Efficient Reduction Rules for Calculating Near-maximum Weighted Independent Set

Jialun Shen
Ruifan Deng
Fudan University
Shanghai, China
16307110030@fudan.edu.cn
18300180053@fudan.edu.cn

ABSTRACT

The maximum independent set (MIS) problem is a famous NP-hard optimization problem in graph theory that has attracted a lot of attention. The maximum weighted independent problem (MWIS) extends the original MIS problem by taking weights on nodes into account, and has wide applications in real senarios. Some reduction rules and greedy algorithms have been proposed to compute a near-maximum weighted independent set. In this paper, we follow the reduction-and-branching strategy, propose a general reduction rule and an exact algorithm on the graph after applying simple reduction rules. Furthermore, we improve existing reduction rules by adding sorting stages, which performs well in the experiment we conduct on several real-life graphs.

PVLDB Reference Format:
Jialun Shen and Ruifan Deng. Efficient Reduction Rules for Calculating Near-maximum Weighted Independent Set. PVLDB, 14(1): XXX-XXX, 2021.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/sgallon-rin/graph-data-final.

1 INTRODUCTION

The maximum independent set (short as MIS) of a graph G is the largest vertex set S, where each two vertices are non-adjacent, in G. It is a classic NP-hard problem in graph theory that has attracted much attention [10]. However, the traditional MIS problem does not take vertex weight into consideration. Vertex weight can have very important information. In real-world graphs, the vertices do not equally contribute to the MIS.

The maximum weighted independent set (short as MWIS) of a weighted graph G is the independent set with largest total weight, which has a wide-range of practical applications in many areas such as map labeling, alignment of biological networks, wireless communication and so on [7]. In MWIS, the weight of an independent set is the sum of weights of all the vertices in S. When all the vertices are identical, the computation of MWIS is the same as the computation of MIS. As a result, computing MWIS is also a NP-hard problem and the existing algorithms designed for computing MIS can not be used to solve MWIS problem directly.

We notice that the reduction-based algorithms behave very efficiently when being applied to reduce the size of graph while the correctness of the results is preserved [3, 6]. The main idea is to determine whether the vertices are definitely in MIS. However, the reduction rule devised for unweighted graphs cannot be used to compute MWIS on weighted graphs. Based on related works, we focus on designing new reduction rules for general weighted graphs. Inspired by single-vertex reduction and two-vertex reduction [14], we propose k-vertex reduction. When no reduction rules can be applied, a heuristic strategy is adopted to delete a vertex from the current graph until one reduction rule can be used.

2 PROBLEM DEFINITION

In the following definitions, the graph G refers to an undirected vertex-weighted graph $G(V, E, \omega)$ with weight function $\omega : V \to \mathbb{R}^+$, without further specification. Let V_G denote the set of vertices in G, E_G denote the set of edges in G. In such a graph G, weight of an independent set canbe defined, which turns it into a weighted independent set. Note that we do not consider edge weights, which can be omitted when computing an independent set or a weighted independent set.

Definition 2.1: (*Independent Set*).
An independent set $IS(G)$ is a set of vertices in a graph G, no two of which are adjacent. Its size $|IS(G)|$ is the number of vertices in it.
Definition 2.2: (*Weight of an Independent Set*).
The weight of $IS(G)$, denoted by $\omega(IS(G))$, is the sum of the weights of the vertices in $IS(G)$, i.e., $\omega(IS(G)) = \sum_{u \in IS(G)} \omega(u)$.
Definition 2.3: (*Maximal Independent Set*).
A maximal independent set is an independent set that is not a proper subset of any other independent set.
Definition 2.4: (*Maximum Independent Set*).
A maximum independent set, denoted by $MIS(G)$, is an independent set of largest possible size for a given graph G.
Definition 2.5: (*Maximum Weighted Independent Set*).
Given a graph G, a maximum weighted independent set, denoted by $MWIS(G)$, is an $IS(G)$ with the largest weight.

Furthermore, note that a node with a self-loop must not be in an independent set according to the definition.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Computing the exact $MWIS(G)$ is a non-trivial task. Thus, it is usually acceptable to compute a near-optimal weighted independent set in real-world applications.

