A 4/3-APPROXIMATION FOR MINIMUM WEIGHT EDGE COVER

A Thesis

Submitted to the Faculty

of

Purdue University

by

Steven A. Gallagher

In Partial Fulfillment of the

Requirements for the Degree

of

Master's of Science

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Alex Pothen, Chair
School of Science

Dr. Pedro Fonseca
School of Science

Dr. Gustavo Rodriguez-rivera
School of Science

Approved by:

Dr. Susanne Hambrusch
Head of the School Graduate Program

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Gallagher, Steven A. M.S., Purdue University, May 2019. A 4/3-approximation for Minimum Weight Edge Cover. Major Professor: Alex Pothen.

This paper addresses the minimum weight edge cover problem (MEC), which is stated as follows: Given a graph $G = (V, E)$, find a set of edges $S : S \subseteq E$ and $\sum_{e \in S} w(e) \leqslant \sum_{e \in Q} w(e)\ \forall Q : \mathrm{Q}$ is an edge cover. Where an edge cover P is a set of edges such that $\forall v \in V$ v is incident to at least one edge in P. An efficient implementation of a 4/3-approximation for MEC is provided. Empirical results obtained experimentally from practical data sets are reported and compared against various other approximation algorithms for MEC.

1. INTRODUCTION

This paper addresses the minimum weight edge cover problem (MEC), which is stated as follows: Given a graph $G = (V, E)$, find an edge cover $S : S \subseteq E$ and $\sum_{e \in S} w(e) \leqslant \sum_{e \in Q} w(e)$ $\forall Q$: Q is an edge cover. Where an edge cover P is a set of edges such that $\forall v \in V$ v is incident to at least one edge in P.

In [1], the authors present a $2/3$-approximation algorithm for the maximum weight edge matching problem (MWM). MWM is stated as follows: Given a graph $G = (V, E)$, find a set of edges $M \subseteq E$ such that M is matching, meaning each edge in M is endpoint disjoint from each other, and $\sum_{e \in M} w(e) \geqslant \sum_{e \in M'} \forall M' : M'$ is matching. This algorithm is discussed in more detail in the succeeding section.

This paper extends the work of [1] by utilizing its algorithms to provide an efficient implementation of a $4/3$-approximation for MEC and reporting empirical results obtained experimentally from practical data sets. The experiment compares the $4/3$-approximation with the optimal solution, and two other approximation algorithms: Nearest Neighbors, which obtains a 2-approximation and Primal Dual, which obtains a $2/3$-approximation.

2. ALGORITHMS

2.1 A Maximum Matching Approximation

This section discusses the results from [1] as we will be using their $^2/_3$-approximation for MWM to implement our $^4/_3$-approximation for MEC. The algorithm originated from [2] as the first $^2/_3 - \epsilon$ approximation for MWM and later simplified by [3]. Before we describe the algorithm, it is best for the reader to first understand a few pieces of terminology.

If the edges of a path P alternate from M and $E \setminus M$, it is said to be **alternating**. P is considered to be an **augmentation** if the symmetric difference $M \oplus P$ is also a matching and $w(M \setminus P) > w(P \setminus M)$. If P has k edges not matching two vertices, P is called a **k-augmentation**. If P is an alternating path, then the **gain** of P is defined to be $g(P) = w(P \setminus M) - w(P \cap M)$. Finally, if P is a 2-augmentation such that all edges of P that are not included in the matching are incident to either v or its mate, P is **centered** at v. Note here that $aug(v)$ denotes a maximum-gain 2-augmentation centered at v. The algorithm, as described in Algorithm 1, takes a graph G as input and returns a matching M. It operates by first permuting the vertices. It then goes through the permuted set, and modifies the matching such that the matched vertices in $aug(v)$ become unmatched and the unmatched vertices in $aug(v)$ become matched. This is repeated for k phases. Algorithm 1 runs in $O(m \log e^{-1})$ time and achieves a $^2/_3 - \epsilon$ approximation [1].

