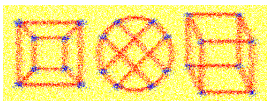

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 29, no. 1, pp. 159–186 (2025)
DOI: 10.7155/jgaa.v29i1.3064

Scalable Algorithms for 2-Packing Sets on Arbitrary Graphs

Jannick Borowitz[1] *Ernestine Großmann*[1] *Christian Schulz*[1] *Dominik Schweisgut*[1]

[1]Heidelberg University, Germany

Submitted: August 2024 Accepted: June 2025 Published: July 2025

Article type: Regular paper Communicated by: Alexander Wolff

Abstract. A 2-packing set for an undirected graph $G = (V, E)$ is defined as a subset $\mathcal{S} \subseteq V$ such that for each pair of vertices $v_1 \neq v_2 \in \mathcal{S}$, the shortest path between v_1 and v_2 has at least length three. Finding a 2-packing set of maximum cardinality is an NP-hard problem. We develop a new approach to solve this problem on arbitrary graphs using its close relation to the independent set problem. Our approach uses new data reduction rules as well as a graph transformation. Experiments show that this technique outperforms the state-of-the-art for arbitrary graphs with respect to solution quality. Furthermore, we can compute solutions multiple orders of magnitude faster than previously possible. Our approach solves 63% of the graphs in the tested data set to optimality in under a second. In contrast, the competitor for arbitrary graphs can only solve 5% of these graphs to optimality even with a 10-hour time limit. Moreover, our approach can solve a wide range of large instances that have previously been unsolved.

1 Introduction

For a given undirected graph G a *2-packing set* is defined as a subset $\mathcal{S}$ of the vertex set of G such that, for each pair of distinct vertices $v_1 \neq v_2 \in \mathcal{S}$, the shortest path between v_1 and v_2 has length at least three. A *maximum 2-packing set* (M2S) is a 2-packing set of the largest cardinality. A generalization of the maximum 2-packing set problem is the maximum k-packing set problem, where the shortest path length is bounded by $k + 1$. For $k = 1$, this results in the *maximum independent set* (MIS) problem. An important application for the maximum 2-packing set problem is given in distributed algorithms. In contrast to the independent set problem, where a given solution vertex is in conflict only with the direct neighborhood, the 2-packing set provides information about a larger area around the vertex. This is important for self-stabilizing algorithms

Research supported by DFG grant SCHU 2567/3-1.

E-mail addresses: jannick.borowitz@uni-heidelberg.de (Jannick Borowitz) e.grossmann@informatik.uni-heidelberg.de (Ernestine Großmann) christian.schulz@informatik.uni-heidelberg.de (Christian Schulz) dominik.schweisgut@stud.uni-heidelberg.de (Dominik Schweisgut)

[20, 21, 39, 42, 48, 46, 50]. In particular, computing large 2-packing sets can be used as a subroutine to ensure mutual exclusion of vertices with overlapping neighborhoods. An example is finding a minimal $\{k\}$-dominating function [20], which has various applications itself as presented by Gairing et al. [22]. Bacciu et al. [5] use large k-packing sets to develop a downsampling approach for graph data. This is particularly useful for deep neural networks. Further, Soto et al. [28] show that the knowledge of the size of a maximum 2-packing set in special graphs can be used for error correcting codes, and Hale et al. [29] indicate that large 2-packing sets can be used to model interference issues for frequency assignment. This can be done by looking at the frequency-constrained co-channel assignment problem. Here, the vertex set consists of locations of radio transmitters, and two vertices share an edge if their frequencies are mutually perceptible. Now, we want to assign a channel to as many radio transmitters as possible to conserve spectrum and avoid interference issues. Vertices assigned to the same channel must have a certain distance. If this distance is two, this can be solved by finding a maximum 2-packing set in the corresponding graph. The maximum 2-packing set problem can be solved to optimality quite fast for small instances. However, since it is an NP-hard problem [31], the running time of exact algorithms grows exponentially with the size of the graph. One powerful technique for tackling NP-hard graph problems is to use *data reduction rules*, that remove or contract local (hyper)graph structures to reduce the input instance to an equivalent, smaller instance. Originally developed as a tool for parameterized algorithms [12], data reduction rules have been effective in practice for computing an (unweighted) maximum independent set [10, 33, 43] / minimum vertex cover [3], maximum clique [9, 51], and maximum k-plex [11, 32], as well as solving graph coloring [38, 51] and clique cover problems [25, 44]. For a detailed overview of data reductions used for other problems, we direct the reader to the survey by Abu-Khzam et al. [2]. The maximum 2-packing set problem can be solved with a reduction to the MIS problem by transforming the instance to the square graph. This is the original graph extended by a set of edges connecting all vertices that share a common neighbor. In recent years several highly scalable algorithms for the maximum (weighted) independent set problem have been presented [13, 16, 24, 26, 27, 33, 34]. However, graphs can get very dense if we directly compute the graph transformation which could prohibit scalability.