3 RELATED WORK

3.1 Exact Algorithms

Exact algorithms usually compute optimal solutions by systematically exploring the solution space. A frequently used paradigm in exact algorithms for combinatorial optimization problems is called branch-and-bound [12, 13]. For the MWIS problem, these types of algorithms compute optimal solutions by case distinctions in which vertices are either included into the current solution or excluded from it, branching into two or more subproblems and resulting in a search tree. There have been many research to improve exact branch-and-bound algorithms. These improvements include different pruning methods and complicated branching schemes with upper and lower bounds to exclude specific subtrees. Warren and Hicks [13] proposed three branch-and-bound algorithms which combine the use of weighted clique covers and a branching scheme introduced by Balas and Yu [1]. Their algorithms are able to quickly solve instances with hundreds of vertices.

In recent years, reduction rules have frequently been added into branch-and-bound methods yielding the branch-and-reduce algorithms. These algorithms are able to improve the worst case runtime of branch-and-bound algorithms by applying the reduction rules to the current graph before each branching step [6].

Specifically, two cases should be considered for a vertex u: u is included in the MWIS; or u is not included in the MWIS.
(1) In the first case, all neighbors of u cannot belong to the MWIS. The Algorithm is recursively invoked to compute the maximum weighted independent set S of $G\backslash(\{u\} \cup N(u))$, where $N(u)$ is the set of neighbors of u and $G\backslash R$ represents the graph remained by removing the vertices in the vertex set R and the incident edges. If it holds that $\omega(u) + \omega(S) > \max W$, we use $S \cup \{u\}$ to update $MWIS(G)$.
(2) In the second case, the process is recursively invoked to compute the maximum weighted independent set S of $G\backslash\{u\}$. If its weight is larger than $\max W$, the MWIS is updated with S.

3.2 Heuristic Algorithms

A widely used heuristic approach is local search, which usually computes an initial solution and then improves it by simple insertion, removal or swap operations. Although local search generally cannot guarantee the solution's quality theoretically, in practice they find high quality solutions significantly faster than exact procedures. Moreover, heuristic algorithms for the minumim weighted vertex cover problem [2, 11] can be helpful to solve the MWIS problem, as vertex cover and independent set are two complementary concepts in graphs.

3.3 Hybrid Algorithms

In order to overcome the drawbacks of both exact and inexact methods, new approaches combining reduction rules with heuristic local search algorithms were proposed [8]. A very successful approach using this paradigm is the reducing-peeling framework. The algorithm works by computing a kernel using practically efficient reduction rules in linear and near-linear time. In addition, they provide an extension of their reduction rules that is able to compute good initial solutions for the kernel. In particular, they greedily select vertices that are unlikely to be in a large independent set, then opening up the reduction space again. Thus, they are able to significantly improve the performance of the local search algorithm that is applied on the kernelized graph.

3.4 Near-Maximum WIS Computation

Inspired by the reducing-and-peeling strategy, a greedy algorithm to compute a near-maximum weighted independent set is proposed. It first applies the reduction rules to add the vertices that must be contained in the $MWIS(G)$, and deletes their neighbors from G. When no reduction rules can be applied to the current graph, we need to delete some the largest-degree vertices so that at least one reduction rule becomes available. Actually, the greedy strategy follows the heuristic: removing the vertices that are less likely to be included in the $MWIS(G)$.

The greedy strategy offers no guarantees for deleting the vertices that cannot be included in the maximum weighted independent set. Actually, the key task is designing effective reduction rules to reduce the size of the input graph. Note that the reduction rules can be also applied to reduce the search space for algorithms that compute the exact MWIS. Zheng et al. [14] proposed the following three theorems. For ease of presentation, let $\omega(S)$ denote the sum of weights of all the vertices in the set S.

