Towards Computing a Near-Maximum Weighted Independent Set on Massive Graphs

Jiewei Gu
Fudan University, China
gujw20@fudan.edu.cn

Weiguo Zheng*
Fudan University, China
zhengweiguo@fudan.edu.cn

Yuzheng Cai
Fudan University, China
yzcai17@fudan.edu.cn

Peng Peng
Hunan University, China
hnu16pp@hnu.edu.cn

ABSTRACT

The vertices in many graphs are weighted unequally in real scenarios, but the previous studies on the maximum independent set (MIS) ignore the weights of vertices. Therefore, the weight of an MIS may not necessarily be the largest. In this paper, we study the problem of maximum weighted independent set (MWIS) that is defined as the set of independent vertices with the largest weight. Since it is intractable to deliver the exact solution for large graphs, we design a reducing and tie-breaking framework to compute a near-maximum weighted independent set. The reduction rules are critical to reduce the search space for both exact and greedy algorithms as they determine the vertices that are definitely (or not) in the MWIS while preserving the correctness of solutions. We devise a set of novel reductions including low-degree reductions and high-degree reductions for general weighted graphs. Extensive experimental studies over real graphs confirm that our proposed method outperforms the state-of-the-arts significantly in terms of both effectiveness and efficiency.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Data mining**.

KEYWORDS

Weighted independent set; Reduction rule; Low-degree reduction; High-degree reduction

ACM Reference Format:
Jiewei Gu, Weiguo Zheng, Yuzheng Cai, and Peng Peng. 2021. Towards Computing a Near-Maximum Weighted Independent Set on Massive Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467232

*Weiguo Zheng is the corresponding author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467232

1 INTRODUCTION

As a classic NP-hard problem in graph theory, the maximum independent set (abbreviated as MIS) problem is to find a vertex set S with the largest cardinality such that each two nodes are not adjacent in G [14]. As discussed in existing work [9, 28], the MIS problem has a wide range of applications, such as collusion detection [3], association rule mining [37], road network routing [24], automated map labeling [15], macromolecular docking [13] and wireless networks [20, 31, 34]. However, computing an MIS ignoring the weights on vertices may result in the loss of some useful information on the nodes. Generally, the nodes are not necessary to contribute to the MIS equally in many real-world scenarios. Let us consider the following motivating examples.

1.1 Motivating Example

Image segmentation and multiobject tracking are fundamental problems in computer vision, attracting great efforts in both academic and industrial communities. Methods based on the maximum weight independent set have been proposed to address such problems [4, 5]. Take image segmentation for example, given a collection of segments extracted from low-level segmenters, the goal is to seek to pick out distinct segments that together partition the image area. Having obtained these segments, a weighted graph is built where nodes represent segments and edges connect nodes sharing overlapped segments in the image. Each node is assigned a weight to capture the distinctiveness of corresponding segments [5]. Hence, the image segmentation task can be performed by finding the maximum weighted independent set of the weighted graph.

The maximum weighted independent set can be applied to schedule wireless networks as well [21, 33]. In the classical network interference model, adjacent transmitters cannot transmit information simultaneously in case of data corruption. By nature, wireless networks can be deemed as weighted graphs, taking transmitters as nodes and transmission channels as edges. The weight of each node represents the throughput of the corresponding transmitter. Therefore, globally maximizing the throughput of entire networks falls into finding a maximum weight independent set.

It is clear that the maximum weighted independent set (abbreviated as MWIS), i.e., the independent set with the largest weight, is very useful in many applications, where the weight of an independent set S is the sum of weights of the vertices in S. Let us consider the graph in Figure 1. The grey vertices v_1, v_4, and v_{13} constitute the maximum weighted independent set with the weight

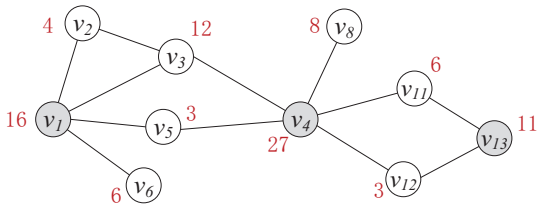

Figure 1: Example of maximum weighted independent set.

$\omega_1 = 16+27+11 = 54$. One traditional maximum independent set is $\{v_3, v_5, v_6, v_8, v_{11}, v_{12}\}$ whose weight is 38. Although the traditional MIS consists of more vertices, its weight is not larger than that of the maximum weighted independent set.

1.2 Previous Algorithms on MIS and MWIS

There have been many efforts devoted to the MIS problem, which can be categorized into three groups, i.e., exact algorithms [1, 10, 12, 36], approximate algorithms [11, 17, 29], and greedy algorithms [2, 9, 16, 23, 25, 28].

Exact algorithms. As it is NP-hard to compute the MIS, the branch-and-reduce framework is widely used in the existing methods [1, 10, 12]. The graph size is reduced by determining the vertices that are definitely (or not) in the MIS based on the reduction rules. Due to the NP-hardness of the MIS problem, several algorithms try to reduce the base of the exponent by designing different branching rules, e.g., $O(1.2201^n)$ [12] and $O(1.1996^n n^{O(1)})$ [36], where $n = |V_G|$ is the number of vertices in graph G. As a state-of-the-art algorithm, Akiba et al. propose an efficient algorithm for the vertex cover problem following the branch-and-reduce paradigm [1], where a minimum vertex cover of the graph G equals the set of remaining vertices by removing the MIS vertices.

Approximate algorithms. Although there does not exist a constant approximation factor for general graphs [18, 32], some efforts have been devoted to deriving approximation ratios [11, 17, 29]. For example, Uriel Feige proposes an algorithm that approximates the maximum clique within a factor of $O(n(\log\log n)^2/(\log n)^3)$. As shown in the work [19], it is NP-hard to compute the *MIS* with the approximation ratio $n^{1-\epsilon}$ or less.

Greedy algorithms. Instead of computing the exact MIS or approximation ratios, a large bunch of algorithms deliver the high-quality (as large as possible, may not be the largest) independent set [2, 16, 23, 25, 28]. They usually apply some heuristic strategies, e.g., vertex swap, to find the local optimal solution [16, 28]. Dahlum et al. propose to perform simple kernelization techniques in an online fashion and find that cutting the high-degree vertices can boost local search performance [10]. Following the intuition that the high-degree vertices are less likely to be included in a maximum independent set [10], Chang et al. remove the vertex with the highest degree when no reduction rules can be applied [9]. Recently, computing the MIS over evolving graphs has also attracted increasing efforts, where the graphs may be dynamically changing by adding/removing vertices/edges [39].

Note that there is a stream of research on the maximum clique problem that finds the largest set of vertices where each of two vertices has an edge [8, 35]. Although computing the *MIS* of a graph equals finding the maximum clique in its complementary graph with edges between all non-neighbors in the original graph, the algorithms designed for maximum cliques are not efficient enough to solve the *MIS* problem as complementary graph of real graphs may be very dense. For example, the graph AstroPh downloaded from SNAP [27] consists of 18,771 vertices and 198,050 edges. Correspondingly, its complementary graph contains 175,967,785 edges.

Existing algorithms on MWIS. As shown in Figure 1, the MIS is not necessary to be the maximum weighted independent set. Actually, the MWIS problem generalizes the MIS problem since when the weights on all the vertices are identical computing the MWIS falls into computing MIS. Hence, it is very expensive to compute the MWIS as well. Similar to computing MIS, reduction rules are desired to reduce the graph for both exact and greedy algorithms [26, 38]. Beyond that, Kako et al. introduce the weighted average degree and weighted inductiveness, and study their effect on MWIS algorithms [22].

