Automatica 48 (2012) 15–24

Contents lists available at SciVerse ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Hearing the clusters of a graph: A distributed algorithm☆

Tuhin Sahai [1], Alberto Speranzon, Andrzej Banaszuk

United Technologies Research Center, East Hartford, CT 06108, USA

ARTICLE INFO

Article history:
Received 20 June 2010
Received in revised form
15 April 2011
Accepted 9 June 2011
Available online 19 October 2011

Keywords:
Decentralized systems
Graph theory
Decomposition methods

ABSTRACT

We propose a novel distributed algorithm to cluster graphs. The algorithm recovers the solution obtained from spectral clustering without the need for expensive eigenvalue/eigenvector computations. We prove that, by propagating waves through the graph, a local fast Fourier transform yields the local component of every eigenvector of the Laplacian matrix, thus providing clustering information. For large graphs, the proposed algorithm is orders of magnitude faster than random walk based approaches. We prove the equivalence of the proposed algorithm to spectral clustering and derive convergence rates. We demonstrate the benefit of using this decentralized clustering algorithm for community detection in social graphs, accelerating distributed estimation in sensor networks and efficient computation of distributed multi-agent search strategies.

1. Introduction

In recent years, there has been great interest in the analysis of large interconnected systems, such as sensors networks, social networks, the Internet, biochemical networks, power networks, etc. These systems are characterized by complex behavior that arises due to interacting subsystems. Graph theoretic methods have recently been applied and extended to study these systems. In particular, spectral properties of the Laplacian matrix associated with such graphs provide useful information for the analysis and design of interconnected systems. The computation of eigenvectors of the graph Laplacian is the cornerstone of spectral graph theory (Chung, 1997; von Luxburg, 2007), and it is well known that the sign of the second (and successive) eigenvectors can be used to cluster graphs (Fiedler, 1973, 1975).

The problem of graph (or data, in general) clustering arises naturally in applications ranging from social anthropology (Kottak, 1991), gene networks (Speer, Fröhlich, Spieth, & Zell, 2005), protein sequences (Paccanaro, Casbon, & Saqi, 2006), sensor networks (Akyildiz, Su, Sankarasubramaniam, & Cayirci, 2002; Ghiasi, Srivastava, Yang, & Sarrafzadeh, 2002; Muhammad & Jadbabaie, 2007), computer graphics (Herman, Melançon, & Marshall, 2000) and Internet routing algorithms (Kempe & McSherry, 2008).

The basic idea behind graph decomposition is to cluster nodes into groups with strong intra-connections but weak inter-connections. If one poses the clustering problem as a minimization of the inter-connection strength (sum of edge weights between clusters), it can be solved exactly and quickly (Stoer & Wagner, 1997). However, the decomposition obtained is often unbalanced (some clusters are large and others small) (von Luxburg, 2007). To avoid unbalanced cuts, size restrictions are typically placed on the clusters, i.e., instead of minimizing inter-connection strength, we minimize the ratio of the inter-connection strength to the size of individual clusters. This, however, makes the problem NP-complete (Wagner & Wagner, 1993). Several heuristics to partition graphs have been developed over the past few decades (Porter, Onnela, & Mucha, 2009) including the Kernighan–Lin algorithm (Kernighan & Lin, 1970), Potts method (Reichardt & Burnholdt, 2004), percolation based methods (Palla, Derényi, Farkas, & Vicsek, 2005), horizontal–vertical decomposition (Varigonda, Kalmar-Nagy, Labarre, & Mezic, 2004) and spectral clustering (Fiedler, 1973, 1975).

1.1. Spectral clustering

Spectral clustering has emerged as a powerful tool of choice for graph decomposition purposes (see von Luxburg, 2007 and references therein). The method assigns nodes to clusters based on the signs of the elements of the eigenvectors of the Laplacian corresponding to increasing eigenvalues (Chung, 1997; Fiedler, 1973, 1975). In Spielman and Teng (2004), the authors have developed a distributed algorithm for spectral clustering of graphs. The algorithm involves performing random walks, and at every step neglecting probabilities below a threshold value. The nodes are then ordered by the ratio of probabilities to node degree and

☆ The material in this paper was partially presented at the 49th IEEE Conference on Decision and Control, December 15–17, 2010, Atlanta, Georgia, USA. This paper was recommended for publication in revised form by Associate Editor Andrea Serrani under the direction of Editor Miroslav Krstic.