Theorem 3.1: (*Single-Vertex Reduction*).
Given a vertex $u \in G$ and its neighbors $N(u)$, if $\omega(u) \geq \omega(N(u))$, u must be contained by one MWIS of G; thus the vertices in $N(u)$ can be removed, and it holds that $MWIS(G) = MWIS(G') \cup \{u\}$, where $G' = G\backslash(N(u) \cup u)$.
Theorem 3.2: (*Two-Vertex Reduction*).
Given two vertices $u, v \in G$ and their neighbors P, u and v must be contained by one MWIS of G if it holds that: (1) $\omega(u) + \omega(v) \geq \omega(P)$; and (2) $\omega(x) < \omega(N(x))$ for each vertex $x \in \{u, v\}$. Thus the vertices in P can be removed, and it holds that $MWIS(G) = MWIS(G') \cup \{u, v\}$, where $G' = G\backslash(P \cup \{u, v\})$.
Theorem 3.3:
Let G be a graph obtained after applying single-vertex reduction. Each two vertices u_1 and u_2 must share one common neighbor at least if they are added into $MWIS(G)$ by the two-vertex reduction.

Here, we provide the proof of Theorem 3.2 since it is omitted in the original paper [14].

Proof. When we consider a pair of nodes (u, v), it implies that they do not belong to S after applying single-vertex reduction. Thus, $\omega(u) < \omega(N(u))$ and $\omega(v) < \omega(N(v))$. Assume u and v do not belong to S after applying two-vertex reduction, then let P' be the MWIS in P, we have $\omega(u) + \omega(v) \geq \omega(P) \geq \omega(P')$. Therefore, we can replace P' with $\{u, v\}$. □

Since the single-vertex reduction is cheaper, it is always applied before the two-vertex reduction. Therefore, the input of two-vertex

Algorithm 1 Single-vertex Reduction with Sort

Input: A graph $G = (V_G, E_G, \omega)$
Output: S: a set of vertices in $MWIS(G)$

initialize $S \leftarrow \emptyset$
sort V_G by weight in descending order so that $\omega(u_1) \geq \cdots \geq \omega(u_n)$
for each vertex $u \in V_G$ **do**
  $\delta(u) \leftarrow 0$
  for each vertex $v \in N(u)$ **do**
    $\delta(u) \leftarrow \delta(u) + \omega(v)$
  if $\omega(u) \geq \delta(u)$ **then**
    $S \leftarrow S \cup \{u\}$
    for earh vertex $v \in N(u)$ **do**
      delete v and its adjacent edges from G
      $T \leftarrow T \cup \{v\}$
while $T \neq \emptyset$ **do**
  $T' \leftarrow T, T \leftarrow \emptyset$
  for each vertex $u \in T'$ **do**
    sort $N(u)$ by weight in descending order so that $\omega(v_1) \geq \cdots \geq \omega(v_m)$
    for each vertex $v \in N(u)$ **do**
      $\delta(v) \leftarrow \delta(v) - \omega(u)$
      if $\omega(v) \geq \delta(v)$ **then**
        $S \leftarrow S \cup \{v\}$
        for each vertex $w \in N(v)$ **do**
          $T \leftarrow T \cup \{w\}$
          delete w and its adjacent edges from G
return S

reduction is the remaining graph that is applied the single-vertex reduction before. Instead of enumerating all pairs of vertices, it just needs to examine the pairs of vertices (u_1, u_2) that share one common neighbor at least.

When no reduction rules can be applied to the current graph, we need to delete some of the largest-degree vertices so that at least one reduction rule becomes available. The greedy strategy follows the heuristic: removing the vertices that are less likely to be included in the $MWIS(G)$.

4 OUR WORK

4.1 Improvement of Reduction Algorithms

Vertices with larger weights are more likely to be in the MWIS. If they do, check them earlier may prune their neighbors earlier, so that there will be less vertices to check in V_G. Therefore, prior to applying the reduction rules, sorting the vertices by weight in descending order can possibly be helpful. However, sorting the vertices brings extra cost, which is tested in our experiment.

Based on the algorithms Zheng et al. [14] proposed, we add sort stages and propose Algorithm 1 and Algorithm 2. Like the original work, instead of checking all the vertices in each iteration, Algorithm 1 just considers the neighbors of each vertex in the set of removed vertices in the previous iteration. Algorithm 2 uses an 'unused' flag to avoid repeated visit of some vertex pairs.

