Dynamic Approximate Vertex Cover and Maximum Matching

Krzysztof Onak*
MIT
konak@mit.edu

Ronitt Rubinfeld†
MIT, Tel Aviv University
ronitt@csail.mit.edu

February 15, 2010

Abstract

We consider the problem of maintaining a large matching or a small vertex cover in a dynamically changing graph. Each update to the graph is either an edge deletion or an edge insertion. We give the first data structure that simultaneously achieves a constant approximation factor and handles a sequence of k updates in $k \cdot \text{polylog}(n)$ time. Previous data structures require a polynomial amount of computation per update.

The starting point of our construction is a distributed algorithm of Parnas and Ron (Theor. Comput. Sci. 2007), which they designed for their sublinear-time approximation algorithm for the vertex cover size. This leads us to wonder whether there are other connections between sublinear algorithms and dynamic data structures.

1 Introduction

Suppose one is given the task of solving a combinatorial problem, such as maximum matching or minimum vertex cover, for a very large and constantly changing graph. In this setting, it is natural to ask, does one need to recompute the solution from scratch after every update?

Such questions have been asked before for various combinatorial quantities—examples include minimum spanning tree, shortest path length, min-cut, and many others (some examples include [EGIN97, EGI97, HK99, Tho05, KS98, Tho01]). Classic works for these problems have shown update times that are sublinear in the input size. For the problem of maximum matching, Sankowski [San07] shows that it can be maintained with $O(n^{1.495})$ computation per update, which for dense graphs is sublinear in the number of edges.

For very large graphs, it may be crucial to maintain the maximum matching with much faster, even polylogarithmic, update time. Note that this might be hard for maximum matching, since obtaining $o(\sqrt{n})$ update time, even in the case when only insertions are allowed, would improve on the 30-year-old algorithm of running time $O(m\sqrt{n})$ due to Micali and Vazirani [MV80], where m is the number of edges in the graph. Therefore, some kind of approximation may be unavoidable. Following similar considerations, Ivković and Lloyd [IL93] give a factor-2 approximation to both vertex cover and maximum matching, by computing a *maximal* matching (which is well known to give the desired approximation for maximum matching and also minimum vertex cover). Their update time is nevertheless still polynomial in n.

*Supported by NSF grants 0732334 and 0728645.

†Supported by NSF grants 0732334 and 0728645, Marie Curie Reintegration grant PIRG03-GA-2008-231077, and the Israel Science Foundation grant nos. 1147/09 and 1675/09.

In this paper, we concentrate on the setting in which slightly weaker, but still $O(1)$ approximation factors are acceptable, but in which it is crucial that update times be extremely fast, in particular, polylogarithmic in the number of vertices.

Interestingly, our data structure uses a technique that Parnas and Ron [PR07] designed for their sublinear-time algorithm as a starting point. We think that it is an interesting direction to explore possible connections between sublinear-time algorithms and dynamic data structures.

2 Problem Statement and Our Results

Recall that in the *maximum matching* problem, one wants to find the largest subset of vertex disjoint edges. In the *vertex cover* problem, one wants to find the smallest set of vertices such that each edge of the graph is incident to at least one vertex in the set.

Our goal here is to design a data structure that handles edge removals and edge insertions. The data structure maintains a list of edges that constitute a large matching or a list of vertices that constitute a small vertex cover. We assume that we always start with an empty graph, and n is known in advance.

The main result of the paper is the following:

> *There is a randomized data structure for maximum matching and vertex cover that*
>
> *(a) achieves a constant approximation factor,*
>
> *(b) with probability* $5/6$*, runs in* $O(k \log^3 n \log\log n)$ *time for a fixed sequence of* k *updates.*

Furthermore, the first step in our presentation is a *deterministic* data structure for vertex cover. The data structure keeps a vertex cover that gives $O(\log n)$ approximation to the minimum vertex cover. The amortized update time of the data structure is $O(\log^2 n)$. Though the approximation factor achieved by this algorithm is relatively weak, the algorithm may be of independent interest because of its relative simplicity and efficient update time.

3 Overview of Our Techniques

We construct our data structure in two stages. We first show a deterministic $O(\log n)$-approximation data structure for vertex cover. Then we modify it, introducing randomization, to achieve a constant approximation factor for both vertex cover and maximum matching.

