https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING

Vol. 00, No. 0, Xxxxx 0000, pp. 000–000

ISSN 0899-1499, EISSN 1526-5528

Accepted to *INFORMS Journal on Computing, May 2026*

A Reduction-Driven Local Search for the Generalized Independent Set Problem

Yiping Liu, Yi Zhou*, Zhenxiang Xu, Mingyu Xiao
University of Electronic Science and Technology of China, Chengdu, China
*Corresponding author
yliu823@aucklanduni.ac.nz; zhou.yi@uestc.edu.cn; zhenxiangxu@std.uestc.edu.cn; myxiao@uestc.edu.cn;

Jin-Kao Hao
LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France, jin-kao.hao@univ-angers.fr

Abstract. The Generalized Independent Set (GIS) problem extends the classical maximum independent set problem by incorporating profits for vertices and penalties for edges. This generalized problem has been identified in diverse applications in fields such as forest harvesting, competitive facility location, social network analysis, and even machine learning. However, solving the GIS problem in large-scale, real-world networks remains computationally challenging. In this paper, we explore data reduction techniques to address this challenge. We first propose 14 reduction rules that can reduce the input graph with rigorous optimality guarantees. We then present a reduction-driven local search (RLS) algorithm that integrates these reduction rules into the pre-processing, the initial solution generation, and the local search components in a computationally efficient way. The RLS is empirically evaluated on 278 graphs drawn from different application scenarios. The results indicate that the RLS is highly competitive – For most graphs, it achieves significantly superior solutions compared to other known solvers, and it effectively provides solutions for graphs exceeding 260 million edges, a task at which every other known method fails. Analysis also reveals that data reduction plays a key role in achieving such a competitive performance.

Funding: This research was supported by the National Natural Science Foundation of China [Grant 62372093] and [Grant 62372095].

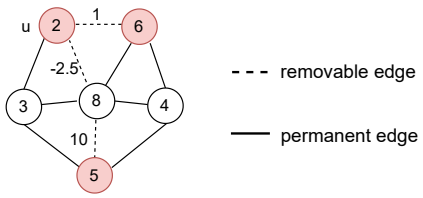
Key words: Generalized independent set; Data reduction; Local search; Large graph

1. Introduction

Graphs are essential tools for modeling complex systems including social networks, geometric networks, and forest management systems. The analysis of graph data has led to the development of numerous combinatorial problems that help address real-world challenges. In this paper, we study

the *Generalized Independent Set* (GIS) problem. Let $G = (V, E = E_p \cup E_r)$ be an undirected graph where V is the vertex set and E is the edge set that is partitioned into a *permanent edge* set E_p and a *removable edge* set E_r. Each vertex $v \in V$ is assigned a profit $w(v) \in \mathbb{R}$. Each removable edge $e \in E_r$ is associated with a penalty $p(e) \in \mathbb{R}$. A *generalized independent set* of the graph G is a vertex set $I \subseteq V$ such that no permanent edge exists between any two vertices of I in G. Then, the GIS problem asks for a generalized independent set I of maximum net benefit, which is the total vertex profits of I minus the total penalties of removable edges whose end vertices are in I. An example is shown in Figure 1.

Figure 1 **A toy example of the GIS problem. The set of red vertices is a generalized independent set of net benefit of** $(2+6+5)-1=12$**.**



When all the edges are permanent and each vertex has a profit of 1, the GIS problem becomes the well-known *maximum independent set problem* (MIS), which asks for a maximum-cardinality set of non-adjacent vertices in a given graph. As the GIS problem allows a more flexible definition of vertex profit and edge penalties, it captures a broader range of real-world scenarios than the classical independent set problem. We give detailed examples of applications in diverse domains in Section 2.1. However, since MIS is NP-hard, W[1]-hard and difficult to approximate (Cygan et al. 2015), the equivalence above indicates that the GIS problem is also computationally hard.

Existing works have proposed various approaches to solve the GIS problem practically. Exact solution methods include mixed integer linear programming (Colombi et al. 2017) and unconstrained binary quadratic programming (Hosseinian and Butenko 2019). They can solve instances with up to 400 vertices and 71,820 edges in three hours so far (Zheng et al. 2024). On the other hand, heuristic search methods have demonstrated their effectiveness in providing high-quality solutions within reasonable computational time (e.g., Colombi et al. (2017), Nogueira et al. (2021)). For example, the most recent algorithm, named *Adaptive Local Search* (ALS), proposed by Zheng et al. (2024) achieves near-optimal solutions on all standard benchmark instances and scales to problems with up to 18,000 vertices. Nevertheless, we observed that these heuristic algorithms struggle to maintain solution quality in even larger instances.

Data reduction, which reduces input size while preserving problem optimality, has recently emerged as a promising technique for improving algorithm scalability on large instances. The technique has been successfully applied to problems such as independent set, vertex cover, and maximum clique (Großmann et al. 2023, Abu-Khzam et al. 2022, Verma et al. 2015, Walteros and Buchanan 2020, Lamm et al. 2019). For the GIS problem, research on data reduction is still at its early stages. There are three main challenges in developing data reduction techniques for the GIS problem: (1) The complex interplay among vertex profits, edge penalties, and the underlying graph structure, (2) Translating reduction rules into efficient, practical algorithms, which often requires incrementally applying the reduction rules (Akiba and Iwata 2016, Chang et al. 2017), (3) Integrating reductions into local search algorithms while balancing search time and efficiency.

In this paper, we address these challenges by proposing a novel reduction-driven local search algorithm for solving the GIS problem. Our contributions are summarized as follows.

1. To the best of our knowledge, this is the first work to apply data reduction techniques to the GIS problem. We introduce 14 reduction rules that are classified into four groups: edge transformation, neighborhood-based reduction, low-degree reduction, and vertex-pair reduction. We provide rigorous proofs that these rules preserve the problem optimality. To efficiently implement these reduction rules, we introduce an ordering strategy and construct a dependency graph to manage their applications.
2. We propose an efficient reduction-driven local search algorithm, named RLS, to solve the GIS problem. We incorporate our proposed reduction rules into three main components of the RLS: the pre-processing, which incrementally reduces the size of the original input instance, the initial solution generation, which interplays the vertex selection and data reduction, and the neighborhood search, which uses a novel reduction-based move operator to guide the search trajectory.
3. We evaluate the proposed the RLS algorithm on a total of 278 instances. On small instances with up to 400 vertices and 80,000 edges, the RLS consistently achieves the known optimal solutions with significantly shorter running time than exact algorithms. On large instances, the RLS exhibits superior scalability over existing heuristic algorithms in solution quality. Notably, the RLS successfully handles instances with over 18 million vertices and over 260 million edges, where existing heuristic methods fail. Our experimental results also demonstrate that data reduction techniques significantly improve the computational efficiency of the RLS.

The rest of the paper is organized as follows: Section 2 reviews applications and related works, followed by Section 3 which introduces the notations and definitions. In Section 4, we present our reduction rules for the GIS problem. Then, in Section 5, we describe the RLS algorithm. Section 6 presents computational results and comparisons with existing methods. In the last section, we conclude the paper and outline future directions.

2. Background

In this section, we provide some application examples of the GIS model and review existing GIS algorithms.

2.1. Applications Related to GIS

In the following, we highlight several important applications of the GIS problem:

Forest harvesting. The planning of forest harvesting involves selecting forest areas (cells) for timber, aiming to maximize benefits like timber value while minimizing ecological damage from excessive cutting. As a GIS problem, forest cells are vertices weighted by harvesting benefits, and edges represent penalties for harvesting adjacent cells (Hochbaum and Pathria (1997)).

Competitive facility location. In determining optimal locations for service facilities, each potential site has a utility value. However, the utility diminishes when multiple facilities compete for customers in the same area. The facility location problem, modeled with vertices as facilities and edges as geographic relationships, can be framed as a GIS problem to maximize utilities (Hochbaum (2004)). Similar spatial conflicts exist in airplane seating (Pavlik et al. 2021) and label placement (Mauri et al. 2010).

Social network analysis. Potentially, the GIS problem has broad applications in social network analysis. A maximum independent set corresponds to the complement of a minimum vertex cover and is equivalent to a maximum clique in the graph's complement (Sanchis and Jagota (1996), Xiao and Nagamochi (2017), Bai and Xiao (2024)). These properties make GIS useful for tasks such as detecting cliques or communities (Gschwind et al. (2021)) and optimizing coverage and reach within social networks (Puthal et al. (2015)).

Other applications. GIS has been applied in machine learning to aggregate low-quality results into better results (Brendel and Todorovic 2010), to track objects in multi-object tracking tasks (Brendel et al. 2011), and to serve as a pooling layer in graph neural networks (Stanovic et al. 2025).

2.2. Existing Algorithms

The GIS problem was first introduced in Hochbaum and Pathria (1997). Existing methods for solving the GIS problem can be mainly divided into exact and heuristic algorithms.

Exact Algorithms. In pursuit of optimal solutions, Hochbaum (2004) introduced an integer linear programming model that assigns variables to each node and edge, and associates each edge with a specific constraint. As a result, the integer linear program for a graph of 2000 vertices (e.g. C2000.5 in the DIMACS competition set) has approximately 500,000 variables and 500,000 constraints, which is quite large for modern ILP solvers like CPLEX. (It fails to even report a solution within 3 hours for this instance.) Then, Colombi et al. (2017) reformulated the GIS problem as an unconstrained binary quadratic program (UBQP), and developed a branch-and-cut algorithm, along with additional valid inequalities. Note that they also generated a total of 216 random instances which often serve as benchmark datasets in the later work. Hosseinian and Butenko (2019) employed another quadratic program to bound the optimal solution, which was later incorporated into a branch-and-bound framework. Recently, Zheng et al. (2024) used a Lagrangian relaxation in their branch-and-bound framework to estimate tight upper bounds. Their experiments showed that their algorithm is able to solve the GIS instances with up to 400 vertices within 3 hours.

Heuristic Algorithms. Kochenberger et al. (2007) also developed a UBQP formulation of the GIS problem and solved it using tabu search. Colombi et al. (2017) enhanced performance by integrating advanced tabu search with UBQP. Nogueira et al. (2021) directly solved the original problem by using an iterated local search heuristic. They explored move operations of adding one or two vertices to the incumbent solution. More recently, Zheng et al. (2024) proposed a more refined local search heuristic"ALS" to explores three move operations – adding one vertex to the solution, dropping one vertex from the solution, and replacing a vertex in the solution with another one not in the solution. The ALS algorithm has been reported to achieve state-of-the-art performance on the standard benchmark set of 216 instances proposed by Colombi et al. (2017). In particular, it is able to find high-quality solutions for instances with up to $17,903$ vertices within 5 minutes. However, its performance on graphs with more than 20,000 vertices has not been systematically evaluated. Our experiments show that ALS quickly becomes impractical on such large-scale instances.

Data Reduction Techniques. Data reduction techniques for integer programming models can be traced back several decades, referred to as variable fixing and coefficient reduction (Crowder et al. 1983). Then, the parameterized complexity community has also adopted this technique, known as *kernelization*, to achieve strong theoretical bounds on the time complexity of many

hard combinatorial graph problems (Cygan et al. 2015). Recently, data reduction techniques are intensively used for practically tackling MIS (Abu-Khzam et al. 2022). For example, Akiba and Iwata (2016) shows that branch-and-reduce outperforms pure branch-and-bound, while Dahlum et al. (2016) and Chang et al. (2017) integrate reductions with heuristics to achieve near-optimal performance. These techniques enable handling graphs with millions of vertices and edges (Lamm et al. 2019, Xiao et al. 2021). Besides, the effectiveness of reductions has also been acknowledged in other fundamental problems like the clique problem (Walteros and Buchanan 2020), and the matching problem (Koana et al. 2021). However, as for the GIS problem, no existing work focuses on reduction rules. Extending existing reductions for the independent set problem to the GIS problem, where both vertices and edges are weighted, remains challenging.

3. Preliminary

According to the definition, an input instance of the GIS problem is an (undirected weighted) graph $G = (V, E = E_p \cup E_r)$ where each vertex $v \in V$ is associated with a profit $w(v) \in \mathbb{R}$ and each removable edge $e \in E_r$ is associated with a penalty $p(e) \in \mathbb{R}$. For a vertex set $S \subseteq V$, we define $nb(S)$ as the *net benefit of* S, i.e.,

$$nb(S) = \sum_{v \in S} w(v) - \sum_{u \in S, v \in S, e=\{u,v\} \in E_r} p(e).$$

Then, a *generalized independent set* of G is a vertex subset $S \subseteq V$ such that $G[S]$ contains no permanent edge. The GIS problem can be formulated as finding a generalized independent set such that its net benefit is maximized. We denote the optimal objective value of the GIS problem in G as $\alpha(G)$, i.e.,

$$\alpha(G) = \max_{S \subseteq V} nb(S) \quad s.t. \quad E_p \cap \{\{u, v\} \mid u, v \in S\} = \varnothing$$

We also denote $MGIS(G)$ as the family of generalized independent sets $S \subseteq V$ that achieve the optimal objective value $\alpha(G)$. Each $S \in MGIS(G)$ corresponds to an optimal solution.

For convenience, we report more notations in Table 1. Note that we use $\tilde{w}(v)$ to represent the profit of v plus the absolute sum of non-positive penalties of edges that are incident to v. Given that $S \subseteq V$ is a generalized independent set, $w^+(S)$ is an upper bound on the net benefit of S. For example, in Figure 1, $\tilde{w}(u) = 2 + 2.5 = 4.5$. When S denotes the set of red vertices in Figure 1, $w^+(S) = 4.5 + 6 + 5 = 15.5$. In the rest of the paper, we use the notation $\coloneqq$ to represent assignment and $=$ to indicate equivalence.

4. Reduction Rules

In this section, we develop reduction rules that identify reducible graph structures without compromising optimality. We rely on four operations to reduce the current instance to a simplified

Table 1 Table of Notations

$G=(V,E=E_p\cup E_r)$	a graph G with vertex set V and edge set E, E_p, E_r represent the set of permanent edges and the set of removable edges, respectively
$w(v), p(\{u,v\})$	the profit of including a vertex v, and the penalty of including both vertices u and v
$N(v), N_p(v), N_r(v)$	subsets of V, representing the sets of vertices connected to v, vertices connected to v by a permanent edge and vertices connected to v by a removable edge, respectively
$\|N(v)\|, \|N_p(v)\|, \|N_r(v)\|$	the degree, permanent degree, and removable degree of v, respectively
$N[v], N_p[v], N_r[v]$	subsets of V, representing the sets $N(v)\cup\{v\}$, $N_p(v)\cup\{v\}$, $N_r(v)\cup\{v\}$, respectively
Δ	the maximum degree of vertices in G
$\tilde{w}(v)$	$\tilde{w}(v)=w(v)+\sum_{x\in N_r(v)}\max(0,-p(\{v,x\}))$
$S, G[S], E(G[S])$	a vertex subset of G, the subgraph induced by S, and the edge set of the induced subgraph
$w(S), w^+(S)$	$w(S)=\sum_{v\in S}w(v)$, $w^+(S)=\sum_{v\in S}\max(0,\tilde{w}(v))$
$nb(S)$	the net benefit of a vertex set S, $nb(S)=\sum_{v\in S}w(v)-\sum_{u\in S,v\in S,e=\{u,v\}\in E_r}p(e)$
$\alpha(G), MGIS(G)$	the largest net benefit value achievable without including any permanent edge, and the family of all subsets of V attaining $\alpha(G)$ value under the same constraint

instance. (1) *Edge transformation*, which identifies removable edges that can be deleted or transformed to permanent edges; (2) *Vertex inclusion*, which identifies vertices that are in at least one set $I\in MGIS(G)$; (3) *Vertex exclusion*, which identifies vertices that do not belong to a set $I\in MGIS(G)$. (4) *Folding*, which identifies vertices that can be merged into a single vertex. Based on the four operations, we introduce a total of 14 reduction rules as follows. For simplicity, we denote the graph obtained after applying a reduction rule by $G'=(V',E'=E'_p\cup E'_r)$. Unless otherwise specified, we assume that $w'(v)$, the profit of a vertex $v\in V'$,keeps its original profit $w(v)$ if v also exists in V, and $p'(e)$, the penalty of a removable edge $e\in E'_r$, keeps its original penalty $p(e)$ if e also exists in E_r.

