normalization procedure that maps the search space of the swarm to the set of doubly stochastic matrices with given sparsity pattern. We extend the method to the problem of finding doubly-stochastic matrices of given dimensions that are as sparse as possible, and attain a given eigenvalue-gap target.

Keywords: Eigenvalue-gap · Particle swarm optimization · Sparse matrix identification · Distributed averaging

1 Introduction

The study of complex networks is a fascinating subject with far reaching results both in terms of their practical applications and in their theoretical foundations. Understanding complex networks is crucial for the cooperation and control of autonomous agents. It is fair to say that complex networks occur in various aspects of life. Interestingly, one of the pioneers of the study of complex networks was a psychiatrist, and founder of group-therapy, Jacob Moreno (1889–1974), with his paper "Statistics of social configurations" [18]. The study of social networks still remains an active area of research [3]. In engineering the applications of complex networks are numerous. For example, after the introduction of the multiprocessor, consensus networks were quickly applied for the task of dynamic load balancing [4]. Modern data centers require more sophisticated solutions to be scalable and reliable [14,25]. Among other applications, energy micro grids is a recent area for distributed control and presents new challenges [11,13]. Additionally, sensor networks have several modern day applications [12], while significant attention has been given to autonomous vehicle control [22]. Complex networks also model many natural processes such as bird flocking [10]. In general, one has to consider the dynamics of the network (how the network structure

D. E. Simos et al. (Eds.): LION 2022, LNCS 13621, pp. 535–548, 2022.
https://doi.org/10.1007/978-3-031-24866-5_38

changes) and the dynamics in the network (how nodes interact). The major concepts can be found, among others, in [1]. A lot of the theoretical interest is on emergent phenomena such as the unidimensionality of beliefs [6].

A basic but integral part of distributed control is *consensus*. The purpose is for all agents in the network to agree on an underlying state of their environment. A special case is distributed averaging. In this instance, the nodes have to agree on the average of their initial measurements by communicating through the network [8]. We consider linear update rules. In each communication round, each node updates its estimate by taking a weighted sum of the estimates of its neighbors. To mention a few applications, the distributed Kalman filter can be seen as a distributed averaging problem [19], while, distributed averaging is heavily leveraged in modern distributed optimisation applications [7].

We are concerned with two key problems related to (a) the *speed of convergence to the average* and (b) the problem of *optimizing the network itself*. In other words, for problem (a) we want to minimize the number of communication rounds required for all the nodes to approximate the average within an error bound. It turns out that the asymptotic speed of convergence is bounded above by the second largest eigenvalue in magnitude. The problem is to optimize the network weights for speed, while respecting the network connectivity constraints. For problem (b), given a target for the asymptotic speed of convergence, we aim to find a network structure that is as *sparse* as possible. We tackle these problems using the *unified particle swarm optimization UPSO* (see [21], the corresponding MATLAB code and references therein), which is an effective and efficient evolutionary algorithm, and can also be considered as a complex network. The algorithm consists of particles that perform random search locally, and exchange information in the framework of their network connectivity in order to minimize a given objective function. The ability of the algorithm to optimize arbitrary functions is an emergent phenomenon of the dynamics between the particles. There are also several advantages. Firstly, the algorithm is easily applied and widely accessible. Secondly, it is incredibly capable in solving hard problems: the objective function can be non-differentiable and even discontinuous, and the algorithm can be applied to spaces with non connected regions. These characteristics appear frequently in difficult real life applications and UPSO can effectively tackle these issues. As a result, we are interested to study how the algorithm performs in aforementioned problems (a) and (b). In any case, we believe that this work is a useful demonstration of the ability of evolutionary algorithms in optimization. Our contribution is in the use of an iterative normalization scheme, which enables the application of evolutionary algorithms in this context.

The paper is structured as follows. Section 2 contains the background material. In Sect. 3 the objective functions are formulated. Finally, Sect. 4 contains experiments, while, in Sect. 5, a synopsis and concluding remarks are given.

2 Preliminaries

This section provides a basic understanding of the background material. To motivate the problem of eigenvalue gap maximization, we examine the effect of eigen-

values in the context of consensus networks [8]. We exhibit that the consensus speed depends on the magnitude of the second largest eigenvalue in modulus (for the diagonalizable case). In general, the consensus problem is grounded in the theory of Markov chains. Details can be found in [15]. Alternatively, we refer the interested reader to [17] for further mathematical treatments of bounds for consensus speed. Furthermore, we describe the unified PSO algorithm (UPSO) [21] with which we tackle the eigenvalue gap maximization problem.

