Fast Approximation Algorithms for Bipartite Vertex-Weighted Matching

Florin Dobrian  Mahantesh Halappanavar  Alex Pothen

We consider the maximum vertex-weight matching problem. Let \( G = (V, E) \) be an undirected graph, where \( V \) is the set of vertices and \( E \) is the set of edges (we assume that there are no isolated vertices and therefore \( |E| = \Omega(|V|) \)). A matching \( M \) represents a subset of independent edges (\( M \subseteq E \)), i.e., no two edges in \( M \) share a vertex. An edge that belongs to \( M \) is called a matching edge and the two vertices that represent its endpoints are said to be matched by it. If \( w : V \rightarrow \mathbb{R}^+ \) is a weight function (real nonnegative weights are assigned to vertices) then the weight of a matching \( M \) is the sum of the weights of the vertices matched by the edges that belong to \( M \). The problem is to find a matching of maximum weight in \( G \).

In this talk we present approximation algorithms for solving the maximum vertex-weight problem for the particular case of bipartite graphs, i.e., \( G = (V, E) \) is bipartite if \( V = S \cup T \), \( S \cap T = \emptyset \) and every edge in \( E \) has one endpoint in \( S \) and the other endpoint in \( T \).

The maximum vertex-weight matching problem occurs in applications such as constructing a basis for a sparse, underdetermined, full rank matrix. If we are required to find a basis that is structurally as sparse as possible then we can employ a bipartite graph model in which every vertex in \( S \) and \( T \) corresponds to a matrix column and row, respectively, and edges correspond to nonzero matrix entries. In addition, weights are assigned only to the vertices in \( S \) such that every weight represents the number of zero entries in the corresponding matrix column. It is then easy to see that the computation of a structurally sparsest basis can be modeled as the computation of a maximum vertex-weight matching in a bipartite graph (actually a restricted version of it because weights are assigned only to the vertices in \( S \)).

We discuss our work in a general matching context in order to relate the maximum vertex-weight matching problem to other matching problems, comparing them from the algorithmic perspective. The audience may be more familiar with two related problems: maximum cardinality matching and maximum edge-weight matching, the latter commonly known as maximum weight matching, but renamed here in order to distinguish it from its vertex-weighted counterpart. For the maximum cardinality matching problem we are given an undirected graph \( G = (V, E) \) without any weight function and we are required to find a matching of maximum cardinality in \( G \). For the maximum edge-weight matching problem we are given the undirected graph \( G = (V, E) \) and a weight function \( w : E \rightarrow \mathbb{R}^+ \) (weights are assigned to edges, not to vertices), and we are required to find a matching of maximum weight in \( G \), this time the weight of a matching \( M \) representing the sum of the weights of the edges that belong to \( M \). Obviously, the maximum cardinality matching problem is just a particular case of the maximum edge-weight matching problem, where all edges are assigned the same weight and therefore the weight function can actually be dropped.

As a particular case, the maximum cardinality matching problem is easier to solve than the more general maximum edge-weight matching problem. The best known time complexity of algorithms (we usually consider algorithms based on the technique of augmentation) that solve the former is \( O(\sqrt{|V||E|}) \) while algorithms that solve the latter have a best known time complexity of \( O(|V||E| + |V|^2 \log(|V|)) \), although algorithms of time complexity \( O(|V||E| \log(|V|)) \) tend to be more practical.

The maximum vertex-weight matching problem can be related to the other two matching problems. It should not be difficult to see that the maximum vertex-weight matching problem can be regarded as a particular case of the maximum edge-weight matching problem. We can reduce the former to the latter by turning the vertex weight function into an edge weight function. The weight of every edge would simply be the sum of the weights of its endpoints. A reduction from the maximum edge-weight matching problem to the maximum vertex-weight matching problem is, however, not possible in general. That would require decomposing the weight of every edge into two terms and assigning the two terms to the endpoints of the edge and, because in a graph the number of edges is usually larger than the number of vertices, assigning weights to vertices
would translate into solving an overdetermined linear system, which generally does not have a solution.