4.1. Edge Transformation

We first introduce two edge transformation rules.

R1 (Edge Removal Reduction). *If there exists an edge $e=\{u,v\}\in E_r$ such that $p(e)=0$, then $\alpha(G)=\alpha(G')$ where G' is obtained by only removing the edge e from G.*

The proofs of the correctness of this reduction rule, and the subsequent reduction rules, are deferred to the online supplementary file. R1 implies removing an edge e with $p(e)=0$ does not change optimality.

R2 (Edge-penalty Reduction). *If there is an edge $e=\{u,v\}\in E_r$ such that $p(e)>min(\tilde{w}(u),\tilde{w}(v))$, then $\alpha(G)=\alpha(G')$ where G' is obtained by only moving e from E_r to E_p in G.*

Compared to a permanent edge, a removable edge is more challenging to reduce because of the difficulties of identifying whether both endpoints of a removable edge are in a set $S\in MGIS(G)$.

R2 allows us to transform a removable edge into a permanent edge without sacrificing optimality, underpinning many further reduction rules.

4.2. Neighborhood-based Reductions

We further consider the local neighborhood of a vertex.

R3 (Neighborhood Weight Reduction). *If there is a vertex $u \in V$ such that $w(u) \geq w^+(N(u))$, then u is in at least one set $I \in MGIS(G)$ and $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq G[V \setminus N_p[u]]$, and $w'(v) \coloneqq w(v) - p(\{u,v\})$ for each $v \in N_r(u)$.*

We extend R3 by considering more neighboring structures to obtain R4-R6.

R4 (Neighborhood Penalty Reduction). *If there exists a vertex $u \in V$ such that $w(u) \geq w^+(N_p(u)) + \sum_{x \in N_r(u)} \max(0, p(\{u,x\}))$, then u is in at least one set $I \in MGIS(G)$ and $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq G[V \setminus N_p[u]]$, and $w'(v) \coloneqq w(v) - p(\{u,v\})$ for each $v \in N_r(u)$.*

R5 (Negative Profit Reduction). *If there exists a vertex $u \in V$ such that $w(u) < 0$ and $\sum_{v \in N_r(u)} \min(0, p(\{u,v\})) > w(u)$, then u is not in any $I \in MGIS(G)$ and $\alpha(G) = \alpha(G')$ where $G' \coloneqq G[V \setminus \{u\}]$.*

Next, we show the *clique reduction* rule. A *clique* of G is a subset of vertices in which every two vertices are connected by a permanent edge.

R6 (Clique Reduction). *If there exists a vertex $u \in V$ such that $N_p(u)$ is a clique of G and $w(u) \geq \sum_{v \in N_r(u)} \max(0, p(\{u,v\})) + \max(0, \max_{v \in N_p(u)} \tilde{w}(v))$, then u is in at least one set $I \in MGIS(G)$, $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq G[V \setminus N_p[u]]$, and $w'(v) \coloneqq w(v) - p(\{u,v\})$ for each $v \in N_r(u)$.*

4.3. Low-degree Reductions

Low-degree vertices such as degree-one and degree-two vertices are common in large real-world graphs. Next, we introduce five reduction rules to reduce these vertices.

4.3.1. Degree-one Vertices Reduction

R7 (Degree-one Vertices Reduction). *Let u be a vertex with a single neighbor v.*

Case 1: $\{u,v\} \in E_r$ and $w(u) \geq p(\{u,v\})$. (1) If $w(u) \geq 0$, then u is in at least one set $I \in MGIS(G)$ and $\alpha(G) = \alpha(G') + w(u)$, where $G' \coloneqq G[V \setminus \{u\}]$ and $w'(v) \coloneqq w(v) - p(\{u,v\})$. (2) If $w(u) < 0$, then $\alpha(G) = \alpha(G')$, where $G' \coloneqq G[V \setminus \{u\}]$ and $w'(v) \coloneqq w(v) - p(\{u,v\}) + w(u)$. u is in at least one set $I \in MGIS(G)$ iff v is in at least one set $I \in MGIS(G')$.

Case 2: $\{u,v\} \in E_p$ or $\{u,v\} \in E_r$ but $w(u) < p(\{u,v\})$. (1) If $w(u) \geq 0$ and $w(u) \geq \tilde{w}(v)$, then u is in at least one set $I \in MGIS(G)$, $v \notin I$, and $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq G[V \setminus \{u,v\}]$. (2) If $w(u) \geq 0$ and $w(u) < \tilde{w}(v)$, then u and v cannot be both included in the same $I \in MGIS(G)$,

we have $\alpha(G) = \alpha(G') + w(u)$*, where* $G' \coloneqq G[V \setminus \{u\}]$ *and* $w'(v) \coloneqq w(v) - w(u)$*. (3) If* $w(u) < 0$*, then* u *is not in any* $I \in MGIS(G)$ *and* $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq G[V \setminus \{u\}]$.

Note that R7 covers all degree-one vertices. In other words, any vertex of degree one can be removed using this rule.

4.3.2. Degree-two Vertices Reductions For ease of describing $R8$ and $R9$, we assume $p(\{u,v\}) = +\infty$ for each $\{u,v\} \in E_p$. This assumption does not affect the optimal objective value in the graph.

As a matter of fact, $R8$ and $R9$ together can reduce all degree-two vertices. To be specific, R8 covers the cases where the two neighbors are connected by a permanent edge, while R9 covers the remaining cases where the two neighbors are connected by a removable edge or not connected. Examples of $R8$ and $R9$ are provided in the online supplementary file.

R8 (Permanently Connected Neighbors Reduction). *Let* u *be a vertex with exactly two neighbors* x, y *and* $\{x,y\} \in E_p$*. The following table shows the rules for reducing* u *under different conditions.*

conditions		*reduction*
Case 1: $w(u) \geq \max(p(\{u,x\}), p(\{u,y\}))$	$w(u) \geq 0$	**R8.1.1:** u *is in at least one set* $I \in MGIS(G)$*,* $\alpha(G) = \alpha(G') + w(u)$*, where* $G' \coloneqq G[V \setminus \{u\}]$*,* $w'(x) \coloneqq w(x) - p(\{u,x\})$*,* $w'(y) \coloneqq w(y) - p(\{u,y\})$.
	$w(u) < 0$	**R8.1.2:** u *is in at least one set* $I \in MGIS(G)$ *iff* $x \in I$ *or* $y \in I$*,* $\alpha(G) = \alpha(G')$*, where* $G' \coloneqq G[V \setminus \{u\}]$*,* $w'(x) \coloneqq w(x) + w(u) - p(\{u,x\})$*,* $w'(y) \coloneqq w(y) + w(u) - p(\{u,y\})$.
Case 2: $p(\{u,x\}) > w(u) \geq p(\{u,y\})$ *w.l.o.g, assume* $p(\{u,x\}) > p(\{u,y\})$	$w(u) \geq 0$ *and* $w(u) \geq \tilde{w}(x)$	**R8.2.1:** u *is in at least one set* $I \in MGIS(G)$ *and* $x \notin I$*,* $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u,x\}]$*, and* $w'(y) \coloneqq w(y) - p(\{u,y\})$.
	$w(u) \geq 0$ *and* $w(u) < \tilde{w}(x)$	**R8.2.2:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u\}]$*,* $w'(x) \coloneqq w(x) - w(u)$*,* $w'(y) \coloneqq w(y) - p(\{u,y\})$*.* u *is in at least one set* $I \in MGIS(G)$ *iff* x *is not in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$	**R8.2.3:** $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq G[V \setminus \{u\}]$*,* $w'(y) \coloneqq w(y) + w(u) - p(\{u,y\})$*.* u *is in at least one set* $I \in MGIS(G)$ *iff* y *is in at least one set* $I' \in MGIS(G')$.
Case 3: $\min(p(\{u,x\}), p(\{u,y\})) > w(u)$ *w.l.o.g, assume* $\tilde{w}(x) \geq \tilde{w}(y)$	$w(u) \geq 0$ *and* $w(u) \geq \tilde{w}(x)$	**R8.3.1:** u *is in at least one set* $I \in MGIS(G)$*,* $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u,x,y\}]$
	$w(u) \geq 0$ *and* $\tilde{w}(y) \leq w(u) < \tilde{w}(x)$	**R8.3.2:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u,y\}]$*,* $w'(x) \coloneqq w(x) - w(u)$*.* u *is in at least one set* $I \in MGIS(G)$ *iff* x *is not in at least one set* $I' \in MGIS(G')$.
	$w(u) \geq 0$ *and* $w(u) < \tilde{w}(y)$	**R8.3.3:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u\}]$*,* $w'(x) \coloneqq w(x) - w(u)$*,* $w'(y) \coloneqq w(y) - w(u)$*.* u *is in at least one set* $I \in MGIS(G)$ *iff both* x *and* y *are not in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$	**R8.3.4:** u *is not in any* $I \in MGIS(G)$*,* $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq G[V \setminus \{u\}]$.

For ease of describing $R9$-$R11$, we further assume $p(\{u,v\}) = 0$ for $\{u,v\} \notin E$. Note that this does not alter the objective value in the graph. Then, for every pair of vertices $\{u,v\}$, there is a corresponding penalty $p(\{u,v\})$.

R9 (Non-permanently Connected Neighbors Reduction). *Let u be a vertex with two neighbors x and y. Assume $p(\{u,x\}) \geq p(\{u,y\})$ without loss of generality. If $\{x,y\} \in E_r$ or $\{x,y\} \notin E$, the following table summarizes the reduction rules under different conditions.*

conditions		*reduction*
Case 1: $w(u) \geq p(\{u,x\})$	$w(u) \geq 0$ *and* $w(u) \geq p(\{u,x\}) + p(\{u,y\})$	**R9.1.1:** u *is in at least one set* $I \in MGIS(G)$, $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq G[V \setminus \{u\}]$, $w'(x) \coloneqq w(x) - p(\{u,x\})$, $w'(y) \coloneqq w(y) - p(\{u,y\})$
	$w(u) \geq 0$ *and* $w(u) < p(\{u,x\}) + p(\{u,y\})$	**R9.1.2:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(x) \coloneqq w(x) - p(\{u,x\})$, $w'(y) \coloneqq w(y) - p(\{u,y\})$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) - p(\{u,x\}) - p(\{u,y\}) + w(u)$ u *is in at least one set* $I \in MGIS(G)$ *iff either* x *or* y *is not in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$	**R9.1.3:** $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(x) \coloneqq w(x) + w(u) - p(\{u,x\})$, $w'(y) \coloneqq w(y) + w(u) - p(\{u,y\})$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) + w(u)$. u *is in at least one set* $I \in MGIS(G)$ *iff either* x *or* y *is in at least one set* $I' \in MGIS(G')$.
Case 2: $p(\{u,x\}) > w(u) \geq p(\{u,y\})$	$w(u) \geq 0$ *and* $w(u) \geq p(\{u,x\}) + p(\{u,y\})$	**R9.2.1:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(y) \coloneqq w(y) - p(\{u,y\})$, $w'(x) \coloneqq w(x) - w(u)$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) + p(\{u,x\}) - w(u)$. u *is in at least one set* $I \in MGIS(G)$ *iff either* x *is not in or* y *is in at least one set* $I' \in MGIS(G')$.
	$w(u) \geq 0$ *and* $w(u) < p(\{u,x\}) + p(\{u,y\})$	**R9.2.2:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(y) \coloneqq w(y) - p(\{u,y\})$, $w'(x) \coloneqq w(x) - w(u)$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) - p(\{u,y\})$. u *is in at least one set* $I \in MGIS(G)$ *iff* x *is not in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$ *and* $w(u) \geq p(\{u,x\}) + p(\{u,y\})$	**R9.2.3:** $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(y) \coloneqq w(y) + w(u) - p(\{u,y\})$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) + p(\{u,x\})$. u *is in at least one set* $I \in MGIS(G)$ *iff* y *is in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$ *and* $w(u) < p(\{u,x\}) + p(\{u,y\})$	**R9.2.4:** $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(y) \coloneqq w(y) + w(u) - p(\{u,y\})$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) + w(u) - p(\{u,y\})$. u *is in at least one set* $I \in MGIS(G)$ *iff* x *is not in and* y *is in at least one set* $I' \in MGIS(G')$.
Case 3: $p(\{u,y\}) > w(u)$	$w(u) \geq 0$	**R9.3.1:** $\alpha(G) = \alpha(G') + w(u)$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $w'(x) \coloneqq w(x) - w(u)$, $w'(y) \coloneqq w(y) - w(u)$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) - w(u)$. u *is in at least one set* $I \in MGIS(G)$ *iff both* x *and* y *are not in at least one set* $I' \in MGIS(G')$.
	$w(u) < 0$ *and* $w(u) < p(\{u,x\}) + p(\{u,y\})$	**R9.3.2:** u *is not in any* $I \in MGIS(G)$, $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq G[V \setminus \{u\}]$.
	$w(u) < 0$ *and* $w(u) \geq p(\{u,x\}) + p(\{u,y\})$	**R9.3.3:** $\alpha(G) = \alpha(G')$ *where* $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,y\}\})$, $p'(\{x,y\}) \coloneqq p(\{x,y\}) - w(u) + p(\{u,x\}) + p(\{u,y\})$. u *is in at least one set* $I \in MGIS(G)$ *iff both* x *and* y *are in at least one set* $I' \in MGIS(G')$.

4.3.3. Low Permanent-degree Reductions We further introduce two reduction rules to reduce high-degree vertices that are connected by only one or two vertices by permanent edges.