Our Results. In addition to the new data reduction rules for the maximum 2-packing set problem, we contribute a novel exact algorithm RED2PACK_B&R and a heuristic RED2PACK_HEURISTIC. Both approaches use our new reductions to compute (near-)optimal maximum 2-packing sets for arbitrary large-scale graphs. These approaches work in three phases. First, the data reductions are applied to the input graph, resulting in a reduced equivalent instance. Afterward, the resulting graph is transformed such that a solution on the transformed graph for the MIS problem corresponds to a solution of the maximum 2-packing set problem for the original graph. The third phase of the approach consists of solving the MIS problem on the transformed graph. In this phase, our two variants differ. We use an exact solver in RED2PACK_B&R while we use a heuristic in RED2PACK_HEURISTIC. Our experiments indicate that our methods outperform the current state-of-the-art heuristic, an evolutionary approach GEN2PACK introduced by Trejo-Sánchez et al. [45], for arbitrary graphs regarding solution quality and running time. For instance, we compute optimal solutions for 63 % of our data set in less than a second, whereas GEN2PACK achieves this only for 5 % of the graphs, even with a 10-hour time limit. Lastly, our method solves many large instances that remained unsolved before.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with $n = |V|$ and $m = |E|$, where $V = \{0, \dots, n-1\}$. When comparing different graphs, we use the notation $n(G)$ and $m(G)$ to specify the number of vertices and edges of a graph G. We extend this definition to a *link-graph* $\mathcal{G} = (G, \mathcal{L})$, which is a tuple of a graph G and a set of *links* $\mathcal{L} \subseteq \binom{V}{2} \setminus E$ connecting two vertices. Note that this link set $\mathcal{L}$ is disjunct to the edge set E, i.e. $\mathcal{L} \cap E = \emptyset$. Two vertices connected by a link are called *linked* vertices. A path p in G is a sequence of adjacent edges. The *length* of a path is equal to its number of edges. In the link-graph $\mathcal{G}$, a path can also contain links. For each link in the path, we add 2 to its length. Therefore, the shortest path between two linked vertices is of length 2. We define the *induced link-subgraph* of a set of vertices $V' \subseteq V$ as $\mathcal{G}[V'] = (G[V'], \mathcal{L}[V'])$ with $\mathcal{L}[V'] = \{\{u, v\} \in \mathcal{L} \mid u, v \in V'\}$. We use the notation $\mathcal{G} - v$ for $\mathcal{G}[V \setminus \{v\}]$ and $\mathcal{G} - V'$ for $\mathcal{G}[V \setminus V']$.

The open neighborhood $N(v)$ of a vertex $v \in V$ is defined as $N(v) = \{u \in V \mid \{u, v\} \in E\}$ and the closed neighborhood $N[v] = N(v) \cup \{v\}$. The notation is extended to a set of vertices $U \subseteq V$ with $N(U) = \cup_{u \in U} N(u) \setminus U$ and $N[U] = N(U) \cup U$. Similarly, we define the *link-neighborhood* of a vertex $v \in V$ as $L(v) = \{u \in V \mid \{u, v\} \in \mathcal{L} \vee u \in N(N[v])\}$. The 2-neighborhood is defined as $N_2[v] = N[v] \cup L(v)$. By this definition, for all vertices $u \in L(v)$, the shortest path from u to v is of length 2. This notation again is extended to a set of vertices $U \subseteq V$ with $L(U) = \cup_{u \in U} L(u) \setminus N[U]$.

The degree of a vertex v is the size of its neighborhood $\deg(v) = |N(v)|$. The *link-degree* of a vertex is defined by the size of its link-neighborhood $\deg_{\mathrm{L}}(v) = |L(v)|$. For a vertex $v \in V$, we define the induced link set $\mathrm{link}(v) = \binom{N(v)}{2} \setminus E$. This set contains all links between two non-adjacent vertices with v as a common neighbor. This set links exactly those vertices that would have to be connected to complete $N(v)$ as a clique. This notation is extended to a set of vertices $V' \subseteq V$ by $\mathrm{link}(V') = \{\{x, y\} \in \binom{N(V')}{2} \setminus E \mid (N(x) \cap N(y)) \cap V' \neq \emptyset\}$, such that $\mathrm{link}(V')$ only contains links connecting vertices in $N(V')$ that have a common neighbor in V'. An important part of the correctness of our data reduction rules introduced in the following sections is extending the set $\mathcal{L}$ by these links for removed vertices.

