Near approximation of maximum weight matching through efficient weight reduction

Cui Di[1] and Andrzej Lingas[1]

Department of Computer Science, Lund University, 22100 Lund, Sweden.
dcdcsunny@gmail.com Andrzej.Lingas@cs.lth.se

Abstract. Let G be an edge-weighted hypergraph on n vertices, m edges of size $O(1)$, where the edges have real weights in an interval $[1,\ W]$. We show that if we can approximate a maximum weight matching in G within factor α in time $T(n,m,W)$ then we can find a matching of weight at least $(\alpha-\epsilon)$ times the maximum weight of a matching in G in time $(\epsilon^{-1})^{O(1)}\max_{1\leq q\leq O(\epsilon\frac{\log\frac{n}{\epsilon}}{\log\epsilon^{-1}})}\max_{m_1+\ldots m_q=m}\sum_1^q T(n,m_j,(\epsilon^{-1})^{O(\epsilon^{-1})})$. In particular, if we combine our result with the recent $(1-\epsilon)$-approximation algorithm for maximum weight matching in graphs with small edge weights due to Duan and Pettie then we obtain $(1-\epsilon)$-approximation algorithm for maximum weight matching in graphs running in time $(\epsilon^{-1})^{O(\epsilon^{-1})}(m+n\log n)$.

1 Introduction

A *hypergraph* G consists of a set V of vertices and a set of subsets of V called edges of G. In particular, if all the edges are of cardinality two then G is a graph. A *matching* of G is a set of pairwise non-incident edges of G. If real weights are assigned to the edges of G then a *maximum weight matching* of G is a matching of G whose total weight achieves the maximum.

The problem of finding a maximum weight matching in a hypergraph is a fundamental generalization of that of finding maximum cardinality matching in a graph. The latter is one of the basic difficult combinatorial problems that still admit polynomial-time solutions. For hypergraphs the decision version of the maximum weight matching problem is NP-hard even if the edges are of size $O(1)$ since it is a generalization of the problem of maximum weight independent set for bounded degree graphs [15]. On the other hand, polynomial-time algorithms yielding $(d-1+1/d)$-approximation of maximum weight matching in hypergraphs with edges of size d are known [3].

The fastest known algorithms for maximum weight matching in graphs have substantially super-quadratic time complexity in terms of the number n of vertices of the input graph G [11, 12, 21]. For these reasons, there is a lot of interest in designing faster approximation algorithms for maximum weight matching [4–6, 14, 19, 20].

Recently, even fast approximation schemes for maximum weight matching in graphs have been presented [1]. The fastest known in the literature is due to Duan and Pettie [7]. It yields a $(1-\epsilon)$-approximation in time $O(m\epsilon^{-2}\log^3 n)$ for a connected graph on n vertices and m edges with real edge weights. The approximation scheme from [7] is a composition of a $(1-\epsilon)$-approximate reduction of the problem in general edge weighted graphs to that in graphs with small edge weights and an efficient $(1-\epsilon)$-approximate algorithm for graphs with small edge weights.

1.1 Our contributions

Let G be an edge-weighted hypergraph on n vertices, m edges of size $O(1)$, where the edges have size real weights in an interval $[1,\ W]$. We show that if we can approximate a maximum weight matching in G within factor α in time $T(n,m,W)$ then we can find a matching of weight at least $\alpha-\epsilon$ times the maximum weight of a matching in G in time
$(\epsilon^{-1})^{O(1)}\max_{1\le q\le O(\epsilon\frac{\log n}{\epsilon})}\max_{m_1+\ldots m_q=m}\sum_1^q T(n,m_j,(\epsilon^{-1})^{O(\epsilon^{-1})})$.

This reduction of maximum weight matching in hypergraphs with arbitralily large edge weights to that in hypergraphs with small edge weights is incomparable to the aforementioned similar reduction for graphs from [7]. In particular, if we combine our reduction with the aforementioned $(1-\epsilon)$-approximation algorithm for maximum weight matching in graphs with small edge weights from [7] then we obtain a $(1-\epsilon)$-approximation algorithm for maximum weight matching in graphs running in time $(\epsilon^{-1})^{-O(\epsilon^{-1})}(m+n\log n)$. In comparison with the approximation scheme from [7], our approximation scheme is more truly linear in $n+m$, as essentially free from the polylogarithmic in n factor but for very sparse graphs, at the cost of super-exponential dependence on ϵ^{-1}.