R10 (Permanent Degree-one Vertices Reduction). *If there exists a vertex u such that u has exactly one neighbor $x \in N_p(u)$ and $w(u) \geq \sum_{v \in N_r(u)} max(0, p(\{u,v\}))$, then $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x,v\} \mid v \in N_r(u) \setminus N_p(x)\})$, $w'(x) \coloneqq w(x) - w(u)$, $w'(v) \coloneqq w(v) - p(\{u,v\})$ for each $v \in N_r(u)$, and $p'(\{x,v\}) \coloneqq p(\{x,v\}) - p(\{u,v\})$ for each $v \in N_r(u) \setminus N_p(x)$. u is in at least one set $I \in MGIS(G)$ iff x is not in at least one set $I' \in MGIS(G')$.*

R11 (Permanent Degree-two Vertices Reduction). *Let u be a vertex with exactly two neighbors $x, y \in N_p(u)$, $\{x, y\} \in E_p$, and $w(u) \geq \sum_{v \in N_r(u)} max(0, p(\{u, v\}))$. W.l.o.g, assume $\tilde{w}(x) \geq \tilde{w}(y)$.*

Case 1: If $w(u) \geq \tilde{w}(x)$, then u must be included in at least one set $I \in MGIS(G)$ and $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq G[V \setminus \{u, x, y\}]$, and $w'(v) \coloneqq w(v) - p(\{u, v\})$ for each $v \in N_r(u)$.

Case 2: If $\tilde{w}(y) \leq w(u) < \tilde{w}(x)$, then y is not in at least one set $I \in MGIS(G)$ and $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq (V \setminus \{u, y\}, E(G[V \setminus \{u, y\}]) \cup \{\{x, v\} \mid v \in N_r(u) \setminus N_p(x)\})$, $w'(x) \coloneqq w(x) - w(u)$, $w'(v) \coloneqq w(v) - p(\{u, v\})$ for each $v \in N_r(u)$, and $p'(\{x, v\}) \coloneqq p(\{x, v\}) - p(\{u, v\})$ for each $v \in N_r(u) \setminus N_p(x)$. u is in at least one set $I \in MGIS(G)$ iff x is not in at least one set $I' \in MGIS(G')$.

Case 3: If $w(u) < \tilde{w}(y)$, then $\alpha(G) = \alpha(G') + w(u)$ where $G' \coloneqq (V \setminus \{u\}, E(G[V \setminus \{u\}]) \cup \{\{x, v\} \mid v \in N_r(u) \setminus N_p(x)\} \cup \{\{y, v\} \mid v \in N_r(u) \setminus N_p(y)\})$, $w'(x) \coloneqq w(x) - w(u)$, $w'(y) \coloneqq w(y) - w(u)$, $w'(v) \coloneqq w(v) - p(\{u, v\})$ for each $v \in N_r(u)$, $p'(\{x, v\}) \coloneqq p(\{x, v\}) - p(\{u, v\})$ for each $v \in N_r(u) \setminus N_p(x)$, and $p'(\{y, v\}) \coloneqq p(\{y, v\}) - p(\{u, v\})$ for each $v \in N_r(u) \setminus N_p(y)$. u is in at least one set $I \in MGIS(G)$ iff both x and y are not in at least one set $I' \in MGIS(G')$.

4.4. Vertex-pair Reductions

We introduce three additional rules that can reduce some specific vertex pairs.

R12 (Permanent Edge Reduction). *If there is an edge $\{u, v\}$ such that $\{u, v\} \in E_p$ and $w(u) \geq \tilde{w}(v) + w^+(N_p(u) \setminus N_p[v]) + \min(w^+(N_r(u) \setminus N_p(v)), \sum_{x \in N_r(u) \setminus N_p(v)} \max(0, p\{u, x\}))$, then v is not in at least one $I \in MGIS(G)$ and $\alpha(G) = \alpha(G')$ where $G' \coloneqq G[V \setminus \{v\}]$.*

R13 (Common Neighbors Reduction). *If there is an edge $\{u, v\}$ such that $\{u, v\} \in E_p$ and $w(v) \geq w^+(N(v)) - \max(0, w(u))$, then $N_p(u) \cap N_p(v)$ is not a subset of at least one $I \in MGIS(G)$ and $\alpha(G) = \alpha(G')$, where $G' \coloneqq G[V \setminus (N_p(u) \cap N_p(v))]$.*

R14 (Twin Vertices Reduction). *If there exist two vertices u, v such that $N_p(u) = N_p(v)$, $w(u) \geq \sum_{x \in N_r(u)} \max(0, p(\{u, x\}))$ and $w(v) \geq \sum_{x \in N_r(v)} \max(0, p(\{v, x\}))$, then $\alpha(G) = \alpha(G')$, where G' is obtained by folding u and v into a vertex v', $w(v') \coloneqq w(u) + w(v) - p(\{u, v\})$, add edge $\{v', x\}$ as a permanent edge in G' for each $x \in N_p(u)$, and add edge $\{v', x\}$ as a removable edge in G' with $p'(\{v', x\}) \coloneqq p(\{u, x\}) + p(\{v, x\})$ for each $x \in N_r(v) \cup N_r(u)$. Both u and v belong to at least one $I \in MGIS(G)$ iff $v' \in MGIS(G')$.*

5. Reduction-driven Local Search

Now, we elaborate on the local search algorithm driven by these 14 reduction rules for solving the GIS problem. This reduction-driven local search (RLS) algorithm is presented in Algorithm 1.

Algorithm 1: The reduction-driven local search for GIS

Input: a graph $G = (V, E)$ with vertex profits and edge penalties, cutoff time

Output: the best solution S^* found

$G, S_{init} \leftarrow Complete\_Reduction(G)$; /* Pre-process to obtain a kernel graph and partial solution via $R1$–$R14$ */

while *cutoff time is not satisfied* **do**

  /* construct an initial solution using $R1$–$R14$ */

  $S \leftarrow S_{init} \cup Random\_Peeling(G)$;

  /* improve S via local search and partial reductions */

  $S \leftarrow Neighborhood\_Search\_with\_Partial\_Reduction(G, S)$;

  /* update the best visited solution */

  if $nb(S) > nb(S^*)$ **then** $S^* \leftarrow S$;

return S^*;

The RLS begins by pre-processing the original graph G. It uses the *Complete_Reduction* procedure which applies all the reduction rules to obtain a *kernel graph*, that is, a graph where no further reduction rules can be applied. Due to the correctness of the reduction rules, solving the GIS problem in the original graph is equivalent to solving it in this kernel graph. The S_{init} is a partial vertex set that must be a subset of at least one optimal solution.

The remaining steps of the RLS follow the traditional iterated local search (Glover and Laguna (1998), Lourenço et al. (2019)): At each iteration, it supplements S_{init} with a random generalized independent set obtained by *Random_Peeling*, then uses *Neighborhood_Search_with_Partial_Reduction* to improve the current solution S, while keeping the best-known solution in S^*. The iteration continues until the cutoff time limit is reached. Compared with traditional local search, RLS seamlessly integrates reduction techniques into the pre-processing, initial solution construction, and neighborhood search.

5.1. Complete Reduction as Pre-processing

The Complete_Reduction algorithm exhaustively applies the 14 reduction rules described in Section 4. A naive implementation checks each rule independently, processes the graph when applicable, and stops when no rules apply. Checking all these rules required $O(|V| + |E|)$ time, as the entire graph must be scanned for each rule. For large real-world graphs, this is inefficient. We

Algorithm 2: The Complete_Reduction algorithm

Input: a graph $G = (V, E)$ with vertex profits and edge penalties

Output: updated graph G and partial solution S

1 $S \leftarrow \emptyset$; /* initial an empty solution */

/* initial candidate sets for 14 reduction rules */

2 $\text{Cand}_i \leftarrow E_r$ for $i = 1, 2$; $\text{Cand}_i \leftarrow V$ for $i = 3, 4, 6$; $\text{Cand}_5 \leftarrow \{v \in V \mid w(v) < 0\}$;

3 $\text{Cand}_7 \leftarrow \{v \in V \mid |N(v)| = 1\}$; $\text{Cand}_8 \leftarrow \{v \in V \mid |N(v)| = 2\}$; $\text{Cand}_9 \leftarrow \text{Cand}_8$;

4 $\text{Cand}_{10} \leftarrow \{v \in V \mid |N_p(v)| = 1\}$; $\text{Cand}_{11} \leftarrow \{v \in V \mid |N_p(v)| = 2\}$;

5 $\text{Cand}_i \leftarrow E_p$ for $i = 12, 13$; $\text{Cand}_{14} \leftarrow V \times V \setminus E_p$;

6 **while** $\exists i : Cand_i \neq \emptyset$ **do**

/* exhaustively apply reduction rules */

7 **for** *each* $i = 1, 2, \ldots, 14$ **do**

8 **for** *each* $elem \in Cand_i$ **do**

9 **if** *the condition of Ri is satisfied by checking* $elem$ **then**

10 change the structure, profits and penalties of G and set S according to Ri;

11 update the Cand_j if Rj reachable from Ri in Figure 2;

12 Remove $elem$ from Cand_i;

13 **return** G, S;

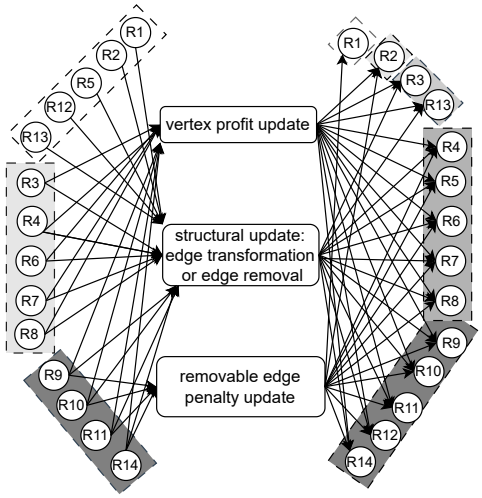
propose a more efficient implementation that incrementally detects applicable rules. Specifically, rules are interdependent, as applying one may trigger others. For example, if only $R1$ is satisfied in graph G (i.e., there exists an edge $\{u, v\} \in E_r$ with $p(\{u, v\}) = 0$), then removing edge $\{u, v\}$ may enable $R3$-14, while other rules remain inapplicable. We analyze these dependencies in depth and summarize them in Figure 2.

In Algorithm 2, we implement this reduction algorithm. It takes a graph G as input and outputs a kernel graph and a set $S \subseteq V$. It initializes a special *candidate set* Cand_i, containing vertices, edges, or vertex pairs, for a rule Ri where $i = 1, ..., 14$. Lines 1-4 show how Cand_i is initialized. When an element $elem \in \text{Cand}_i$ satisfies the condition of Ri, the current graph G is changed and the current set S is updated according to Ri. Along with the change of graph G, if another reduction rule Rj can be applied (i.e., Rj is reachable from Ri in the dependency graph in Figure 2), then the vertices, edges, or vertex pairs which are "neighbor elements" of $elem$ are moved into Cand_j. The process

continues recursively until all candidate sets are empty. Therefore, the algorithm checks whether each reduction rule is satisfied by tracking the candidate set in an incremental manner.

In our implementations, $R1$ and $R2$ are prioritized because the removal of edges may enable more reduction rules. $R3$-$R6$ are assigned secondary priority, as they directly identify vertices that are in at least one set in MGIS(G), which may further shrink the candidate set. $R7$-$R11$ are of the third priority because they can identify vertices that are in at least one set in MGIS(G) in some cases. $R12$-$R14$ are put at the last because they are not able to identify such vertices.

Figure 2 **The dependency graph of the reduction rules. A directed edge on the left side indicates how the reduction rule changes the graph G to G'. A directed edge on the right side indicates what reduction rules can be satisfied due to this change.**



Algorithm 3: The Random_Peeling algorithm

Input: a graph $G = (V, E)$ with vertex profits and edge penalties

Output: a solution S to the GIS problem

initialize an empty set S;
while $|V \setminus N_p[S]| > 0$ **do**
    $u \leftarrow$ randomly select a vertex from $V \setminus N_p[S]$;
    $G, S \leftarrow Complete\_Reduction(G[V \setminus \{u\}], S)$;
extend S to be a maximal generalized independent set;
return S;

5.2. Random Peeling for Initial Solutions

Random-peeling is a simple random procedure for generating an initial solution. It starts with an empty vertex set S. When $V \setminus N_p[S]$ is not empty (meaning that S can possibly be a larger generalized independent set), a vertex u is picked randomly from $V \setminus N_p[S]$ and u is moved into S. After the removal of u from G, the Complete_Reduction is used again to further shrink the remaining graph G (line 4). When $V \setminus N_p[S]$ becomes empty, we further extend S by randomly picking a vertex v from the randomly removed vertices such that $v \notin N_p[S]$ and $nb(S \cup \{v\}) > nb(S)$. The S is returned when there is no other generalized independent set S' such that $S \subset S'$ and $nb(S') \geq nb(S)$. Clearly S is a maximal generalized independent set.

5.3. Neighborhood Search

To improve the current solution, we use an iterated tabu search with three *move operators*.

Move operators. In the RLS, there are three move operators that update the current solution: ADD, SWAP, and the novel REDUCE operator based on two reduction rules. Given the current solution $S \subseteq V$, the ADD(v) operator displaces a vertex v (if any) from $V \setminus N_p[S]$ to the S and the SWAP(u, v) operator adds a vertex $v \in N_p(S)$ if $|N_p(v) \cap S| = 1$ and removes the only vertex $u \in N_p(v) \cap S$ from the current solution S. Indeed, ADD and SWAP are widely used in existing local search methods such as Zhou et al. (2017), Zhou and Hao (2017). The REDUCE($V \setminus N_p[S]$) operator applies $R3$ and $R4$ to include candidate vertices that are guaranteed to be in a set in $MGIS(G)$.

Fast Calculation of Move Gain. For each vertex $u \in V$, we maintain $B_u = w(u) - \sum_{v \in S, \{u,v\} \in E_r} p(\{u, v\})$. Then, the move gain of $ADD(v)$—the gain of the net benefit after applying a $ADD(v)$, is B_v, and the move gain of $SWAP(u, v)$ is $B_v - B_u$. After an $ADD(v)$ move, for each $u \in N_r(v)$, we update B_u as $B_u \coloneqq B_u - p(\{u, v\})$. After a $SWAP(u, v)$ move, for each $x \in N_r(u)$, $B_x \coloneqq B_x + p(\{x, u\})$ and for each $y \in N_r(v)$, $B_y \coloneqq B_y - p(\{y, v\})$. The time complexity of performing ADD or $SWAP$ is $O(\Delta)$, and of REDUCE is $O(|E|)$. The overall complexity of applying all three operators is $O(|E|)$. Notably, the REDUCE operator is not limited to the GIS problem and can enhance the efficiency of local search in other optimization problems.

Neighborhood Search with Tabu and Perturbation. Neighborhood search in RLS iteratively performs the three basic operators above to improve the current solution S, as shown in Algorithm 4.