The *square graph* $G^2 = (V, E^2)$ of a graph $G = (V, E)$ is defined as a graph with the same vertex set and an edge for every pair of vertices that are connected by a path of length at most 2 in G.

For $0 < k \in \mathbb{N}$ a *k-packing set* is defined as a subset $\mathcal{S} \subseteq V$ such that between each pair of vertices in $\mathcal{S}$ the shortest path has length at least $k + 1$. For $k = 1$, we refer to the set as the *independent set*, where all vertices in $\mathcal{S}$ are non-adjacent. The *maximum* independent set (MIS) problem is finding an independent set of maximum cardinality. Our work mainly focuses on the case $k = 2$, where the shortest path between each pair of vertices in $\mathcal{S}$ is at least length three. A *maximal* 2-packing set is a 2-packing set $\mathcal{S} \subseteq V$ that cannot be extended by any further vertex $v \in V$ without violating the 2-packing set condition. The *maximum* 2-packing set problem (M2S) is finding a 2-packing set of maximum cardinality. Analogously to the independence number $\alpha(G)$ for the maximum independent set, we define $\alpha^2(\mathcal{G})$ as the size of the solution to the maximum 2-packing set problem for a given link-graph $\mathcal{G}$.

A distance-2-clique is a set of vertices in $\mathcal{G}$ pairwise connected by a shortest path of length at most 2. A vertex v is *distance-2-isolated* if the vertices of $N_2[v]$ form a distance-2-clique.

3 Related Work

In Section 3.1, we summarize the most important related work on the maximum 2-packing set problem. Since our approach also utilizes independent set solvers, we cover related work on that problem in Section 3.2.

3.1 2-Packing Set Algorithms

Trejo-Sánchez et al. [45] are the first and as far as we know only authors to have proposed a sequential heuristic for the maximum 2-packing set problem on connected arbitrary graphs. They developed a genetic approach in which they use local improvements in each round and a penalty function. Ding et al. [14] propose a self-stabilizing algorithm for arbitrary graphs. The algorithm consists basically of two operations, namely entering the solution candidate and exiting the solution candidate for each vertex in the graph. If a vertex enters the solution, its neighbors get locked, so they cannot enter the solution and cause a conflict. The decision to enter or exit the solution is based on the simple criteria to check whether a vertex causes a conflict or not. In general, most of the contributions to the 2-packing set problem on arbitrary graphs are in the context of distributed algorithms [20, 39, 42]. Further, there are some contributions to distributed algorithms for the M2S problem for special graphs. Flores-Lamas et al. [19] present a distributed algorithm that finds a maximal 2-packing set in an undirected non-geometric Halin graph in linear time. They use reduction rules by Eppstein [17] to determine a partition of the vertex set on which they base a coloring scheme to determine a maximal 2-packing set. However, the reduction of the graph is used only temporarily to compute the vertex partition; the coloring phase of the algorithm works on the original graph. Fernández-Zepeda et al. [47] present a distributed algorithm for an M2S in geometric outerplanar graphs. Mjelde [40] presents a self-stabilizing algorithm for the maximum k-packing set problem on tree graphs. Further, Mjelde [40] present a sequential algorithm for the M2S problem on tree graphs using dynamic programming. For special graphs, there are also some non-distributed algorithms for the M2S problem [18, 28, 49]. Trejo-Sánchez et al. [49] present an approximation algorithm called Apx-2p + Imp2p using graph decompositions and LP-solvers. The approximation ratio is related to how the algorithm decomposes the input graph into smaller subgraphs, which is inspired by Baker [6]. They also mention the possibility of solving the maximum 2-packing set problem by using a graph transformation to the square graph and then applying an independent set solver. However, this is not used for their algorithm. This equivalence was stated by Halldórsson et al. [30].

3.2 Independent Set Algorithms

Our algorithm uses data reductions in a problem-specific context as well as for solving the MIS problem on the transformed graph. Therefore, we summarize some related work on this topic with respect to the MIS problem as well. In recent years, the branch-and-reduce paradigm has been shown to be an effective method for solving the maximum independent set problem to optimality, as well as its complementary problem, the minimum vertex cover [3]. By this paradigm, we mean branching algorithms that utilize reduction rules to reduce the input size. These reduced instances are equivalent to the original input instances, and an optimal solution on the reduced instance can be extended to an optimal solution on the original instance. Akiba and Iwata show that this approach yields good results in comparison to other exact approaches for the minimum vertex cover problem and the maximum independent set problem [3]. Furthermore, this approach is successfully applied to the maximum weight independent set problem. Lamm et al. [34] use this approach for an exact algorithm, while data reductions are also used for heuristics. Großmann et al. [26] use reduction rules in combination with an evolutionary approach for solving the maximum weight independent set problem on huge sparse networks. Gao et al. [23] use inexact reduction rules by performing multiple rounds of a local search algorithm to determine vertices that are likely to be part of a solution to the MIS problem. Further, some contributions combine local search algorithms with data reductions [36] and graph neural networks [35]. Moreover, Gu et al. [27]