Throughout the remainder of the paper, the structure of a network is represented by a graph $G(V, E)$ where $V = \{1, 2, \ldots, n\}$ is the vertex set and E is the ordered set containing the edges $E = \{(i, j) : \text{node } i \text{ listens to node } j\}$. The set of weights that nodes give to their neighbors can be represented by a matrix W with elements $W_{i,j}$ such that:

$$W_{i,j} = \begin{cases} W_{i,j} > 0, & \text{if } \{i, j\} \in E, \\ 0, & \text{if } \{i, j\} \notin E. \end{cases} \tag{1}$$

2.1 Consensus and Eigenvalues

Suppose $W \in \mathbb{R}^{n \times n}$ is a stochastic matrix with non-negative entries so that each row sums to one, i.e. $\sum_j^n W_{i,j} = 1$, for all $i \in \{1, 2, \ldots, n\}$. Given any $x(0) \in \mathbb{R}^n$, we are concerned with the convergence speed of the sequence $x(t)$, $t = 0, 1, \ldots$ defined by:

$$x(t + 1) = Wx(t) = W^{t+1}x(0). \tag{2}$$

This is known as the *consensus model of DeGroot* and was initially formulated as a method for pooling probability distributions [5]. Intuitively $x(0)$ can be seen as the vector of beliefs of each node at time 0 (their initial approximations), and W is the matrix of the network's edge weights. The application of W updates the belief of the nodes by taking a linear combination of the beliefs of their neighbors. The *consensus vector* denoted by x_c is given by the following limit, provided it exists:

$$x_c = \lim_{t \to \infty} x(t) = \lim_{t \to \infty} W^t x(0). \tag{3}$$

It can be seen that the existence of the limit in Eq. (3) depends on the existence and uniqueness of a left eigenvector with eigenvalue 1. Suppose $\pi \in \mathbb{R}^n$ such that $\pi X = \pi$ is the unique eigenvector (up to normalization) with eigenvalue 1. Then we have that:

$$\pi x(t + 1) = \pi(Wx(t)) = (\pi W)x(t) = \ldots = \pi x(0). \tag{4}$$

Taking the limit as $t \to \infty$ above we obtain:

$$\pi x_c = \pi x(0).$$

At this point it can be seen that if W is doubly-stochastic (rows and columns sum to one), then $\pi = [1/n, 1/n, \ldots, 1/n]^\top$ is a left eigenvector, and the system converges to the average of the values of $x(0)$.

We can derive sufficient conditions for the existence of the limit in Eq. (3) using linear algebra. It is known that the dominant eigenvalue of any stochastic matrix has modulus one. Additionally, we know from the Perron-Frobenius theorem that strictly positive matrices have simple dominant eigenvalues. Since the eigenpairs of the matrices W^t are the same for all $t = 1, 2, \ldots$, we simply require that W^{t_0} is strictly positive for some $t_0 \in \mathbb{N}$. In terms of the graph induced by W, the interpretation of this condition is that for all $t \geqslant t_0$, any vertex can be reached from any other vertex in exactly t steps. This is equivalent to requiring that the graph of the network is irreducible and aperiodic. These conditions are also necessary for consensus. To this end, consider, for example, the following periodic matrix:

$$T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The beliefs of the nodes will swap in every iteration, thus, never reaching consensus. If, on the other hand, the network is reducible, the independent strongly connected components of the network can reach consensus on their respective averages, but not on the total average of all nodes in the network (since information does not flow from some independent component to others).

We showcase that the convergence speed depends directly on the second largest eigenvalue for diagonalizable matrices. We do this by considering the spectral decomposition. Supposing W is diagonalizable, we can write:

(a) $W = PDP^{-1}$,
(b) $W^t = PD^tP^{-1}$,

where D is a diagonal matrix $D = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, $P = [f_1, f_2, \ldots, f_n]^\top$ where f_i's are the right eigenvectors of W, and $P^{-1} = [\pi_1, \pi_2, \ldots, \pi_n]$ where π_i's are the left eigenvectors of W. Assuming, W is generic (i.e. non-singular) we have that:

$$f_i\pi_j = \begin{cases} 0, & \text{if} \quad i \neq j, \\ 1, & \text{if} \quad i = j. \end{cases}$$

We can define matrices M_i for $i = 1, 2, \ldots, n$ such that:

$$M_k = f_k\pi_k^\top = \begin{bmatrix} f_k(1)\,\pi_k(1) & \ldots & f_k(1)\,\pi_k(n) \\ \vdots & \ddots & \vdots \\ f_k(n)\,\pi_k(1) & \ldots & f_k(n)\,\pi_k(n) \end{bmatrix}.$$