Algorithm 1: A $^2/_3$-approximation for MWM [1].

Input: G : G is a graph, k is the number of phases

Result: M : M is a matching

```
1  begin
2      M := any matching;
3      for i = 1; i < k; i++ do
4          V = permute(V);
5          foreach v ∈ V do
6              M := M ⊕ aug(v);
7          end
8      end
9      Return M;
10 end
```

2.2 An Approximation Preserving Reduction

It is possible to obtain a minimum weight edge cover of G by transforming the weights such that $\forall e = (u, v) \in G, w(u, v) = w(\mu(u)) + w(\mu(v)) - w(u, v)$, where $\mu(t) = (s, t)$ is the edge incident to t with minimum weight $\forall t \in V$; finding a maximum weight matching using the transformed edges; then greedily expanding the matching until an edge cover is found. This algorithm is described in further detail below [4].

Algorithm 2: Minimum Weight Edge Cover through Maximum Weight Matching

Input: G : G is a graph, ALG : ALG is a function that returns a matching M

Result: C : C is an edge cover

1 **begin**

2 $\forall v \in V$, Let $\mu(v) = (u, v)$ be the edge incident to v with minimum weight;

3 $\forall v \in V$, Let $w(u, v)$ be the weight of the edge between vertices u and v;

4 $\forall e = (u, v) \in G, e = w(\mu(u)) + w(\mu(v)) - w(u, v)$;

5 $M = ALG(G)$;

6 $C := \emptyset$;

7 $\forall e = (u, v) \in M, C = C \cup e$;

8 $\forall v \in V \setminus M,\ C = C \cup \{\mu(v))\}$;

9 Return C;

10 **end**

If ALG returns a perfect matching, the above algorithm will result in a minimum weight edge cover. The correctness of this procedure is proved in [4], details of this proof are summarized in theorem 2.2.1. This procedure was proven to be approximation preserving in [5], consequently, we can substitute the exact algorithm for MWM with [1]'s $2/3$-approximation for MWM. This results in a $4/3 + \epsilon$ approximation for MEC, which is proved in theorem 2.2.2.

Theorem 2.2.1 *If ALG is an optimal MWM algorithm, then Algorithm 2 obtains a minimum weight edge cover.*

Proof: Lines 1-3 transform the weights of the input graph. There are three cases to consider under this transformation.

Case 1: $w(\mu(u)) = w(\mu(v)) = w(u, v) \iff w'(u, v) = w(u, v)$

Case 2: $w(\mu(u)) = w(u, v) > w(\mu(v)) \iff w'(u, v) = w(\mu(v)) < w(u, v)$

Case 3: $w(u,v) > w(\mu(v)) > w(\mu(u)) \iff w'(u,v) < w(\mu(u)) < w(u,v)$

There are also three additional symmetric cases.

Line 7 appends the matching to the edge cover. Line 8 greedily expands the matching to find a complete edge cover.

$\therefore$ If ALG is an optimal MWM algorithm, then Algorithm 2 obtains a minimum weight edge cover [4] $\square$

Theorem 2.2.2 *If ALG is a $2/3 - \epsilon$ approximation for MWM, then Algorithm 2 obtains a $4/3 + \epsilon$ approximation for MEC.*

Proof: The correctness of Algorithm 2 is proved in Theorem 3.1. We now prove the approximation ratio it will obtain with ALG $= 2/3 - \epsilon$ approximation.

It has been proved in [4] that if ALG is a $1 - \epsilon$ approximation for MWM, then Algorithm 2 would obtain a $1 + \epsilon$ approximation for MEC.

If ALG obtains a $2/3$-approximation, it follows,

$$
\begin{aligned}
& 1 - \epsilon = 2/3 - \delta \\
\iff & -\epsilon = -1/3 - \delta \\
\iff & \epsilon = 1/3 + \delta
\end{aligned}
$$

We have, $1 + \epsilon = 1 + 1/3 + \delta = 4/3 + \delta$

$\therefore$ If ALG is a $2/3 - \epsilon$ approximation for MWM, then Algorithm 2 obtains a $4/3 + \epsilon$ approximation for MEC $\square$

2.3 Redundant Edges

The $4/3$-approximation, along with Nearest Neighbors and Primal Dual, suffers from cases in which edges that are not required for an edge cover are included in the solution. These are referred to as redundant edges. An edge is redundant in an edge cover if we can remove the edge and maintain an edge cover.