At each iteration in lines 1-13, it first builds M_1, the set of vertices that can be added into S by ADD, and M_2, the set of vertices that can be swapped into S by $SWAP$. By evaluating M_1 and M_2, ADD or $SWAP$ with the largest move gain is applied, as presented in lines 4-8. After performing

Algorithm 4: The Neighborhood_Search_with_Partial_Reduction algorithm

Input: a graph $G = (V, E)$ with vertex profits and edge penalties, initial solution S, search tolerance $L \in \mathbb{N}$, perturbation parameter $\epsilon \in (0, 1)$

Output: a solution S to the GIS problem

1 $S^* \leftarrow S$, $tabu\_list \leftarrow \emptyset$;
2 **while** *True* **do**
3 $M_1 \leftarrow V \setminus N_p[S]$, $M_2 \leftarrow \{v \in N_p(S) \mid u \in S, \{u, v\} \in E_p, |N_p(v) \cap S| = 1\} \setminus tabu\_list$
4 $v \leftarrow$ a vertex with the largest move gain from $M_1 \cup M_2$;
5 **if** $v \in M_1$ **then** $S \leftarrow S \cup \{v\}$, $M_1 \leftarrow M_1 \setminus \{v\}$ /* add v into S */
6 **else if** $v \in M_2$ **then**
7 $u \leftarrow N_p(v) \cap S$, $S \leftarrow S \cup \{v\} \setminus \{u\}$; /* Swap v and u */
8 add u into $tabu\_list$ with tabu tenure $T_u = 10 + random(0, |M_2|)$
9 $REDUCE(M_1)$ and reassign S; /* apply $R3$ and $R4$ */
10 **if** $nb(S) > nb(S^*)$ **then** $S^* \leftarrow S$, $l \leftarrow 0$ **else** $l \leftarrow l + 1$
11 **if** $l > |S|$ *or* $M_1 = \emptyset \& M_2 = \emptyset$ **then**
12 sequentially drop $\epsilon|S|$ vertices with the least move gain from S, ties breaking randomly; /* Perturbation */
13 **if** $l > L$ **then** break;
14 **return** S^*;

ADD or $SWAP$, we additionally use the REDUCE operator to add more vertices to S. This narrows the search space $M_1 = V \setminus N_p[S]$ of ADD.

To avoid short-term search cycles, a tabu list is used to exclude recently visited vertices for successive iterations (line 8). The tabu tenure for a vertex is set to $10 + random(0, |M_2|)$, as commonly used in local search algorithms (Wu et al. 2023, Zhou et al. 2017, Zhou and Hao 2017). To further avoid local optima, a perturbation step is applied during neighborhood search (lines 11-12). Specifically, when we detect that no operators can be applied or the best solution remains unchanged after a certain number of iterations, we remove a portion of low-contribution vertices. The whole neighborhood search terminates when the consecutive number of iterations without improvement reaches the search tolerance parameter L (line 13).

6. Computational Experiments

This section is dedicated to a comprehensive evaluation of the proposed reduction rules and heuristic approaches. To evaluate the algorithm performance, we compare both solution quality and runtime between the RLS and other benchmark algorithms. To evaluate the effect of the reduction rules, we compare the algorithms with and without reductions; we also conduct an ablation study to evaluate the effect of each reduction rule.

6.1. Datasets

The proposed reduction rules and the RLS algorithm are evaluated on three sets of $216 + 47 + 15 = 278$ instances, covering various application scenarios.

SET-1. The first set of instances consists of 216 standard instances with up to 400 vertices and up to $71,820$ edges. These instances were first introduced by Colombi et al. (2017) and tested in Nogueira et al. (2021), Hosseinian and Butenko (2019), Zheng et al. (2024). This set of instances is available at https://or-dii.unibs.it/instances/Instances_GISP.zip.

SET-2. The second set of instances consists of 47 medium-sized instances with up to $17,903$ vertices and up to $1,997,732$ edges, introduced by Zheng et al. (2024) as benchmark instances for the GIS problem (available at https://github.com/m2-Zheng/GISP/tree/main).

SET-3. To further evaluate the algorithms on large instances and diverse real-world scenarios, we introduce 15 instances with up to 18 million vertices and 261 million edges. The first 4 smaller instances represent ecosystem and road/facility networks, while the remaining 11 larger instances are drawn from hyperlinks, citations, and social networks. The instances were generated using the same method for generating SET-2 (available at Liu et al. (2026)).

6.2. Experimental Settings

All experiments are conducted on a machine with an Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz and 2TB RAM.

Benchmark Algorithms. To evaluate the effectiveness of our proposed reduction rules and the RLS algorithm, we compare them with the best-performing algorithms for the GIS problem.

- CB&B: The exact branch-and-bound algorithm using the UBQP formulation proposed by Hosseinian and Butenko (2019).
- LA-B&B: The exact branch-and-bound algorithm using an Lagrange relaxation techniques by Zheng et al. (2024).
- CPLEX: The mixed integer programming solver, version 22.12 (released in December 2024), using a standard formulation (as shown in the online supplementary file).

- ILS-VND: The iterative local search algorithm proposed by Nogueira et al. (2021).
- ALS: The adaptive local search algorithm proposed by Zheng et al. (2024)
- RLS: Our proposed reduction-driven local search algorithm. We implement RLS in C++.

The source codes of CB&B, LA-B&B, ILS-VND and ALS were made available by the authors. All source codes and figures are available at Liu et al. (2026).

Parameter Settings. The RLS algorithm requires two parameters: the search tolerance L and the perturbation parameter $\epsilon \in (0, 1)$. We set $L = 10|V|$ where $|V|$ is the number of vertices of the input graph and $\epsilon = 0.2$. This setting was obtained using the general IRACE automatic parameter configuration tool (López-Ibáñez et al. 2016). For large instances like *soc-pokec, LiveJ* and *uk2002*, we decrease the search tolerance as $L = 0.5|V|$ to encourage explorations within the cutoff time.

6.3. Performance Comparison with Existing Algorithms

We first compared five algorithms on the SET-1 instances. The two branch & bound algorithms CB&B and LA-B&B ran with a cutoff time of 3 hours (10,800 seconds) for each instance. If an exact algorithm fails to finish within the cutoff time, then we report the best solution found so far. The remaining three heuristic algorithms ran 10 trials with a cutoff time of 30 seconds for each instance. We reported the largest value among the best-known objective values obtained in these trials. The setting is consistent with that in (Hosseinian and Butenko 2019, Nogueira et al. 2021).

In Table 2, we summarize the computational results on SET-1. Extended results are provided in Liu et al. (2026). For each algorithm, we report the number of instances (*#Best*) where the algorithm achieves its best-known objective value (BKV), and the mean running time for the algorithm to first hit the BKV (*htime*). Note that the runtime of the RLS refers to the wall-clock time, including both pre-processing and local search. We observe that, in all instances, the best-known objective values obtained by local search algorithms are comparable to those obtained by exact algorithms. In particular, if an exact algorithm obtains the optimal solution within 3 hours, local search algorithms are also able to find an optimal solution. However, when neither of the exact algorithms finds the optimal solution, local search algorithms always produce a better solution. Besides, the exact LA-B&B dominates CB&B whereas the heuristics ALS and the RLS dominate ILS-VND in terms of both solution quality and runtime. Therefore, we only consider LA-B&B, ALS, and the RLS for even larger instances in SET-2 and SET-3.

Next, we compared the RLS with LA-B&B, ALS and CPLEX on SET-2 and SET-3. Each of the two heuristic ALS and the RLS were run 10 trials and reported its best known objective value. Each trial was given a cutoff time of 30 seconds. In total, each heuristic algorithm ran 300 seconds

Table 2 Summary of comparative results on SET-1 instances

#Total	CB&B		LA-B&B		ILS-VND		ALS		RLS	
	#Best	htime(s)	#Best	htime(s)	#Best	htime(s)	#Best	htime(s)	#Best	htime(s)
216	126	5,816.0	155	4,157.9	215	2.0	216	1.0	216	1.0

per instance in SET-2. The settings are the same as in Zheng et al. (2024). The cutoff time of exact algorithms was directly set to 3 hours (10,800 seconds) per instance in SET-2. However, for SET-3, the heuristic trial time was increased to 200 seconds. For large instances like *soc-pokec* and *LiveJ*, the cutoff time of ALS and the RLS were given 1,000 seconds per trial. For the larger *uk2002*, the cutoff time of LA-B&B and CPLEX was 100,000 seconds, and the cutoff time of ALS and the RLS is 10,000 seconds per trial. In particular, for the three instances *soc-pokec*, *LiveJ* and *uk2002*, the RLS excluded $R6$ and $R12$–14, as these reductions exhibited considerably slower performance compared to the other rules in the experiments. For LA-B&B and CPLEX, if the algorithm did not finish within the total running time, the best solution found so far was reported.

The summary of comparative results on SET-2 and SET-3 are presented in Tables 3 and 4. Detailed results are provided in Tables 1 and 2 in the online supplementary file. The first column groups instances into subgroups according to the number of edges. For RLS, we additionally provide the number of empty kernel graphs ($\#|V_{\text{ker}}| = 0$), the average ratio between kernel graph size and the original graph size ($|V_{\text{ker}}|/|V|$), and the average time of generating a kernel ($time_{ker}$). Note that RLS's htime includes the pre-processing time for building graph kernels. OOT indicates no feasible solution is found within the cutoff time.

Table 3 Summary of comparative results of RLS on SET-2 instances

Instance Subgroup	#Total	CPLEX		LA-B&B		ALS		RLS				
		#Best	htime(s)	#Best	htime(s)	#Best	htime(s)	#Best	htime(s)	$\#\lvert V_{\text{ker}}\rvert = 0$	$\lvert V_{\text{ker}}\rvert/\lvert V\rvert$	time_{ker}(s)
$1.9K < \lvert E_p \cup E_r\rvert \le 40K$	18	18	34.4	0	1,944.9	2	9.1	16	2.4	11	1.8%	<1
$40K < \lvert E_p \cup E_r\rvert \le 800K$	25	6	9,187.3	1	2,140.7	13	7	21	7.5	0	58.8%	1.4
$800K < \lvert E_p \cup E_r\rvert \le 4M$	4	0	OOT	0	2,396	3	8	4	11	0	100%	3
All instances	47	24	5,819.2	1	2,087.4	18	7.9	**41**	**5.9**	–	–	1.4

Table 4 Summary of comparative results of RLS on SET-3 instances

Instance Subgroup	#Total	CPLEX		LA-B&B		ALS		RLS				
		#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	$\#\lvert V_{\text{ker}}\rvert = 0$	$\lvert V_{\text{ker}}\rvert/\lvert V\rvert$	time_{ker}(s)
$2K < \lvert E_p \cup E_r\rvert \le 800K$	7	7(0)	27.7	2(2)	1,256	3(1)	15.7	6(0)	7.3	3	0.6%	1
$800K < \lvert E_p \cup E_r\rvert \le 262M$	8	4(2)	19,794.5	0(8)	OOT	0(8)	OOM	5(0)	1,528.8	0	20.8%	1,156.9
All instances	15	11(2)	10,570.1	2(10)	1,256	3(9)	15.7	**11(0)**	818.7	–	–	617.5

Table 3 shows that RLS outperforms other methods on SET-2 instances in both best-known objective value and runtime. It is known that the RLS proves optimality if all vertices are reduced

Table 5 Summary of comparative results of algorithms with or without reduction rules

	Instance Subgroup	#Total	CPLEX				LA-B&B				RLS			
			without reduction		with reduction		without reduction		with reduction		without reduction		with reduction	
			#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)	#Best(#Fail)	htime(s)
SET-2	$1.9K < \|E_p \cup E_r\| \leq 40K$	18	18(0)	34.4	18(0)	33.7	0(0)	1,944.9	13(0)	240.7	4(0)	11.3	16(0)	2.4
	$40K < \|E_p \cup E_r\| \leq 800K$	25	4(0)	9,187.3	6(0)	9187.0	1(0)	2,140,7	2(0)	1,913.2	12(0)	8.9	21(0)	7.5
	$800K < \|E_p \cup E_r\| \leq 4M$	4	0(4)	10,800	0(4)	10,800	0(0)	2,396	0(0)	2,396	3(0)	10	4(0)	11
SET-3	$2K < \|E_p \cup E_r\| \leq 800K$	7	7(0)	27.7	7(0)	1	2(2)	1,256	5(0)	42.4	3(0)	35	6(0)	7.3
	$800K < \|E_p \cup E_r\| \leq 262M$	8	2(2)	19,794.5	5(0)	19,296.5	0(8)	OOM	1(3)	4,595.8	0(0)	382.5	4(0)	1,528.8
All instances		62	31(6)	7,096.5	36(4)	6,901.0	3(10)	2,007.5	21(3)	1,441.1	22(0)	**60.8**	**51(0)**	202.6

(i.e., $|V_{kernel}| = 0$), hence 11 out of 47 instances are solved to optimality. In contrast, LA-B&B solves only 1. Similarly, CPLEX fails on instances with more than 40,000 variables. Moreover, CPLEX even fails to report feasible solutions for 4 large instances. In contrast, RLS and ALS reach the BKV in an average of no more than 8 seconds.

Table 4 compares algorithms on SET-3 instances. For each algorithm, we additionally report the number of instances where the algorithm fails to report a feasible solution (*#Fail*). OOM indicates out of memory. Both ALS and LA-B&B quickly become infeasible as instance size grows. CPLEX again fails to report a feasible solution for large instances. In contrast, RLS computes feasible solutions even for large instances with 260 million edges such as *uk2002*. For instances where the other algorithms are feasible, RLS consistently achieves the best net benefit. For the infeasible instance *Email*, RLS even finds the optimal solution.

6.4. Evaluation of Reduction Rules

To evaluate the effectiveness of the reduction rules, we conducted the following two experiments.

Comparative results with vs. without reduction. This experiment evaluates algorithms with and without data reduction. For LA-B&B and CPLEX with reductions, we used kernel graphs as input. For RLS without reduction, all reductions were skipped, and random initialization was applied. ALS cannot handle kernel graphs due to unsupported negative vertex profits and edge penalties.

The summary of comparative results are shown in Table 5. More detailed results are provided in the Table 3 in the online supplementary file. In general, the reduction rules can help all algorithms to find better solutions. With the reductions, LA-B&B can solve instances it cannot solve without them, e.g., *WikiTalk* (1.6 million vertices, 11.1 million edges). For some large instances where the original LA-B&B runs out-of-memory, the new LA-B&B with reductions can find feasible solutions or solve them to optimality within the cutoff time. CPLEX also benefits using adding reductions. It solves some instances faster. For two large instances *LiveJ* and *uk2002* where CPLEX fails, CPLEX with reductions can report feasible solutions. For RLS, the reductions consistently lead to better results across all instances, with a shorter time than without them. with reduction rules, it consistently finds better results across all instances in a shorter time than without them.