1.3 Our Approach and Contributions

Generally, the reduction-based algorithms generate a kernel graph G' by applying the reduction rules to the graph G, where the kernel graph G' may be significantly smaller than G while preserving the correctness of the results. The fundamental principle is determining whether the vertices are definitely (or not) included in an MWIS. Although several reduction rules have been designed, e.g., two-vertex reduction [38] and isolated vertex removal [26], we argue that these reductions may not be cost-effective enough since (1) it takes time to determine whether a reduction can be applied to the graph; (2) the graph is not guaranteed to be reduced after expensive checking. For example, the isolated vertex removal requires computing the cliques of G, which is extremely time-consuming especially when the graph is very large. In contrast, the degree-one reduction adds the vertex u to MIS only if u has just one neighbor [12]. It is efficient to use, but it is particularly designed for unweighted graphs (or the equally weighted graphs). Thus, it cannot be applied to computing the MWIS on general weighted graphs. For instance, the vertices v_6 and v_8 in Figure 1 can be included in an MIS according to the degree-one reduction, but they do not belong to the MWIS.

Therefore, we focus on designing new reduction rules for general weighted graphs. Specifically, we propose two kinds of reduction rules, i.e., low-degree reductions and high-degree reductions. The intuition is that low-degree vertices form a vast majority in most real graphs that follow the power-law degree distribution. Thus, our proposed low-degree reductions generalize the unweighted degree-one reduction and unweighted degree-two reduction to a general weighted graph (Section 3). More importantly, the proposed low-degree reductions exhibit a good nature of completeness, that is, the reduced graph G' will not contain any vertices with degree less than three. Furthermore, low-degree reductions are easy to compute as they just need to consider degree-one and degree-two vertices. To further reduce the graph, we devise two high-degree reductions (Section 4). We also prove that the proposed single-edge reductions have a larger pruning power than the weighted domination reduction [26]. When no reduction rules can be applied, a heuristical strategy is adopted to delete a vertex from the current graph until one reduction rule can be used. To that end, we present a systematic tie-breaking framework.

In summary, we make the following contributions.

- In order to compute a near-MWIS efficiently, we propose a systematic tie-breaking framework that can incorporate different policies.
- We devise efficient low-degree reductions which exhibit a good nature of completeness for general weighted graphs.
- To enhance the pruning ability, we propose a set of high-degree reductions including the single-edge reduction and extended single-edge reduction.
- We evaluate the proposed methods through extensive experiments on a bunch of graphs to confirm the effectiveness and efficiency.

2 PRELIMINARIES AND FRAMEWORK

We focus on undirected weighted graphs in the paper. An undirected weighted graph G consists of tuples (V_G, E_G, ω), where V_G is the set of vertices, E_G is the set of edges, and ω is the function that assigns the weight $\omega(u)$ for the vertex $u \in V_G$. A subgraph G_S induced by a set of vertices S is the subgraph of G such that $e(u,v) \in G_S$ iff $e(u,v) \in G$ for any two vertices u and v in S. Let $N(u)$ denote the set of neighbors of u and $N[u] = N(u) \cup \{u\}$.

2.1 Problem Definition

An independent set of a graph G, denoted by $IS(G)$, is a set of vertices in which any two vertices have no edge in G. A maximal independent set of a graph G is the independent set that is not contained in any other independent set. A maximum independent set of a graph G, denoted by $MIS(G)$, is a largest independent set of G. Since each vertex is assigned a weight, we can compute the weight of an independent set.

Definition 2.1. *(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 2.2. *(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. Let $\lambda(G)$ denote the weight of a maximum independent set of G.*

Example 1. *Let us consider the graph in Figure 1. Its maximum weighted independent set consists of vertices v_1, v_4, and v_{13}. The maximum independent set is $\{v_3, v_5, v_6, v_8, v_{11}, v_{12}\}$.*

Clearly, the maximum weighted independent set $MWIS(G)$ of a graph G must be a maximal independent set since it will not be the maximum one if it is not maximal. However, the maximum independent set may not be the maximum weighted independent set. Specially, when the weights of all the vertices in V_G are identical to each other, the maximum independent set equals the maximum weighted independent set.

As proved in [38], it is NP-hard to compute $MWIS(G)$ for a graph G. Nevertheless, it is acceptable to return a near-optimal weighted independent set in real applications. Hence, we design a general framework for computing a near-maximum weighted independent set in the following sections.

Problem Statement 1. *The task of the paper is to find one maximum or near-maximum weighted independent set of G.*

2.2 Previous Reduction Rules

Reducing-and-peeling is a state-of-the-art strategy to compute a high-quality independent set [9]. The main idea is to reduce the graph by employing some reduction rules and perform the peeling operation (i.e., delete the largest-degree vertex) when no reduction rules can be applied. The process proceeds iteratively until the graph is empty. To make it self-contained, we include several reduction rules devised for computing a maximum weighted independent set. Let "$A \setminus B$" denote the vertices in set A but not in set B.

Theorem 2.1. *(Single-vertex Reduction [38]) Given a vertex $u \in G$ and its neighbors $N(u)$, if $\omega(u) \geq \omega(N(u))$, u must be contained in one MWIS of G; thus the vertices in $N(u)$ can be removed, and it holds that $MWIS(G) = MWIS(G') \cup \{u\}$, where $G' = G \setminus (N(u) \cup u)$.*

Theorem 2.2. *(Two-vertex Reduction [38]) Given two non-adjacent vertices $u, v \in G$ and their neighbors P, u and v must be contained in 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 $x \in G$. Thus the vertices in P can be removed, and it holds that $MWIS(G) = MWIS(G') \cup \{u, v\}$, where $G' = G \setminus (P \cup \{u, v\})$.*

Theorem 2.3. *(Neighbor Removal [26]) Let $v \in V_G$. For any $u \in N(v)$, if $\lambda(G[N(v) \setminus N[u]]) + \omega(u) \leq \omega(v)$, then u can be removed from G, as there is some $MWIS(G)$ that excludes u, and $\lambda(G) = \lambda(G[V \setminus \{u\}])$.*

Theorem 2.4. *(Weighted Domination [26]) Let $u, v \in V_G$ be vertices such that $N[u] \supseteq N[v]$. If $\omega(u) \leq \omega(v)$, there is an MWIS in G that excludes u and $\lambda(G) = \lambda(G[V \setminus \{u\}])$. Therefore, u can be removed from G.*

As discussed in [26], neighbor removal reduction is very expensive since it has to compute a maximum independent set of a subgraph. Thus, several simplified reductions, e.g., single-vertex reduction, two-vertex reduction and domination reduction, have been proposed. However, we argue that they are not cost-effective enough since determining whether reductions can be applied takes time itself and they are not guaranteed to be applicable.

2.3 Near-Maximum WIS Computation

We propose a general framework of computing a near-maximum weighted independent set as shown in Algorithm 1. The framework consists of two phases, i.e., the tie-breaking phase (lines 2-4) and the reducing phase (lines 5-6). In the reducing phase, it applies the reduction rules to add the vertices (also called target 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 break the tie by deleting some vertices so that at least one reduction rule becomes available. Actually, the tie-breaking operation follows the heuristic removing the vertices that are less likely to be included in the $MWIS(G)$.

Tie-breaking policies. Note that there may be multiple strategies of choosing the vertices to delete in the tie-breaking phase. We devise three strategies in the paper.

(1) Degree-oriented: Delete the vertex with the largest degree. The intuition is that deleting the largest-degree vertices is more likely to generate more available reductions.

Algorithm 1: Reducing and Tie-breaking Framework

Input : A graph $G = (V_G, E_G, \omega)$
Output: A near-maximum weighted independent set $NWIS(G)$

```
while G is not empty do
    if there is no reduction to use then
        choose a vertex u from V_G by a tie-breaking policy;
        delete u and the neighbors of u from G;
    else
        apply reductions;
        // add target vertices into NWIS(G) and delete their neighbors
return NWIS(G)
```

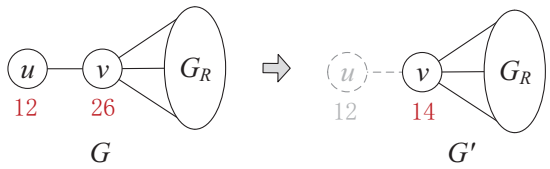

Figure 2: Example of degree-one reduction.