The succeeding algorithm operates by iteratively removing edges induced by the subgraph created from over-saturated vertices. A vertex is considered to be over-saturated if it is covered by more than one edge. The algorithm iterates over each edge in the subgraph and removes the edge if both endpoints are over-saturated. This process repeats until no over-saturated vertices remain. The procedure obtains a perfect matching [4].

Algorithm 3: Removing Redundant Edges

Input: G : G is an edge cover

Result: rWeight : rWeight is the weight of the redundant edges in G

```
1  begin
2      continue := true;
3      rWeight := 0;
4      degrees := array of length n where the i^th entry refers to the degree of i^th
         vertex;
5      while continue do
6          continue = false;
7          foreach v in G do
8              foreach u adjacent to v do
9                  if degrees[u] > 1 and degrees[v] > 1 then
10                     rWeight+ = w(u, v);
11                     degrees[u]− = 1;
12                     degrees[v]− = 1;
13                     continue := true;
14             end
15         end
16     end
17     Return rWeight;
18 end
```

3. EXPERIMENTS

3.1 Preparing The Experiment

Through a reduction to MWM, we compute a minimum weight edge cover, which employs [1]'s $2/3$-approximation for MWM using one phase. We compare these results to those obtained by Nearest Neighbors (N.N.), and Primal Dual (P.D.) [6]. We draw these comparisons using LEDA's optimal implementation [7]. Table 3.1 summarizes each algorithm's approximation ratio and complexity [4]. It can be observed that the $4/3$-approximation ($4/3$) should obtain the most accurate results, while consuming the most time. It is also worth noting Nearest Neighbors should obtain the least accurate results, while consuming the least time.

Table 3.1.: Approximation ratio and complexity of each algorithm used in the experiment [4].

Algorithm	**Approx. Ratio**	**Complexity**
$4/3$	$4/3+\epsilon$	$O(m \log \epsilon^{-1})$
N.N.	2	$O(m)$
P.D.	$3/2$	$O(m \log n)$

The data used for this experiment consists of a set of sparse symmetric graphs collected from SuiteSparse: a suite of sparse matrix software. Table 3.2 shows the graphs and their associated count of vertices and edges.

Table 3.2.: Structural properties of graphs sorted in increasing order of edges.

Graph	**Vertices**	**Edges**
Fault_639	638 802	2 097 150
mouse_gene	45 101	14 461 095
Serena	1 391 349	31 570 176
bone010	986 703	35 339 811
dielFilterV3real	1 102 824	44 101 598
Flan_1565	1 564 794	57 920 625
kron_g500-logn21	2 097 152	91 040 932
hollywood-2011	2 180 759	114 492 816
G500_21	2 097 150	118 594 475
SSA_21	2 097 152	123 097 397

The experiment was conducted using Purdue University's Rice Community Cluster. Each job was submitted using 20 computing nodes. Each computing node utilizes two 10 core, 2.60GHz, Haswell CPUs and 64GB of memory. The data is collected by randomizing each graph, submitting a new job for each algorithm to run on each graph and repeating this for five iterations. Random weights assigned are strictly positive, integer values between 1 and 100. The means of these five iterations are reported in the succeeding section.

3.2 Experimental Results

Table 3.3 and Figure 1 present a calculated gap with respect to the weight obtained from the exact algorithm. This gap is calculated as $(approx - exact)/exact * 100$. The $4/3$-approximation out performs the other algorithms on each graph in our experimental data set. It is notable that on larger graphs, the $4/3$-approximation is able to obtain a near exact minimal weight edge cover.

Table 3.3.: Gap from optimal weight.