Figure 3 **Variation of kernel sizes when ablating reductions. Each cell shows the difference in kernel size between using and not using the corresponding reductions; higher values indicate larger effects.**

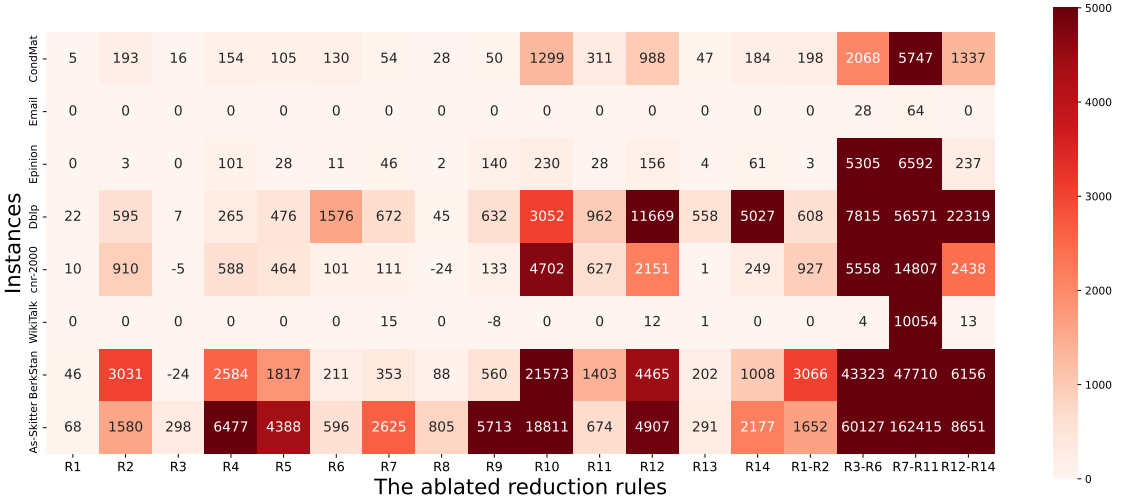
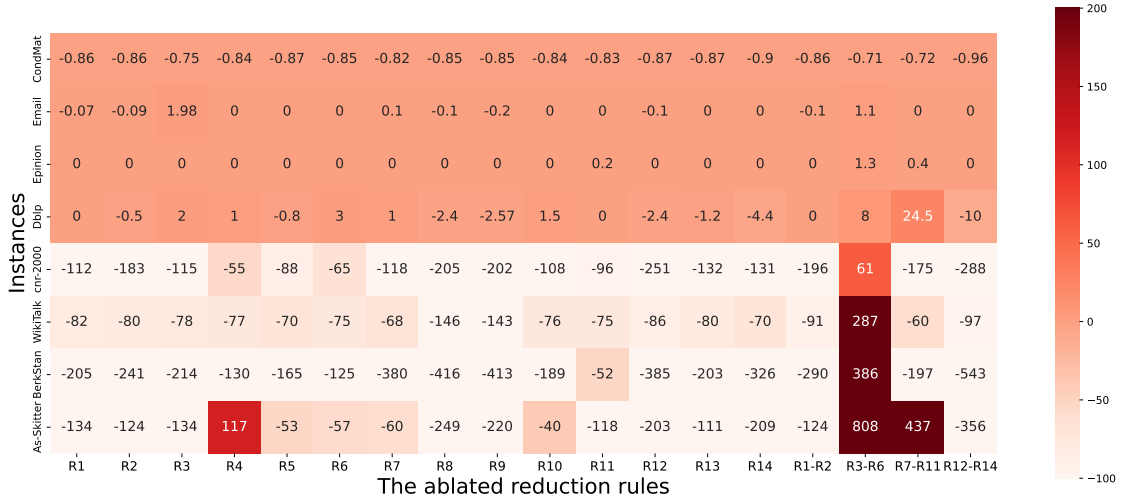


Figure 4 **Variation of pre-processing time when ablating reductions. Each cell shows the difference in time (seconds) between using and not using the corresponding reductions; higher values indicate larger effects.**



Ablation study of reduction rules. We further compare kernel sizes (number of vertices) and pre-processing time (runtime of Algorithm 2) after ablating different reduction rules. Normally, a smaller kernel indicates a smaller search space. Experimental results are shown in Figures 3 and 4.

Figure 3 shows that the impacts on the kernel size vary across reduction rules. Among individual reductions, $R10$ and $R12$ are the most effective in reducing kernel size. Among groups of reductions, the degree-based rules ($R7$–$R11$) are the strongest. The neighborhood-based rules ($R3$–$R6$) are generally more effective than vertex-pair rules ($R12$–$R14$) except for *Dblp* and *WikiTalk*. Although $R1$ and $R2$ do not remove vertices, they still reduce kernel size significantly. In particular, adding $R2$

to the RLS decreases over 3,000 vertices for the *BerkStan* instance. Intuitively, ablating a reduction rule can decrease overall pre-processing time, but may result in a larger kernel. However, Figure 4 shows that ablating $R3$–$R6$ or $R7$–$R11$ actually increase pre-processing time, , as the remaining reduction rules become more time-consuming. For the four larger instances *As-Skitter*, *Berkstan*, *WikiTalk* and *cnr-2000*, including all reductions significantly improves pre-processing time.

7. Conclusion and Future Works

In this paper, we proposed a novel approach to the Generalized Independent Set (GIS) problem using data reduction techniques to tackle the complexity of large graphs. We developed 14 reduction rules, categorized into edge transformation, neighborhood-based, low-degree, and vertex-pair reductions, and designed a reduction-driven local search (RLS) algorithm based on them. Experiments on benchmark instances showed that the RLS algorithm outperformed state-of-the-art methods, especially on large graphs with tens of thousands of vertices. Future work will focus on developing more advanced reduction techniques to further improve efficiency and extending our algorithms to real-world scenarios such as dynamic graphs with changing weights.

References

Abu-Khzam FN, Lamm S, Mnich M, Noe A, Schulz C, Strash D (2022) Recent advances in practical data reduction. *Algorithms for Big Data* 97–133.

Akiba T, Iwata Y (2016) Branch-and-reduce exponential/fpt algorithms in practice: A case study of vertex cover. *Theoretical Computer Science* 609:211–225.

Bai T, Xiao M (2024) Exact algorithms for maximum independent set problem on hypergraphs. *SCIENTIA SINICA Informationis* 54(12):2709.

Brendel W, Amer M, Todorovic S (2011) Multiobject tracking as maximum weight independent set. *CVPR 2011*, 1273–1280 (IEEE).

Brendel W, Todorovic S (2010) Segmentation as maximum-weight independent set. *Proceedings of the 24th International Conference on Neural Information Processing Systems-Volume 1*, 307–315.

Chang L, Li W, Zhang W (2017) Computing a near-maximum independent set in linear time by reducing-peeling. *Proceedings of the 2017 ACM International Conference on Management of Data*, 1181–1196.

Colombi M, Mansini R, Savelsbergh M (2017) The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research* 260(1):41–55.

Crowder H, Johnson EL, Padberg M (1983) Solving large-scale zero-one linear programming problems. *Operations Research* 31(5):803–834.

Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized algorithms*, volume 5 (Springer).

Dahlum J, Lamm S, Sanders P, Schulz C, Strash D, Werneck RF (2016) Accelerating local search for the maximum independent set problem. *Experimental Algorithms: SEA, Proceedings 15*, 118–133 (Springer).

Glover F, Laguna M (1998) *Tabu search* (Springer).

Großmann E, Lamm S, Schulz C, Strash D (2023) Finding near-optimal weight independent sets at scale. *Proceedings of the Genetic and Evolutionary Computation Conference*, 293–302.

Gschwind T, Irnich S, Furini F, Calvo RW (2021) A branch-and-price framework for decomposing graphs into relaxed cliques. *INFORMS Journal on Computing* 33(3):1070–1090.

Hochbaum DS (2004) 50th anniversary article: Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today. *Management Science* 50(6):709–723.

Hochbaum DS, Pathria A (1997) Forest harvesting and minimum cuts: a new approach to handling spatial constraints. *Forest Science* 43(4):544–554.

Hosseinian S, Butenko S (2019) Algorithms for the generalized independent set problem based on a quadratic optimization approach. *Optimization Letters* 13(6):1211–1222.

Koana T, Korenwein V, Nichterlein A, Niedermeier R, Zschoche P (2021) Data reduction for maximum matching on real-world graphs: Theory and experiments. *Journal of Experimental Algorithmics (JEA)* 26:1–30.

Kochenberger G, Alidaee B, Glover F, Wang H (2007) An effective modeling and solution approach for the generalized independent set problem. *Optimization Letters* 1:111–117.

Lamm S, Schulz C, Strash D, Williger R, Zhang H (2019) Exactly solving the maximum weight independent set problem on large real-world graphs. *ALENEX*, 144–158 (SIAM).

Liu Y, Zhou Y, Xu Z, Xiao M, Hao JK (2026) A reduction-driven local search algorithm for generalized independent set problem. URL `http://dx.doi.org/10.1287/ijoc.2025.1203.cd`, available for download at https://github.com/INFORMSJoC/2025.1203.

López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T (2016) The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3:43–58.

Lourenço HR, Martin OC, Stützle T (2019) Iterated local search: Framework and applications. *Handbook of metaheuristics* 129–168.

Mauri GR, Ribeiro GM, Lorena LA (2010) A new mathematical model and a lagrangean decomposition for the point-feature cartographic label placement problem. *Computers & Operations Research* 37(12):2164–2172.

Nogueira B, Pinheiro RG, Tavares E (2021) Iterated local search for the generalized independent set problem. *Optimization Letters* 15:1345–1369.

Pavlik JA, Ludden IG, Jacobson SH, Sewell EC (2021) Airplane seating assignment problem. *Service Science* 13:1–18.

Puthal D, Nepal S, Paris C, Ranjan R, Chen J (2015) Efficient algorithms for social network coverage and reach. *2015 IEEE International Congress on Big Data*, 467–474 (IEEE).

Sanchis LA, Jagota A (1996) Some experimental and theoretical results on test case generators for the maximum clique problem. *INFORMS Journal on Computing* 8(2):87–102.

Stanovic S, Gaüzère B, Brun L (2025) Graph neural networks with maximal independent set-based pooling: Mitigating over-smoothing and over-squashing. *Pattern Recognition Letters* 187:14–20.

Verma A, Buchanan A, Butenko S (2015) Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on computing* 27(1):164–177.

Walteros JL, Buchanan A (2020) Why is maximum clique often easy in practice? *Operations Research* 68(6):1866–1895.

Wu J, Li CM, Wang L, Hu S, Zhao P, Yin M (2023) On solving simplified diversified top-k s-plex problem. *Computers & Operations Research* 153:106187.

Xiao M, Huang S, Zhou Y, Ding B (2021) Efficient reductions and a fast algorithm of maximum weighted independent set. *Proceedings of the Web Conference 2021*, 3930–3940.

Xiao M, Nagamochi H (2017) Exact algorithms for maximum independent set. *Information and Computation* 255:126–146.

Zheng M, Hao JK, Wu Q (2024) Exact and heuristic solution approaches for the generalized independent set problem. *Computers & Operations Research* 164:106561.

Zhou Y, Hao JK (2017) Frequency-driven tabu search for the maximum s-plex problem. *Computers & Operations Research* 86:65–78.

Zhou Y, Hao JK, Goëffon A (2017) Push: A generalized operator for the maximum vertex weight clique problem. *European Journal of Operational Research* 257(1):41–54.

Online Supplement for "A Reduction-Driven Local Search for the Generalized Independent Set Problem"

1. Technical Proofs

Proof of R1. Consider whether u and v are included in a generalized independent set S. There are four cases: $u \in S$ and $v \in S$, $u \in S$ and $v \notin S$, $u \notin S$ and $v \in S$, and $u \notin S$ and $v \notin S$. When either $u \notin S$ or $v \notin S$, the edge $e = \{u, v\}$ does not exist in $G[S]$, and thus removing e has no effect on the net benefit $nb(S)$. If both $u \in S$ and $v \in S$, the edge $\{u, v\}$ has a zero penalty, which has no effect on the net benefit $nb(S)$ either. □

Proof of R2. Assume, for the sake of contradiction, that there exists $I \in MGIS(G)$ such that $u \in I$ and $v \in I$. Assume, without loss of generality, that $\tilde{w}(u) \leq \tilde{w}(v)$. We have $nb(I) - nb(I \setminus \{u\}) = w(u) - \sum_{x \in N_r(u) \cap I} p(\{u, x\}) \leq w(u) - p(\{u, v\}) + \sum_{x \in N_r(u)} max(0, -p(\{u, x\})) = \tilde{w}(u) - p(\{u, v\}) < 0$. In conclusion, $I \setminus \{u\}$ has a larger net benefit without containing permanent edges, which contradicts the optimality of I. Therefore, u and v cannot be simultaneously included in any $S \in MGIS(G)$, which is equivalent to the case where $\{u, v\} \in E_p$. □

Proof of R3. We prove that u is in at least one set in $MGIS(G)$. Assume, for the sake of contradiction, that u is not included in any set in $MGIS(G)$. Let I be an arbitrary set in $MGIS(G)$. It is clear that at least one vertex in $N(u)$ must be included in I. Otherwise, $I \cup \{u\}$ implies with a larger net benefit without including any permanent edge, a contradiction, since $w(u) \geq w^+(N(u))$, $nb(I \cup \{u\} \setminus N(u)) \geq nb(I)$. So u is in at least one set in $MGIS(G)$. It holds that $\alpha(G) = \alpha(G') + w(u)$. □

Proof of R4. We prove that u is in at least one set in $MGIS(G)$. Assume, for the sake of contradiction, that u is not included in any set in $MGIS(G)$. Let I be an arbitrary set in $MGIS(G)$. Let $S := N_p(u) \cap I$ and $S' := N_r(u) \cap I$. Now, we remove S from I and add u to I to obtain I'. Consider the difference between two net benefits $nb(I') - nb(I) \geq w(u) - \sum_{v \in S} \tilde{w}(v) - \sum_{v \in S'} p(\{u, v\}) \geq 0$. Moreover, $G[I']$ does not contain permanent edges. Therefore, $I' \in MGIS(G)$, which contradicts the assumption. So u is in at least one set in $MGIS(G)$. It holds that $\alpha(G) = \alpha(G') + w(u)$. □

Proof of R5. We prove that u is not in any set in $MGIS(G)$. Given an arbitrary $I \in MGIS(G)$, there are two cases. When none of the neighbors of u is included in I, I does not include u because $w(u) < 0$. When any subset of neighbors of u is included in I, I does not include u either because $w(u) - \sum_{v \in S} p(\{u, v\}) < 0$ for any subset $S \subseteq N(u)$. Since I is an arbitrary set in $MGIS(G)$, u will not appear in any $I \in MGIS(G)$ and we can directly remove u from G. □

Proof of R6. Observe that there exists a vertex set $I \in MGIS(G)$ such that $I \cap N_p[u] \neq \emptyset$, because if $I \cap N_p[u] = \emptyset$ then $nb(I \cup \{u\}) = nb(I) + w(u) \geq nb(I)$ implies $I \cup \{u\} \in MGIS(G)$.