We are therefore inclined to think that the maximum vertex-weight matching problem, as a particular case, is easier to solve than the more general maximum edge-weight matching problem. Indeed, the best known time complexity of algorithms that solve the maximum vertex-weight matching problem is $O(\sqrt{|V||E|} \log(|V|))$, with algorithms of time complexity $O(|V||E|)$ being more practical. Of course, this also makes the maximum vertex-weight matching problem more difficult to solve than the maximum cardinality matching problem. Interestingly enough, the two latter problems are closely related, algorithms that compute maximum vertex-weight matchings being just specialized maximum cardinality matching algorithms.

Note that the computational complexity results mentioned so far correspond to algorithms that compute optimal (exact) solutions. We refer to such algorithms as exact algorithms, in order to distinguish them from approximation algorithms, which compute suboptimal (approximate) solutions. However, our interest, as already mentioned, lies with approximation algorithms.

While approximation algorithms were originally designed for problems for which no polynomial time exact algorithms seem to exist, there is a recent trend in designing approximation algorithms for problems for which polynomial time algorithms are known but are not fast enough. Note for example that all the computational complexity results mentioned so far are polynomial but not linear. One may prefer a linear time algorithm if the computed solution is reasonably good, although not optimal.

A very simple example of an approximation algorithm is available for the maximum cardinality matching problem: compute a maximal matching rather than a maximum cardinality one. Such an algorithm guarantees an approximation ratio (the factor that indicates how good the suboptimal solution is, compared to the optimal one) of $1/2$, for a time complexity of $\Theta(|E|)$, thus linear. It is actually easy to tune the approximation ratio for the maximum cardinality matching as $(k + 1)/(k + 2)$, where $k \in \mathbb{Z}, k \geq 0$, being thus able to get arbitrarily close to the optimal solution, at the expense of increased time complexity.

Linear time complexity $1/2$-approximation algorithms were recently designed for the maximum edge-weight matching problem. Several attempts were also made for $2/3$-approximation but, although one can get arbitrarily close to the $2/3$ approximation ratio, this also increases the time complexity.

A closer look at such matching algorithms suggests short augmentation as a basic technique for approximation. For the maximum cardinality matching problem, for example, approximation algorithms restrict the length of augmenting paths to $2k + 1$ in order to achieve an approximation ratio of $(k + 1)/(k + 2)$. It is worth noting that short augmentations tend to have a local nature and therefore can be good candidates for parallel matching algorithms.

We employ short augmentations for vertex-weighted approximation matching algorithms as well. In this talk we describe three such algorithms for the particular case of bipartite graphs: two $1/2$-approximation algorithms and one $2/3$-approximation algorithm. One of the $1/2$-approximation algorithms and the $2/3$-approximation algorithm require sorting each set of vertices $(S$ and $T)$ in decreasing order of their weights and therefore their time complexity is not linear. The other $1/2$-approximation algorithm has linear time complexity, $\Theta(|E|)$.

The difference between our vertex-weighted approximation matching algorithms and the recently proposed approximation algorithms for edge-weighted matching is determined by the ability to achieve $2/3$-approximation. In the vertex-weighted case we are able to do so by properly restricting the length of the augmenting paths, something that does not seem possible in the edge-weighted case.

The existence of a reasonably fast, although not linear time, $2/3$-approximation algorithm for the bipartite maximum vertex-weight matching problem indicates that this problem is easier to solve approximately than its edge-weighted counterpart. However, it looks like our $2/3$-approximation algorithm cannot be generalized to $(k + 1)/(k + 2)$-approximation (even pushing it to $3/4$-approximation does not seem possible, at least based on the techniques we use in order to achieve $2/3$-approximation), which indicates that the maximum vertex-weight problem is more difficult to solve approximately than the maximum cardinality matching problem.

In this talk we illustrate the techniques used by our approximation algorithms and we provide elements of the proofs of the approximation ratios. We also supplement this theoretical work with experimental results obtained through actual implementations, providing thus practical comparisons between various algorithms as well.