Algorithm 1: High-level view of RED2PACK.

```
Data: Graph G = (V, E)
Result: M2S
Procedure RED2PACK(G):
    K, L ← REDUCE(G)
    K² ← TRANSFORM(K, L)
    return MISSOLVE(K²)
```

use data reduction and a tie-breaking policy to apply data reductions repeatedly until they reach an empty graph. Lamm et al. [33] present a branch-and-reduce approach combined with an evolutionary algorithm for the MIS problem. A different, widely used technique is local search. The main idea behind it is to start from an initial solution and then improve this solution by swaps. The algorithm proposed by Andrade et al. [4] has proven to be a successful approach in this area. The basic idea is to perform $(1, 2)$-swaps, i.e. to delete one vertex from the solution and add two new vertices. Hence, the solution size increases by one. Other local search approaches have also proven successful [7, 8, 37, 41].

4 The Algorithm red2pack

We now give an overview of the components of our algorithm. The idea of our approach is to build the square graph on which a maximum independent set is equivalent to a maximum 2-packing set on the original graph (see Theorem 1).

On this transformed graph, we apply well-studied maximum independent set solvers to find optimal solutions. During the transformation, we increase the number of edges in the graph. Since this results in a more dense graph, this approach can necessitate a substantial amount of memory and yield longer running times.

To alleviate this issue, we add a preprocessing step. There, we apply new problem-specific data reductions exhaustively to the graph to obtain a reduced link-graph $\mathcal{K}$. An (optimal) solution $\mathcal{K}$ can be used to construct an (optimal) solution on the original instance. On this reduced instance $\mathcal{K}$, we apply the transformation, resulting in a significantly smaller square graph. On this, a maximum independent set solver is applied to obtain an (optimum) solution. In the end, the solution is transformed into an (optimum) solution for the input instance. Overall, this results in the algorithm RED2PACK (see Algorithm 1). In the following, we introduce our new data reductions, then present the used graph transformation, and finally give details about the maximum independent set solvers used.

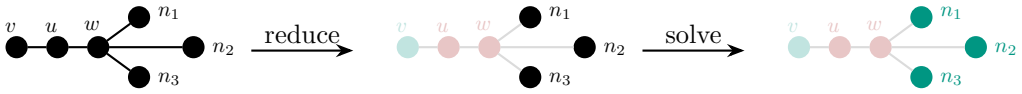
4.1 Data Reduction Rules

In this section, we introduce a set of new data reduction rules for the maximum 2-packing set problem following the scheme explained in the following.

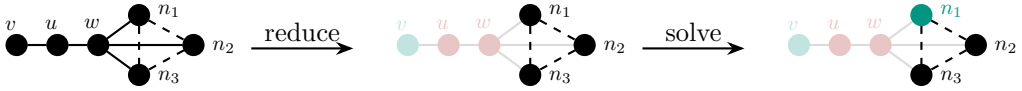
Reduction Scheme [Reduction Name] by [Authors]
Description of the pattern that can be reduced.

Figure 1: Demonstration of the vital role of retaining 2-neighborhood information for non-reduced vertices in maintaining a valid solution to the M2S problem. Reduced vertices and edges have less opacity. Green vertices are included, and red vertices are excluded from the solution.

(a) Including v without links $\mathcal{L}$, thereby loosing the 2-neighborhood information yielding an invalid solution.



(b) Including v with necessary 2-neighborhood information via $\mathcal{L}$ resulting in a valid M2S.



Link Set	*Which additional links $\mathcal{L}$ have to be stored*
Reduced Graph	*How to build the reduced graph $\mathcal{G}'$*
Offset	*Difference in solution size on $\mathcal{G}$ and $\mathcal{G}'$*
Reconstruction	*How to reconstruct the solution $\mathcal{S}$ for the original graph given the solution $\mathcal{S}'$ on the reduced graph G'*

First, we give the name of the reduction rule. Then, we define the pattern that this rule can reduce. Finally, we give details on how to perform the actual reduction. This last information consists of four parts. We start with information on how to extend the link set $\mathcal{L}$. Note that initially, the set $\mathcal{L}$ is empty. Then, the construction of the reduced link-graph $\mathcal{G}'$ from $\mathcal{G}$ already containing the extended link set is defined. The *offset* describes the difference between the size of an M2S on the reduced graph $\alpha^2(\mathcal{G}')$ and the size of an M2S on the original graph $\alpha^2(\mathcal{G})$. Lastly, the information on how the solution on the reduced instance, $\mathcal{S}'$, can be lifted to a solution on the original graph, $\mathcal{S}$, is provided.