(2) Weight-oriented: Delete the vertex with the smallest weight. The intuition is that the smallest-weight vertex is likely to contribute the least to *MWIS(G)*.
(3) Hybrid: Delete the vertex u with the largest $\phi(u)$, where $\phi(u)$ is the weight difference between u's neighbors $N(u)$ and u, i.e., $\phi(u) = \sum_{v \in N(u)} \omega(v) - \omega(u)$.

These greedy strategies are not guaranteed to delete the vertices that cannot be included in the $MWIS(G)$. Hence, the delivered solution may not be the exact $MWIS(G)$. Actually, the key of the framework 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. Thus we focus on the reduction rules in the next sections.

3 LOW-DEGREE REDUCTIONS

The goal of reductions is to determine the vertices that are definitely included or excluded in the $MWIS(G)$. In this section, we devise low-degree reductions including degree-one reduction (Section 3.1) and degree-two reduction (Section 3.2).

3.1 Degree-one Reduction

For ease of presentation, let $\omega(S)$ denote the sum of weights of all the vertices in the set S, i.e., $\omega(S) = \sum_{v \in S} \omega(v)$. Let $|P|$ denote the size of a vertex set P, i.e., the number of vertices in P. The basic idea of degree-one reduction is: analyze each degree-one vertex u and compare its weight with the weight of its neighbor v. The aim is reducing the graph G by removing or including u or v without sacrificing the optimality of the maximum weighted independent set. Formally, we can derive the degree-one reduction.

Theorem 3.1. ***(Degree-one Reduction)*** *Let u be a degree-one vertex with the neighbor v in G.*

- *Case 1: if $\omega(u) \geq \omega(v)$, u must be contained in one MWIS of G; thus v can be removed from G, i.e., $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by removing both u and v;*
- *Case 2: if $\omega(u) < \omega(v)$, $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by removing u and updating the weight of v as $\omega(v) = \omega(v) - \omega(u)$. It holds that $v \in MWIS(G)$ iff $v \in MWIS(G')$.*

Proof. Please refer to Appendix A for details. □

Clearly, the graph G can be reduced if it contains degree-one vertices regardless of whether their weights are larger than their neighbors or not.

Example 2. *Let us consider the graph G in Figure 2. u and v are adjacent vertices, while the other vertices of G and their inner edges are denoted as G_R for simplicity. Since the degree of u is one and $\omega(u) < \omega(v)$, it falls into Case 2 in degree-one reduction (Theorem 3.1). We can safely remove u and update the weight of v as $\omega(v) = \omega(v) - \omega(u) = 14$ to obtain G'.*

The proposed degree-one reduction generalizes the previously classical unweighted degree-one reduction devised in [12].

Theorem 3.2. *The unweighted degree-one reduction proposed in [12] is a special case of degree-one reduction when $\omega(u)$ equals $\omega(v)$ for any two vertices u and v in V_G.*

Proof. Assume that $\omega(u) = \omega(v)$ for any two vertices u and v. When u has only one neighbor w, u must be contained in $MIS(G)$, which accords with Theorem 3.1 as $\omega(u) \geq \omega(N(u)) = \omega(w)$. □

Algorithm 2 presents the details of degree-one reduction procedure. Lines 3-5 iterate each degree-one vertex u, and invoke function *Degree-one Reduction* to compare $\omega(u)$ with $\omega(v)$, where v is the only neighbor of u. Since we remove some edges during iteration, the set of degree-one vertices $V_{=1}$ should be updated. Instead of checking all the vertices to update $V_{=1}$, we just need to consider the neighbors of deleted vertices in each iteration (line 11 and 14).

Algorithm 2: Degree-one Reduction

Input : A graph $G = (V_G, E_G, \omega)$
Output: A reduced graph $G' = (V_{G'}, E_{G'}, \omega)$

```
1  G' ← G
2  Let V_{=1} be the set of degree-one vertices
3  while V_{=1} ≠ ∅ do
4      Extract a vertex u from V_{=1}
5      Degree-one Reduction(u)
6  return G'
7  Function Degree-one Reduction (u)
8  v ← the only neighbor of u
9  Remove u and edge e(u, v)
10 if ω(u) ≥ ω(v) then
11     Remove v and its edges, update V_{=1}
12 else
13     ω(v) ← ω(v) − ω(u)
14     Check degree of v, update V_{=1}
```

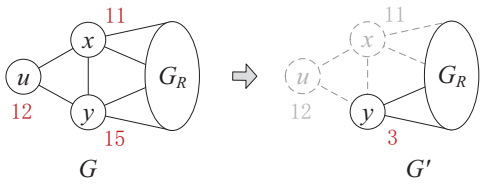

Figure 3: Example of triangle reduction.

Time complexity. In the iterations of Algorithm 2 (lines 3-5), each vertex $u \in V_{=1}$ is visited once, and neighbors of v are processed in line 11. Since each edge can only be removed once, the overall time cost is $O(|V_G| + |E_G|)$.

3.2 Degree-two Reduction

When the graph G has no degree-one vertices anymore, we consider the degree-two vertices and compare them with their neighbors.

Theorem 3.3. ***(Triangle Reduction)*** *Let u be a degree-two vertex with two neighbors x and y in G, where edge $e(x,y) \in E_G$. $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by (1) removing u, (2) removing $w \in \{x,y\}$ s.t. $\omega(w) \leq \omega(u)$, and (3) updating $\omega(w) = \omega(w) - \omega(u)$ s.t. $\omega(w) > \omega(u)$. And it holds that $w \in MWIS(G)$ iff $w \in MWIS(G')$, where $w \in \{x,y\}$.*

Proof. Please refer to Appendix A for details. □

Example 3. *Let us consider the vertex u and its adjacent neighbors x and y in Figure 3. The other vertices of G and their inner edges are denoted as G_R for simplicity. Since the degree of u is two, $e(x,y) \in E_G$ and $\omega(x) < \omega(u) < \omega(y)$, according to triangle reduction (Theorem 3.3), we can safely remove u and x and update the weight of y as $\omega(y) = \omega(y) - \omega(u) = 3$ to obtain G'.*

Generally, a degree-two vertex u is not guaranteed to belong to a triangle, that is, the two neighbors may not be adjacent.

Lemma 3.1. ***(Accompany rule)*** *Given two non-adjacent vertices u and v in graph G, if $N(v) \subseteq N(u)$, u is accompanied by v when u is contained in an MWIS of G.*

Proof. If u exists in an MWIS, we can remove its neighbors from graph G, leading v to being an independent vertex. Since independent vertices naturally belong to valid members of MWIS solutions, this lemma holds. □

This rule is straightforward, but it lays theoretical foundations for our diverse graph transformations below.