Graph	**Opt. Weight**	**Gap**		
		4/3	**P.D.**	**N.N.**
Fault_639	1 233 199.60	0.67	3.37	5.86
mouse_gene	191 847.60	0.60	1.99	3.75
Serena	2 690 749.20	0.49	3.19	5.73
bone010	1 395 886.40	0.74	3.35	5.74
dielFilterV3real	1 552 988.20	0.67	3.37	5.86
Flan_1565	2 031 408.40	0.83	6.05	3.39
kron_g500-logn21	26 557 937.20	0.00	0.12	0.18
hollywood-2011	9 403 471.80	0.11	1.88	3.68
G500_21	25 880 291.80	0.00	0.17	0.11
SSA21	8 323 301.00	0.26	2.87	4.06
Geo. Mean		0.13	1.66	2.31

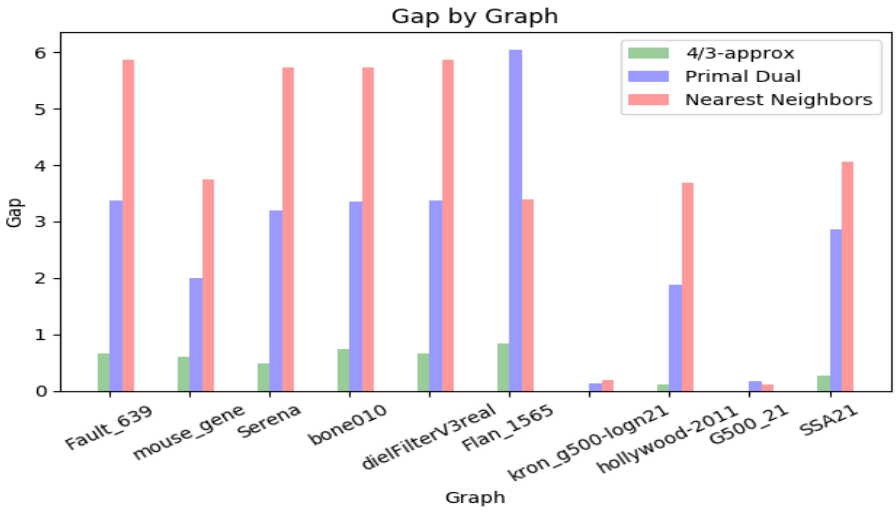

Fig. 3.1.: Gap from optimal weight.

Table 3.4 and Figure 3.2 present relative time with respect to the exact algorithm. Relative time is calculated as approx/exact. Nearest Neighbors achieves the best running times, but not significantly better than Primal Dual. The 4/3-approximation is the slowest.

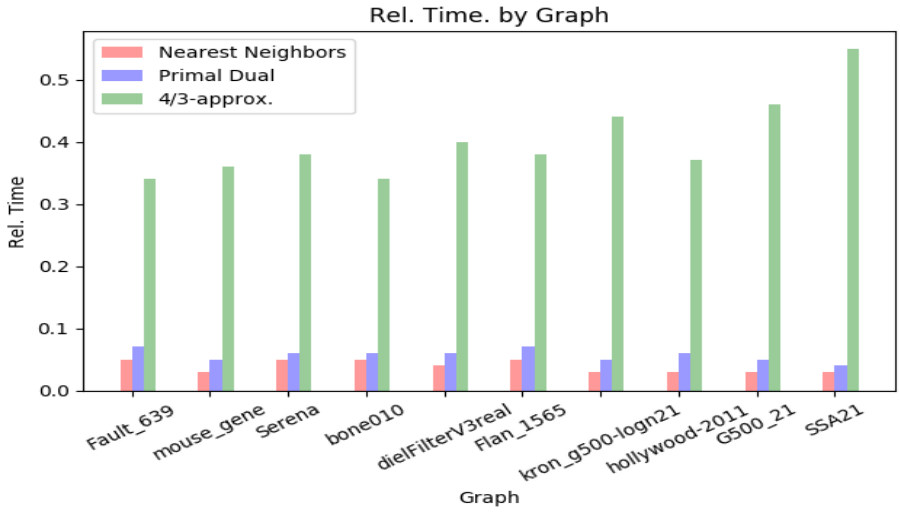

Fig. 3.2.: Relative time w.r.t exact algorithm.