In general, our reductions allow us to identify vertices as (1) part of a solution to the maximum 2-packing set problem (included) and (2) as non-solution vertices (excluded). Before presenting the reductions, we give a lemma that generally describes how to change the graph for including or excluding a vertex and discuss the importance of the link set $\mathcal{L}$.

4.1.1 Initial Reduction Information

We briefly motivate the importance of the edge set $\mathcal{L}$ for our reductions. When a reduction includes a vertex v, its 2-neighborhood has to be excluded to obtain a valid maximum 2-packing set of the original graph from the solution on the reduced instance. In Figure 1a, we perform a reduction that includes the vertex v without the use of the link set $\mathcal{L}$. Here, we do not get a valid solution since the information that n_1, n_2, and n_3 belong to the same 2-neighborhood is lost. When we apply the reduction with the additional connecting links in $\mathcal{L}$ for the excluded vertex w, we obtain a valid solution. This is illustrated in Figure 1b. In Lemma 1, we show how to include or exclude a vertex v in a link-graph $\mathcal{G} = (G, \mathcal{L})$.

Lemma 1 *Let $\mathcal{G}$ be a link-graph with vertex set V and $v \in V$.*

1. *If there is an M2S in $\mathcal{G}$ not containing v, i.e. we can exclude v from the solution. Then, we can perform the reduction by the following steps.*

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(v)$
Reduced Graph	$\mathcal{G}' = \mathcal{G}[V \setminus \{v\}]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}')$
Reconstruction	$\mathcal{S} = \mathcal{S}'$

2. *If there is an M2S in $\mathcal{G}$ containing v, i.e. we can include v into the solution. Then, we can perform the reduction by the following steps.*

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G}[V \setminus N_2[v]]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be a vertex in a link-graph $\mathcal{G}$. First, assume there is an M2S not containing v, i.e. v can be excluded (Lemma 1, Case 1). The link set $\mathcal{L}$ is extended by links between vertices in the 2-neighborhood of v, i.e. $\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$. This way, we maintain the 2-neighborhood information for all remaining vertices. The reduced graph $\mathcal{G}'$ is then obtained by removing the vertex v from the vertex set V and all incident edges and links to v. Since there exists an M2S not containing v, there is an M2S $\mathcal{S}'$ in $\mathcal{G}'$ that is also an M2S in $\mathcal{G}$. We get $\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}')$, and $\mathcal{S} = \mathcal{S}'$.

Second, assume there is an M2S $\mathcal{S}$ containing v, i.e. v can be included, (Lemma 1, Case 2). First, links have to be added to link vertices in the link-neighborhood of v. Note that we do not need to add links for the link-neighborhood of direct neighbors of v since these only link to removed vertices. This way, the 2-neighborhood information is maintained for all remaining vertices. Then, all vertices in the 2-neighborhood $N_2[v]$ along with their incident edges and links have to be removed to avoid a conflict resulting in $\mathcal{G}' = \mathcal{G}[V \setminus N_2[v]]$. Since, there is an M2S $\mathcal{S}'$ in $\mathcal{G}'$ that can be extended by v to an M2S in $\mathcal{G}$, we get $\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$, and $\mathcal{S} = \mathcal{S}' \cup \{v\}$. □

In the following, we first introduce our two *main* reductions. Afterward, we present more efficient special cases of these reductions.

Reduction 4.1 (Domination)
Let $u, v \in V$ be vertices such that $N_2[v] \subseteq N_2[u]$, then exclude u.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(u)$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - u$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}')$
Reconstruction	$\mathcal{S} = \mathcal{S}'$

Proof: Let $u, v \in V$ and $N_2[v] \subseteq N_2[u]$. Further, assume that $\mathcal{S}$ is an M2S in $\mathcal{G}$ containing u. Since $N_2[v] \subseteq N_2[u]$, it holds for all vertices $x \in N_2[v] \setminus \{u\}$ that $x \notin \mathcal{S}$. We define $\mathcal{S}' = (\mathcal{S} \setminus \{u\}) \cup \{v\}$ and it follows that $|\mathcal{S}| = |\mathcal{S}'|$. Moreover, $\mathcal{S}'$ is still a valid 2-packing set since there is no vertex in $N_2[v] \setminus \{v\}$ that is also an element of $\mathcal{S}'$. By construction $\mathcal{S}'$ has the same size and, therefore, is an equivalent solution to M2S not containing u. Using Lemma 1, we get the desired reduction. □

An example of the Domination Reduction is illustrated in Figure 2a.