Theorem 3.4. ***(V-shape Reduction)*** *Let u be a degree-two vertex with two neighbors x and y in G, where the edge $e(x,y) \notin E_G$. Without loss of generality, assume $\omega(x) \leq \omega(y)$.*

- *Case 1: if $\omega(u) \geq \omega(y)$, $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by removing u, and contracting x and y into a super node v such that $N(v) = N(x) \cup N(y)$ and $\omega(v) = \omega(x) + \omega(y) - \omega(u)$. It holds that $\{x,y\} \subseteq MWIS(G)$ iff $v \in MWIS(G')$;*
- *Case 2: if $\omega(x) \leq \omega(u) < \omega(y)$, $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by removing u, updating $N(x) = N(x) \cup N(y)$ and $\omega(y) = \omega(y) - \omega(u)$. It holds that $\forall w \in \{x,y\}$, $w \in MWIS(G)$ iff $w \in MWIS(G')$;*
- *Case 3: if $\omega(x) > \omega(u)$, $\lambda(G) = \lambda(G') + \omega(u)$, where G' is the graph obtained by updating $\omega(x) = \omega(x) - \omega(u)$, $\omega(y) = \omega(y) - \omega(u)$, and $N(u) = N(x) \cup N(y)$. It holds that $\{x,y\} \subseteq MWIS(G)$ iff $\{x,y\} \subseteq MWIS(G')$.*

Proof. Please refer to Appendix A for details. □

Example 4. *Let us consider the graphs in Figure 4. Vertices u, x and y are local neighbors, while the other vertices of G and their inner edges are denoted as G_R for simplicity. Since the degree of u is two and $e(x,y) \notin E_G$ w.r.t. each graph G in Figure 4, we can apply V-shape reduction (Theorem 3.4).*

In graph G of Figure 4(a), $\omega(x) < \omega(y) < \omega(u)$ indicates that it falls into case 1. We can safely remove u, then contract x and y into a super node v such that $N(v) = N(x) \cup N(y)$ and $\omega(v) = \omega(x) + \omega(y) - \omega(u) = 8$ to obtain G'.

In graph G of Figure 4(b), $\omega(x) < \omega(u) < \omega(y)$ indicates that it falls into case 2. To obtain G', u should be removed, while $N(y) = N(y) \cup N(x)$ and $\omega(y) = \omega(y) - \omega(u) = 3$.

In graph G of Figure 4(c), $\omega(u) < \omega(x) < \omega(y)$ indicates that it falls into case 3. We should update $\omega(x) = \omega(x) - \omega(u) = 3$, $\omega(y) = \omega(y) - \omega(u) = 5$, and $N(u) = N(x) \cup N(y)$ to obtain G'.

Actually, case 1 in V-shape reduction may contain a special instance that $\omega(u) > \omega(x) + \omega(y)$. Thus the super node v cannot be contained in the maximum independent set since the weight of v is negative, which corresponds to a special instance of single-vertex reduction [38]. Algorithm 4 (in Appendix B) presents the details of performing the degree-two reduction.

Theorem 3.5. *The time complexity of merging edges of V-shape reduction is $min\{O(K|V_G|), O(K^2\Delta)\}$, where K is the number of processed degree-two vertices and Δ is the largest degree in G.*

Proof. In the first iteration, merging edges of x and y costs $O(2\Delta)$, and the merged one has degree within $(0, 2\Delta]$. In the second iteration, time cost is $O(2\Delta + \Delta) = O(3\Delta)$, and the merged one has degree within $(0, 3\Delta]$. Similarly, we can derive the time complexity of each iteration, and the total cost can be $O(\sum_{k=2}^{K+1} k\Delta) = O(K^2\Delta)$. Moreover, since $\Delta < |V_G|$, the total cost can also be $O(K|V_G|)$. Hence, the overall time complexity is $min\{O(K|V_G|), O(K^2\Delta)\}$. □

Time complexity. In the iterations of Algorithm 4 (lines 3-13), each vertex u s.t. $u \in V_{=1} \vee u \in V_{=2}$ is visited once, and edges of x and y may be processed in function *Triangle Reduction*. In function *V-shape Reduction*, merging edges of x and y takes the cost $min\{O(K|V_G|), O(K^2\Delta)\}$ according to Theorem 3.5. Hence, the overall time cost is $min\{O(K|V_G| + |E_G|), O(K^2\Delta + |V_G| + |E_G|)\}$.

After performing degree-one and degree-two reductions, all vertices with degree less than three can be reduced. Note that most networks in real applications follow the power-law degree distribution, which indicates low-degree vertices account for the sizeable majority. Hence, lots of vertices will be reduced, and our experiments on real-world graphs show that nearly 70% vertices are reduced on average with help of degree-one and degree-two reductions.

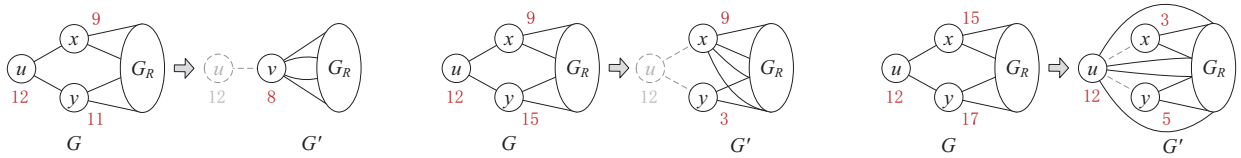

Figure 4: Examples of V-shape reduction.

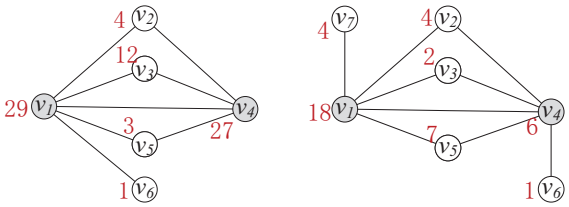

Figure 5: Single-edge reductions.

4 HIGH-DEGREE REDUCTIONS

In this section, we further devise some reduction rules to prune some vertices with degree greater than two. These reduction rules can be used to further reduce the graph.

4.1 Single-Edge Reduction

First, let us consider an edge $e(u, v)$. If the weight of u is larger than the sum of weights of v and u's neighbors that are not adjacent to v, it indicates that there must be one $MWIS(G)$ excluding v. This is because adding u into a maximum weighted independent set can be more beneficial than including v. Therefore, v can be removed safely.

Theorem 4.1. ***(Basic Single-edge Reduction)*** *Given an edge $e(u, v) \in E_G$, if $\omega(v) + \omega(N(u) \setminus N(v)) \leq \omega(u)$, it holds that $\lambda(G) = \lambda(G')$, where G' is obtained by removing v from G.*

Proof. If one $MWIS(G)$ contains v, we can replace v with u. After this replacement, we also remove all vertices in both $MWIS(G)$ and $N(u) \setminus N(v)$. Then, we obtain a new set of vertices, $MWIS'(G)$. It is obvious that $MWIS'(G)$ is also an independent set. Because $\omega(v) + \omega(N(u) \setminus N(v)) \leq \omega(u)$, the weight of $MWIS'(G)$ is larger than the weight of $MWIS(G)$. This is conflict with the premise that $MWIS(G)$ is a maximum weighted independent set. □

Example 5. *Let us consider the the edge $e(v_1, v_4)$ in Figure 5(a). There is only one vertex v_6 that is adjacent to v_1 but not adjacent to v_4, while $w(v_4) + w(v_6) = 27 + 1 < w(v_1) = 29$. Thus, v_4 can be removed safely.*

Actually, the proposed basic single-edge reduction above generalizes weighted domination reduction that is devised in [26].