Algorithm 2 Two-vertex Reduction with Sort

Input: A graph $G = (V_G, E_G, \omega)$
Output: S: a set of vertices in $MWIS(G)$

initialize $S \leftarrow \emptyset$
initialize status of each vertex as 'unused'
sort V_G by weight in descending order so that $\omega(u_1) \geq \cdots \geq \omega(u_n)$
for each vertex $u \in V_G$ **do**
  status(u) $\leftarrow$ 'used'
  $T \leftarrow \emptyset$
  for each vertex $v \in N(u)$ **do**
    for each 'unused' vertex $w \in N(v)$ **do**
      if $e(u, w) \notin E_G$ **then**
        $T \leftarrow T \cup \{w\}$
  sort T by weight in descending order so that $\omega(w_1) \geq \cdots \geq \omega(w_m)$
  for each vertex $v \in T$ **do**
    if $\omega(u) + \omega(v) \geq \omega(N(u) \cup N(v))$ **then**
      $S \leftarrow S \cup \{u, v\}$
      remove $\{u, v\}$ and $N(u) \cup N(v)$ from G
      break
return S

4.2 Generalization of the Reduction Rule

On the basis of theorem 3.1 and 3.2, we generalized it to k-vertex reduction as the following theorem:

Theorem 4.1: (*k-Vertex Reduction*).
Given a set of vertices $\{u_1, u_2, \cdots, u_k\} \in G$ and their neighbors $P = \bigcup_{i=1}^{n} N(u_i)$, $\{u_1, u_2, \cdots, u_k\}$ must be contained by a MWIS of G if it holds that: (1) $\omega(u_1) + \omega(u_2) + \cdots + \omega(u_k) \geq \omega(P)$; and (2) $\omega(x) < \omega(N(x))$ for each vertex $x \in \{u_1, u_2, \cdots, u_k\}$. Thus the vertices in P can be removed, and it holds that $MWIS(G) = MWIS(G') \cup \{u_1, u_2, \cdots, u_k\}$, where $G' = G \backslash (P \cup \{u_1, u_2, \cdots, u_k\}), k \geq 2$.

Proof. When we consider a set of k vertices $\{u_1, u_2, \cdots, u_k\}$, it implies that they do not belong to S after applying single-vertex to $(k-1)$-vertex reduction. Thus, $\omega(u_i) < \omega(N(u_i))$ for all $i \in \{1, 2, \cdots, k\}$. Assume $u_1, u_2, \cdots, u_k$ do not belong to S after applying k-vertex reduction, then let P' be the MWIS in P, we have $\sum_{i=1}^{k} \omega(u_i) \geq \omega(P) \geq \omega(P')$. Therefore, we can replace P' with $\{u_1, u_2, \cdots, u_k\}$. □

For a graph after applying $(k-1)$-vertex reduction, it has the following properties, given in theorem 4.2 and 4.3.

Theorem 4.2:
Let G be a graph obtained after applying $(k-1)$-vertex reduction. Each set of k vertices $\{u_1, u_2, \cdots, u_k\}$ must share one common neighbor at least if they are added into $MWIS(G)$ by k-vertex reduction ($k \geq 2$).

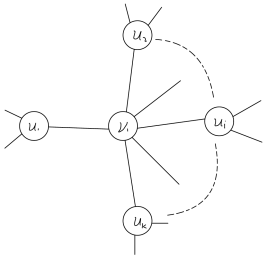

Figure 1: a common neighbor

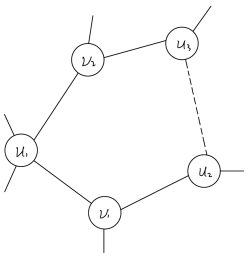

Figure 2: 2-hop neighbors

Proof. Let us consider a set of k vertices $\{u_1, u_2, \cdots, u_k\}$ that have no common neighbors, i.e., $\bigcap_{i=1}^{n} N(u_i) = \emptyset$. Since $\omega(u_i) < \omega(N(u_i))$ for all $i \in \{1, 2, \cdots, k\}$, it holds that

$$\sum_{i=1}^{n} \omega(u_i) < \sum_{i=1}^{n} \omega(N(u_i)) = \omega(P)$$

Thus the proof is achieved. □

However, if $k \geq 3$, 'used" status in Algorithm 2 may no longer be helpful. For a center vertex u_1, its 2-hop neighbours u_2 and u_3 can either be neighbors or not neighbors, as shown in Figure 2.