Then we can check that:

$$M_iM_j = \begin{cases} 0, & \text{if} \quad i \neq j, \\ M_i, & \text{if} \quad i = j. \end{cases} \tag{5}$$

with these definitions, one can see that the above relation $W = PDP^{-1}$ is equivalent to:

$$W = PDP^{-1} = \lambda_1M_1 + \lambda_2M_2 + \cdots + \lambda_nM_n.$$

By taking powers of the above equation, and due to property (5), we can see that:

$$W^t = \lambda_1^t M_1 + \lambda_2^t M_2 + \cdots + \lambda_n^t M_n. \tag{6}$$

Eq. (6) shows that the convergence in Eq. (2) directly depends on the eigenvalues of W. For simplicity, index the eigenvalues so that:

$$|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|.$$

Since we are working with doubly stochastic matrices, $\lambda_1 = 1$ is the dominant eigenvalue. By the Perron-Frobenius theorem, λ_1 is simple, i.e. $|\lambda_2| < |\lambda_1| = 1$, and so $\lim_{t\to\infty} |\lambda_2|^t = 0$. Clearly, the smaller the magnitude $|\lambda_2|$, the faster the decay, and the faster the convergence of Eq. (2). The works cited in the beginning of this section provide several proofs for the general case of irreducible and aperiodic Markov Chains. In general, the second largest eigenvalue in magnitude bounds the convergence speed of Eq. (2) from above.

2.2 Relevant Approaches

The previous subsection demonstrated that in order to maximize the asymptotic speed of convergence of the DeGroot model, one has to minimize the second largest eigenvalue in modulus. In the literature, the second largest eigenvalue magnitude is often abbreviated as SLEM and its minimization solves the *Fastest Mixing Markov Chain* (FMMC) problem. It has been solved for medium and large, symmetric problems by Boyd *et al.* using a sub-gradient method [2,26]. In fact, they show that SLEM minimization can be cast as a *Semi-Definite Program* (SDP). To see the formulation of the problem one can simply observe that the complete graph with equal weights yields the fastest possible network. It corresponds to full connectivity (all nodes communicate with all other nodes), and it converges to the average in one iteration. Obviously this particular case is of no practical interest. We would like to impose the network's communication restrictions. The maximisation of the asymptotic convergence speed and the per-step convergence speed are defined by the following two problems:

$$\begin{aligned} &\min\ \varrho(W - \mathbf{1}\mathbf{1}^\top n^{-1}), \\ &\text{s.t.}\quad W \in S, \\ &\qquad \mathbf{1}^\top W = \mathbf{1}^\top, \\ &\qquad W\mathbf{1} = \mathbf{1}. \end{aligned} \qquad \text{and} \qquad \begin{aligned} &\min\ \left\|W - \mathbf{1}\mathbf{1}^\top n^{-1}\right\|, \\ &\text{s.t.}\quad W \in S, \\ &\qquad \mathbf{1}^\top W = \mathbf{1}^\top, \\ &\qquad W\mathbf{1} = \mathbf{1}. \end{aligned} \tag{7}$$

respectively. The matrix $\mathbf{1}\mathbf{1}^\top n^{-1}$ is known as the *averaging matrix* and corresponds to a complete graph with all weights equal to $1/n$. The first problem minimises the spectral radius (i.e., the largest absolute value of the eigenvalues of the matrix) and is generally hard to solve because it is non-convex and not Lipschitz continuous [20]. The second problem minimises the spectral norm, that

is, $\|W\|$ is the largest singular value of W. If W is constrained to be symmetric, then the two problems coincide. The set S is the network's communication restrictions and it corresponds to Eq. (1).

The problem of optimizing the network itself is concerned with finding the sparsest possible communication graph, on a given set of nodes, for a given asymptotic convergence speed target (given in terms of the modulus of the second largest eigenvalue). Our solution to this problem is inspired by the work of authors in [16]. Their work is concerned with continuous time distributed consensus, but it is easily relatable to the discreet equivalent. Their aim is to solve the following problem:

$$\min J(W) + \gamma \, \mathbf{card}(W), \tag{8}$$

where $J(W)$ is a measure of performance, $\mathbf{card}(W)$ is the number of non-zero elements of the matrix W, and γ is a scalar factor. They do this by considering a relaxation of the $\mathbf{card}(\cdot)$ function and forming an SDP. They solve problem (8), and follow with a "polishing step" (of the sparse network) to optimize for convergence speed. Our work differs in the fact that the "sparsification" of the network is done simultaneously with the convergence speed optimization. Moreover, we treat the convergence speed as an input: a target for asymptotic convergence speed given in terms of SLEM, and the final output is a network optimized for sparsity with given speed.