Theorem 4.2. *The weighted domination proposed in [26] is a special case of the basic single-edge reduction.*

Proof. Given an edge $e(u, v) \in E_G$, the weighted domination indicates that there exists a maximum weighted independent set excluding v if $w(v) \leq w(u) \cap N(u) \subseteq N(v)$. It can be easily derived that $N(u) \subseteq N(v)$ is equivalent to $N(u) \setminus N(v) = \emptyset$. Therefore, single-edge reduction was reduced to weighted domination when $N(u) \setminus N(v) = \emptyset$, indicating that the latter is just a special case of the former. □

Algorithm 3: Single-edge Reduction

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

$S \leftarrow \emptyset$
for *each edge* $(u, v) \in E_G$ **do**
  $\delta_1(u) \leftarrow 0, \delta_1(v) \leftarrow 0, \delta_2(u) \leftarrow 0, \delta_2(v) \leftarrow 0$
  for *each vertex* $w \in N(u)$ **do**
    if $w \notin N(v)$ **then**
      $\delta_1(u) \leftarrow \delta_1(u) + \omega(w)$
    $\delta_2(u) \leftarrow \delta_2(u) + \omega(w)$
  for *each vertex* $w \in N(v)$ **do**
    if $w \notin N(u)$ **then**
      $\delta_1(v) \leftarrow \delta_1(v) + \omega(w)$
    $\delta_2(v) \leftarrow \delta_2(v) + \omega(w)$
  $\delta_2(u) \leftarrow \delta_2(u) - \omega(v), \delta_2(v) \leftarrow \delta_2(v) - \omega(u)$
  if $\omega(u) \geq \delta_1(u) + \omega(v)$ **then**
    delete v and its adjacent edges from G
    $T \leftarrow T \cup \{v\}$
  if $\omega(v) \geq \delta_1(v) + \omega(u)$ **then**
    delete u and its adjacent edges from G
    $T \leftarrow T \cup \{u\}$
  if $\omega(u) \geq \delta_2(u)$ *or* $\omega(v) \geq \delta_2(v)$ **then**
    delete all vertices in $N(u) \cap N(v)$ and their adjacent edges
    $T \leftarrow T \cup (N(u) \cap N(v))$
return S

4.2 Extended Single-Edge Reduction

In addition, given an edge $e(u, v)$, if the total weight of all neighbors of v except u is smaller than the weight of v, all common neighbors of u and v can be ensured not be contained in a maximum weighted independent set. This is because including all common neighbors of u and v in a maximum weighted independent set cannot be more beneficial than containing v (or u) and its own neighbors. Therefore, all common neighbors of u and v can be removed safely. Formally, we have the following theorem.

Theorem 4.3. ***(Extended Single-edge Reduction)*** *Given an edge $e(u, v) \in E_G$, if $\omega(v) \geq \omega(N(v)) - \omega(u)$, it holds that $\lambda(G) = \lambda(G')$, where G' is obtained by removing all vertices in $N(u) \cap N(v)$.*

Proof. Suppose that $v, u \notin MWIS(G)$, which implies that there exists a subset of $N(v) \setminus \{u\}$, namely P, is contained in $MWIS(G)$. Since $\omega(v) \geq \omega(N(v)) - \omega(u) \geq \omega(P)$, we can obtain a new set of vertices $MWIS'(G)$ whose weight is not smaller than that of

Table 1: Comparison with existing works, where rt and mem represent the response time (seconds) and memory cost, respectively. Numbers in bold and labeled with stars means they are the best amongst all the solvers.

Graphs	*HILS*			*DynWVC*			*KaMIS*			*HtWIS*		
	weight	*rt (s)*	*men (MB)*	*wegiht*	*rt (s)*	*men (MB)*	*weight*	*rt (s)*	*men (MB)*	*weight*	*rt (s)*	*men (MB)*
GrQC	**289356***	46.252	4.08	**289356***	5.996	4.2	**289356***	0.005	5.7	**289356***	**0.002***	**3.55***
com-dblp	17320996	>1000	54.39	17608914	>1000	64	**17625916***	0.11	83	**17625916***	**0.06***	**23.43***
CondMat	1145256	>1000	7.46	1145188	>1000	8.12	**1145292***	0.023	13.99	**1145292***	**0.017***	**5.66***
Gowalla	12224804	>1000	40.98	12259297	>1000	51	**12276798***	>1000	47.33	12276781	**0.035***	**14.42***
temporal	106071157	>1000	159.62	106215786	>1000	198	**106216158***	1.237	440	**106216158***	**0.318***	**123.1***
wiki-topcats	97380337	>1000	704	105722441	>1000	893	106324142	>1000	2197	**106635460***	**6.41***	**423.74***
Wikitalk	235485822	>1000	316.25	235834861	>1000	400	**235835582***	1.825	907	**235835582***	**0.603***	**255.63***
Email-Enron	2457506	>1000	10.59	2457500	>1000	12.3	**2457578***	0.032	21.84	**2457578***	**0.021***	**7.77***
com-youtube	87435714	>1000	167.27	90004215	>1000	205	**90285974***	0.869	459.21	90285968	**0.422***	**121.74***
NotreDame	25852907	>1000	55.8	25959316	>1000	68	25924706	>1000	6066	**25962184***	**0.144***	**39.449***
web-BerkStan	41310274	>1000	201.6	43626371	>1000	253	43822023	>1000	2356	**43901478***	**16.916***	**121.94***
nh2010	586919	>1000	9.91	584276	>1000	11.13	581588	>1000	1962	**587059***	**0.035***	**7.83***
ri2010	457069	>1000	6.83	457145	>1000	7.4	445976	>1000	3037	**457108***	**0.038***	**5.609***
ca2010	15104847	>1000	100.64	16503330	>1000	118	16554197	>1000	2281	**6792827***	**0.585***	**75.53***
fl2010	8139134	>1000	69.3	8605746	>1000	82	8632754	>1000	2603	**8719272***	**0.379***	**52.25***
ga2010	4533128	>1000	42.96	4593516	>1000	50.5	4631281	>1000	2406	**4639891***	**0.18***	**32.13***
il2010	5559637	>1000	64.45	5880086	>1000	75	5839811	>1000	1937	**5963974***	**0.407***	**47.48***

$MWIS(G)$ by replacing P with v. This is conflict with the premise that $MWIS(G)$ is the optimal solution. Therefore, either v or u should be in some $MWIS(G)$, resulting in that their common neighbors, namely $N(u) \cap N(v)$, are removed from the graph. □

Example 6. *Let us consider the the edge $e(v_1, v_4)$ in Figure 5(b). Except v_4, there are four neighbors of v_1: v_2, v_3, v_5 and v_7. Their total weight is $w(v_2) + w(v_3) + w(v_5) + w(v_7) = 4 + 2 + 7 + 4 = 17 < w(v_1) = 18$. Thus, the common neighbors of v_1 and v_4 (v_2, v_3 and v_5) can be removed safely.*

Algorithm 3 gives the details of performing the above two single-edge reduction rules. In general, we scan edges one by one, and check whether we can use any one of the above two single-edge reduction rules.

Time complexity. For each vertex pair (u, v) such that $e(u, v) \in E_G$, we need to compute the common neighbors between u and v . So the total time cost is $O(m * \Delta)$, where Δ represent the maximum degree among all vertices.

5 EMPIRICAL STUDIES

The graphs and evaluation metrics used in the experiments are introduced in Section 5.1. Section 5.2 reports the results in terms of effectiveness and efficiency.

5.1 Experimental Setting

Graphs. We use 23 real graphs to evaluate the algorithms. The statistics are listed in Table 5 (in Appendix C).

There are two different ways of generating weights for vertices in our experiments, i.e., (1) the weights are randomly generated between a and b, and (2) Similar to the previous work [7], the weight is generated according to $\omega(v) = (v.id \bmod c) + 1$, where $v.id$ is the identifier of the vertex v and c is a positive integer. The parameters a, b, and c are set to 1, 200, and 200 by default, respectively.

Competitors. The following algorithms are compared in the paper.

- *KaMIS*: The state of the art exact algorithm based on branch and reduce paradigm [26];
- *HILS*: A heuristic algorithm based on local seach [30];
- *DynWVC*: A well-known advanced algorithm with dynamical local search techniques [6];
- *DtWIS, WtWIS, HtWIS*: the degree-oriented, weight-oriented, or hybrid tie-breaking policies equipped with our proposed reductions.

Metrics. Since it is hard to obtain the exact solution over large graphs, we employ a normalized metric, relative precision (abbreviated as *rp*), as defined in Equation (1) to measure the precision of each algorithm, where $maxAlg(G)$ is the independent set of the largest weight detected by all the adopted algorithms.