As another corollary from our approximate edge-weight reduction for hypergraphs, we obtain also some results on approximating maximum weight independent set in graphs of bounded degree.

1.2 Other related results

As the problem of finding maximum weight matching in graphs is a classical problem in combinatorial optimization there is an extensive literature on it. It includes such milestones as an early algorithm of Kuhn [18] just in the bipartite case, an algorithm of Edmond and Karp [8] running in time $O(nm^2)$, where n

[1] In a preliminary version of this paper presented at SOFSEM Student Forum (January 2010), an $O(n^\omega\log n)$-time approximation scheme for maximum weight matching in graphs has been presented.

is the number of vertices and m is the number of edges in the input graph, an $O(\sqrt{n}m\log(nW))$ and $O(\sqrt{n\log n}m\log(nW))$ time bounds respectively in bipartite and general graphs due to Gabow and Tarjan [11, 12], assuming integer edge weights in $[-W, W]$. The Hungarian algorithm [18] can be implemented in time $O(mn + n^2\log n)$ with the help of Fibonacci heaps [9] and this upper bound can be extended to include general graphs [10].

More recently, Sankowski designed an $O(n^\omega W)$-time algorithm for the weighted matching problem where ω stands for the exponent of fast matrix multiplication known to not exceed 2.376, the edge weights are positive integers and the input graph is bipartite [21].

There is also an extensive literature on fast approximation algorithms for maximum weight matching in graphs [4–6, 14, 19, 20]. Typically they yield an approximation within a constant factor between $\frac{1}{2}$ and almost $\frac{4}{5}$, running in time of order $m\log^{O(1)} n$. Already the straightforward greedy approach yields $\frac{1}{2}$-approximation in time $O(m\log n)$.

The maximum weight matching problem in hypergraphs is known also as a set packing problem in combinatorial optimization [15]. By duality it is equivalent to maximum weight independent set and hence extremely hard to approximate in polynomial time [13]. The most studied case of maximum weight matching in hypergraphs is that for d-uniform hypergraphs where each edge is of size d. Then a polynomial-time $(d-1+1/d)$-approximation is possible [3]. By duality, one obtains also a polynomial-time $(d-1+1/d)$-approximation of maximum weight independent set in graphs of degree d (cf. [15]).

2 Simple edge weight transformations

In this section, we describe two simple transformations of the edge weights in the input hypergraph G such that an α-approximation of maximum weight matching in the resulting hypergraph yields an $(\alpha-\epsilon)$-approximation of maximum weight matching of G. We assume w.l.o.g. throughout the paper that G has n vertices, m edges, and real edge weights not less than 1. The largest edge weight in G is denoted by W.

Lemma 1. *Suppose that there is an α-approximation algorithm for maximum weight matching in G running in time $T(n, m, W)$. Then, there is an $O(n+m)$-time transformation of G into an isomorphic hypergraph G^* with edge weights in the interval $[1, \frac{n}{\epsilon}]$ such that the aforementioned algorithm run on G^* yields an $(\alpha-\epsilon)$-approximation of maximum weight matching in G in time $T(n, m, \frac{n}{\epsilon})$.*

Proof. We may assume w.l.o.g that $W > \frac{n}{\epsilon}$. Note that the total weight of maximum weight matching in G is at least W. Hence, if we transform G to a hyper-

graph G' by raising the weight of all edges in G of weight smaller than $\frac{W\epsilon}{n}$ to $\frac{W\epsilon}{n}$ then the following holds:

1. the maximum weight of a matching in G' is not less than that in G;
2. any matching in G' induces a matching in G whose weight is smaller by at most ϵW.