2.3 Unified Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic optimization method inspired by the aggregating behaviors of populations. Given an objective function to be minimized, a population of candidate solutions (particles) is moved around in the search space in accordance with a set of mathematical formulas which dictate the particles' positions and velocities. The movement of each particle of the swarm is influenced by its local best-known position, and also by the best-known positions of the swarm. Iterating the process is expected to shift the swarm towards a good candidate solution. We can distinguish between two approaches: the local approach and the global approach. In the global approach, particles take into account the overall best position ever found by all particles in the swarm. Known as *global PSO variant* (gbest), this approach has a good convergence ability towards the global best positions found during the optimization process, and is distinguished for its exploitation abilities. On the other hand, in the *local PSO variant* (lbest), particles take into account the best position ever found by neighboring particles only, thus, having better exploration ability. The local variant is better at detecting the most favorable regions of the search space. The *Unified Particle Swarm Optimization* (UPSO) [21] variant harnesses both properties. Through the use of a new parameter $u \in (0, 1)$, called *unification factor*, it controls the impact of exploitative and exploratory characteristics. Suppose N is the swarm size, n is the dimension of the problem at hand, V_i is the velocity of the particle x_i, X_i is the position of the particle x_i, P_i is the best position ever visited by particle x_i. Also, let $G_i(t+1)$ denote the velocity update

of the i-th particle, x_i, for the global PSO variant with constriction coefficient χ, which is defined as:

$$G_i(t+1) = \chi\left(V_i(t) + c_1 r_1(P_i(t) - X_i(t)) + c_2 r_2(P_g(t) - X_i(t))\right), \tag{9}$$

and $L_i(t+1)$ denote the corresponding velocity update for the local PSO variant:

$$L_i(t+1) = \chi\left(V_i(t) + c_1' r_1'(P_i(t) - X_i(t)) + c_2' r_2'(P_{li}(t) - X_i(t))\right), \tag{10}$$

where r_1, r_2, r_1', r_2' are stochastic parameters uniformly distributed within the range [0,1], χ is a parameter called constriction coefficient or constriction factor (with typical value $\chi = 0.729$), c_1, c_2, c_1', c_2' are weighting constants called cognitive and social parameter respectively (usually set to 2.05), g denotes the index of the overall best position, and l_i denotes the best position in the neighborhood of x_i. Then, the velocity V_i and the position of the particle X_i are updated as follows:

$$V_i(t+1) = u\, G_i(t+1) + (1-u)\, L_i(t+1), \tag{11}$$

$$X_i(t+1) = X_i(t) + V_i(t+1). \tag{12}$$

Eqs. (11) and (12) indicate that the new shifted position of a particle in UPSO is made up of a weighted combination of the Global and Local PSO position shifts. Consequently, both the local's inherent exploration capabilities and the global's inherent exploitation capabilities contribute. In the special cases where the unification factor $u = 0$ and $u = 1$, UPSO coincides with the original local and global PSO variant, correspondingly. Values around the middle point, $u = 0.5$, are expected to produce more balanced behaviors with respect to the exploration/exploitation abilities. Several additional PSO methods, together with standard parameter settings have been proposed (see e.g., [21]).

3 Problem Formulation

In this section we formulate objective functions for the key problems presented in the introduction: (a) eigenvalue gap maximization, and (b) the maximally sparse network given asymptotic convergence speed target. Our approach is relatively straight forward. We utilize the UPSO algorithm because it can be effectively applied to difficult problems, including among others, problems with a discontinuous function. It is also worth noting that an effective solution of aforementioned problem (b) is enabled by an effective solution of the problem (a).

Start by fixing a graph $G(V, E)$, and a corresponding doubly-stochastic weight matrix $W \in \mathbb{R}^{n\times n}$. We index the eigenvalues of W so that:

$$1 = |\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|.$$

Note that $|\lambda_1| = |\lambda_2|$ occurs in three cases: (i) W induces a periodic graph, (ii) W induces a graph with more than one strongly connected components, (iii)

both (i) and (ii). As indicated in Subsect. 2.1 these cases are of no interest to us. Denote the eigenvalue gap of W as:

$$e_{\text{gap}}(W) = ||\lambda_1| - |\lambda_2||\,, \tag{13}$$

we define the function:

$$L(W) = \begin{cases} -e_{\text{gap}}(W), & \text{if} \quad e_{\text{gap}}(W) \neq 0, \\ \infty, & \text{if} \quad e_{\text{gap}}(W) = 0. \end{cases} \tag{14}$$