Whenever a vertex x in $N_p(u)$ is included in I, other vertices in the clique $N_p[u] \setminus \{x\}$ must not be included in I. Then $I \cup \{u\} \setminus \{x\} \in MGIS(G)$ because $G[I \cup \{u\} \setminus \{x\}]$ does not contain permanent edges, and $nb(I \cup \{u\} \setminus \{x\}) \geq nb(I) + w(u) - w(x) + \sum_{v \in N_r(x)} p(\{v,x\}) - \sum_{v \in N_r(u)} max(0, p(\{u,v\})) \geq nb(I) + w(u) - \sum_{v \in N_r(u)} max(0, p(\{u,v\})) - \max(0, \tilde{w}(x)) \geq nb(I)$. In conclusion, u is in at least one set in $MGIS(G)$. □

Proof of R7. Given an arbitrary $I \in MGIS(G)$.

When $w(u) \geq p(\{u,v\})$ and $w(u) \geq 0$. If $u \notin I$ and $v \in I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) = nb(I) + w(u) - p(\{u,v\}) \geq nb(I)$. If $u \notin I$ and $v \notin I$, then $I \cup \{u\} \in MGIS(G)$ because $G[I \cup \{u\}]$ does not contain permanent edges, and $nb(I \cup \{u\}) = nb(I) + w(u) \geq nb(I)$. In summary, u is included in at least one set in $MGIS(G)$.

When $w(u) < 0$ and $w(u) \geq p(\{u,v\})$. If $v \in I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) = nb(I) + w(u) - p(\{u,v\}) \geq nb(I)$ for $u \notin I$, and $G[I \cup \{u\}]$ does not contain permanent edges. If $v \notin I$, then $nb(I \cup \{u\}) < nb(I \setminus \{u\})$. In other words, there exists a set $S \in MGIS(G)$ such that $u \in S$ if and only if there exists a set $S' \in MGIS(G')$ such that $v \in S'$. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, v that could appear in a set in $MGIS(G)$.

When $w(u) \geq 0$, $w(u) < p(\{u,v\})$ and $w(u) \geq \tilde{w}(v)$. If $u \notin I, v \in I$, then $nb(I \cup \{u\}) \setminus \{v\}) - nb(I) = w(u) - w(v) - \sum_{x \in N_r(v) \cap I} max(0, -p(\{v,x\})) = w(u) - \tilde{w}(v) \geq 0$, implying $I \cup \{u\} \setminus \{v\} \in MGIS(G)$. Otherwise, if $u \notin I, v \notin I$, then $nb(I \cup \{u\}) - nb(I) = w(u) \geq 0$, thus $I \cup \{u\} \in MGIS(G)$. In summary, u is included in at least one vertex set $I' \in MGIS(G)$ and v is not in I'.

When $w(u) \geq 0$, $w(u) < p(\{u,v\})$ and $w(u) < \tilde{w}(v)$. u and v cannot be both included in I. Indeed, if $u, v \in I$, then $nb(I \setminus \{u\}) = nb(I) + p(\{u,v\}) - w(u) > nb(I)$. Moreover, $G[I \setminus \{u\}]$ does not contain permanent edges. This contradicts the optimality of I. Furthermore, if $v \notin I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) - nb(I \setminus \{u\}) = w(u) \geq 0$. We conclude that there exists a vertex set in $MGIS(G)$ that includes u iff there exists a vertex set in $MGIS(G')$ that does not contain v. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, v that could appear in a set in $MGIS(G)$.

Finally, when $w(u) < 0$ and $w(u) < p(\{u,v\})$. We have $nb(I \cup \{u\}) < nb(I \setminus \{u\})$, therefore u is not in any set in $MGIS(G)$ and we can directly remove u. □

Proof of R8. Given an arbitrary $I \in MGIS(G)$. Because that $\{x,y\} \in E_p$, it holds that x and y cannot be both included in I.

Case 1: $w(u) \geq \max(p(\{u,x\}), p(\{u,y\}))$.

When $w(u) \geq 0$. If both x and y are not included in I and $u \notin I$, then $nb(I \cup \{u\}) = nb(I) + w(u) \geq nb(I)$. If $u \notin I$ and either $x \in I$ or $y \in I$, $nb(I \cup \{u\}) \geq nb(I) + w(u) - \max(p(\{u,x\}), p(\{u,y\})) \geq nb(I)$. In summary, there exists a vertex set in $MGIS(G)$ that includes u.

When $w(u) < 0$. If both x and y are not included in I, then $u \notin I$ because $nb(I \cup \{u\}) = nb(I \setminus \{u\}) + w(u) < nb(I \setminus \{u\})$. If either $x \in I$ or $y \in I$ and $u \notin I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) \geq nb(I) + w(u) - \max(p(\{u,x\}), p(\{u,y\})) \geq nb(I)$. We conclude that there exists a vertex set in $MGIS(G)$ that includes u iff there exists a vertex set in $MGIS(G')$ that includes either x or y. In other words, we can defer the decision of u after determining x and y. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, x, y that could appear in a set in $MGIS(G)$.

Case 2: $p(\{u,y\}) \leq w(u) < p(\{u,x\})$. In this case, if y is in I, then $x \notin I$ and $I \cup \{u\} \in MGIS(G)$.

When $w(u) \geq 0$ and $w(u) \geq \tilde{w}(x)$. If x is in I, then y, u are not in I and $I \cup \{u\} \setminus \{x\} \in MGIS(G)$ because $nb(I \cup \{u\} \setminus \{x\}) \geq nb(I) + w(u) - \tilde{w}(x) \geq nb(I)$. If neither x nor y is included in I, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) = nb(I) + w(u) \geq nb(I)$ for $u \notin I$. In summary, there exists a vertex set $I' \in MGIS(G)$ that includes u and not includes x.

When $w(u) \geq 0$ and $w(u) < \tilde{w}(x)$. There are five cases: $x \in I$ and $u, y \notin I$, $y \in I$ and $u, x \notin I$, $u \in I$ and $x, y \notin I$, $u, y \in I$ and $x \notin I$, and $u, x \in I$ and $y \notin I$. But u, x cannot appear in the same I, otherwise $nb(I \setminus \{u\}) = nb(I) + p(\{u,x\}) - w(u) > 0$ without containing permanent edges, which contradicts the optimality of I. If $y \in I$ and $u, x \notin I$, then $I \cup \{u\} \in MGIS(G)$. In summary, there exists a vertex set $I' \in MGIS(G)$ such that either $x \in I'$, or both $u, y \in I'$, or $u \in I'$. Since u only connects to x and y in G, we conclude that there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ that does not include x. In other words, we can defer u after the decision of x. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, x, y that could appear in a set in $MGIS(G)$.

When $w(u) < 0$. If $y \notin I$, then $u \notin I$ because otherwise $nb(I \setminus \{u\}) > nb(I)$. Since u only connects to x and y in G, we conclude that there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ that includes y. In other words, we can defer u after the decision of y. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, x, y that could appear in a set in $MGIS(G)$.

Case 3: $w(u) < p(\{u,x\})$ and $w(u) < p(\{u,y\})$. When $w(u) \geq 0$, the three vertices form a triangle where every edge has penalty M. The result follows from theorem 3.3 of Gu et al. (2021). In particular, (1) if $w(u) \geq 0$ and $w(u) \geq \tilde{w}(x)$, then u must be in at least one set in $MGIS(G)$. (2)

If $w(u) \geq 0$ and $\tilde{w}(y) \leq w(u) < \tilde{w}(x)$, then y must not be in at least one set in $MGIS(G)$. There exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ that does not include x. (3) If $w(u) \geq 0$ and $w(u) < \tilde{w}(y)$, then there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ that does not include both x and y. (4) When $w(u) < 0$, the result follows from R5. It is easy to verify $\alpha(G') = \alpha(G)$ in all conditions and all combinations of u, x, y that could appear in a set in $MGIS(G)$. □

Proof of R9. Given an arbitrary $I \in MGIS(G)$. Notice that x and y can be both included in I.

Case 1: $w(u) \geq p(\{u,x\})$. In this case, if one of x, y is in I, say x, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) - nb(I) \geq w(u) - p(\{u,x\}) \geq 0$ when $u \notin I$.

When $w(u) \geq 0$ and $w(u) \geq p(\{u,x\}) + p(\{u,y\})$. If either $x \in I$ or $y \in I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) - nb(I) \geq w(u) - \max(p(\{u,x\}) + p(\{u,y\}), p(\{u,x\})) \geq 0$ for $u \notin I$. If none of x, y is in I, then $I \cup \{u\} \in MGIS(G)$ because $w(u) \geq 0$. Therefore, u must belong to at least one set in $MGIS(G)$.

When $w(u) \geq 0$ and $w(u) < p(\{u,x\}) + p(\{u,y\})$. If both x, y are in I, then $nb(I \cup \{u\}) < nb(I)$. If either $x \notin I$ or $y \notin I$, then $nb(I \cup \{u\}) \geq nb(I)$. In other words, including u has negative profit only if both x and y are included in the same set in $MGIS(G)$. we conclude that there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that either $x \notin S'$ or $y \notin S'$. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, x, y that could appear in a set in $MGIS(G)$.

When $w(u) < 0$, it holds that $w(u) \geq p(\{u,x\}) + p(\{u,y\})$ because $w(u) \geq p(\{u,x\})$ and $p(\{u,y\}) < 0$. If either $x \in I$ or $y \in I$, then $nb(I \cup \{u\}) \geq nb(I)$. If both $x \notin I$ and $y \notin I$, then $nb(I \cup \{u\}) < nb(I)$. In other words, including u has positive profit if and only if either x or y is included. we conclude that there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that either $x \in S'$ or $y \in S'$. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of u, x, y that could appear in a set in $MGIS(G)$.

Case 2: $p(\{u,y\}) \leq w(u) < p(\{u,x\})$. In this case, we consider whether the three vertices u, x, y can be included in I.

When $w(u) \geq 0$ and $w(u) \geq p(\{u,x\}) + p(\{u,y\})$, the all three vertices may be included in I. All combinations of x, y, u may appear in I. Except that if x is included and y is not included in I, then $u \notin I$ because $nb(I \cup \{u\}) = nb(I \setminus \{u\}) + w(u) - p(\{u,x\}) < nb(I \setminus \{u\})$. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$

such that either $x \notin S'$ or $y \in S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

When $w(u) \geq 0$ and $w(u) < p(\{u,x\}) + p(\{u,y\})$, the three vertices can not be included in I. Moreover, u and x cannot be both included by I because otherwise $nb(I \setminus u) > nb(I)$ without containing permanent edges, which contradicts the optimality of I. If $x \notin I$, then $I \cup \{u\} \in MGIS(G)$ because $nb(I \cup \{u\}) \geq nb(I \setminus \{u\})$. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

When $w(u) < 0$ and $w(u) \geq p(\{u,x\}) + p(\{u,y\})$, the three vertices can be included in I. If $y \notin I$, then $nb(I \cup \{u\}) \leq nb(I \setminus \{u\})$ regardless of whether $x \in I$. If $y \in I$, then $nb(I \cup \{u\}) \geq nb(I \setminus \{u\})$ regardless of whether $x \in I$ In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $y \in S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

When $w(u) < 0$ and $w(u) < p(\{u,x\}) + p(\{u,y\})$, the three vertices can not be included in I. u is in I iff y is included and x is not included in I. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$ and $y \in S'$. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

Case 3: $w(u) < p(\{u,y\})$. When $w(u) \geq 0$. u is in at least one set $I \in MGIS(G)$ iff neither x nor y is included in I. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$ and $y \notin S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

When $w(u) < 0$ and $w(u) < p(\{u,x\}) + p(\{u,y\})$. u is not in any $I \in MGIS(G)$ because otherwise $nb(I \setminus \{u\}) > nb(I)$ without containing permanent edges, a contradiction. Therefore, we can remove u directly.

When $w(u) < 0$ and $w(u) \geq p(\{u,x\}) + p(\{u,y\})$. u is in at least one set $I \in MGIS(G)$ iff both x and y are included in I. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \in S'$ and $y \in S'$. It is easy to verify $\alpha(G') = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$. □

Proof of R10. Observe that there exists a vertex set $I \in MGIS(G)$ that includes at least one of x, u must be included in I. Because if none of them is included in at least one set $I \in MGIS(G)$, then $nb(I \cup \{u\}) - nb(I) \geq w(u) - \sum_{v \in N_r(u)} \max(0, p(\{u,v\})) \geq 0$ and therefore $I \cup \{u\} \in MGIS(G)$.

Given an arbitrary $I \in MGIS(G)$. Furthermore, x and u cannot be both included in I since $\{u,x\} \in E_p$. In summary, there are two possible states in I: either $x \in I, u \notin I$ or $u \in I, x \notin I$. Therefore, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, u that could appear in a set in $MGIS(G)$. □

Proof of R11. Observe that there exists a vertex set $I \in MGIS(G)$ that includes at least one of u, x, y. If none of them is included in I, then $nb(I \cup \{u\}) - nb(I) \geq w(u) - \sum_{v \in N_r(u)} \max(0, p(\{u,v\})) \geq 0$ In addition, at most one of u, x, y can be included in I because $\{u,x\} \in E_p, \{u,y\} \in E_p, \{x,y\} \in E_p$.

Given an arbitrary $I \in MGIS(G)$. When $w(u) \geq \tilde{w}(x)$. If one of x, y is included in I,say x, then $nb(I \cup \{u\} \setminus \{x\}) \geq nb(I)$. In summary, u must be included in at least one set in $MGIS(G)$.

When $\tilde{w}(y) \leq w(u) < \tilde{w}(x)$. If y is included in I, then $u \notin I$ and $nb(I \cup \{u\} \setminus \{y\}) \geq nb(I)$. Therefore, there is a vertex set $I' \in MGIS(G)$ such that $y \notin I'$ and exactly one of u, x is included in I'. We conclude that there exists a vertex set $S \in MGIS(G)$ such that $u \in S$ iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$.

When $w(u) < \tilde{w}(y)$. u is in at least one set $I \in MGIS(G)$ iff both x, y are not included in I. In other words, there exists a vertex set $S \in MGIS(G)$ that includes u iff there exists a vertex set $S' \in MGIS(G')$ such that $x \notin S'$ and $y \notin S'$. It is easy to verify $\alpha(G') + w(u) = \alpha(G)$ in all combinations of x, y, u that could appear in a set in $MGIS(G)$. □

Proof of R12. Given an arbitrary $I \in MGIS(G)$.

When $w(u) \geq \tilde{w}(v) + w^+(N(u) \setminus N_p[v])$. If $v \in I$, then $u \notin I$. We can replace $N(u) \cap I$ with u to get a larger or equal net benefit because $nb(I \cup \{u\} \setminus N(u)) - nb(I) \geq w(u) - \tilde{w}(v) + w^+(N(u) \setminus N_p[v]) \geq w(u) - \tilde{w}(v) + w^+(N(u) \cap I) \geq 0$. Therefore, $I \cup \{u\} \setminus N(u) \in MGIS(G)$. Since $v \in N_p(u)$, there must exist at least one set in $MGIS(G)$ that does not include v.