To find an α-approximation of maximum weight matching in G', we can simply rescale the edge weights in G' by multiplying them by $\frac{n}{W\epsilon}$. Let G^* denote the resulting graph. Now it is sufficient to run the asumed algorithm on G^* to obtain an $(\alpha - \epsilon)$-approximation of maximum weight matching in G. Note that the application of the algorithm will take time $T(n, m, \frac{n}{\epsilon})$. □

Lemma 2. *Suppose that there is an $(\alpha - \epsilon)$-approximation algorithm for maximum weight matching in G running in time $T'(n', m', W', \epsilon)$. By rounding down each edge weight to the nearest power of $1 + \epsilon$ and then running the $(\alpha - \epsilon)$-approximation algorithm on the resulting graph, we obtain an $(\alpha - O(\epsilon))$-approximation of maximum weight matching in G in time $T'(n, m, W, \epsilon) + O(n + m)$.*

Proof. Let e be any edge in G. Denote its weight in G by $w(e)$ and its weight in the resulting graph by $w'(e)$. We have $w'(e)(1 + \epsilon) \geq w(e)$. Consequently, we obtain $w'(e) \geq w(e) - \epsilon w'(e) \geq (1 - \epsilon)w(e)$. It follows that a maximum weight matching in the resulting graph has weight at least $1-\epsilon$ times the weight of a maximum weight matching in G. Thus, if we run the asumed $(\alpha - \epsilon)$-approximation algorithm on the resulting graph then the produced matching with edge weights restored back to their original values will yield an $(\alpha - 2\epsilon)$-approximation. □

3 A transformation into an $(\alpha - \epsilon)$-approximation algorithm

A *subhypergraph* of a hypergraph H is any hypergraph that can be obtained from H by deleting some vertices and some edges. A class C of hypergraphs such that any subhypergraph of a hypergraph in C also belongs to C is called *hereditary*.

In this section, we present a transformation of a hypothetic α-approximation algorithm for maximum weight matching in a hereditary family of hypergraphs with edges of size $O(1)$ into a $(\alpha - \epsilon)$-approximation algorithm. The running time of the $(\alpha - \epsilon)$-approximation algorithm is close to that of the α-approximation algorithm in case the largest edge weight is $\epsilon^{-O(\epsilon^{-1})}$.

Theorem 1. *Suppose that there is an algorithm for a maximum weight matching in any hypergraph having edges of size $O(1)$ and belonging to the same hereditary class as G running in time $T(n', m', W') = \Omega(n' + m')$, where n', m' are respectively the number of vertices and edges, and $[1, W']$ is the interval to which all edge weights belong. There is an $(\alpha - \epsilon)$-approximation algorithm for a maximum weight matching in G running in time* $(\epsilon^{-1})^{O(1)} \max_{1 \leq q \leq O(\epsilon \frac{\log \frac{n}{\epsilon}}{\log \epsilon^{-1}})} \max_{m_1 + \ldots m_q = m} \sum_1^q T(n, m_j, (\epsilon^{-1})^{O(\epsilon^{-1})})$.

Proof. We may assume w.l.o.g that $W = O(n/\epsilon)$ and any edge weight is a nonnegative integer power of $1 + \epsilon$ by Lemmata 1, 2. Order the values of the edge weights in G in the increasing order. Set $k = O(\epsilon^{-1})$ and $l = \lceil \log_{1+\epsilon} \frac{2}{\epsilon} \rceil$. By the form of the edge weights and the setting of l, the following holds.

Remark 1: For any two different edge weights w_1 and w_2, if the number of w_1 is greater than that of w_2 by at least l in the aforementioned ordering then $\frac{\epsilon}{2} w_1 \geq w_2$.

In order to specify our $(\alpha - \epsilon)$-approximation algorithm, we partition the ordered edge weights into consecutive closed basic intervals, each but perhaps for the last, containing exactly l consecutive edge weights, see Fig. 1.

