Matching through Embedding in Dense Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Finding optimal matchings in dense graphs is of general interest and of particular importance in social, transportation and biological networks. While developing optimal solutions for various matching problems is important, the running times of the fastest available optimal matching algorithms are too costly. However, when the vertices of the graphs are point-sets in $\mathbb{R}^d$ and edge weights correspond to the euclidean distances, the available optimal matching algorithms are substantially faster. In this paper, we propose a novel *network embedding* based heuristic algorithm to solve various matching problems in dense graphs. In particular, using existing network embedding techniques, we first find a low dimensional representation of the graph vertices in $\mathbb{R}^d$ and then run faster available matching algorithms on the embedded vertices. To the best of our knowledge, this is the first work that applies network embedding to solve various matching problems. Experimental results validate the efficacy of our proposed algorithm.

1 Introduction

Matching in graphs is one of the most well-studied problems in economics and computer science. This is an ubiquitous problem with application areas ranging from transportation networks [2] to social networks [5], and computational chemistry [9]. For instance, in the case of social networks, tag suggestions and localization problems for images can be transformed into a weighted bipartite graph matching problems [5]. Similarly, the processes of de-anonymization and privacy inference in social networks can be reduced to finding the maximum weighted bipartite matching of the corresponding knowledge graph [20].

Given a graph $G = (V, E)$, a matching M is a subset of edges such that no two edges in M share a common vertex. The matching M is *perfect* if every $v \in V$ belongs to an edge of M. For weighted graphs, it is often required to compute a perfect matching that is optimal with respect to some criterion. Given a real weight w_e for each edge $e \in E$, a *minimum cost* matching (MCM) minimizes $\sum_{e \in M} w_e$ among all feasible perfect matchings M for G. Similarly, a bottleneck matching (BM) minimizes $\max_{e \in M} w_e$ while a *uniform matching* (UM) minimizes $\max_{e \in M} w_e - \min_{e \in M} w_e$. A *minimum deviation* matching (MDM) minimizes $(1/|V| \sum_{e \in M} w_e) - \min_{e \in M} w_e$.

A plethora of work has been done to develop efficient algorithms for obtaining optimal solutions for various matchings. However, for many real-world problems with larger graphs [3, 17], the running times of the fastest available matching algorithms are too costly. For example, the best known algorithm for the minimum cost matching problem runs in $O(n^3)$ time for a dense graph with n vertices [14]. For massive graphs, this is quite inefficient. Thus designing efficient approximate algorithms permit the solution of large instances of matching problems that arise in practical situations.

There exist many constant factor approximation algorithms for the maximum weight matching (MWM) problem. However, there is no such algorithm for the minimum cost matching (MCM) problem. The greedy heuristic for the MCM problem, attempts to construct a minimum cost per-

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

Graph Type	Matching Type	Algo.	Algo. Type	Complexity
Bipartite	Minimum Cost (MCM)	Kuhn et al. [14]	Non-Euclidean	$O(n^3)$
		Agarwal et al. [1]	**Euclidean-$\mathbb{R}^d$**	$\boldsymbol{O(n^{2+\epsilon})}$
Bipartite	Bottleneck (BM)	Punnen et al. [19]	Non-Euclidean	$O(n^2\sqrt{n})$
		Efrat et al. [7]	**Euclidean-$\mathbb{R}^d$**	$\boldsymbol{O(n^{1.5}\log^d n)}$
Bipartite	Uniform (UM)	Martello et al. [16]	Non-Euclidean	$O(n^4)$
		Efrat et al. [7]	**Euclidean**	$\boldsymbol{O(n^{10/3})}$
Bipartite	Minimum Deviation (MDM)	Efrat et al. [6]	Non-Euclidean	$O(n^4)$
		Efrat et al. [7]	**Euclidean**	$\boldsymbol{O(n^{10/3})}$
Non-bipartite	Minimum Cost (MCM)	Gabow et al. [11]	Non-Euclidean	$O(nm + n^2\log n)$
		Varadarajan et al. [22]	**Euclidean-$\mathbb{R}^d$**	$\boldsymbol{O(n^{1.5}\log n)}$
Non-bipartite	Bottleneck (BM)	Gabow et al. [10]	Non-Euclidean	$O(m\sqrt{n}\log n)$
		Efrat et al. [8]	**Euclidean-$\mathbb{R}^d$**	$\boldsymbol{O\left(n^{2-2/(\lceil d/2\rceil+1)+\epsilon}\right)}$

Table 1: Running time complexities of various optimal matching algorithms.

fect matching by starting with an empty matching and iteratively adding a minimum weight edge between two exposed nodes. While the running time complexities of various greedy matching algorithms are generally linear in terms of the number of edges (m), the corresponding approximation ratios are high. The greedy heuristic runs in $O(m \log n)$ time and finds a solution with cost at most $4/3\ n^{\log 3/2}$ times the optimum cost [21]. [12] developed an $O(m)$ heuristic that constructs a matching with cost at most $2n^{\log_3 7/3}$ times the optimum cost. Thus one should focus on designing efficient approximate algorithms with good performance guarantees.