When $w(u) \geq \tilde{w}(v) + w^+(N_p(u) \setminus N_p[v]) + \sum_{x \in N_r(u) \setminus N_p(v)} \max(0, p(\{u,x\}))$. If $v \in I$, then $u \notin I$. We can replace $N_p(u) \cap I$ with u to get a larger net benefit because $nb(I \cup \{u\} \setminus N_p(u)) - nb(I) \geq w(u) - \tilde{w}(v) + w^+(N_p(u) \setminus N_p(v)) + \sum_{x \in N_r(u) \setminus N_p(v)} \max(0, p(\{u,x\})) \geq w(u) - \tilde{w}(v) + w^+(N(u) \cap I) + \sum_{x \in N_r(u) \cap I} \max(0, p(\{u,x\})) \geq 0$. Since $v \in N_p(u)$, there must exist a set in $MGIS(G)$ that does not include v. □

Proof of R13. Assume, for the sake of contradiction, that neither u nor v is included in any set in $MGIS(G)$. If for any $I \in MGIS(G), N(v) \cap I = \emptyset$, then $I \cup \{v\} \in MGIS(G)$ because $nb(I \cup \{v\}) \geq$

$nb(I)$ without containing permanent edges, a contradiction. Note that $u \notin N(v) \cap I$. If there exists a set $I \in MGIS(G)$ such that $N(v) \cap I \neq \emptyset$, then $w(v) \geq w^+(N(v)) - max(0, w(u)) \geq w^+(S)$. $nb(I \cup \{v\} \setminus N(v)) \geq nb(I)$ without containing permanent edges, again a contradiction. Therefore, there exists a set in $MGIS(G)$ that includes at least one of u, v. We conclude that $N_p(u) \cap N_p(v)$ must not be included in at least one set in $MGIS(G)$. □

Proof of R14. Given an arbitrary $I \in MGIS(G)$. If u is included in I, then $N_p(v) \cap I = \emptyset$. $I \cup \{v\} \in MGIS(G)$ because $nb(I \cup \{v\}) - nb(I \setminus \{v\}) = w(v) - \sum_{x \in N_r(v)} \max(0, p(\{v, x\})) \geq 0$. The same argument holds when v is included in I. Therefore, we can fold u and v into a vertex v'. There exists $S \in MGIS(G)$ such that $u \in S$ and $v \in S$ iff there exists $S' \in MGIS(G')$ such that $v' \in S'$. □

2. Examples of Reduction 8 and 9

Examples of $R8$ are presented in Figure 1. For example, in the first plot, u satisfies case 1 and $w(u) \geq 0$. Then by $R8.1.1$, we can remove u from the original graph G (left) to get the reduced graph G'(right) and $\alpha(G) = \alpha(G') + 10$. Similarly, examples of $R9$ are presented in Figure 2.

Figure 1 **Examples of $R8$. Suppose that vertex u is adjacent to x and y, and $\{x, y\} \in E_p$. G_R represents the residual graph $G[V \setminus \{u, x, y\}]$. α and α' indicate the optimal objective value in the original graph and the graph after reduction, respectively.**

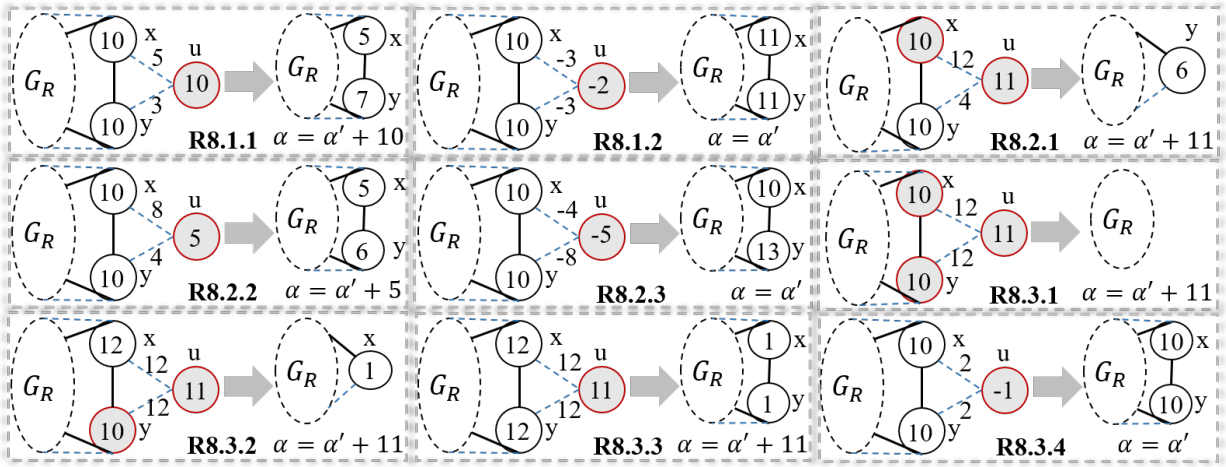
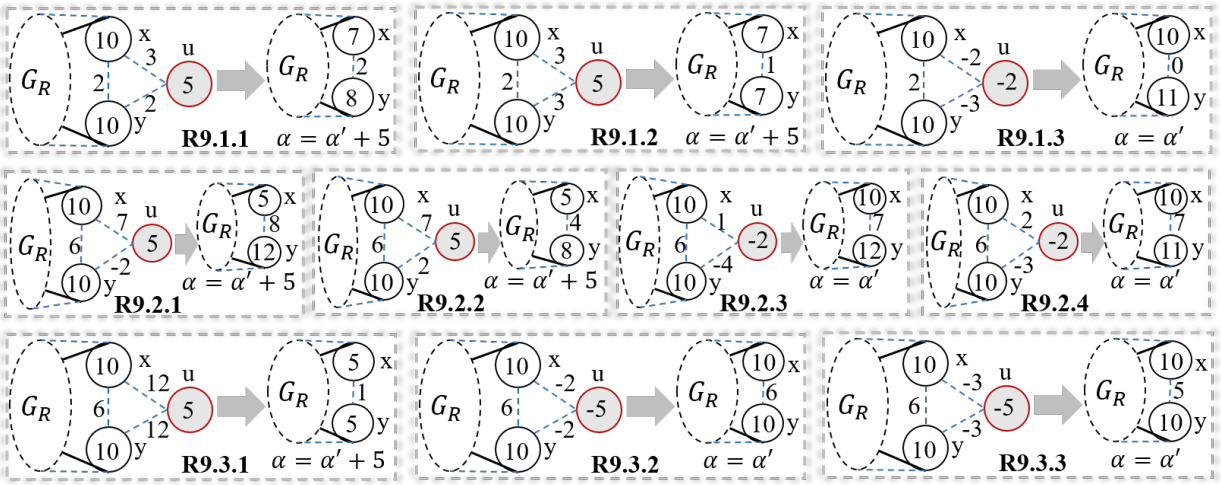


Figure 2 **Examples of $R9$. Suppose that vertex u is adjacent to vertices x and y, and $\{x,y\} \in E_r$ or $\{x,y\} \notin E$.**



3. Integer Programming of the GIS Problem

The GIS problem can be conveniently formulated as follows (Colombi et al. 2017):

$$\alpha(G) = \max \quad \sum_{i \in V} w(i)x_i - \sum_{\{i,j\} \in E_r} p(\{i,j\})y_{ij} \tag{1}$$

$$\text{s.t.} \quad x_i + x_j \leq 1 \qquad \{i,j\} \in E_p \tag{2}$$

$$x_i + x_j - y_{ij} \leq 1 \qquad \{i,j\} \in E_r \tag{3}$$

$$x_i \geq y_{ij}, x_j \geq y_{ij} \qquad \{i,j\} \in E_r \tag{4}$$

$$x_i \in \{0,1\} \qquad i \in V \tag{5}$$

$$y_{ij} \in \{0,1\} \qquad \{i,j\} \in E_r \tag{6}$$

where binary variables x_i indicate whether the vertex i is selected, and binary variables y_{ij} indicate whether the removable edge $\{i,j\}$ is included.

4. Detailed Computational Results

The detailed comparative results on SET-2 and SET-3 are presented in Tables 1 and 2. The first four columns list instance information. The remaining columns show the best-known objective value (*BKV*), the number of vertices in the BKV solution ($|V_I|$), the mean running time for the heuristic algorithm to first hit the BKV (*htime*), the kernel graph size ($|V_{\text{ker}}|$) and the time of generating a kernel ($time_{ker}$). The last two rows report statistical results. Values marked with $*$ denote the optimal objective value. Note that RLS's htime includes the pre-processing time for building graph kernels. The detailed ablation results on SET-2 and SET-3 are presented in Tables 3.

Table 1 Comparative results of RLS on SET-2 instances

Instance	$\|V\|$	$\|E_p\|$	$\|E_r\|$	CPLEX			LA-B&B			ALS			RLS				
				BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	$\|V_{\text{ker}}\|$	time$_{\text{ker}}$(s)
bio-yeast	1,458	980	968	**68,574***	1,111	<1	66,545	1,062	2	68,572	1,113	9	**68,574***	1,111	<1	0	<1
soc-wiki-Vote	889	727	2,187	**37,358***	648	<1	35,973	615	<1	**37,358***	648	8	**37,358***	648	<1	0	<1
vc-exact_131	2,980	1,191	4,169	**109,641***	2,098	<1	107,812	2,045	14	109,392	2,121	14	**109,641***	2,098	<1	0	<1
web-edu	3,031	4,813	1,661	**114,140***	1,752	<1	109,856	1,644	10	113,781	1,746	16	**114,140***	1,753	<1	98	<1
MANN_a81	3,321	3,118	3,362	**117,327***	2,084	<1	114,616	1,999	16	116,907	2,125	14	**117,327***	2,084	<1	0	<1
tech-routers-rf	2,113	1,539	5,093	**97,799***	1,668	<1	93,690	1,588	54	**97,799***	1,668	11	**97,799***	1,667	<1	0	<1
vc-exact_039	6,795	2,574	8,046	**259,542***	5,018	<1	254,941	4,908	253	258,137	5,164	9	**259,542***	5,018	<1	0	<1
ca-GrQc	4,158	3,179	10,243	**174,782***	2,991	9	167,846	2,871	45	174,654	2,997	15	**174,782***	2,994	11	282	<1
vc-exact_038	786	10,321	3,703	**12,015***	208	589	10,406	168	1,994	12,014	208	12	**12,015***	207	4	697	<1
vc-exact_078	11,349	4,289	13,450	**438,671***	8,330	<1	431,371	8,128	1,074	435,412	8,671	<1	**438,671***	8,330	<1	0	<1
vc-exact_087	13,590	5,126	16,114	**518,500***	10,025	2	509,734	9,809	2,319	517,500	10,083	2	**518,500***	10,026	<1	0	<1
vc-exact_107	6,402	15,646	5,594	**407,878***	6,763	<1	396,456	6,488	10,646	400,223	6,846	1	**407,878***	6,763	<1	0	<1
vc-exact_151	15,783	12,144	12,519	**530,138***	9,526	2	516,445	9,135	2,402	522,462	9,764	1	**530,138***	9,526	<1	0	<1
vc-exact_167	15,783	18,235	6,428	**465,178***	7,797	1	449,579	7,438	10,469	457,272	7,923	2	**465,178***	7,797	<1	0	<1
bio-dmela	7,393	18,897	6,672	**302,992***	4,952	<1	280,897	4,474	170	302,449	4,959	10	**302,992***	4,951	<1	44	<1
vc-exact_011	9,877	19,102	6,871	**303,793***	5,369	1	286,137	4,950	4,818	301,789	5,487	7	**303,793***	5,387	<1	172	<1
web-spam	4,767	8,089	28,386	**190,507***	3,285	4.61	180,446	3,130	53	190,344	3,299	14	190,459	3,289	13	487	<1
vc-exact_026	6,140	27,159	9,608	**201,509***	3,887	1	177,902	3,372	668	201,316	3,922	17	201,503	3,900	<1	300	<1
vc-exact_001	6,160	19,655	20,552	**215,941***	4,129	2	194,984	3,709	9,575	215,668	4,175	13	**215,941***	4,128	<1	134	<1
vc-exact_024	7,620	11,367	35,926	**236,109**	4,632	10,783	226,540	4,438	244	235,440	4,735	14	236,018	4,659	12	1,255	<1
C1000.9	1,000	11,989	37,432	5,263	99	10,800	6,104	116	430	7,229	126	12	**7,239**	125	10	1,000	<1
tech-WHOIS	7,476	42,181	14,762	**325,584***	5,368	5	309,903	5,050	218	325,327	5,367	14	325,581	5,371	13	338	<1
p-hat700-3	700	30,398	31,242	4,025	68	10,800	3,030	52	7,436	**4,473**	78	<1	**4,473**	78	<1	700	<1
vc-exact_008	7,537	35,678	37,155	**247,091***	4,800	3	210,126	4,089	9,093	246,771	4,839	14	**247,091***	4,810	<1	574	<1
keller5	776	17,800	56,910	2,373	58	10,800	2,860	54	6,327	**3,877**	69	1	**3,877**	69	2	776	<1
vc-exact_194	1,150	39,520	41,331	3,734	77	10,708	4,386	73	1,324	**5,419**	86	<1	**5,419**	86	<1	1,150	<1
hamming10-4	1,024	66,369	23,231	3,256	52	8,108	3,073	45	2,069	**3,660**	54	<1	**3,660**	54	<1	1,024	<1
brock800_2	800	26,629	84,805	1,177	49	10,800	1,996	36	10	**2,473**	44	<1	**2,473**	43	<1	800	<1
brock800_4	800	26,831	85,126	1,583	48	10,800	1,979	39	7,215	**2,501**	43	<1	**2,501**	42	<1	800	<1
ca-HepPh	11,204	57,721	59,898	**372,096**	6,094	10,800	348,738	5571	422	370,779	6,083	6	372,079	6,098	17	2,165	<1
p-hat700-2	700	90,788	32,134	2,870	41	10,140	2,345	37	2,078	**2,971**	42	<1	**2,971**	42	<1	700	<1
vc-exact_196	1,534	93,273	32,809	3,916	62	10,800	4,273	57	66	**5,321**	76	<1	**5,321**	76	<1	1,534	<1
p-hat700-1	700	135,804	47,847	730	11	10,800	**844***	11	5	**844***	11	<1	**844***	11	<1	700	<1
ca-AstroPh	17,903	47,135	149,837	**604,288**	10,468	10,800	558,736	9,752	2,544	602,320	10,521	2	604,019	10,491	14	4,985	1
C2000.9	2,000	147,548	51,920	3,958	65	9,489	5,477	73	11	**6,396**	86	3	**6,396**	86	3	2,000	<1
DSJC1000.5	1,000	59,694	189,980	925	39	7,645	1,515	30	694	**1,861**	32	1	**1,861**	32	<1	1,000	<1
socfb-MIT	6,402	123,064	128,166	7,098	124	10,800	110,677	1,779	20	129,807	2,142	17	**129,917**	2,132	16	5,494	<1
p-hat1500-3	1,500	66,206	210,800	4,097	132	10,800	3,877	83	704	**7,501**	148	14	**7,501**	148	16	1,500	<1
socfb-UCSB37	14,917	356,918	125,297	16,708	287	10,800	219,965	3,286	256	254,965	4,013	16	**255,924**	3,994	22	13,655	3
socfb-Duke14	9,885	374,731	131,706	11,565	198	10,800	125,940	1,887	39	151,169	2,415	15	**151,365**	2,390	18	9,003	4
p-hat1500-2	1,500	411,372	143,918	3,542	53	10,800	2,611	43	2,150	**4,253**	68	<1	**4,253**	68	<1	1,500	<1
socfb-Stanford3	11,586	136,823	431,486	7,250	122	10,800	215,381	3,932	265	251,350	4,441	12	**251,932**	4,436	14	9,380	2
socfb-UConn	17,206	448,038	156,829	16,996	279	10,800	231,718	3,471	322	274,637	4,329	12	**275,320**	4,299	18	16,025	5
p-hat1500-1	1,500	411,744	427,583	0	0	10,800	1,184	18	5,499	**1,287**	19	<1	**1,287**	19	1	1,500	<1
C2000.5	2,000	489,894	509,270	0	0	10,800	1,581	24	865	**1,849**	26	2	**1,849**	26	6	2,000	1
keller6	3,361	759,816	266,766	0	0	10,800	3,545	55	1,561	**5,236**	73	10	**5,236**	73	16	3,361	<1
C4000.5	4,000	1,957,884	2,039,848	0	0	10,800	1,611	25	1,659	1,948	28	20	**1,952**	28	22	4,000	9
# of best / # of instances				24/47	–	–	1/47	–	–	18/47	–	–	**41**/47	–	–	–	–
average running time(s)				–	–	5,819.2	–	–	2,087.4	–	–	7.9	–	–	**5.9**	–	1.4