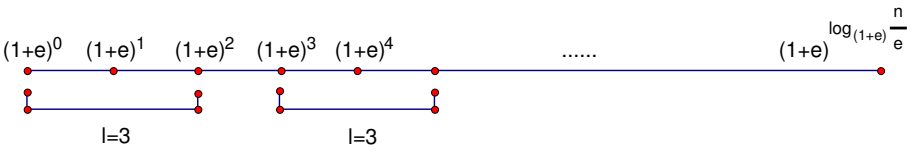

Fig. 1. Partitioning of edge weights ($l = 3$)

Next, we group k-tuples of consecutive basic intervals into large intervals composed of $k-1$ consecutive basic intervals followed by a single basic interval called a gap. This partition corresponds to the situation when the so called shift parameter x is set to 0. For $x \in \{1, .., k-1\}$, the partition into alternating large intervals and gaps is shifted by x basic intervals from the right, so the first large interval from the right is composed solely of $k - 1 - x$ basic intervals, see Fig. 2. The maximal subgraph of G containing solely edges in the large intervals in the partition is denoted by G_x.

For our $(\alpha - \epsilon)$-approximation algorithm for a maximum weight matching in G see Fig. 3. We shall assume the definitions of the subgraphs $G'_x, G_{x,j}, M_x$ from the algorithm.

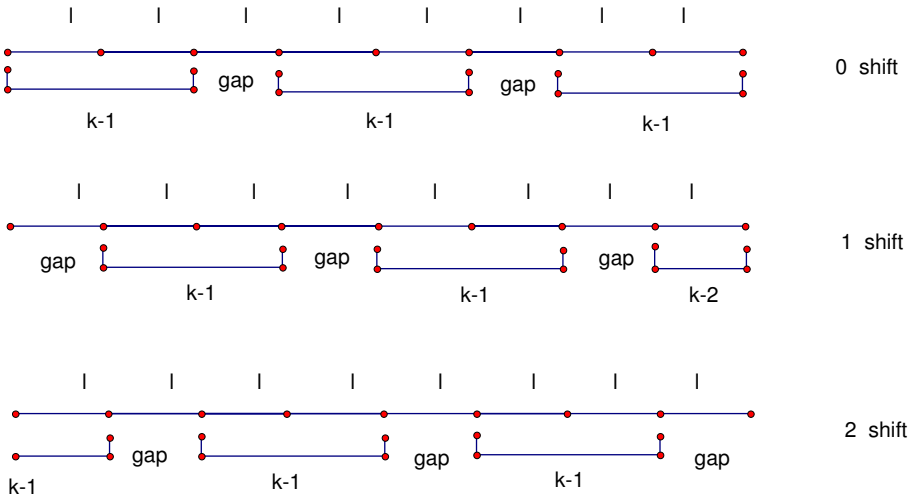

Fig. 2. An example of shift: l=3, k=3

Since the union of the gaps over all shifts covers all weights there must a shift where the gaps cover at most $\frac{1}{k}$ of the weight of optimal matching of G. Hence, there must be a shift x such that the weight of optimal matching in G_x is at least $(1 - 1/k)$ of the weight of optimal matching of G. Thus, it is sufficient to show that M_x closely approximates an α-approximate weight matching of G_x.

Consider a maximum weight matching OM_x of G_x and the α-approximation $M_{x.j}$ of a maximum weight matching of $G_{x,j}$, respectively. Note that $M_{x,j}$ has total weight not smaller than α times the total weight of OM_x restricted to the edges in $G_{x,j}$. On the other hand, each edge e in $M_{x,j}$ can eliminate at most $O(1)$ edges of OM_x from all $G_{x,i}$ for $i > j$. The total weight of the at most $O(1)$ edges is only at most the ϵ fraction of the weight of e by Remark 1. Let EOM_x denote the set of all edges in OM_x eliminated by $M_x = \bigcup_j M_{x,j}$. The following two inequalities follow:

$$weight(M_x) + weight(EOM_x) \geq \alpha \times weight(OM_x)$$

$$\epsilon \times weight(M_x) \geq weight(EOM_x)$$

Consequently, we obtain:

$$weight(M_x) \geq \alpha \times weight(OM_x) - \epsilon \times weight(M_x) \geq (\alpha - \epsilon) \times weight(OM_x)$$

Algorithm 1

1. **for** $x \leftarrow 1$ **to** $k-1$ **do**
2. $M_x \leftarrow \emptyset$;
3. $G'_x \leftarrow G_x$;
4. **for** $j \leftarrow 1$ **to** $O(\log_{1+\epsilon} \frac{n}{\epsilon})$ **do**
5. **begin**
6. Set $G_{x,j}$ to the sub-hypergraph of G'_x induced by the edges whose weights
7. fall in the jth interval from the right;
8. Run the α-approximation algorithm for maximum weight matching $M_{x,j}$ of $G_{x,j}$;
9. $M_x \leftarrow M_x \bigcup M_{x,j}$;
10. Remove all edges incident to $M_{x,j}$ from G'_x;
11. **end**
12. Return the heaviest among the matchings M_x

Fig. 3. The $(\alpha - \epsilon)$-approximation algorithm.