Theorem 4.3:
If a set of k vertices $\{u_1, u_2, \cdots, u_k\}$ is added into $MWIS(G)$ by k-vertex reduction ($k \geq 2$), then it holds that $\forall u_i$, $i \in \{1, 2, \cdots, k\}$, $\{u_1, \cdots, u_{i-1}, u_{i+1}, \cdots, u_k\}$ must be an independent set of the set $T_i = \{v | v \text{ is } u_i\text{'s 2-hop neighbor}\}$.

Proof. By the definition of an independent set, any pair of vertices (u_i, u_j) $(i \neq j)$ from $\{u_1, u_2, \cdots, u_k\}$ must not be neighbors; and by theorem 4.2, $\{u_1, u_2, \cdots, u_k\}$ share at least one common neighbor. Therefore, an arbitrary pair of vertices (u_i, u_j) $(i \neq j)$ from $\{u_1, u_2, \cdots, u_k\}$ are 2-hop neighbors of each other. Thus the proof is achieved. □

For an exact solution, we propose the optimal k-vertex reduction algorithm shown as Algorithm 3, which can be applied directly after applying single-vertex reduction. Here, k is not a parameter, but the size of a locally optimal maximal independent set that has the largest weight. Calculating a maximal independent set can be costly, but it is studied more and thus an easier problem. We propose this exact algorithm without further experiment. Instead, we propose the exact k-vertex reduction algorithm shown as Algorithm 4, where $k \geq 2$ is a parameter, as approximate solution. When $k = 2$, the algorithm reduces to Two-vertex Reduction. Exact k-vertex reduction should be applied after applying Single-vertex reduction and exact $\{2, \cdots, k-1\}$-vertex reductions, since k-vertex reduction with a smaller k is always cheaper. Instead of enumerating all pairs of vertices, it just needs to examine the set of k vertices $\{u_1, \cdots, u_k\}$ that share one common neighbor at least. Theoretically, using a larger k will lead to a more precise MWIS. When no reduction rules can be applied to the current graph, we need to delete some of the largest-degree vertices so that at least one reduction rule becomes available.

Algorithm 3 Optimal k-vertex Reduction

Input: A graph $G = (V_G, E_G, \omega)$
Output: S: a set of vertices in $MWIS(G)$

```
initialize S ← ∅
sort V_G by weight in descending order so that ω(u_1) ≥ ··· ≥ ω(u_n)
for each vertex u ∈ V_G do
    T ← ∅
    for each vertex v ∈ N(u) do
        for each vertex w ∈ N(v) do
            if e(u, w) ∉ E_G then
                T ← T ∪ {w}
    calculate the maximal independent sets: M = {M_1, ··· , M_a}, M_i ⊆ T, i = 1, ··· , a
    sort M in descending order by weight so that ω(M_1) ≥ ··· ≥ ω(M_a)
    for M_i ∈ M do
        if ω(u) + ω(M_i) ≥ ω(N(u) ∪ N(M_i)) then
            S ← S ∪ {u} ∪ M_i
            remove {u}, M_i and N(u) ∪ N(M_i) from G
            break
return S
```