Table 2 Comparative results of RLS on SET-3 instances

Instance	$\|V\|$	$\|E_p\|$	$\|E_r\|$	CPLEX			LA-B&B			ALS			RLS				
				BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	BKV	$\|V_I\|$	htime(s)	$\|V_{\text{ker}}\|$	time$_{\text{ker}}$(s)
foodweb-wet	128	1,509	566	**2,464***	44	<1	**2,464***	44	8	**2,464***	44	<1	**2,464***	44	<1	125	<1
foodweb-dry	128	1,571	535	**2,226***	42	<1	**2,226***	42	8	**2,226***	42	<1	**2,226***	42	<1	125	<1
USAir97	332	1,589	537	**10,308***	194	<1	9,507	178	407	**10,308***	194	<1	**10,308***	194	<1	0	<1
powergrid	4,941	1,727	4,867	**193,339***	3,710	<1	190,537	3,684	108	193,095	3,754	59	**193,339***	3,537	<1	0	<1
CondMat	23,133	46,769	46,670	**711,517***	12,821	12	685,442	12,061	5,749	705,901	1,306	14	711,334	12,870	45	1,648	<1
Email	265,009	91,093	273,388	**13,054,427***	257,435	118	OOM	–	–	OOM	–	–	**13,054,427***	257,394	1	0	1
Epinion	75,879	202,653	203,087	**3,052,059***	57,976	60	OOM	–	–	3,038,533	58,267	18	**3,052,059***	58,008	1	253	1
Dblp	317,080	787,268	262,598	**9,737,911***	171,421	790	OOM	–	–	OOM	–	–	9,734,653	171,403	64	11,851	11
cnr-2000	325,557	685,686	2,053,283	13,367,460	260,331	10,800	OOM	–	–	OOM	–	–	**13,368,670**	260,716	482	59,052	399
WikiTalk	2,394,385	3,494,873	1,164,692	**118,708,326***	2,342,835	3,566	OOM	–	–	OOM	–	–	**118,708,326***	2,342,861	333	8	333
BerkStan	685,230	1,662,833	4,986,637	**26,036,210**	501,197	10,800	OOM	–	–	OOM	–	–	26,031,704	501,410	1,112	114,191	1,031
As-Skitter	1,696,415	5,550,661	5,544,637	**66,603,360**	1,264,844	10,800	OOM	–	–	OOM	–	–	66,582,254	1,265,266	723	31,173	636
soc-pokec	1,632,803	11,152,945	11,149,019	4,146,172	84,217	10,800	OOM	–	–	OOM	–	–	**43,999,402**	833,989	934	881,319	134
LiveJ	4,846,609	10,710,579	32,140,658	0	0	10,800	OOM	–	–	OOM	–	–	**168,431,056**	3,258,831	904	401,307	469
uk2002	18,483,186	65,451,024	196,336,234	0	0	100,000	OOM	–	–	OOM	–	–	**687,089,901**	13,716,043	7,678	4,806,660	6,243
# of best / # of instances				**11**/15	–	–	3/15	–	–	3/15	–	–	**11**/15	–	–	–	–
average running time(s)				–	–	10,570.1	–	–	1,256	–	–	15.7	–	–	818.7	–	617.5

References

Colombi M, Mansini R, Savelsbergh M (2017) The generalized independent set problem: Polyhedral analysis and solution approaches. *European Journal of Operational Research* 260(1):41–55.

Gu J, Zheng W, Cai Y, Peng P (2021) Towards computing a near-maximum weighted independent set on massive graphs. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 467–477.

Table 3 Comparative results of algorithms with or without reduction rules

Instance	$\|V\|$	$\|V_{ker}\|$	CPLEX				LA-B&B				RLS			
			without reduction		with reduction		without reduction		with reduction		without reduction		with reduction	
			BKV	htime(s)	BKV	htime(s)	BKV	htime(s)	BKV	htimes	BKV	htime(s)	BKV	htime(s)
bio-yeast	1,458	0	**68,574***	<1	**68,574***	<1	66,545	2	**68,574***	<1	**68,574***	1	**68,574***	<1
soc-wiki-Vote	889	0	**37,358***	<1	**37,358***	<1	35,973	<1	**37,358***	<1	**37,358***	1	**37,358***	<1
vc-exact_131	2,980	0	**109,641***	<1	**109,641***	<1	107,812	14	**109,641***	<1	109,567	9	**109,641***	<1
web-edu	3,031	98	**114,140***	<1	**114,140***	< 1	109,856	10	**114,140***	21	114,080	12	**114,140***	<1
MANN_a81	3,321	0	**117,327***	<1	**117,327***	<1	114,616	16	**117,327***	<1	117,275	14	**117,327***	<1
tech-routers-rf	2,113	0	**97,799***	<1	**97,799***	<1	93,690	54	**97,799***	<1	**97,799***	1	**97,799***	<1
vc-exact_039	6,795	0	**259,542***	<1	**259,542***	<1	254,941	253	**259,542***	<1	259,475	13	**259,542***	<1
ca-GrQc	4,158	282	**174,782***	9	**174,782***	6	167,846	45	174,525	2,998	174,768	7	**174,782***	11
vc-exact_038	786	697	**12,015***	589	**12,015***	585	10,406	1,994	10,782	28	**12,015***	13	**12,015***	4
vc-exact_078	11,349	0	**438,671***	<1	**438,671***	1	431,371	1,074	**438,671***	1	438,306	16	**438,671***	<1
vc-exact_087	13,590	0	**518,500***	2	**518,500***	<1	509,734	2,319	**518,500***	<1	518,069	15	**518,500***	<1
vc-exact_107	6,402	0	**407,878***	<1	**407,878***	<1	396,456	10,646	**407,878***	<1	406,379	14	**407,878***	<1
vc-exact_151	15,783	0	**530,138***	2	**530,138***	<1	516,445	2,402	**530,138***	<1	529,003	18	**530,138***	<1
vc-exact_167	15,783	0	**465,178***	1	**465,178***	<1	449,579	10,469	**465,178***	<1	463,446	12	**465,178***	<1
bio-dmela	7,393	44	**302,992***	<1	**302,992***	< 1	280,897	170	**302,992***	< 1	302,793	18	**302,992***	<1
vc-exact_011	9,877	172	**303,793***	1	**303,793***	< 1	286,137	4,818	303,589	1271	303,352	9	**303,793***	<1
web-spam	4,767	487	**190,507***	4.61	**190,507***	< 1	180,446	53	189,897	< 1	190,460	18	190,460	13
vc-exact_026	6,140	300	**201,509***	1	**201,509***	< 1	177,902	668	200,868	< 1	201,458	13	201,503	<1
vc-exact_001	6,160	134	**215,941***	2	**215,941***	< 1	194,984	9,575	**215,941***	2	215,934	13	**215,941***	<1
vc-exact_024	7,620	1,255	**236,109**	10,783	**236,109**	10,800	226,540	244	234,933	1	235,911	14	236,018	12
C1000.9	1,000	1,000	5,263	10,800	5,263	10,800	6,104	430	6,104	430	7,235	16	**7,237**	10
tech-WHOIS	7,476	338	**325,584***	5	**325,584***	< 1	309,903	218	325,119	255	325,480	16	325,581	13
p-hat700-3	700	700	4,025	10,800	4,025	10,800	3,030	7,436	3,030	7,436	**4,473**	1	**4,473**	<1
vc-exact_008	7,537	574	**247,091***	3	**247,091***	< 1	210,126	9,093	245,289	6,627	247,036	16	**247,091***	<1
keller5	776	776	2,373	10,800	2,373	10,800	2,860	6,327	2,860	6,327	**3,877**	2	**3,877**	2
vc-exact_194	1,150	1,150	3,734	10,708	3,734	10,800	4,386	1,324	4,386	1,324	**5,419**	2	**5,419**	<1
hamming10-4	1,024	1,024	3,256	8,108	3,256	10,800	3,073	2,069	3,073	2,069	**3,660**	1	**3,660**	<1
brock800_2	800	800	1,177	10,800	1,177	10,800	1,996	10	1,996	10	**2,473**	1	**2,473**	<1
brock800_4	800	800	1,583	10,800	1,583	10,800	1,979	7,215	1,979	7,215	**2,501**	1	**2,501**	<1
ca-HepPh	11,204	2,165	372,096	10,800	**372,229**	10,800	348,738	422	368,637	2	372,079	14	372,079	17
p-hat700-2	700	700	2,870	10,140	2,870	10,140	2,345	2,078	2,345	2,078	**2,971**	1	**2,971**	<1
vc-exact_196	1,534	1,534	3,916	10,800	3,916	10,800	4,273	66	4,273	66	**5,321**	2	**5,321**	<1
p-hat700-1	700	700	730	10,800	730	10,800	**844***	5	**844***	5	**844***	1	**844***	<1
ca-AstroPh	17,903	4,985	604,288	10,800	**604,342**	10,800	558,736	2,544	593,899	37	603,732	15	603,945	14
C2000.9	2,000	2,000	3,958	9,489	3,958	10,800	5,477	11	5,477	11	**6,396**	3	**6,396**	3
DSJC1000.5	1,000	1,000	925	7,645	925	10,800	1,515	694	1,515	694	**1,861**	1	**1,861**	<1
socfb-MIT	6,402	5,494	7,098	10,800	118,114	10,800	110,677	20	113,623	10	129,831	16	**129,872**	16
p-hat1500-3	1,500	1,500	4,097	10,800	4,097	10,800	3,877	704	3,877	704	7,500	15	**7,501**	16
socfb-UCSB37	14,917	13,655	16,708	10,800	57,084	10,800	219,965	256	223,780	150	255,887	21	**255,898**	22
socfb-Duke14	9,885	9,003	11,565	10,800	127,145	10,800	125,940	39	127,269	2,231	151,321	14	**151,326**	18
p-hat1500-2	1,500	1,500	3,542	10,800	3,542	10,800	2,611	2,150	2,611	2,150	**4,253**	1	**4,253**	<1
socfb-Stanford3	11,586	9,380	7,250	10,800	127,708	10,800	215,381	265	223,398	7,756	251,788	19	**251,833**	14
socfb-UConn	17,206	16,025	16,996	10,800	51,328	10,800	231,718	322	235,104	240	275,006	16	**275,034**	20
p-hat1500-1	1,500	1,500	0	10,800	0	10,800	1,184	5,499	1,184	5,499	**1,287**	2	**1,287**	1
C2000.5	2,000	2,000	0	10,800	0	10,800	1,581	865	1,581	865	**1,849**	5	**1,849**	6
keller6	3,361	3,361	0	10,800	0	10,800	3,545	1,561	3,545	1,561	5,229	19	**5,236**	16
C4000.5	4,000	4,000	0	10,800	0	10,800	1,611	1,659	1,611	1,659	**1,951**	15	**1,951**	22
foodweb-wet	128	125	**2,464***	< 1	**2,464***	< 1	**2,464***	8	**2,464***	6	**2,464***	<1	**2,464***	<1
foodweb-dry	128	125	**2,226***	< 1	**2,226***	< 1	**2,226***	8	**2,226***	8	**2,226***	<1	**2,226***	<1
USAir97	332	0	**10,308***	< 1	**10,308***	< 1	9,507	407	**10,308***	< 1	**10,308***	<1	**10,308***	<1
powergrid	4,941	0	**193,339***	< 1	**193,339***	< 1	190,537	108	**193,339***	< 1	193,194	52	**193,339***	<1
CondMat	23,133	1,648	**711,517***	12	**711,517***	< 1	685,442	5,749	708,907	84	709,927	54	711,334	45
Email	265,009	0	**13,054,427***	118	**13,054,427***	1	OOM	–	**13,054,427***	1	13,054,390	69	**13,054,427***	1
Epinion	75,879	253	**3,052,059***	60	**3,052,059***	1	OOM	–	3,051,256	196	3,050,415	67	**3,052,059***	1
Dblp	317,080	11,851	**9,737,911***	790	**9,737,911***	39	OOM	–	9,715,860	307	9,577,609	46	9,734,653	64
cnr-2000	325,557	59,052	13,367,460	10,800	**13,641,278**	10,800	OOM	–	11,347,241	10,796	13,361,167	84	13,368,670	482
WikiTalk	2,394,385	8	**118,708,326***	3,566	**118,708,326***	333	OOM	–	**118,708,326***	333	118,672,055	56	**118,708,326***	333
BerkStan	685,230	114,191	26,036,210	10,800	**26,055,767**	10,800	OOM	–	22,591,394	1,031	25,813,846	200	26,031,704	1,112
As-Skitter	1,696,415	31,173	66,603,360	10,800	**66,830,857**	10,800	OOM	–	65,907,243	10,512	59,886,304	200	66,582,254	723
soc-pokec	1,632,803	881,319	4,146,172	10,800	27,037,851	10,800	OOM	–	OOM	–	36,540,838	350	**43,999,402**	934
LiveJ	4,846,609	401,307	0	10,800	163,061,840	10,800	OOM	–	OOM	–	148,758,359	318	**168,431,056**	904
uk2002	18,483,186	4,806,660	0	100,000	567,666,923	100,000	OOM	–	OOM	–	633,365,020	1,806	**687,089,901**	7,678
# of best / # of instances			31/62	–	36/62	–	3/62	–	21/62	–	22/62	–	**51**/62	–
average running time(s)			–	7,096.5	–	6,901.0	–	2,007.5	–	1,441.1	–	**60.8**	–	202.6