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

All the experiments are conducted on a Linux machine with an Intel(R) Xeon(R) E5-2678 v3 CPU @2.5GHz and 220G RAM. The programmes above are implemented in C++ with -O3 Optimization.

5.2 Experimental Results

5.2.1 *Evaluation of Solution Quality.*

To validate the performance of our framework, we compare our algorithms with three heuristic competitors, namely, KaMIS [26], DynWVC [6] and HILS [30]. KaMIS is driven by reductions following the classical branch-and-reduce paradigm. HILS and DynMwc are two local search based frameworks, aiming to iteratively search for solutions with swap techniques. However, it is time-consuming to deliver an exact solution, so the algorithm is forced to terminate if it takes more than 1000 seconds. Table 1 reports the performance. It is observed that HtWIS outperforms three competitors in most cases as it takes less time and memory to produce a larger solution. In some graphs, e.g., web-BerStan and ca2010, HtWIS is faster than other solvers by three magnitudes. Therefore, HtWIS exhibits significant advantages compared to the existing algorithms.

Table 2: Effect of high-degree reductions (gap to HtWIS).

Graphs	NotreDame	com-dblp	as-skitter	roadNet-PA	Gowalla	amazon	Wikitalk	web-Google	soc-epinions1	web-Berkstan	GrQc	CondMat	com-youtube
HtWIS w/o BSE	27773	46180	20216	4886	0	23706	0	255329	2567	261737	572	4812	723
HtWIS w/o ESE	718	4	1524	2859	0	362	0	424	0	2939	0	0	0

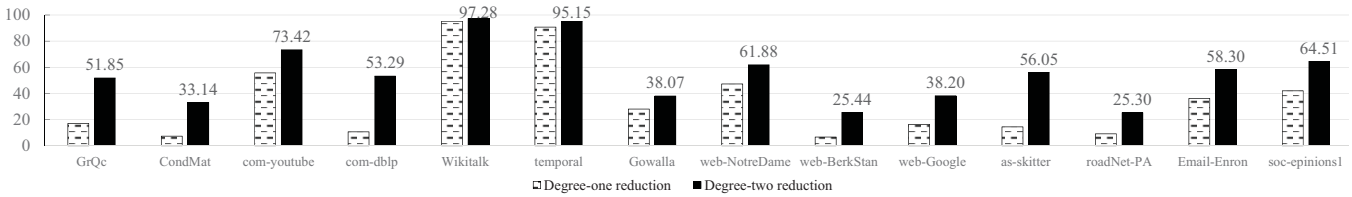

Figure 6: Effect of low-degree reductions (the percentage of reduced vertices to graph size, %)

5.2.2 Evaluation of Reduction Rules.

Low-degree reductions are designed to reduce the graph size as fast as possible, so as to reduce search space for high-degree reductions efficiently. Since low-degree vertices can be maintained within a queue and are easy to detect after iterative update on graphs, the time cost is quite cheap and we put more attention to its effectiveness. As such, the best way to evaluate its effectiveness is to measure the percentage of reduced vertices to graph size. As shown in Figure 6, over 50% of vertices can be removed in most graphs, which accords with the fact that real-world graphs tend to follow power-law distributions. In graphs like Wikitalk and temporal, a considerable number of vertices accounting for 90% of the graph size can be removed. Therefore, we can conclude that low-degree reductions are of significance and successfully serve our purpose.

High-degree reductions are to handle residual graphs returned by low-degree reductions. As mentioned above, graph structures are too complicated for a single high-degree reduction to cover and deal with, and this gives rise to diverse high-degree reductions to ensure the robustness of our framework. We conduct an ablation test to evaluate the capability and importance of each high-degree reduction. Two high-degree reductions are disabled from HtWIS in turn and we obtain two suboptimal algorithms, i.e., HtWIS without BSE and HtWIS without ESE, where BSE and ESE denote the basic single-edge and extended single-edge reductions, respectively. The gaps of their delivered independent sets to the HtWIS algorithm are presented in Table 2, where a larger gap indicates a worse result but greater importance of the reduction. Without BSE reduction, it is clear that more vertices remain in graphs, revealing that BSE reduction plays the vital role in high-degree reductions. However, in some graphs, e.g., com-dblp and roadNet-PA, the optima is not achieved at the absence of either of them. This indicates the necessity of the two reductions. In conclusion, we observe that the BSE reduction is more useful while both of them matter for the sake of better performance.

5.2.3 Evaluation of tie-breaking policies.

Tie-breaking policies activate reductions by picking out vertices that are unpromising to appear in the solution. Figure 7 reports the effect of different policies. The degree-oriented policy loses information hidden in node weights, resulting in poor performance on vertex-weighted graphs. Having taken weight into account, hybrid (shorted as Ht) and weight-oriented (Wt) tie-breaking policies are designed specially. However, Wt policy performs worse than Ht and Dt (degree-oriented) since it lacks the ability to boost reductions compared to the other two policies. In other words, it initially tries to reduce loss of accuracy by removing the vertex contributing least to solutions. Meanwhile, we unexpectedly find that more vertex pickings bring much more loss to the final solution. In contrast, Ht benefits from advantages of reductions as payback, achieving the best performance.

Table 3: rp (%) $vs.$ the scale (*HtWIS*)

Graphs	b-a=50	b-a=100	b-a=200	b-a=500	b-a=800	b-a=1000
GrQc	100	100	100	100	100	100
com-youtube	100	100	100	100	100	100
com-dblp	100	100	100	100	100	100
as-skitter	100	99.98	100	100	100	99.99
Email-Enron	100	100	100	100	100	100
web-Google	100	99.93	100	100	100	100
Gowalla	100	100	100	100	100	100

5.2.4 Evaluation of Weight Assignments.

Since there are two different ways of weight assignments for a graph, i.e., (1) the weights are randomly generated between $[a, b]$ and (2) the weights are given according to $\omega(v) = (v.id \bmod c) + 1$. In this subsection, we study the effect of the scale $(b - a)$ and c. Table 3 reports the effect of $(b - a)$ on the relative precision, where the scale is varied from 50 to 1000. Note that the relative precision is computed according to the solution of the largest weight among the solutions delivered by all the methods. It is obvious that HtWIS always produces the best solution with the growth of $b - a$. Table 4 reports the effect of the parameter c on the relative precision, where c is varied from 50 to 1000. As c grows, we find that the relative precision hardly decreases on most graphs. Neither of these different weight assignments stops algorithm *HtWIS* producing the best solution.

6 CONCLUSIONS

In this paper, we study the problem of computing a near-maximum weighted independent set over the general weighted graphs. A general reducing and tie-breaking framework of computing a near-maximum weighted independent set is presented. We devise two kinds of effective reduction rules, i.e., the low-degree reductions and

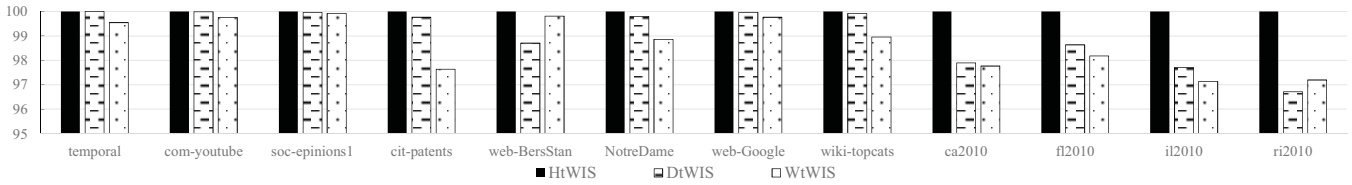

Figure 7: Relative precision (%) of different policies

Table 4: *rp* (%) *vs.* the integer *c* (*HtWIS*)

Graphs	c=50	c=100	c=200	c=500	c=800	c=1000
GrQc	100	100	100	100	100	100
com-youtube	100	100	100	100	100	100
com-dblp	100	100	100	100	100	100
as-skitter	100	99.98	100	100	100	100
Email-Enron	100	100	100	100	100	100
web-Google	100	100	100	100	100	100
Gowalla	100	100	100	100	100	100

high-degree reductions, which generalize the existing unweighted degree-one reduction and weighted domination reduction rule. Extensive experiments over a wide range of graphs confirm the effectiveness and efficiency of the proposed methods.