Thus, M_x approximates within $\alpha - \epsilon$ a maximum weight matching of G_x, and consequently the heaviest of the matchings M_x approximates within $(1 - \epsilon)(1 - 1/k)$ a maximum weight matching of G. By setting $k = \Omega(\frac{1}{\epsilon})$, we obtain an $(1 - O(\epsilon))$-approximation of the optimum.

It remains to estimate the time complexity of our method. Note that the weight of heaviest edge in $G_{x,j}$ is at most

$$(1+\epsilon)^{lk} = O(\epsilon^{-1})^{O(\epsilon^{-1})} = (\epsilon^{-1})^{O(\epsilon^{-1})}$$

times larger than that of the lightest one. Let $m_{x,j}$ denote the number of edges in $G_{x,j}$. Hence, by rescaling the weights in $G_{x,j}$, we can find $M_{x,j}$ in time $T(n, m_{x,j}, (\epsilon^{-1})^{O(\epsilon^{-1})})$ for $j = 1.., O(\log_{1+\epsilon} \frac{n}{\epsilon}/lk)$ and $x = 0, ..., k-1$. Note that $\log_{1+\epsilon} \frac{n}{\epsilon} = \frac{\log \frac{n}{\epsilon}}{\log 1+\epsilon} = \Theta(\epsilon^{-1} \log \frac{n}{\epsilon})$ and similarly $lk = \log_{1+\epsilon} \frac{2}{\epsilon} \Theta(\epsilon^{-1}) = \Theta(\frac{\log \frac{2}{\epsilon}}{\log 1+\epsilon} \epsilon^{-1}) = \Theta(\epsilon^{-2} \log \epsilon^{-1})$. It follows that for a given x, the largest value of j, i.e., the number of the subgraphs $G_{x,j}$ is $O(\epsilon \frac{\log \frac{n}{\epsilon}}{\log \epsilon^{-1}})$.

Note that $\sum_j m_{x,j} \leq m$ since each edge of G belongs to at most one hypergraph $G_{x,j}$. Thus, the total time taken by finding all $M_{x,j}$ for $j = 1, ..., O(\epsilon \frac{\log \frac{n}{\epsilon}}{\log \epsilon^{-1}})$ for a fixed x is
$\max_{1 \leq q \leq O(\epsilon \frac{\log \frac{n}{\epsilon}}{\log \epsilon^{-1}})} \max_{m_1 + ... m_q = m} \sum_1^q T(n, m_j, (\epsilon^{-1})^{O(\epsilon^{-1})})$. Recall that x ranges over $O(\epsilon^{-1})$ possible values.

By the assumed form of the edge weights in G, we can apply a standard radix sort with $O(\epsilon^{-1} \log \frac{n}{\epsilon})$ buckets to sort the edges of G by their weights in

time $O(m + \epsilon^{-1} \log \frac{n}{\epsilon})$. The latter is also $O(\epsilon^{-2}T(n, m, (\epsilon^{-1})^{O(\epsilon^{-1})}))$ by the assumptions on T.

In order to efficiently construct the graphs $G_{x,j}$, the sorted edge list is kept in array and there are double links between an occurrence of an edge in the adjacency lists representing G and its occurrence in the sorted edge list. To determine the edges inducing $G_{x,j}$, we just scan a consecutive fragment of the sorted list from left to right. Given a list of edges of $G_{x,j}$, an adjacency representation of the sub-hypergraph can be constructed in time $O(n + m) = O(T(n, m, (\epsilon^{-1})^{O(\epsilon^{-1})}))$ by using the aforementioned double links.