There are several things to note. First, eigenvalues need to be calculated, a task which is computationally expensive. We can ease the computational burden by using Lanczos algorithm for the approximation of dominant eigenvalues (see for example [23]). Secondly, the loss function disregards reducible and/or aperiodic matrices by giving them maximum penalty. This should not be a problem for evolutionary algorithms. Last but not least, the loss function (14) has to be minimized for weight matrices satisfying the following restrictions:

(a) $W\mathbf{1} = \mathbf{1}^\top W = \mathbf{1}$ (doubly-stochastic).
(b) $W_{i,j} = 0$ if $\{i, j\} \notin E$.

The second restriction can be satisfied naturally by a vectorization of the sparse matrix W. If there are $k = |E|$ edges (here $|E|$ denotes the cardinality of the edge set E), then $\mathbf{vec}(W) \in \mathbb{R}^k$. The first restriction is crucial and needs to be discussed. There are several ways by which one might try to restrict the attention of the particles to doubly-stochastic matrices. One of them is to penalize deviations of the sum of the rows and columns from unity. According to our experience this procedure is not efficient. Allowing the particles to search the k-dimensional Euclidean space (where $k = |E|$) makes it very difficult to randomly come across a doubly stochastic matrix. Even if that happens, the evolution of the velocity vectors makes it very difficult to reach a different doubly-stochastic matrix. To get around this issue, we find a map from the Euclidean space to the set of doubly-stochastic matrices with a given sparsity pattern. This way, the particles will be able to search the Euclidean space, then map their position into the set of doubly-stochastic matrices, and retain the relevance of the velocity vectors. The loss function, then, essentially becomes a composition of Eq. (14) with that mapping.

We present an iterative normalization scheme for constructing a doubly stochastic matrix. The idea is to normalize the rows and the columns of W iteratively until the matrix becomes doubly stochastic. The proposed iterative normalization scheme is exhibited in Algorithm 1.

Iterative normalization is related to the Sinkhorn-Knopp algorithm [24]. It is shown that it converges in linear time and the only requirement is that the initial matrix W has support. It should also be seen that the sparsity pattern of W remains unchanged.

Denoting the iterative normalization function as $N(\cdot)$ there are two ways to incorporate iterative normalization in the UPSO algorithm and both of them

Algorithm 1. Iterative Normalization I

```
function NORMALIZE(W, k = 10)
    dim ← the dimension of W, i.e. n
    for i = 0, 1, ..., k do
        for j = 0, 1, ..., dim do
            W_j ← W_j / (Σ_{k=1}^{dim} W_{j,k})
        W ← W^⊤
    if W is doubly-stochastic then
        return W
    else
        Normalize(W)
```

are viable. One way is to incorporate it into the loss function. Then, the loss for problem (a) becomes:

$$L_a(W) = L \circ N(W) = L(N(W)). \tag{15}$$

The other is to incorporate $N(\cdot)$ into the UPSO step. In that case the UPSO equations for particle i become:

$$\begin{aligned} G_i(t+1) &= \chi\left(V_i(t) + c_1 r_1 (P_i(t) - X_i(t)) + c_2 r_2 (P_g(t) - X_i(t))\right), \\ L_i(t+1) &= \chi\left(V_i(t) + c_1' r_1' (P_i(t) - X_i(t)) + c_2' r_2' (P_{li}(t) - X_i(t))\right), \\ V_i(t+1) &= u\, G_i(t+1) + (1-u)\, L_i(t+1), \\ X_i^{\text{temp}}(t+1) &= X_i(t) + V_i(t+1), \\ X_i(t+1) &= N\big(X_i^{\text{temp}}(t+1)\big). \end{aligned} \tag{16}$$

In our experience, without iterative normalization, the velocity vector is tasked with determining the absolute change necessary of the state vector $X_i = \mathbf{vec}(W_i)$. In contrast, the output of $N(W_i)$ is determined by the relative magnitudes of the elements of W_i. By composing the loss function (14) with $N(\cdot)$, the velocity vectors are now tasked with determining favorable changes for the relative magnitudes of the elements of $X_i(t+1)$ instead. To elaborate a little further, the elements of the velocity vectors may contain positive elements only (although not necessarily). The relative magnitudes of these elements translate to changes in the relative magnitudes of the elements of the state vectors X_i^{temp}. These, in turn, determine a doubly-stochastic matrix through iterative normalization. Both schemes (15) or (16), results in effective solutions for eigenvalue-gap maximization with UPSO and other evolutionary algorithms.