While for non-Euclidean graphs the running time complexities of optimal matching algorithms are high, the available optimal matching algorithms are substantially faster for the Euclidean case, i.e. when the vertices of the graph are point sets in $\mathbb{R}^d$ and edge weights corresponds to euclidean distances. For example, the best known algorithm for the bottleneck matching problem runs in $O(n^{1.5}\log^d n)$ time for a bipartite Euclidean graph as compared to an $O(n^{2.5})$ for its non-Euclidean counterpart. Thus the following natural question arises. *Can we leverage Euclidean matching techniques to obtain near-optimal solutions to non-Euclidean matching problems?*

In this work, we propose a *network embedding* based algorithm to obtain approximate solutions to non-Euclidean matching problems. More precisely, using existing linear time network embedding techniques, we embed the vertices of the non-Euclidean graph into points in $\mathbb{R}^d$ such that the neighborhood of the vertices are approximately preserved. We then run faster available Euclidean matching algorithms on the embedded vertices. To the best of our knowledge, this is the first work that applies network embedding to solve various matching problems. Empirical results show the efficacy of our proposed algorithm.

2 Technical Preliminaries

We consider the problem of finding approximate solutions to optimal matching problem in complete bipartite and general graphs[1]. We denote the graph as $G = (V, E)$ with $E \subset V \times V$. Here, V denotes the vertex set and E denotes the edge set. We assign a real weight w_e to each edge $e \in E$. A matching $M \subset E$ is defined to be a set of edges such that no vertex of G is incident to more than one edge of M. Denote $|V| = n$ and $|E| = m$. Our goal is to compute a perfect matching that is near-optimal with respect to optimality criterion defined in the previous Section.

2.1 Geometry Helps in Matching

A summary of the running time complexities of various optimal matching algorithms in Euclidean and non-Euclidean setting is given in Table 1. Note that, bipartite matching is a special case of general graph matching. Computing an optimal bipartite matching is more challenging than computing

[1]We assume the graph to be sufficiently dense for the case when it is not complete. In such a case, we can add $O(1)$ edges each with infinite costs to the graph to make it complete without affecting the overall running time complexity.

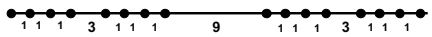

Figure 1: Adversarial model for assigning edge weights with $n = 16$ [21].

an optimal matching on a complete non-bipartite graph in a Euclidean setting. For example, MCM on a set of $2n$ points can be computed in $O(n^{1.5} \log n)$ time, while the best known algorithm for computing MCM between two point sets of size n each takes $O(n^{2+\epsilon})$ time. Also note that, there is considerable amount of savings in terms of running times in the Euclidean setting as compared to the non-Euclidean setting. For instance, MCM on a non-Euclidean non-bipartite complete graph runs in $O(n^3)$ time where as MCM on a Euclidean non-bipartite complete graph runs in $O(n^{1.5} \log n)$ time. Similar savings can be observed for many other matching problems, such as BM, UM and MDM. Below we transform the non-Euclidean matching problem into a Euclidean one through network embedding.

3 Our Approach: Matching through Embedding

Given a graph G, we aim to learn a *representation* of V in a low-dimensional vector space $\mathbb{R}^d$, i.e., find a map $f : V \rightarrow \mathbb{R}^d$ such that the neighborhoods of nodes are approximately preserved. This allows us to execute matching algorithms on the set of vectors instead of doing that on G itself. Below we present two network embedding techniques available from literature.

- *Deep Walk* [18]: As a homogeneous network embedding method, DeepWalk performs uniform random walks to get a corpus of vertex sequences. Then the word2vec is applied on the corpus to learn vertex embeddings.
- *Node2Vec* [13]: This method extends DeepWalk by performing biased random walks to generate the corpus of vertex sequences. The hyper-parameters p and q can be set to different values.

