Discrete Applied Mathematics 161 (2013) 1699–1706

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

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

Bisimplicial edges in bipartite graphs

Matthijs Bomhoff*, Bodo Manthey

Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 16 September 2010
Received in revised form 9 December 2010
Accepted 3 March 2011
Available online 31 March 2011

Keywords:
Bipartite graphs
Random graphs
Algorithms
Gaussian elimination

ABSTRACT

Bisimplicial edges in bipartite graphs are closely related to pivots in Gaussian elimination that avoid turning zeroes into non-zeroes. We present a new deterministic algorithm to find such edges in bipartite graphs. Our algorithm is very simple and easy to implement. Its running-time is $O(nm)$, where n is the number of vertices and m is the number of edges. Furthermore, for any fixed p and random bipartite graphs in the $G_{n,n,p}$ model, the expected running-time of our algorithm is $O\left(n^2\right)$, which is linear in the input size.

1. Introduction

When applying Gaussian elimination to a square $n \times n$ matrix M containing some elements with value zero, the choice of pivots can often determine the amount of zeroes turned into non-zeroes during the process. This is called the *fill-in*. Some matrices even allow Gaussian elimination without any fill-in. Avoiding fill-in has the nice property of bounding the required space for intermediate results of the Gaussian elimination to the space required for storing the input matrix M. This is often important for processing very large sparse matrices. Even when fill-in cannot be completely avoided, it is still worthwhile to avoid it for several iterations, motivating the search for pivots that avoid fill-in.

If we assume subtracting a multiple of one row of M from another turns at most one non-zero into a zero, we can represent the relevant structure of our problem using only $\{0, 1\}$ matrices. (This assumption is quite natural, as it holds with probability one for a random real-valued matrix.) Given such a square matrix M, we can construct the bipartite graph $G[M]$ with vertices corresponding to the rows and columns in M, where the vertex corresponding to row i and the one corresponding to column j are adjacent if and only if $M_{i,j}$ is non-zero. We denote the number of non-zero elements of M by m. Furthermore, we assume M has no rows or columns containing only zeroes, so the associated bipartite graph has no isolated vertices and $n \leq m \leq n^2$. Fig. 1 shows an example.

The $\{0, 1\}$ matrices that allow Gaussian elimination without fill-in correspond to the class of *perfect elimination bipartite* graphs [3]. Central to the recognition of this class of graphs is the notion of a *bisimplicial* edge: a bisimplicial edge corresponds to an element of M that can be used as a pivot without causing fill-in. The fastest known algorithm for finding bisimplicial edges has a running-time of $O(nm)$ for sparse instances and $O(n^{\omega})$ in general [2,6], where $\omega \leq 2.376$ is the matrix multiplication exponent [1]. However, fast matrix multiplication using the algorithm of Coppersmith and Winograd [1] has huge hidden constants, which makes it impractical for applications.

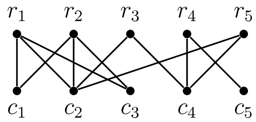
We present a new deterministic algorithm for finding all bisimplicial edges in a bipartite graph. Our algorithm is very fast in practice, and it can be implemented easily. Its running-time is $O(nm)$. In addition, we analyze its expected running-time on random bipartite graphs. For this, we use the $G_{n,n,p}$ model. This model consists of bipartite graphs with n vertices in each

* Corresponding author. Fax: +31 53 4894858.
E-mail addresses: m.j.bomhoff@utwente.nl (M. Bomhoff), b.manthey@utwente.nl (B. Manthey).

0166-218X/$ – see front matter
doi:10.1016/j.dam.2011.03.004

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

a

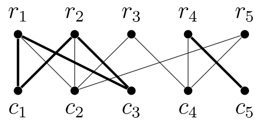


b

Fig. 1. An example of a {0, 1}-matrix M and its bipartite graph $G[M]$.

$$\begin{pmatrix} \boxed{1} & 1 & \boxed{1} & 0 & 0 \\ \boxed{1} & 1 & \boxed{1} & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & \boxed{1} \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

a



b

Fig. 2. Bisimplicial edges in M and its bipartite graph $G[M]$ (bisimplicial edges are bold, the corresponding matrix entries are dashed).

vertex class, where edges are drawn independently, and each possible edge is present with a probability of p. We show that the expected running-time of our algorithm on $G_{n,n,p}$ graphs for fixed $p \in (0, 1)$ is $O\left(n^2\right)$, which is linear in the input size. (The input size of a random $G_{n,n,p}$ graph is $\Theta(n^2)$ with high probability.)

2. Bisimplicial edges

We denote by $\Gamma(u)$ the neighbors of a vertex u and by $\delta(u)$ its degree.

Definition 2.1. An edge (u, v) of a bipartite graph $G = (U, V, E)$ is called *bisimplicial*, if the induced subgraph $G[\Gamma(u) \cup \Gamma(v)]$ is a complete bipartite graph.

Clearly, we can determine in $O(m)$ time if an edge (u, v) is bisimplicial: we simply have to check all edges adjacent to it. So a simple algorithm to find a bisimplicial edge in a bipartite graph G, if one exists, takes $O\left(m^2\right)$ time. The bisimplicial edges in our example matrix M and associated graph $G[M]$ are shown in Fig. 2. As mentioned above, Goh and Rotem [2] have presented a faster algorithm based on matrix multiplication that can be implemented in either $O(n^\omega)$ or $O(nm)$.

We present a different approach that first selects a set of candidate edges. The candidate edges are not necessarily bisimplicial and not all bisimplicial edges are marked as candidates. However, knowing which candidates, if any, are bisimplicial allows us to quickly find all other bisimplicial edges as well. By bounding the number of candidates, we achieve an improved expected running-time. The following observation is the basis of our candidate selection procedure.

Lemma 2.2. *If an edge (u, v) of a bipartite graph $G = (U, V, E)$ is bisimplicial, we must have $\delta(u) = \min_{u' \in \Gamma(v)} \delta\left(u'\right)$ and $\delta(v) = \min_{v' \in \Gamma(u)} \delta\left(v'\right)$.*

Proof. Let $(u, v) \in E$ be a bisimplicial edge, and let $A = G[\Gamma(u) \cup \Gamma(v)]$ be the complete bipartite graph it induces. Now assume that there is a vertex $u' \in U_A$ with $\delta\left(u'\right) < \delta(u)$. Then there must be a $v' \in V_A$ with $u'v' \notin E_A$. But this would mean A is not a complete bipartite graph, leading to a contradiction. □

Translated to the matrix M, this means that if $M_{i,j} = 1$, it can only correspond to a bisimplicial edge if row i has a minimal number of 1s over all the rows that have a 1 in column j and column j has a minimal number of 1s over all the columns having a 1 in row i. In what follows, we will call the row (column) in M with the minimal number of 1s over all the rows (columns) in M the *smallest* row (column). Using this observation, we construct an algorithm to pick candidate edges that may be bisimplicial.

Algorithm 2.3. Perform the following steps:

1. Determine the row and column sums for each row i and column j of M.
2. Determine for each row i the index c_i of the smallest column among those with $M_{i,c_i} = 1$ (breaking ties by favoring the lowest index); or let $c_i = 0$ if row i has no 1.