Algorithm 4 Exact k-vertex Reduction

Input: A graph $G = (V_G, E_G, \omega)$, a parameter k
Output: S: a set of vertices in $MWIS(G)$

```
initialize S ← ∅
sort V_G by weight in descending order so that ω(u_1) ≥ ··· ≥ ω(u_n)
for each vertex u ∈ V_G do
    T ← ∅
    for each vertex v ∈ N(u) do
        for each vertex w ∈ N(v) do
            if e(u, w) ∉ E_G then
                T ← T ∪ {w}
    calculate independent sets of size k: IS(k) = {IS_1, ··· , IS_a}, IS_i ⊆ T, i = 1, ··· , a
    sort IS(k) in descending order by weight so that ω(IS_1) ≥ ··· ≥ ω(IS_a)
    for IS_i ∈ IS(k) do
        if ω(u) + ω(IS_i) ≥ ω(N(u) ∪ N(IS_i)) then
            S ← S ∪ {u} ∪ IS_i
            remove {u}, IS_i and N(u) ∪ N(IS_i) from G
            break
return S
```

5 EXPERIMENTAL SUPPORT

5.1 Experimental Settings

Graphs. We tested our algorithm on several graphs at different scales. The graphs are downloaded from *Stanford Network Analysis Platform* [9] (short as SNAP), *The SuiteSparse Matrix Collection* [5]

Table 1: Statistics of Graphs Used for Experiment

ID	Graph Name	|V(G)|	|E(G)|	Source	Weight
G01	sw-full-inter	110	398	ICON	original
G02	GD98_c	112	168	SSMC	random
G03	USAir97	332	2 126	SSMC	random
G04	ca-GrQC	5 242	14 496	SNAP	random
G05	ca-AstroPh	18 772	198 110	SNAP	random

(short as SSMC) and *The Index of Complex Networks* [4] (short as ICON). The self-loops are not removed. Table 1 lists the statistics.

These networks are popular benchmark instances commonly used for the maximum independent set problem. However, most of them are unweighted, and comparable weighted instances are very scarce. Therefore, a common approach in literature is to assign vertex weights uniformly at random from a fixed size interval [2, 11]. To keep our results in line with existing work [8], we assigned vertex weights uniformly at random from the interval [1, 200]. Also, we deleted all the self-loops from the graphs.

Algorithms. *DtSingle* / *DtSingleS*. Single Vertex Reduction with / without sort. *DtTwo* / *DtTwoS*. Single and Two-Vertex Reduction with / without sort. *EDtk*. Single and Exact 2 to k-Vertex Reduction. We tested $k = 3, 4, 5$.

Metrics. As the exact solution is hard to achieve, we use relative precision (rp) as metrics. Let $Alg(G)$ denote the independent set detected by algorithm Alg, and $\max Alg(G)$ denote the independent set with the largest weight detected by all the greedy algorithms.

$$rp = \frac{\omega(Alg(G))}{\omega(maxAlg(G))} \tag{1}$$

5.2 Experimental Results

We have implemented the exact recursive MWIS algorithm as *ExactMWIS*. Its results are not included below, as it is so inefficient that it even takes several minutes to run on a graph as small as GD98_c. So does the exact algorithm Algorithm 3. For *EDtk*, we tested it only on small graphs. The experiment results are shown in Table 2, Table 3 and Table 4.

From the results, we can conclude that Algorithm 1 and Algorithm 2 which add sorting stages into the original algorithms can improve the quality of the obtained weighted independent set without significant difference in running time. Algorithm 4 can get a even better solution, with some loss in efficiency, as it needs to calculate independent sets. As for the effect of k, choosing $k = 3$ can be enough, as an independent set with a larger size is rarer.

6 CONCLUSION

In this paper, we study the maximum weighted independent set problem, both its exact and approximate computation. We improve existing reduction rules by adding sorting stages. We propose and prove a new general reduction rule and algorithm, k-vertex reduction, for both exact and approximate computation of MWIS. Through experiments on real-life graphs, we show the effectiveness and correctness of the proposed methods.

ACKNOWLEDGMENTS

This work was the coursework of course *Graph Data Management and Mining* (No. DATA130045.01) at Fudan University.

REFERENCES

[1] Egon Balas and Chang Sung Yu. 1986. Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.* 15, 4 (1986), 1054–1068.

[2] Shaowei Cai, Wenying Hou, Jinkun Lin, and Yuanjie Li. 2018. Improving Local Search for Minimum Weight Vertex Cover by Dynamic Strategies. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18.*

[3] Lijun Chang, Wei Li, and Wenjie Zhang. 2017. Computing a near-maximum independent set in linear time by reducing-peeling. In *Proceedings of the 2017 ACM International Conference on Management of Data.* 1181–1196.