Reduction 4.2 (Clique)
Let $v \in V$ be distance-2-isolated in $\mathcal{G}$, then include v into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be distance-2-isolated and $\mathcal{S} \subseteq V$ be an M2S in G. Then, at least one vertex $w \in N_2[v]$ is contained in $\mathcal{S}$, otherwise $\mathcal{S}$ is not maximal. It follows that $u \notin \mathcal{S}$ for all $u \in N_2[v] \setminus \{w\}$. Additionally, since v is distance-2-isolated $N_2[v] \subseteq N_2[w]$. Therefore, a new solution $\mathcal{S}' = (\mathcal{S} \setminus \{w\}) \cup \{v\}$ of the same size containing v can be constructed. This way, there is always an equivalent or better solution when including v, and therefore, the vertex v is in some M2S of G. Reducing the graph by including v results in $\alpha^2(\mathcal{G}) = 1 + \alpha^2(\mathcal{G}[V \setminus N_2[v]])$. Using Lemma 1, we get the desired reduction. □

These two reductions require knowledge about the 2-neighborhood as a prerequisite. The 2-neighborhood of a vertex v can become quite large, and verifying these conditions is computationally expensive. Hence, we have sought out different special cases where only constraints on the direct neighbors and the maintained link-degree need to be satisfied. As a result, the links of a link-neighborhood do not need to be considered in a reduction test. The following lemma helps us to show that these special cases are instances of the more general case.

Lemma 2 *Let $u, v \in V$ be neighbors in $\mathcal{G}$ with $N[v] \subseteq N[u]$ such that $\deg_L(v) + \deg(v) \leq \deg(u)$. Then, $L(v) = N[u] \setminus N[v]$, i.e. all link-neighbors of the vertex v, are also neighbors of the vertex u.*

Proof: Let $u, v \in V$ be neighbors with $N[v] \subseteq N[u]$ such that $\deg_L(v) + \deg(v) \leq \deg(u)$. By definition of the link-neighborhood and since u and v are neighbors, we know that $N[u] \setminus N[v] \subseteq L(v)$. Therefore, it follows that $\deg(u) + 1 - (\deg(v) + 1) \leq \deg_L(v)$ which is equivalent to $\deg(u) \leq \deg_L(v) + \deg(v)$. Since u and v are neighbors, it follows that $\deg_L(v) + \deg(v) \geq \deg(u)$, and because of $\deg_L(v) + \deg(v) \leq \deg(u)$ it follows that $\deg_L(v) + \deg(v) = \deg(u)$. Because of $N[u] \subseteq N[v] \cup L(v)$, the two sets $N[u] \setminus N[v]$ and $L(v)$ must be equal. □

Note that, during the reduction process, all direct neighbors of a vertex v may be removed, resulting in $\deg(v) = 0$, but there are remaining links to the vertex v, yielding $\deg_L(v) > 0$. This case is considered in the following reduction.

Reduction 4.3 (Degree Zero Reduction)
Let $v \in V$ be a degree zero vertex with $\deg_L(v) \leq 1$, then include v into the solution.

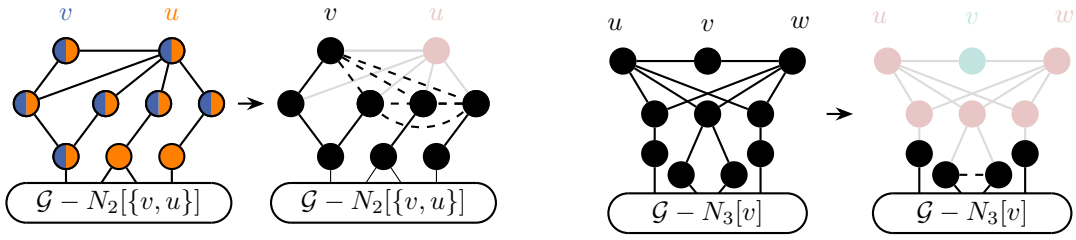
Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be a vertex with $\deg(v) = 0$ and $\deg_L(v) \leq 1$. For the case of $\deg_L(v) = 0$, there is no vertex in the 2-neighborhood of v, and therefore, v can be included in the solution. In the case of $\deg_L(v) = 1$, let the link-neighbor of v be $u \in L(v)$. If there is a solution not containing u, the vertex v can safely be included in the solution. In the case of u being part of an M2S $\mathcal{S}$ on the original instance, we show there always exists another M2S $\tilde{\mathcal{S}}$ containing v. We can create $\tilde{\mathcal{S}}$

Figure 2: Original link-graph on the left reduced to the link-graph on the right.