For the problem of optimizing the network itself, we would like to provide the number of nodes n, and a target eigenvalue gap denoted by $e_{\text{gap}}^{\text{tgt}} \in (0, 1)$. The task is to find a positive weight matrix $W^* \in \mathbb{R}^{n \times n}$ which satisfies the following:

(a) It is doubly-stochastic: $W^*\mathbf{1} = \mathbf{1}^\top W^* = \mathbf{1}$.

(b) It attains the target eigenvalue gap: $L(W^*) = e_{\text{gap}}^{\text{tgt}}$.
(c) It is as sparse as possible, i.e. $\textbf{card}(W^*)$ is as small as possible.

We achieve the above by allowing for a trade-off between the deviation from the target eigenvalue gap, and the number of non-zero elements of the corresponding weight matrix W. When the trade-off is favorable, particles in UPSO will remove an edge of W, and continue subsequent iterations by improving the loss until the next trade-off is favorable. Note that this is happening with a single initialization, a single UPSO swarm. Doing this is enabled by UPSO's ability to perform well in problem (a). Furthermore, to facilitate the process, we find a suitable relaxation for the $\textbf{card}(\cdot)$ function that penalizes small entries of W. First we add a safety loop in the iterative normalization scheme. If an entry of W is sufficiently small, we turn it to zero. To this end we give the following Algorithm 2:

Algorithm 2. Iterative Normalization II

```
function NORMALIZE(W, k = 10, threshold = 0.01)
    dim ← the dimension of W, i.e. n
    for every entry of W do
        if W_{i,j} < thresholds then W_{i,j} ← 0
    for i = 0, 1, ..., k do
        for j = 0, 1, ..., dim do
            W_j ← W_j / (Σ_{k=1}^{dim} W_{j,k})
        W ← W^⊤
    if W is doubly-stochastic then
        return W
    else
        Normalize(W)
```

This will enable us to reliably identify the graph associated with the weight matrix W. For the remainder of the paper at hand, for the sake of readability, we simply write W for the matrix that results after iterative normalization (c.f., Algorithm 2). We define the adjacency matrix A^W associated with W whose entries are given by:

$$A_{i,j}^W = \begin{cases} 0, & \text{if} \quad W_{i,j} = 0, \\ 1. & \text{if} \quad W_{i,j} > 0. \end{cases}$$

Then, the relaxation of the $\textbf{card}(\cdot)$ function for a particular W matrix is given by:

$$C(W) = \sum_{i,j} (A_{i,j}^W - W_{i,j}). \tag{17}$$

Clearly, $C(\cdot)$ is positive and increasing with the number of non-zero elements of W. Additionally, the smaller the entry $W_{i,j}$, the larger its contribution

to $C(W)$. Thus, C essentially penalizes small entries. When a small entry $W_{i,j}$ falls below the threshold value, i.e. $W_{i,j} < \text{threshold}$, Algorithm 2 turns it to zero, and $C(W') \approx C(W) - (1 - \text{threshold})$. Of course, $C(\cdot)$ is a discontinuous function.

To control the eigenvalue gap, we define the following function:

$$D(W) = \min\left(\frac{|L(W)|}{e_{\text{gap}}^{\text{tgt}}},\ \frac{e_{\text{gap}}^{\text{tgt}}}{|L(W)|}\right). \tag{18}$$

That is, $0 < D(W) < 1$ if the eigenvalue gap of W deviates from the target, while $D(W) = 1$ if W attains the target eigenvalue gap. Then, the loss function can be written as follows:

$$L_b(W) = \frac{C(W)}{D(W)^2}. \tag{19}$$

In the loss function (19), the denominator is maximized when W attains the target eigenvalue-gap, and the numerator is minimized when W is as sparse as possible. The square in the denominator guarantees that reaching the target gap is prioritized over removing edges from the graph associated with W. In our experience, when a certain entry of the W matrix approaches the user defined threshold value (in Algorithm 2), the particle swarm faces a trade-off between the numerator and the denominator. If the edge associated with that entry of W is removed, then the numerator is reduced by approximately $(1 - \text{threshhold})$, however, a deviation from the eigenvalue-gap target is induced, and is reflected in the denominator. The next section shows that UPSO performs nicely in both problems: eigenvalue gap maximization, and, maximization of network sparsity given an asymptotic convergence speed target.