[4] Aaron Clauset, Ellen Tucker, and Matthias Sainz. 2016. The Colorado Index of Complex Networks. https://icon.colorado.edu/.

[5] Timothy A. Davis and Yifan Hu. 2011. The university of Florida sparse matrix collection. *Acm Transactions on Mathematical Software* 38, 1 (2011), 1–25.

[6] Fedor V Fomin, Fabrizio Grandoni, and Dieter Kratsch. 2009. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM (JACM)* 56, 5 (2009), 1–32.

[7] Alexander Gellner, Sebastian Lamm, Christian Schulz, Darren Strash, and Bogdán Zaválnij. 2021. Boosting Data Reduction for the Maximum Weight Independent Set Problem Using Increasing Transformations. In *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX).* SIAM, 128–142.

[8] Sebastian Lamm, Christian Schulz, Darren Strash, Robert Williger, and Huashuo Zhang. 2019. Exactly Solving the Maximum Weight Independent Set Problem on Large Real-World Graphs. In *Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments, ALENEX 2019.* SIAM, 144–158. https://doi.org/10.1137/1.9781611975499.12

[9] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[10] Harry R Lewis. 1983. Computers and intractability. A guide to the theory of NP-completeness.

[11] Yuanjie Li, Shaowei Cai, and Wenying Hou. 2017. An Efficient Local Search Algorithm for Minimum Weighted Vertex Cover on Massive Graphs. In *Asia-Pacific Conference on Simulated Evolution and Learning.*

[12] Patric RJ Östergård. 2002. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics* 120, 1-3 (2002), 197–207.

[13] Jeffrey S Warren and Illya V Hicks. 2006. Combinatorial branch-and-bound for the maximum weight independent set problem. *Relatório Técnico, Texas A&M University, Citeseer* 9 (2006), 17.

[14] W. Zheng, J. Gu, P. Peng, and J. X. Yu. 2020. Efficient Weighted Independent Set Computation over Large Graphs, In 2020 IEEE 36th International Conference on Data Engineering (ICDE). *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 1970–1973. https://doi.org/10.1109/ICDE48307.2020.00216

Table 2: Experiment Result - Single Vertex Reduction

Graph	*DtSingle*				*DtSingleS*			
	size	weight	time (s)	rp	size	weight	time (s)	rp
sw-full-inter	1	8	**0.0025**	**1**	1	8	0.0030	**1**
GD98_c	9	1 485	0.0015	**0.439**	9	1 450	**0.0013**	0.428
USAir97	63	7 941	**0.0108**	0.6141	85	9 805	0.0122	**0.7583**
ca-GrQC	953	125 932	**0.1027**	0.7294	1 047	131 071	0.1084	**0.7591**
ca-AstroPh	1 197	158 285	**1.3402**	0.9807	1 272	161 407	1.4132	**1**

Table 3: Experiment Result - Two Vertex Reduction

Graph	*DtTwo*				*DtTwoS*			
	size	weight	time (s)	rp	size	weight	time (s)	rp
sw-full-inter	1	8	**0.1198**	**1**	1	8	0.1256	**1**
GD98_c	21	3 185	0.0039	0.9423	23	3 380	**0.0026**	**1**
USAir97	91	11 127	0.3573	0.8605	111	12 931	**0.2847**	**1**
ca-GrQC	1 269	164 298	**1.4322**	0.9516	1 363	172 656	1.6521	**1**

Table 4: Experiment Result - k Vertex Reduction

Graph	*EDtk (k=3)*				*EDtk (k=4)*				EDtk (k=5)			
	size	weight	time (s)	rp	size	weight	time (s)	rp	size	weight	time (s)	rp
GD98_c	34	5 155	0.0111	**1**	34	5 155	0.0146	**1**	34	5 155	0.0203	**1**
USAir97	133	16 702	16.665	**1**	133	16 702	16.758	**1**	133	16 702	16.853	**1**