ACKNOWLEDGMENTS

This work was substantially supported by National Natural Science Foundation of China (Grant No. 61902074), Science and Technology Committee Shanghai Municipality (Grant No. 19ZR1404900), and National Key Research and Development Program of China (Grant No. 2019YFB1406401).

REFERENCES

[1] Takuya Akiba and Yoichi Iwata. 2016. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theor. Comput. Sci.* 609 (2016), 211–225.
[2] Diogo Vieira Andrade, Mauricio G. C. Resende, and Renato Fonseca F. Werneck. 2012. Fast local search for the maximum independent set problem. *J. Heuristics* 18, 4 (2012), 525–547.
[3] Filipe Araujo, Jorge Farinha, Patrício Domingues, Gheorghe Cosmin Silaghi, and Derrick Kondo. 2011. A maximum independent set approach for collusion detection in voting pools. *J. Parallel Distrib. Comput.* 71, 10 (2011), 1356–1366.
[4] William Brendel, Mohamed R. Amer, and Sinisa Todorovic. [n.d.]. Multiobject tracking as maximum weight independent set. In *CVPR*. 1273–1280.
[5] William Brendel and Sinisa Todorovic. 2010. Segmentation as Maximum-Weight Independent Set. In *NIPS*, Vol. 23. 307–315.
[6] Shaowei Cai, Wenying Hou, Jinkun Lin, and Yuanjie Li. [n.d.]. Improving Local Search for Minimum Weight Vertex Cover by Dynamic Strategies. In *IJCAI*, Jérôme Lang (Ed.). 1412–1418.
[7] Shaowei Cai and Jinkun Lin. 2016. Fast Solving Maximum Weight Clique Problem in Massive Graphs. In *IJCAI*. 568–574.
[8] Lijun Chang. 2019. Efficient Maximum Clique Computation over Large Sparse Graphs. In *KDD*. 529–538.
[9] Lijun Chang, Wei Li, and Wenjie Zhang. 2017. Computing A Near-Maximum Independent Set in Linear Time by Reducing-Peeling. In *SIGMOD*. 1181–1196.
[10] Jakob Dahlum, Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. 2016. Accelerating Local Search for the Maximum Independent Set Problem. In *SEA*. 118–133.
[11] Uriel Feige. 2004. Approximating Maximum Clique by Removing Subgraphs. *SIAM J. Discrete Math.* 18, 2 (2004), 219–225.
[12] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. 2009. A measure & conquer approach for the analysis of exact algorithms. *J. ACM* 56, 5 (2009), 25:1–25:32.
[13] E. J. Gardiner, P Willett, and P. J. Artymiuk. 2000. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information & Computer Sciences* 40, 2 (2000), 273.
[14] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.
[15] Andreas Gemsa, Martin Nöllenburg, and Ignaz Rutter. 2014. Evaluation of Labeling Strategies for Rotating Maps. In *Experimental Algorithms - 13th International Symposium, SEA 2014*. 235–246.
[16] Andrea Grosso, Marco Locatelli, and Wayne J. Pullan. 2008. Simple ingredients leading to very efficient heuristics for the maximum clique problem. *J. Heuristics* 14, 6 (2008), 587–612.
[17] Magnús M. Halldórsson and Jaikumar Radhakrishnan. 1994. Greed is good: approximating independent sets in sparse and bounded-degree graphs. In *ACM Symposium on Theory of Computing*. 439–448.
[18] Magnús M. Halldórsson and Jaikumar Radhakrishnan. 1997. Greed is Good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs. *Algorithmica* 18, 1 (1997), 145–163.
[19] Johan Håstad. 1996. Clique is Hard to Approximate Within $n^{1-epsilon}$. In *Annual Symposium on Foundations of Computer Science*. 627–636.
[20] Kyomin Jung and Devavrat Shah. 2007. Low Delay Scheduling in Wireless Network. In *IEEE International Symposium on Information Theory*.
[21] K. Jung and D. Shah. 2007. Low Delay Scheduling in Wireless Network. In *2007 IEEE International Symposium on Information Theory*. 1396–1400.
[22] Akihisa Kako, Takao Ono, Tomio Hirata, and Magnús M. Halldórsson. 2009. Approximation algorithms for the weighted independent set problem in sparse graphs. *Discret. Appl. Math.* 157, 4 (2009), 617–626.
[23] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh V. Vazirani. 1998. On Syntactic versus Computational Views of Approximability. *SIAM J. Comput.* 28, 1 (1998), 164–191.
[24] Tim Kieritz, Dennis Luxen, Peter Sanders, and Christian Vetter. 2010. Distributed Time-Dependent Contraction Hierarchies. In *Experimental Algorithms*. 83–93.
[25] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F. Werneck. 2016. Finding Near-Optimal Independent Sets at Scale. In *ALENEX*. 138–150.
[26] 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 *ALENEX*, Stephen G. Kobourov and Henning Meyerhenke (Eds.). SIAM, 144–158.
[27] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.
[28] Yu Liu, Jiaheng Lu, Hua Yang, Xiaokui Xiao, and Zhewei Wei. 2015. Towards Maximum Independent Sets on Massive Graphs. *PVLDB* 8, 13 (2015), 2122–2133.
[29] George L. Nemhauser and Leslie E. Trotter Jr. 1975. Vertex packings: Structural properties and algorithms. *Math. Program.* 8, 1 (1975), 232–248.
[30] Bruno C. S. Nogueira, Rian G. S. Pinheiro, and Anand Subramanian. 2018. A hybrid iterated local search heuristic for the maximum weight independent set problem. *Optim. Lett.* 12, 3 (2018), 567–583.
[31] Ioannis Ch. Paschalidis, Fuzhuo Huang, and Wei Lai. 2015. A Message-Passing Algorithm for Wireless Network Scheduling. *IEEE/ACM Trans. Netw.* 23, 5 (2015), 1528–1541.
[32] J. M. Robson. 1986. Algorithms for Maximum Independent Sets. *J. Algorithms* 7, 3 (1986), 425–440.
[33] S. Sanghavi, D. Shah, and A. S. Willsky. 2009. Message Passing for Maximum Weight Independent Set. *IEEE Transactions on Information Theory* 55, 11 (2009), 4822–4834.
[34] Peng-Jun Wan, Boliu Xu, Lei Wang, Sai Ji, and Ophir Frieder. 2015. A new paradigm for multiflow in wireless networks: Theory and applications. In *INFOCOM*. 1706–1714.
[35] Jingen Xiang, Cong Guo, and Ashraf Aboulnaga. 2013. Scalable maximum clique computation using MapReduce. In *ICDE*. 74–85.
[36] Mingyu Xiao and Hiroshi Nagamochi. 2013. Exact Algorithms for Maximum Independent Set. In *ISAAC*. 328–338.
[37] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. 1997. New algorithms for fast discovery of association rules. In *KDD*. 283–286.
[38] Weiguo Zheng, Jiewei Gu, Peng Peng, and Jeffrey Xu Yu. 2020. Efficient Weighted Independent Set Computation over Large Graphs. In *ICDE*. IEEE, 1970–1973.
[39] Weiguo Zheng, Chengzhi Piao, Hong Cheng, and Jeffrey Xu Yu. 2019. Computing a Near-Maximum Independent Set in Dynamic Graphs. In *ICDE*. 76–87.