(a) Reduction 4.1 (Domination): $N_2[u]$ is colored with orange, $N_2[v]$ with blue. The vertex u dominates v and can therefore be excluded.

(b) Reduction 4.10 (Twin): The neighbors $u, w \in N(v)$ are twins. We can include v into the solution and exclude $N_2(v)$.



by swapping the vertex u for v, i.e. $\tilde{\mathcal{S}} = \mathcal{S} \setminus \{u\} \cup \{v\}$. It follows that $\tilde{\mathcal{S}} \cap N_2[v] = \{v\}$, since u is the only 2-neighbor of v in $\mathcal{G}$. Furthermore, $\tilde{\mathcal{S}}$ has the same size as $\mathcal{S}$ and is therefore also an M2S. Therefore, there exists an M2S containing v, and v can be included in the solution. Using Lemma 1, we get the described reduction. □

Reduction 4.4 (Degree Zero Triangle)
Let $v \in V$ be a degree zero vertex. Furthermore, let $\deg_L(v) = 2$ with link-neighbors $L(v) = \{u, w\}$ that are also adjacent or linked, that is $u \in N_2[w]$, then include v into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be a vertex of $\deg(v) = 0$ and $\{u, w\} = L(v)$ such that its link-neighbors are adjacent or linked, i.e. $u \in N_2[w]$. The vertices u and w dominate vertex v and can therefore be excluded by Reduction 4.1. Now, Reduction 4.3 is applicable, and vertex v can be included in the solution. With Lemma 1, we get the final reduction described. □

Reduction 4.5 (Degree One)
Let $v \in V$ be a degree one vertex with $N(v) = \{u\}$. Furthermore, let $\deg_L(v) \leq \deg(u) - 1$, then include v into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

.

Proof: Let $v \in V$ be a vertex with $\deg(v) = 1$ and $N(v) = \{u\}$ with $\deg_L(v) \leq \deg(u) - 1$. We can apply Lemma 2, and it holds that $N_2[v] = N[u]$. Therefore, this represents a special case of the distance-2-clique and Reduction 4.2 can be applied. □

Reduction 4.6 (Degree Two V-Shape)
Let $v \in V$ be a vertex of $\deg(v) = 2$ with $N(v) = \{u, w\}$ and $\deg_L(v) = 0$, then include v into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be a vertex of $\deg(v) = 2$ with $N(v) = \{u, w\}$ and $\deg_L(v) = 0$. First, we show that $N_2[v] \subseteq N_2[u]$. Since u is a direct neighbor of v, it holds that $N[v] \subseteq N_2[u]$. With the additional assumption that $\deg_L(v) = 0$, it follows that $L(v) = \emptyset$. Therefore, we get $N[v] = N_2[v] \subseteq N_2[u]$. With the same argument, we also get $N_2[v] \subseteq N_2[w]$. Then, the vertices w and u can be excluded by Reduction 4.1. Since $|L(v)| = 0$, vertex v can safely be included. □

Reduction 4.7 (Degree Two Triangle)
Let $v \in V$ be a vertex of $\deg(v) = 2$ with $N(v) = \{u, w\}$ and $\deg(u) = \deg(w) = 2$. Furthermore, let $\deg_L(v) = 0$, then include v into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let the vertices $v, u, w \in V$ all have degree two and $N(v) = \{u, w\}$ and $\deg_L(v) = 0$. In this case, the vertices u, v, and w form a triangle. Since $N_2[v] \subseteq N_2[u]$ and $N_2[v] \subseteq N_2[w]$ the vertices w and u can be excluded by Reduction 4.1. Since $|L(v)| = 0$, vertex v can safely be included. □

Reduction 4.8 (Degree Two 4-Cycle)
Let $u, v, w, x \in V$ be vertices with $\deg(v) = \deg(u) = \deg(w) = 2$, $N(v) = \{u, w\}$ and $L(v) = \{x\}$. Furthermore, let $x \in N(u)$ and $x \in N(w)$. Then, the vertices build a 4-cycle and v can be included into the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let the above-stated assumptions hold such that the vertices u, v, w, and the one link-neighbor $x \in L(v)$ form a 4-cycle. It holds that $N_2[v] = \{u, v, w, x\} \subseteq N_2[u]$, therefore, we can exclude the vertex u by Reduction 4.1. Similarly, we can exclude vertex w. Assume x is part of an M2S $\mathcal{S}$. Then, we can create a new solution $\mathcal{S}' = \mathcal{S} \setminus \{x\} \cup \{v\}$ of same size. This way, we always find an M2S including v. By applying Lemma 1, we get the desired reduction. □