A final note concerns the restriction of search to symmetric matrices. The simplest way to do this is to augment the Eqs. (16) with an extra equation to "symmetrize" the doubly-stochastic matrix. That is, supposing $W_i(t+1)$ is the matrix corresponding to vector $X_i(t+1)$, the following step is taken directly after iterative normalization:

$$W_i^{'}(t+1) = \big(W_i(t+1) + W_i^{\top}(t+1)\big)/2. \tag{20}$$

The new vectorized position of the particle is $X_i^{'}(t+1) = \mathbf{vec}(W_i^{'}(t+1))$.

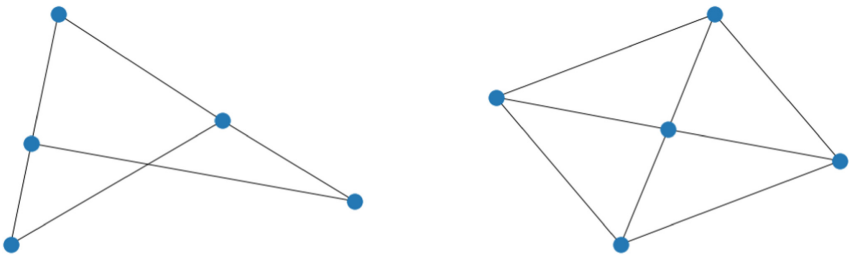
4 Experiments

The experiments carried out are restricted to symmetric edge weights. We consider two 5×5 graphs with known optimal eigenvalue-gaps. The graphs are shown in Fig. 1. We use loss function in Eq. (14) with the augmented UPSO Eqs. (16) together with Eq. (20) for "symmetrization". In our experiments we find optimal or near-optimal eigenvalue-gaps. We employ the widely used convention of $4\,|E|$ particles. Then, we test objective (19) on the sparse matrix identification problem. We do this on a graph of 30 nodes, with eigenvalue-gap target of 0.2. The

result is comparable to that of [16], however, in contrast to [16], we have chosen the asymptotic speed of convergence. In addition we have observed that UPSO on the "sparsification" problem with the objective function of Eq. (19) usually finds effective solutions with way less than $4\,|E|$ particles. In all cases, the swarm particles communicate with 4 neighbors in a cyclic manner.

With optimal SLEM values of 0.4286 and 0.25, the optimal eigenvalue-gaps of the graphs in Fig. 1 are 0.5714 and 0.75 respectively [2]. Our experiments indicate that UPSO can find the optimal, or near-optimal values. The results can be seen in Table 1.

For the considered sparsification problem, on a graph of 30 nodes, UPSO with objective function of Eq. (19) and $e_{\text{gap}}^{\text{tgt}} = 0.2$ produced a graph with 57 bidirectional edges. The number of particles used was 200, and the number of iterations was 400. The final eigenvalue-gap is exactly 0.2 (Fig. 2).



(a) Optimal eigenvalue-gap: 4/7. (b) Optimal eigenvalue-gap: 3/4.

Fig. 1. Two small networks with known optimal eigenvalue-gaps e^* and optimal edge weights given in [2].

Table 1. Maximal eigenvalue-gaps produced by UPSO for several unification parameter values u. The optimal eigenvalue gap of Graph (a) is 0.5714 and the optimal eigenvalue gap of Graph (b) is 0.5. Number of iterations: 400. Bold-face entries indicate optimal values.

u	Graph (a)	Graph (b)
0	0.5634	**0.7500**
0.25	**0.5714**	**0.7500**
0.5	0.5660	0.7489
0.75	**0.5714**	0.7492
1	**0.5714**	0.7498

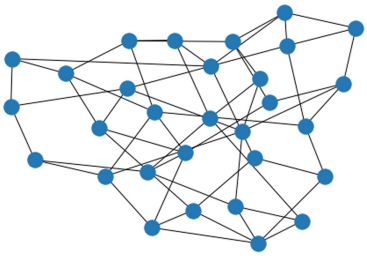


Fig. 2. Graph on 30 nodes with symmetric weight matrix. It contains 57 bidirectional edges and 5 non-zero diagonal elements.

In a future correspondence, we will also examine the case of asymmetric weight matrices. A relevant work can be found in [9].

5 Synopsis and Concluding Remarks

We tackled two difficult problems using unified particle swarm optimization (UPSO). Namely, (a) eigenvalue-gap maximization on doubly-stochastic matrices with sparsity constraints, and, (b) sparse doubly-stochastic matrix identification with given eigenvalue-gap target. The intricacy we faced is that the domains of these problems are hard to navigate. We get around this issue by using an iterative normalization procedure that maps the Euclidean space to the domain of the respective problem. UPSO seems to provide optimal or near optimal solutions to problem (a) with symmetric edge weights, and yields effective solutions to problem (b). In a future correspondence, we would like to study the properties of the iterative normalization scheme, and assess the ability of UPSO to find optimal eigenvalue-gaps for matrices with asymmetric edge weights, and compare with other evolutionary algorithms.