A PROOFS

Proof of Theorem 3.1

Proof. Case 1 is easy to prove since it is special instance of single-vertex reduction [38]). Let us consider case 2 next.
(1) When $v \in MWIS(G)$, u can be safely removed from G. It is straightforward to have $\lambda(G) = \lambda(G') + \omega(u)$. And v must be contained in one $MWIS$ of G', otherwise there exists a larger $MWIS(G) = \{u\} \cup MWIS(G')$, which contradicts the premise that $MWIS(G)$ is a maximum weighted independent of G.
(2) When $v \notin MWIS(G)$, u must be contained in $MWIS(G)$, and $\lambda(G) = \lambda(G') + \omega(u)$ holds. Also, $v \notin MWIS(G')$, otherwise letting v belong to $MWIS(G)$ will lead to larger $MWIS(G)$, i.e., $\lambda(G')+\omega(u) > \lambda(G)$, which contradicts the premise that $MWIS(G)$ is a maximum weighted independent of G. □

Proof of Triangle Reduction and V-shape Reduction

Proof. Typically, we provide the proof of Case 3 of V-shape reduction. Proofs of Case 1, Case2, and triangle reduction, are quite similar. Thus they are omitted for space limitation. For a given vertex u with two neighbors x and y, considering the possibility of each vertex's being in the solution, there are at most four cases to discuss, of which the one with largest weight is our target. Thus we can formulate the problem as follows,

$$\omega(MWIS(G)) = max \begin{cases} \omega(u) + \omega(MWIS(G_{N[u]})), \\ \omega(x) + \omega(MWIS(G_{N[x]})), \\ \omega(y) + \omega(MWIS(G_{N[y]})), \\ \omega(x) + \omega(y) + \omega(MWIS(G_{N[y]\cup N[x]})) \end{cases} \tag{2}$$

where G_P represents the residual graph by removing each vertex in set P from G. By extracting $\omega(u)$ for each item in the brackets, we can rewrite this formula,

$$\omega(MWIS(G)) = \omega(u) + max\{case\ a, case\ b, case\ c, case\ d\} \tag{3}$$

where $case\ a = \omega(MWIS(G_{N[u]}))$, $case\ b = \omega(x) - \omega(u) + \omega(MWIS(G_{N[x]}))$, $case\ c = \omega(y) - \omega(u) + \omega(MWIS(G_{N[y]}))$, $case\ d = \omega(u) + \omega(x) - \omega(u) + \omega(y) - \omega(u) + \omega(MWIS(G_{N[y]\cup N[x]}))$.

It is observed that, the last term in equation 3, i.e, $max\{\cdot\}$, which has a similar form as equation 2, also implies a potential MWIS solution to a new graph G' which can be derived from the original graph G, thus we can further obtain that:

$$\omega(MWIS(G)) = \omega(u) + \omega(MWIS(G')) \tag{4}$$

However, to construct the new graph G' is non-trivial, we need to adjust edges and weights of vertices of graph G to hold the equivalency throughout the transformations. Aside from the necessity of weight adjustments of x and y by subtracting $\omega(u)$, case d requires that x and y are selected into the solution iff u accompanies in new graph G'. Resorting to Lemma 3.1, we reorganize u's neighborhood by reconnecting it with neighbors of x and y to meet this goal. By this mean can we bridge the gap between graph G' and G without sacrificing the equivalency in terms of computing the MWIS solution. Note that this reduction can be easily generalized to high-degree vertices, but we dismiss it due to the explosive number of combinations as the degree increases. □

B PSEUDOCODE

Algorithm 4: Degree-two Reduction

Input : A graph $G = (V_G, E_G, \omega)$
Output: A reduced graph $G' = (V_{G'}, E_{G'}, \omega)$

```
1  G' ← G
2  Let V_=1, V_=2 be the set of degree-one and degree-two
     vertices, respectively
3  while V_=1 ≠ ∅ or V_=2 ≠ ∅ do
4      if V_=1 ≠ ∅ then
5          Extract a vertex u from V_=1
6          Degree-one Reduction(u)
7      else
8          Extract a vertex u from V_=2
9          x, y ← the two neighbors of u, ensure ω(x) ≤ ω(y)
10         if e(x, y) ∈ E_G' then
11             Triangle Reduction(u, x, y)
12         else
13             V-shape Reduction(u, x, y)
14 return G'
15 Function Triangle Reduction (u, x, y)
16 Remove u and its edges, update V_=1 and V_=2
17 if ω(x) ≤ ω(u) then Remove x, update V_=1 and V_=2
18    else Update ω(x) = ω(x) − ω(u)
19 if ω(y) ≤ ω(u) then Remove y, update V_=1 and V_=2
20    else Update ω(y) = ω(y) − ω(u)
21 Function V-shape Reduction (u, x, y)
22 if ω(u) ≥ ω(y) then
23     Contract x, y into v, s.t. N(v) = N(x) ∪ N(y)
24     Remove u and its edges, update V_=2
25     ω(v) ← ω(x) + ω(y) − ω(u)
26 else
27     if ω(u) ≥ ω(x) then
28         N(x) ← N(x) ∪ N(y), ω(y) ← ω(y) − ω(u)
29         Remove u and its edges, update V_=1 and V_=2
30     else
31         N(u) ← N(x) ∪ N(y)
32         ω(x) ← ω(x) − ω(u), ω(y) ← ω(y) − ω(u)
```

As shown in Algorithm 4. Lines 3-13 iterate all vertices with degree less than three, and decide which reduction should be invoked. Function *Triangle Reduction* (lines 15-20) follows Theorem 3.3, while function *V-shape Reduction* (lines 21-32) follows Theorem 3.4. Since some edges may be removed in function *Triangle Reduction*, the set of degree-one vertices $V_{=1}$ and degree-two vertices $V_{=2}$ should be updated in lines 16 and 17.

Table 5: Statistics of graphs

ID	Graph	$\|V(G)\|$	$\|E(G)\|$	*source*
G01	ga2010	291086	709028	SSMC
G02	il2010	451554	1082232	SSMC
G03	fl2010	484481	1173147	SSMC
G04	ca2010	710145	1744683	SSMC
G05	nh2010	48837	117275	SSMC
G06	ri2010	25181	62875	SSMC
G07	GrQC	5242	14496	SNAP
G08	com-dblp	317080	1049866	SNAP
G09	CondMat	23133	93439	SNAP
G10	Gowalla	196591	950327	SNAP
G11	as-skitter	1696415	11095298	SNAP
G12	temporal	1094018	2787967	SNAP
G13	Wikitalk	2394385	4659565	SNAP
G14	Email-Enron	36692	183831	SNAP
G15	cit-patents	3774768	16518947	SNAP
G16	com-youtube	1134890	2987624	SNAP
G17	NotreDame	325729	1090108	SNAP
G18	web-BerkStan	685230	6649470	SNAP
G19	web-Google	875713	4322051	SNAP
G20	roadNet-PA	1088092	1541898	SNAP
G21	soc-epinions1	75879	405740	SNAP
G22	amazon	334863	925872	SNAP
G23	wiki-topcats	1791489	28511807	SNAP

C EXPERIMENTAL SETTINGS

Datasets: The real graphs ga2010, ca2010, fl2010, il2010, ri2010, and nh2010 are downloaded from The SuiteSparse Matrix Collection (denoted by SSMC, https://sparse.tamu.edu/), where each vertex is assigned a weight. The other graphs, such as GrQC, Email-Enron, Gowalla, com-youtube, wiki-topcats, and cit-Patents, etc., come from the Stanford Network Analysis Platform [27], denoted by SNAP.

Reproducibility: The datasets used in the papar are available at: http://snap.stanford.edu/ and https://sparse.tamu.edu/ . The source codes are available at: https://github.com/mwis-abc/mwis-source-code .