E-mail addresses: SahaiT@utrc.utc.com (T. Sahai), SperanA@utrc.utc.com (A. Speranzon), BansaszA@utrc.utc.com (A. Banaszuk).

[1] Tel.: +1 860 610 7763; fax: +1 860 622 0666.

0005-1098/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2011.09.019

grouped into clusters. Since this algorithm is based on random walks, it suffers, in general, from slow convergence.

Since the clustering assignment is computed using the eigenvectors/eigenvalues of the Laplacian matrix, one can use standard matrix algorithms for such computation (Golub & Loan, 1996). However, as the size of the matrix (and thus the corresponding network) increases, the execution of these standard algorithms becomes infeasible on monolithic computing devices. To address this issue, algorithms for distributed eigenvector computations have been proposed (Kempe & McSherry, 2008). These algorithms, however, are also (like the algorithm in Spielman & Teng, 2004) based on the slow process of performing random walks on graphs.

1.2. Wave equation method

In a theme similar to Mark Kac's question "Can one hear the shape of a drum?" (Kac, 1966), we demonstrate that by evolving the wave equation in the graph, nodes can "hear" the eigenvectors of the graph Laplacian using only local information. Moreover, we demonstrate, both theoretically and on examples, that the wave equation based algorithm is orders of magnitude faster than random walk based approaches for graphs with large mixing times. The overall idea of the wave equation based approach is to simulate, in a distributed fashion, the propagation of a wave through the graph and capture the frequencies at which the graph "resonates". In this paper, we show that by using these frequencies one can compute the eigenvectors of the Laplacian, thus clustering the graph. We also derive conditions that the wave must satisfy in order to cluster graphs using the proposed method.

The paper is organized as follows: in Section 2 we describe current methodologies for distributed eigenvector/clustering computation based on the heat equation. In Section 3 the new proposed wave equation method is presented. In Section 4 we determine bounds on the convergence time of the wave equation. In Section 5 we show some numerical clustering results for a few graphs, including a large social network comprising of thousands of nodes and edges. We then show, in Section 6, how the wave equation can be used to accelerate distributed estimation in a large-scale environment such as a building. In Section 7 we show how the proposed distributed clustering algorithm enables one to efficiently transform a centralized search algorithm into a decentralized one. Finally, conclusions are drawn in Section 8.

2. From heat to wave equation: related work

Let $\mathcal{G} = (V, E)$ be a graph with vertex set $V = \{1, \ldots, N\}$ and edge set $E \subseteq V \times V$, where a weight $\mathbf{W}_{ij} \geq 0$ is associated with each edge $(i, j) \in E$, and $\mathbf{W}$ is the $N \times N$ weighted adjacency matrix of $\mathcal{G}$. We assume that $\mathbf{W}_{ij} = 0$ if and only if $(i, j) \notin E$. The (normalized) graph Laplacian is defined as,

$$\mathbf{L}_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\mathbf{W}_{ij} \Big/ \sum_{\ell=1}^{N} \mathbf{W}_{i\ell} & \text{if } (i, j) \in E \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

or equivalently, $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$ where $\mathbf{D}$ is the diagonal matrix with the row sums of $\mathbf{W}$.

Note that in this work we only consider undirected graphs. The smallest eigenvalue of the Laplacian matrix is $\lambda_1 = 0$, with an associated eigenvector $\mathbf{v}^{(1)} = \mathbf{1} = [1, 1, \ldots, 1]^T$. Eigenvalues of $\mathbf{L}$ can be ordered as, $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \cdots \leq \lambda_N$ with associated eigenvectors $\mathbf{1}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}, \ldots, \mathbf{v}^{(N)}$ (von Luxburg, 2007). It is well known that the multiplicity of λ_1 is the number of connected components in the graph (Mohar, 1991). We assume in the following that $\lambda_1 < \lambda_2$ (the graph does not have disconnected clusters). We also assume that there exist unique cuts that divide the graph into k clusters. In other words, we assume that there exist k distinct eigenvalues close to zero (Luxburg, Bousquet, & Belkin, 2004).