To remove an edge from G'_x, we locate it on the sorted edge list by using the double links with the adjacency lists and then link its predecessor with its successor on the sorted list. We conclude that the updates of G'_x take time $O(m) = O(T(n, m, (\epsilon^{-1})^{O(\epsilon^{-1})}))$. □

4 Applications

There are at least two known exact algorithms for maximum weight matching in bipartite graphs with integer edge weights for which the upper time bounds on their running time in linear fashion depend on the maximum edge weight W [16, 21]. Recently, Duan and Pettie have provided substantially more efficient $1-\epsilon$ approximation algorithm for maximum weight matching in general graphs with integer edge weights, whose running time also depends on W in linear fashion.

Fact 1 (Duan and Pettie, see the second section in [7]). *An $(1-\epsilon)$-approximation of maximum weight matching in a connected graph on m edges and positive integer weights not exceeding W can be found deterministically in time $O(mW/\epsilon)$.*

We can trivially generalize the upper time bound of Fact 1 to include a non-necessarily connected graph by extending it by an additive factor of $O(n)$.

There is one technical difficulty in combining Facts 1 with Theorem 1. Namely, in the theorem we assume that there is available an α-approximation algorithm for maximum weight matching for graphs belonging to the same hereditary class as G with arbitrary real edge weights not less than 1 whereas the algorithm of Facts 1 assumes integer weights. In fact, even if the input graph got positive integer weights the preliminary edge weight transformations in the proof of Theorem 1 would result in rational edge weights. There is a simple remedy for this. We may assume w.l.o.g that ϵ is an inverse of a positive integer and through all the steps of our approximation scheme round down the edge weights to the nearest fraction with denominator $O(\epsilon^{-1})$ and then multiply them by the common denominator to get integer weights. This will increase the maximum weight solely by $O(\epsilon^{-1})$ and will preserve close approximability.

Hence, Fact 1 combined in this way with Theorem 1 yield our main application result by straightforward calculations.

Theorem 2. *There is an approximation scheme for a maximum weight matching in a graph on n vertices and m edges running in time $(\epsilon^{-1})^{O(\epsilon^{-1})}(m + n \log n)$.*

5 Extensions

Note that Theorem 1 includes as a special case the problem of finding a maximum weight independent set in a graph G of maximum degree d which is equivalent to the problem of finding a maximum weight matching in the dual hypergraph with edges corresponding to the vertices of G and *vice versa*.

Several combinatorial algorithms for maximum independent set achieving the approximation ratio of $O(d)$, where d is the maximum or average degree are known in the literature [15]. Here, we demonstrate that by using the method of Theorem 1 they can be simply transformed into good approximation algorithms for maximum weight independent set.

Lemma 3. *Suppose that there is an $\alpha(d)$-approximation algorithm for maximum independent set in a graph on n vertices and maximum (or average degree, respectively) degree d running in time $S(n, d)$, where the function S is non-decreasing in both arguments. There is an $\alpha(dW)$-approximation algorithm for maximum weight independent set in a graph on n vertices, maximum (or average degree, respectively) degree d, positive integer weights not exceeding an integer W, running in time $S(nW, dW)$.*

Proof: Let G be the input vertex weighted graph G. We form the auxiliary unweighted graph G^* on the base of G as follows. In G^*, we replace each vertex v in G with the number of its copies equal to the weight of v. We connect each copy of v by an edge with each copy of each neighbor of v. Next, we run the assumed algorithm for maximum unweighted independent set on G^*. Note that any maximal independent in G^* is in one-to-one correspondence with an independent set in G since whenever a copy of v is in the independent set then all other copies of v can be inserted into it without any conflicts. □

The drawback of Lemma 3 is that the approximation factor and/or the running time of the resulting algorithm for the weighted case can be very large in case the maximum weight W is large. However, we can plug Lemma 3 in the method of Theorem 1 to obtain much more interesting approximation algorithms in the weighted case.