References

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.-U.: Complex networks: structure and dynamics. Phys. Rep. **424**(4–5), 175–308 (2006)
2. Boyd, S., Diaconis, P., Xiao, L.: Fastest mixing Markov chain on a graph. SIAM Rev. **46**(4), 667–689 (2004)
3. Chandrasekhar, A.G., Larreguy, H., Xandri, J.P.: Testing models of social learning on networks: evidence from two experiments. Econometrica **88**(1), 1–32 (2020)
4. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors. J. Parallel Distrib. Comput. **7**(2), 279–301 (1989)
5. DeGroot, M.H.: Reaching a consensus. J. Am. Stat. Assoc. **69**(345), 118–121 (1974)
6. DeMarzo, P.M., Vayanos, D., Zwiebel, J.: Persuasion bias, social influence, and unidimensional opinions. Q. J. Econ. **118**(3), 909–968 (2003)
7. Duchi, J.C., Agarwal, A., Wainwright, M.J.: Dual averaging for distributed optimization: convergence analysis and network scaling. IEEE Trans. Autom. Control **57**(3), 592–606 (2011)

8. Hadjicostis, C.N., Domínguez-García, A.D., Charalambous, T.: Distributed averaging and balancing in network systems. Now Foundations (2018)
9. Hao, H., Barooah, P.: Improving convergence rate of distributed consensus through asymmetric weights. In: 2012 American Control Conference (ACC), pp. 787–792. IEEE, (2012)
10. Jadbabaie, A., Lin, J., Morse, A.S.: Coordination of groups of mobile autonomous agents using nearest neighbor rules. IEEE Trans. Autom. Control **48**(6), 988–1001 (2003)
11. Jirdehi, M.A., Tabar, V.S., Ghassemzadeh, S., Tohidi, S.: Different aspects of microgrid management: a comprehensive review. J. Energ. Storage **30**, 101457 (2020)
12. Kandris, D., Nakas, C., Vomvas, D., Koulouras, G.: Applications of wireless sensor networks: an up-to-date survey. Appl. Syst. Innov. **3**(1), 14 (2020)
13. Khan, M.R.B., Jidin, R., Pasupuleti, J.: Multi-agent based distributed control architecture for microgrid energy management and optimization. Energ. Convers. Manag. **112**, 288–307 (2016)
14. Koponen, T., et al.: Onix: a distributed control platform for large-scale production networks. In: 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10), (2010)
15. Levin, D.A., Peres, Y.: Markov chains and mixing times. Am. Math. Soc. 107, (2017)
16. Lin, F., Fardad, M., Jovanović, M.R.: Identification of sparse communication graphs in consensus networks. In: 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 85–89. IEEE, (2012)
17. Montenegro, R., Tetali, P.: Mathematical aspects of mixing times in Markov chains. Foundations and Trends® in Theoretical Computer Science 1(3), 237–354 (2006). https://doi.org/10.1561/0400000003
18. Moreno, J.L., Jennings, H.H.: Statistics of social configurations. Sociometry, pp. 342–374 (1938)
19. Olfati-Saber, R.: Distributed Kalman filter with embedded consensus filters. In: Proceedings of the 44th IEEE Conference on Decision and Control, pp. 8179–8184. IEEE (2005)
20. Overton, M.L., Womersley, R.S.: On minimizing the special radius of a nonsymmetric matrix function: optimality conditions and duality theory. SIAM J. Matrix Anal. Appl. **9**(4), 473–498 (1988)
21. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization and intelligence: advances and applications. Information Science Publishing (IGI Global) (2010)
22. Peng, Z., Wang, J., Wang, D., Han, Q.-L.: An overview of recent advances in coordinated control of multiple autonomous surface vehicles. IEEE Trans. Ind. Inform. **17**(2), 732–745 (2020)
23. Saad, Y.: Numerical methods for large eigenvalue problems: revised edition. SIAM (2011)
24. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific J. Math. **21**(2), 343–348 (1967)
25. Thramboulidis, K., Perdikis, D., Kantas, S.: Model driven development of distributed control applications. Int. J. Adv. Manuf. Technol. **33**(3), 233–242 (2007)
26. Xiao, L., Boyd, S.: Fast linear iterations for distributed averaging. Syst. Control Lett. **53**(1), 65–78 (2004)