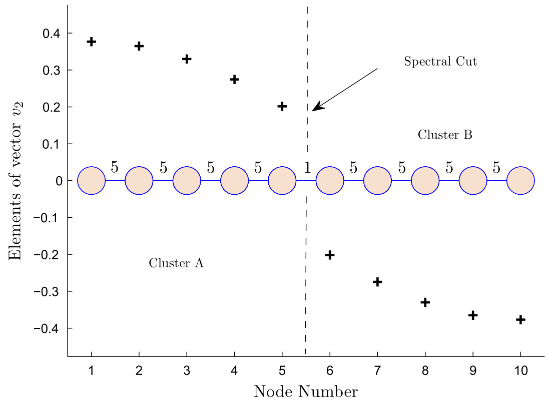

Fig. 1. Spectral clustering: The sign of the i-th element of eigenvector v_2 determines the cluster assignment of the i-th vertex, demonstrated on a simple line graph example (shown in the center). With $+$ we plot the value of the components of v_2.

Given the Laplacian matrix $\mathbf{L}$, associated with a graph $\mathcal{G} = (V, E)$, spectral clustering divides $\mathcal{G}$ into two clusters by computing the sign of the N elements of the second eigenvector $\mathbf{v}^{(2)}$, or Fiedler vector (Fiedler, 1975; von Luxburg, 2007). This process is depicted in Fig. 1 for a line graph where one edge (the edge (5, 6)) has lower weight than other edges.

More than two clusters can be computed from signs of the elements of higher eigenvectors, i.e. $\mathbf{v}^{(3)}$, $\mathbf{v}^{(4)}$, etc. von Luxburg (2007). Alternatively, once the graph is divided into two clusters, the spectral clustering algorithm can be run independently on both clusters to compute further clusters. This process is repeated until either a desired number of clusters is found or no further clusters can be computed. This method can also be used to compute a hierarchy of clusters.

There are many algorithms to compute eigenvectors, such as the Lanczos method or orthogonal iteration (Golub & Loan, 1996). Although some of these methods are distributable, convergence is slow (Golub & Loan, 1996) and the algorithms do not consider/take advantage of the fact that the matrix for which the eigenvalues and eigenvectors need to be computed is the adjacency matrix of the underlying graph. In Kempe and McSherry (2008), the authors propose an algorithm to compute the first k largest eigenvectors (associated with the first k eigenvalues with greatest absolute value)[2] of a symmetric matrix. The algorithm in Kempe and McSherry (2008) emulates the behavior of orthogonal iteration. To compute the first k eigenvectors of a given matrix $\mathbf{J}$, at each node in the network, matrix $\mathbf{V}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{J}_{ij}\mathbf{Q}_j$ is computed, where $\mathbf{Q}_j \in \mathbb{R}^{N \times k}$ is initialized to a random matrix and $\mathcal{N}(i)$ is the set of neighbors of node i (including node i itself). Orthonormalization is achieved by the computation of matrix $\mathbf{K}_i = \mathbf{V}_i^T\mathbf{V}_i$ at every node, followed by computation of matrix $\mathbf{K}$, which is the sum of all the $\mathbf{K}_i$ matrices in the network. Once matrix $\mathbf{K}$ is computed, $\mathbf{Q}_i = \mathbf{V}_i\mathbf{R}^{-1}$ is updated at each node, where $\mathbf{R}$ is a unique matrix such that $\mathbf{K} = \mathbf{R}^T\mathbf{R}$ (Cholesky decomposition). The above iteration is repeated until $\mathbf{Q}_i$ converges to the i-th eigenvector. The sum of all the matrices $\mathbf{K}_i$ is done in a decentralized way, using gossip (Shah, 2009), which is a deterministic simulation of a random walk on the network. In particular, at each node one computes the matrix $\mathbf{K}$ as

[2] Note that in the case of spectral clustering we desire to compute the smallest k eigenvectors of $\mathbf{L}$. The algorithm is still applicable if we consider the matrix $\mathbf{I} - \mathbf{L}$.