Table 3.4.: Relative time w.r.t. exact algorithm.

Graph	**Exact(s)**	**Rel. Time**		
		N.N.	**P.D.**	**4/3**
Fault_639	12.22	0.05	0.07	0.34
mouse_gene	13.94	0.03	0.05	0.36
Serena	28.05	0.05	0.06	0.38
bone010	28.75	0.05	0.06	0.34
dielFilterV3real	38.79	0.04	0.06	0.40
Flan_1565	45.25	0.05	0.07	0.38
kron_g500-logn21	140.33	0.03	0.05	0.44
hollywood-2011	116.88	0.03	0.06	0.37
G500_21	176.84	0.03	0.05	0.46
SSA21	216.46	0.03	0.04	0.55
Geo. Mean		0.04	0.06	0.40

Table 3.5 presents the weight retained after redundant edges are removed. This is calculated by dividing the weight with redundant edges excluded by the weight with redundant edges included. The results are reported as a percentage. Nearest Neighbors suffers from the most redundant edges. The $4/3$-approximation encounters the least redundant edges; obtaining almost none in some cases.

Table 3.5.: Percentage of weight retained after removing redundant edges.

Graph	**Weight Retained**		
	N.N.	**P.D.**	**$4/3$**
Fault_639	92.18	96.62	99.99
mouse_gene	96.02	97.25	99.91
Serena	92.14	95.51	99.91
bone010	90.92	95.26	99.91
dielFilterV3real	91.26	94.70	99.92
Flan_1565	92.18	95.62	99.91
kron_g500-logn21	99.43	99.16	100.00
hollywood-2011	95.86	96.79	99.97
G500_21	99.45	99.21	100.00
SSA21	95.20	95.20	99.95
Geo. Mean	94.42	96.42	99.95

4. CONCLUSION

Employing the MWM algorithm of [1] and the approximation preserving results of [5], this paper has presented a 4/3-approximation algorithm for the minimum weight edge cover problem. We provide a practical implementation and empirically compare the results of this implementation against various other MEC approximations. The results suggest that the 4/3-approximation obtains edge covers with significantly smaller weights than the others, however, it also runs significantly slower. The 4/3-approximation finds the fewest redundant edges among the algorithms compared. Nonetheless, it is able to find slightly better solutions at the cost of efficiency by removing these edges. If accuracy is of greater priority than efficiency, the data collected supports the application of the 4/3-approximation as a practical algorithm.

REFERENCES

REFERENCES

[1] J. Maue and P. Sanders, "Engineering algorithms for approximate weighted matching," in *Experimental Algorithms*, C. Demetrescu, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 242–255.

[2] D. E. D. Vinkemeier and S. Hougardy, "A linear-time approximation algorithm for weighted matchings in graphs," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 107–122, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1077464.1077472

[3] S. Pettie and P. Sanders, "A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching," *Information Processing Letters*, vol. 91, no. 6, pp. 271 – 276, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019004001565

[4] A. Pothen, S. M. Ferdous, and F. Manne, "Approximation algorithms in combinatorial scientific computing," *Acta Numerica*, vol. 28, p. 541–633, 2019.

[5] D. Huang and S. Pettie, "Approximate generalized matching: f-factors and f-edge covers," *CoRR*, vol. abs/1706.05761, 2017. [Online]. Available: http://arxiv.org/abs/1706.05761

[6] S. M. Ferdous, A. Pothen, and A. Khan, *New Approximation Algorithms for Minimum Weighted Edge Cover*, pp. 97–108. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975215.10

[7] K. Mehlhorn and S. Näher, *LEDA: A Platform for Combinatorial and Geometric Computing*. New York, NY, USA: Cambridge University Press, 1999.