A Deterministic $O(\log n)$-Approximation Data Structure. We construct a data structure that makes use of a carefully designed partition of vertices into a logarithmic number of subsets. The partition is inspired by a simple distributed algorithm of Parnas and Ron [PR07]. In [PR07], the first subset in the partition corresponds to removing vertices of degree approximately greater than n. The second subset corresponds to removing vertices of degree approximately greater than $n/4$ from the modified graph. In general, the i-th subset is a set of vertices that are approximately greater than $n/4^{i-1}$ in the graph with all previous subsets of vertices removed. Finally, after a logarithmic number of steps, the remaining graph has no edges. This implies that the union of all subsets removed so far constitutes a vertex cover. For each of the removed subsets, it is easy to show that the subset size is bounded by $O(\mathrm{VC}(G))$, where $\mathrm{VC}(G)$ is the size of the minimum vertex cover. Hence the total vertex cover is bounded by $O(\mathrm{VC}(G) \cdot \log n)$.

The main idea behind our data structure is to modify the partition of Parnas and Ron in order to allow efficient maintenance of this partition. While this is not possible in the partition of Parnas and Ron, it is possible in our relaxed version of it. As edges are inserted and removed, we want to move vertices between subsets. In order to determine whether to move a vertex, we associate a potential function with every vertex, and we allow a vertex to jump from one set to another only if it has collected enough potential. To do this, we set two thresholds $\tau_1 < \tau_2$ for each subset. A vertex can move into the subset from a subset corresponding to a lower degree if its number of neighbors in a specific graph is at least τ_2. Then the vertex can move back to a subset corresponding to a lower degree only if its number of edges decreases to τ_1 in the same graph. A slight technical difficulty is presented by the fact that moving vertices may increase the potential of other vertices. We overcome this obstacle by carefully selecting constants in the potential function so that the potential of the vertex that moves is spent on increasing the potential of its neighbors whenever needed.

A Randomized $O(1)$-Approximation Data Structure. In this case, we redesign the partition, building upon the previous one. In the process of defining the partition, whenever we remove a large subset W of vertices of degree approximately greater than $n/4^i$, we also show the existence of a matching M which is smaller than W by at most a constant factor. To build the next set of the partition, we not only remove W but also all vertices matched in M. In this way we achieve a matching and a vertex cover of sizes that are within a constant factor of each other. Therefore, both give a constant factor approximation to their respective optimal solutions.

Efficient maintenance of the new partition is more involved, as we are sometimes forced to recompute a new matching. This can happen, for instance, when many edges in the old matchings are deleted from the graph. Unfortunately, the creation of the new matching is expensive, since we have modify the set of the vertices matched in M that are deleted together with W. If the edges in the matching are deleted too quickly, we would have to create a new matching often, in which case we do not know how to maintain small update time. Fortunately, by picking a *random matching*, we can ensure that it is unlikely that many edges from the matching get deleted in a short span of time. Thus, by the time the matching gets deleted, we are likely to have collected enough potential to pay for the creation of a new matching.

4 Other Related Work

A sequence of papers [FKM$^+$05, McG05, Zel08] considers computing a large matching or a large weight matching (in the weighted case) in the semi-streaming model. The stream is a sequence of edges, and the goal of an algorithm is to compute a large matching in a small number of passes over the stream, using $\tilde{O}(n)$ space, and preferably at most polylog(n) update time. Results in this model correspond to results for dynamically changing graphs in which only edge insertions occur, except that the matching is only output once at the end of the processing. To the best of our knowledge, it is not known how to achieve a better approximation factor than 2 in one pass in $\tilde{O}(n)$ space for the maximum matching problem.

Lotker, Patt-Shamir, and Rosén [LPR07] show how to maintain a large matching in a distributed network.

References

[EGI97] David Eppstein, Zvi Galil, and Giuseppe F. Italiano. *Dynamic graph algorithms*. CRC Press, 1997.

[EGIN97] David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997.

[FKM+05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

[HK99] Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *J. ACM*, 46(4):502–516, 1999.

[IL93] Zoran Ivković and Errol L. Lloyd. Fully dynamic maintenance of vertex cover. In *WG*, pages 99–111, 1993.

[KS98] Philip N. Klein and Sairam Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998.

[LPR07] Zvi Lotker, Boaz Patt-Shamir, and Adi Rosén. Distributed approximate matching. In *PODC*, pages 167–174, 2007.

[McG05] Andrew McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.

[MV80] Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

[PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007.

[San07] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *SODA*, pages 118–126, 2007.

[Tho01] Mikkel Thorup. Fully-dynamic min-cut. In *STOC*, pages 224–230, 2001.

[Tho05] Mikkel Thorup. Worst-case update times for fully-dynamic all-pairs shortest paths. In *STOC*, pages 112–119, 2005.

[Zel08] Mariano Zelke. Weighted matching in the semi-streaming model. In *STACS*, pages 669–680, 2008.