We construct the embedded graph $G'(V', E')$ as follows. We let $V' = \{f(v), \forall v \in V\}$. Also, we let $E' = \{(f(u), f(v)), \forall (u, v) \in E\}$. We then define weight of an edge $e' = (u', v') \in E'$ as $w_{e'} = ||u' - v'||_2$. Clearly G' is a Euclidean graph in dimension d. We then apply available optimal Euclidean matching algorithms on G' to obtain a matching M'. We output M' as the approximate solution to the optimal matching problem on G.

Remark 1. *Note that, the above mentioned embedding techniques can learn the representation of V in $O(n \log n)$ time[4]. Thus the overall time complexity of the proposed algorithm is equal to that of the corresponding Euclidean matching algorithm.*

4 Empirical Results

In this section we present experiments on several synthetic datasets.

4.1 Synthetic Datasets and Parameters

For synthetic datasets, we vary the distributions of edge weights. We compare the performance of our proposed heuristic algorithm to that of the greedy heuristic under two models based on generating edge weights. First, we consider an adversarial model. Under this model, edge weights are generated according to [21] and as shown in Figure 1. We set $n = 2^{t-1}$ with t being the number of clusters of nodes and the largest distance between adjacent nodes is set to 3^{t-2}. Under the adversarial model, the greedy heuristic finds a solution with cost $O(n^{\log 3/2})$ times the optimum cost [21]. Next, we consider the Lomax distribution (a long tail distribution) [15] for generating the weights since they are widely used in real-world applications [23]. We chose the Lomax distribution parameter from the set $\{2, 3, 5, 10, 50\}$. The parameter settings are motivated by [23, 13]. We set the number of walks per node and length of each random walk to 20 for both deepWalk and node2vec. The embedding number of dimensions, return (p) and in-out parameter (q) are set to $10, 0.5$ and 2.0 respectively. The experiments are repeated 5 times for each parameter setting and the 95% confidence interval plots are reported. If not mentioned, in our experiments, n = 100 and m = 4950 for non-bipartite graphs. We focus on MCM and BM algorithms in non-bipartite setting. We get similar results for bipartite graphs and hence omit them here.

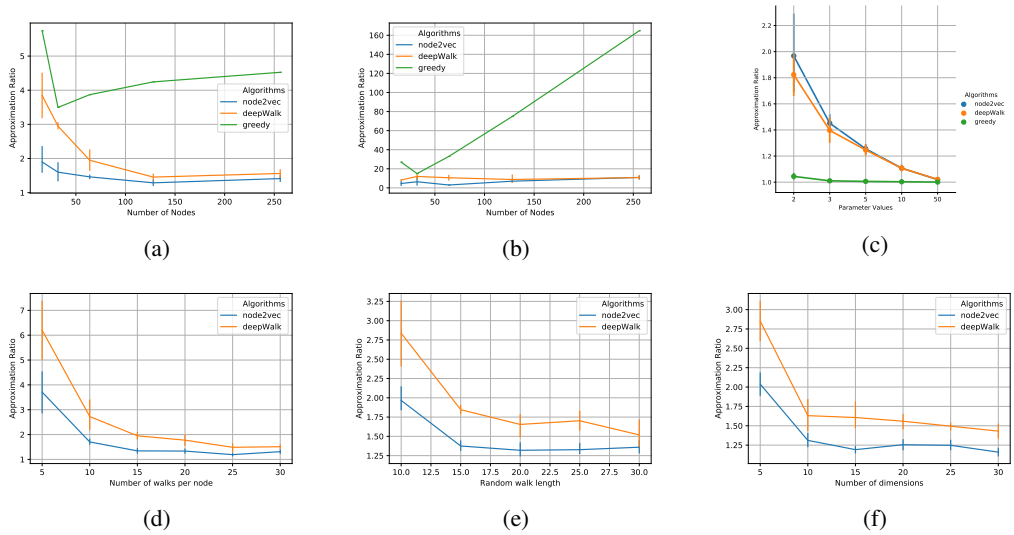

Figure 2: Performance comparison of our proposed algorithm to that of greedy for (a) MCM (b) BM under adversarial model and (c) MCM under lomax model. (d)-(f) Parameter sensitivity of the proposed algorithm under adversarial model.

4.2 Results

We present the simulation results in Figure 2.

4.2.1 Edge weight distribution

We first present the results for MCM and BM under adversarial model as shown in Figures 2(a) and (b) respectively. We plot the approximation ratios of various matching algorithms as a function of number of nodes in the graph. For both MCM and BM, the proposed node2vec and deepWalk based algorithms yield lower approximation ratio, hence better performance compared to greedy heuristic. Also, as the number of nodes increases, the performance of greedy heuristic decreases while that of embedding based techniques increases. One possible explanation for better performance of the embedding based techniques is due to their ability to learn the inherent one-dimensional structure of the adversarial model. However, the nature of the plots are reversed for MCM under the Lomax distribution model as shown Figure 2 (c). In this case, greedy beats both deepWalk and node2vec based algorithms. Note that, as the Lomax distribution parameter increases, both embedding based techniques gradually perform better and finally converge to the greedy solution.