Theorem 3. *Suppose that there is an $\alpha(d)$-approximation algorithm for maximum independent set in a graph on n vertices and maximum degree d running in time $S(n,d)$, where the function S is non-decreasing in both arguments and $S(n,d) = \Omega(nd\log n)$. There is an $(\alpha(d\epsilon^{-1})^{O(\epsilon^{-1})}) - d\epsilon)$-approximation algorithm for maximum weight independent set in a graph on n vertices, with maximum degree d, positive integer vertex weights, running in time $O(\epsilon\frac{\log(n/\epsilon)}{\log\epsilon^{-1}}S(n(\epsilon^{-1})^{O(\epsilon^{-1})}, d(\epsilon^{-1})^{O(\epsilon^{-1})}))$.*

Proof. sketch. Recall that the problem of maximum (weighted or unweighted) independent set is equivalent to the problem of maximum (weighted or unweighted, respectively) matching in the dual hypergraph. In the dual hypergraph, the edges have size not exceeding the maximum vertex degree in the input graph. We run the method of Theorem 1 on the dual hypergraph using as the black box algorithm the result of the application of Lemma 3 to the assumed algorithm and its adaptation to the maximum matching problem in the dual hypergraph. □

6 Acknowledgments

The authors are grateful to anonymous referees for valuable comments on a preliminary version of the paper.

References

1. P. Berman. A $d/2$ Approximation for Maximum Weight Independent Set in d-Claw Free Graphs. Proc. 7th SWAT, Lecture Notes in Computer Science, Springer, Volume 1851, pp. 31-49, 2000.
2. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms. *2nd edition, McGraw-Hill Book Company, Boston, MA, 2001.*
3. Y.H. Chan and L.C. Lau. On Linear and Semidefinite Programming Relaxations for Hypergraph Matching. Proc.
4. D. Drake and S. Hougardy. A simple approximation algorithm for the weighted matching problem. *Info. Proc. Lett.*, 85:211-213, 2003.
5. D. Drake and S. Hougardy. Linear time local improvements for weighted matchings in graphs. *International Workshops on Experimental and Efficient Algorithms (WEA), LNCS 2647*, pages 107-119, 2003.
6. D. Drake and S. Hougardy. Improved linear time approximation algorithms for weighted matchings. *7th International Workshops on Randomization and Approximation Techniques in Computer Science (APPROX), LNCS 2764*, pages 14-23, 2003.
7. R. Duan andS. Pettie. Approximating Maximum Weight Matching in Near-linear Time. Proc. FOCS 2010.
8. J. Edmonds and R. M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM*, 19(2):248-264, 1972.
9. M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM, vol. 23, no. 2, pp. 596-615, 1987.

10. H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. *First Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 434-443, 1990.
11. H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013-1036, 1989.
12. H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. ACM*, 38(4):815-853, 1991.
13. J. Hastad. Clique is Hard to Approximate within $n^{1-\epsilon}$. Acta Math 182(1), pp. 105-142, 1999.
14. Hanke and Hougardy. $3/4 - \epsilon$ and $4/5 - \epsilon$ approximate MWM algorithms running in $O(mlogn)$ and $O(mlog^2n)$ time University of Bonn, Research Institute for Discrete Mathematics Report No. 101010.
15. D. S. Hochbaum, Approximating Covering and Packing Problems: Set Cover, Vertex Cover, Independent Set, and Related Problems in Approximation Algorithms for NP-hard Problems, D.S. Hochbaum (ed.), PWS Publishing Company, Boston, 1997.
16. M.-Y. Kao, T.-W. Lam, W.-K. Sung and H.-F. Ting. A Decomposition Theorem for Maximum Weight Bipartite Matchings with Applications to Evolutionary Trees. Proc. European Symposium on Algorithms (ESA 1999), LNCS 1643, Springer Verlag, pp. 438-449, 1999.
17. R. M. Karp, E. Upfal and A. Wigderson. Constructing a perfect matching is in random nc. *Combinatorica*, 6(1):35-48, 1986.
18. H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83-97, 1955.
19. S. Pettie and P. Sanders. A simple linear time 2/3-ϵ approximation for maximum weight matching. *Information Processing Letters*, 91:271-276, 2004.
20. R. Preis. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. *Proc. 16th Ann. Symp. on Theoretical Aspects of Computer Science (STACS), LNCS 1563*, pages 259-269, 1999.
21. P. Sankowski. Weighted bipartite matching in matrix multiplication time. *LNCS 4051*, pages 274-285, 2006.