Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 99 (2006) 54–58
Information Processing Letters
www.elsevier.com/locate/ipl

Approximate maximum weight branchings

Amitabha Bagchi [a,*], Ankur Bhargava [b], Torsten Suel [c]

[a] *Department of Computer Science and Engineering, Indian Institute of Technology, Hauz Khas, New Delhi 110016, India*
[b] *Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA*
[c] *Department of Computer and Information Science, Polytechnic University, 6 Metrotech Center, Brooklyn, NY 11201, USA*

Received 22 September 2005; received in revised form 13 February 2006; accepted 20 February 2006

Available online 20 March 2006

Communicated by K. Iwama

Abstract

We consider a special subgraph of a weighted directed graph: one comprising only the k heaviest edges incoming to each vertex. We show that the maximum weight branching in this subgraph closely approximates the maximum weight branching in the original graph. Specifically, it is within a factor of $k/(k+1)$. Our interest in finding branchings in this subgraph is motivated by a data compression application in which calculating edge weights is expensive but estimating which are the heaviest k incoming edges is easy. An additional benefit is that since algorithms for finding branchings run in time linear in the number of edges our results imply faster algorithms although we sacrifice optimality by a small factor. We also extend our results to the case of edge-disjoint branchings of maximum weight and to maximum weight spanning forests.

Keywords: Analysis of algorithms; Graph algorithms

1. Introduction

Given a graph $G = (V, E)$, (V, B) is a *branching* if B is a subset of E such that each vertex in (V, B) has in-degree at most one and there are no cycles. Branchings are basic graph structures which have found applications in various fields of computer science. Motivated by a data compression problem [12] we prove the following general theorem about weighted branchings:

Define G_k to be a subgraph of a directed graph G where each node only retains its k heaviest incoming edges. If $w(G_k)$ is the weight of a maximum weight branching on G_k and $w(G)$ is the weight of a maximum weight branching on the entire graph G, then

$$\frac{w(G_k)}{w(G)} \geqslant 1 - \frac{1}{k+1}.$$

Thus, we can compute a branching with weight almost as large as the maximum possible weight on a dense graph by only considering a few incoming edges for each vertex. Since algorithms for computing maximum weight branchings [15,3] depend at least linearly on the number of edges in the graph, this implies faster algorithms for approximate maximum weight branching after appropriate preprocessing of the dense graph. More importantly, in many scenarios that can be modeled as graph problems, the main cost is in computing appropriate edge weights for the input graph rather than in the actual graph computation; our result implies that

* Corresponding author.
E-mail addresses: bagchi@cse.iitd.ernet.in (A. Bagchi), ankur@cs.jhu.edu (A. Bhargava), suel@poly.edu (T. Suel).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2006.02.011

for branching problems it suffices to exactly compute only the weights of the heaviest edges in the reduction.

1.1. A simple application of maximum weight branching

Consider the following simple application in the context of data compression [12], which provided the initial motivation for our work. We are interested in compressing a collection of files where there is a significant degree of similarity (or redundancy) between many of the files. For example, web pages from the same site frequently share certain elements in their page layout and menu structure, and may also contain some similar and repeated content. This inter-file redundancy can be exploited to achieve better overall compression of the collection.

The process of compressing one file (the *target file*) with respect to another file (the *reference file*) is called *delta compression* or *differential compression* [7,9,17, 10]. But given a collection of n files, in what order should we apply delta compression between pairs of files to minimize the overall size? There is an exponential number of possible orderings of the pairwise compression steps. Obviously, it is beneficial to compress each file with respect to another very similar file. However, we have to avoid cycles, such as A being compressed with respect to B and B being compressed with respect to A, since it would be impossible to uncompress the resulting data.

This scenario [12], as well as related problems in the compression of web graphs [1] and multispectral images [16], can be modeled as a maximum weight branching problem on a directed graph. Finding an optimal set of delta compression steps is equivalent to finding a maximum weight branching on a complete directed graph with one node v_A for each file A in the collection, where edge (v_A, v_B) has a weight equal to the savings in bits obtained by delta compressing B with respect to A instead of compressing B by itself [12].

One problem with this reduction is that the resulting graph has n^2 edges, slowing down the maximum weight branching computation. However, in practice a much more significant challenge is the computation of the edge weights: The only way to determine the precise weight of an edge is to actually run the delta compressor on the two files involved. Of course, the final branching usually contains mostly fairly heavy edges, and thus it would be highly desirable to compute for each node only the weights of these most promising edges, instead of materializing the entire graph. Fortunately, in our scenario efficient techniques are known for estimating the similarities among pairs of files and for finding for each file the k most similar other files (or k-nearest neighbors) under various definitions of file similarity [11,2,6,8].

Thus, a promising approach would be to compute only the weights of the incoming edges from the k most similar files for each node, and then compute the maximum weight branching on this subgraph. But can we show that this results in a compression scheme that is almost as good as using the complete graph? There are two issues here. First, the techniques for finding k-nearest neighbors in [11,2,6,8] assume certain formal similarity measures between files that do not precisely model the benefit obtained by an actual delta compressor that uses a combination of various state-of-the-art compression techniques to minimize size. However, this issue is unlikely to be completely resolved, and in practice the known techniques seem to be able to identify promising references files for each file. The second question is, assuming that we have correctly identified for each file the k best references files, if we run a branching computation on this subgraph are we guaranteed to get a compression scheme whose benefit approximates that of a scheme based on the complete graph?

This question lead us to the following very simple and natural conjecture about maximum weight branchings: If $w(G_k)$ is the weight of a maximum weight branching on a subgraph of G where each node only retains its k heaviest incoming edges, and $w(G)$ the weight of a maximum weight branching on the entire graph, we conjecture that $w(G_k)/w(G) \geqslant 1 - 1/(k+1)$.

In this paper we settle the above conjecture in the affirmative. We also show that this result can be extended in a natural way to c edge disjoint branchings [4] of maximum total weight, and to maximum weight spanning forests in undirected graphs.

2. Maximum weight branchings

We consider a directed graph $G = (V, E)$ with an edge weight function $w : E \to \mathbb{R}^+$. A *branching* (V, B) is a subgraph of G with an edge set $B \subseteq E$ such that (V, B) is acyclic and the in-degree of any vertex of (V, B) is at most 1. Note that in general, a branching forms a forest of rooted directed trees. The weight of a branching B is $w(B) = \sum_{e \in B} w(e)$. A *maximum weight branching* is a branching with weight at least that of any other branching.

We define the *k-heavy subgraph* of G, denoted G_k, as the subgraph that contains only the k heaviest edges incoming to each vertex. If the in-degree of a vertex is