4.2.2 Parameter Sensitivity

The embedding based matching algorithms involve a number of parameters. Below, we examine the effect of different parameters on the overall performance of embedding based techniques under adversarial model as shown in Figures 2 (d)-(f). All other parameters assume default values, except for the parameter being examined. We measure the approximation ratio as a function of number and length of walks per node, number of dimensions. The performance of both node2vec and deepWalk increase with the number, length of walks per node and number of dimensions as expected. Again, node2vec beats deepWalk across all values of parameters.

5 Conclusion

In this paper, we developed approximate solutions for various matching problems on dense graphs. More precisely, we proposed a network embedding based heuristic algorithm using existing network embedding techniques. We also performed simulations on synthetic datasets to obtain comparison results for empirical approximation ratios across different proposed and existing matching algorithms. Future directions include to consider the effect of other non-random walk based embedding techniques on the overall performance of the proposed algorithm.

References

[1] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *In Proc. 11th Annu. ACMSympos. Comput. Geom.*, pages 39–50, 1995.

[2] A. Aggarwal, A. Barnoy, S. Khuller, D. Kravets, and B. Schieber. Efficient minimum cost matching using quadrangle inequality. *Journal of Algorithms*, 19:116–143, 1995.

[3] R. Beier and J. F. Sibeyn. A powerful heuristic for telephone gossiping. *In Proceedings of the 7th Colloquium on Structural Information and Communication Complexity*, pages 17–35, 2000.

[4] H. Chen, Y. Hu, B. Perozzi, and S. Skiena. HARP: Hierarchical representation learning for networks. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 2127–2134, 2018.

[5] W. T. Chu, C. J. Li, and J. Y. Yu. Tag suggestion and localization for images by bipartite graph matching. *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2013*, 2013.

[6] A. Efrat. Geometric Location Optimization. *Ph.D. Dissertion, Tel-Aviv University*, 1998.

[7] A. Efrat, M. Itai, and M. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001.

[8] A. Efrat and M. Katz. Computing euclidean bottleneck matchings in higher dimensions. *Information Processing Letters*, 75:169–174, 2000.

[9] H. Frohlich, J. K. Wegner, and A. Zell. Assignment kernels for chemical compounds. *IJCNN*, 2005.

[10] H. N. Gabow, , and R. E. Tarjan. Algorithms for Two Bottleneck Optimization Problems. *J. Algorithms*, 9:411–417, 1988.

[11] H. N. Gabow. Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[12] M. D. Grigoriadis and B. Kalantari. A new class of heuristic algorithms for weighted perfect matching. *J. Assoc. Comput. Mach.*, 35:769–776, 1988.

[13] A. Grover and J. Leskovec. node2vec: Scalable Feature Learning for Networks. *KDD*, pages 855–864, 2016.

[14] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[15] K. S. Lomax. Business Failures; Another example of the analysis of failure data. *Journal of the American Statistical Association*, 49:847–852, 1954.

[16] S. Martello, W. R. Pulleyblank, P. Toth, and D. Werra. Balanced optimization problems. *Operations Research Letters*, 3:275–278, 1984.

[17] B. Monien, R. Preis, and R. Diekmann. Quality matching and local improvement for multilevel graph-partitioning. *Paral. Comput.*, 26:1609–1634, 2000.

[18] B. Perozzi and S. Skiena. DeepWalk: Online Learning of Social Representations. *KDD*, 2014.

[19] A. P. Punnen, , and K. P. Nair. Improved Complexity Bound for the Maximum Cardinality Bottleneck Bipartite Matching Problem. *Discrete Applied Mathematics*, 55:91–93, 1994.

[20] J. Qian, X. Y. Li, C. Zhang, and L. Chen. De-anonymizing social networks and inferring private attributes using knowledge graphs. *Proceedings - IEEE INFOCOM*, 2016-July, 2016.

[21] E. M. Reingold and R. E. Tarjan. On a greedy heuristic for complete matching. *SIAM J. Comput.*, 10:676–681, 1981.

[22] K. R. Varadarajan. A divide-and-conquer algorithm for min-cost perfect matching in the plane. *In Proc. FOCS*, 1998.

[23] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. *Proceedings - International Conference on Data Engineering*, 2019-April:1478–1489, 2019.