Reduction 4.9 (Fast Domination)
Let $u, v \in V$ be vertices such that $N[v] \subseteq N[u]$ and $\deg_L(v) + \deg(v) \leq \deg(u)$, then exclude u from the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(u)$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - u$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}')$
Reconstruction	$\mathcal{S} = \mathcal{S}'$

Proof: Let $u, v \in V$ and $N[v] \subseteq N[u]$ with $\deg_L(v) + \deg(v) \leq \deg(u)$. By the assumption $N[v] \subseteq N[u]$, it follows that u and v are neighbors, and we can use Lemma 2. With this, we know $L(v) = N[u] \setminus N[v]$. It holds that $N_2[v] \subseteq N_2[u]$ and Reduction 4.1 is applicable. □

Reduction 4.10 (Twin)
Let $v \in V$ be a vertex with $\deg(v) = 2$ and $u, w \in V$ be its neighbors with $N(u) = N(w)$. Furthermore, let $\deg_L(v) \leq \deg(u) - 1$. Then, u and w are twins, and v can be included in the solution.

Link Set	$\mathcal{L} = \mathcal{L} \cup \text{link}(N_2[v])$
Reduced Graph	$\mathcal{G}' = \mathcal{G} - N_2[v]$
Offset	$\alpha^2(\mathcal{G}) = \alpha^2(\mathcal{G}') + 1$
Reconstruction	$\mathcal{S} = \mathcal{S}' \cup \{v\}$

Proof: Let $v \in V$ be a vertex with $\deg(v) = 2$ and $u, w \in V$ be its neighbors with $N(u) = N(w)$. Since $u \notin N(u) = N(w)$, the vertices u and w are non-adjacent. We now show that all link-neighbors of v are adjacent to the vertex u and w. By definition of the link-neighborhood and since $w \notin N(u)$, we know $N(u) \setminus \{v\} \subseteq L(v)$. This yields $\deg(u) - 1 = |N(u) \setminus \{v\}| \leq |L(v)| = \deg_L(v)$. The additional assumption $\deg_L(v) \leq \deg(u) - 1$ ensures that the sets are equal and therefore $\mathcal{G}[N_2[v]]$ forms a distance-2-clique. Consequently, we can apply Reduction 4.2. □

An example of the Twin Reduction is given in Figure 2.

4.2 The Reduction Process

In our preprocessing, the introduced reductions are applied exhaustively in a predefined order. If a reduction is successful, we start again by applying the first reduction. We define two reduction list configurations, which we later compare to the configuration called 2PACK, which does not include any of our proposed reductions. In the configuration MAIN, we only use the clique and domination reduction (Reduction List 1). In preliminary experiments, this order of reductions proved to be the most effective.

Reduction List 1 $R_1 = [4.2, 4.1]$

For the second variant FAST we apply the full set of all special case reductions as well as the Domination and Clique Reduction. The order of reductions for this variant is given in Reduction List 2.

Reduction List 2 $R_2 = [4.3, 4.4, 4.5, 4.7, 4.8, 4.6, 4.10, 4.9, 4.1, 4.2]$

With this ordering, we put our special case reductions before the Domination and Clique Reductions to test efficient reductions first and speed up the reduction process. These faster reduction rules can potentially reduce the number of vertices before applying more computationally expensive ones. Note that we did not experiment with different orderings for the reductions since Großmann et al. show in [26] that for the MWIS problem, an intuitive ordering worked best, and small changes do not affect the solution quality and running time significantly.

Figure 3: The graph $\mathcal{G}$ on the left with links marked as dashed lines. This instance is transformed into the square graph on the right. The green vertices present the M2S (left) and the MIS (right).

4.3 On-Demand Link-Neighborhood

Our reductions operate on a dynamic graph data structure based on adjacency arrays. Internally, for each vertex, we store edges and links separately. Overall, this results in two vectors of vectors to store the adjacent and linked neighbors for each vertex. Undirected edges are stored as two directed edges. A key observation of our reductions is that we often do not need to know the link-neighborhood to exclude a vertex. This can be seen, for example, in the Degree One Reduction applied to a vertex v with its neighbor u. Then, we do not need to consider $L(u)$. In general, we only compute and store the link-neighborhood of a vertex v on demand. This also leads to less effort compared to initially constructing the link-neighborhood for the whole graph. When a vertex u is removed from the graph, we delete every edge and link pointing to it. Removing all incoming edges and links can take $\mathcal{O}(\Delta^4)$ time where Δ is the highest degree in the graph. This is because $|N_2[v]| \in \mathcal{O}(\Delta^2)$ for each $v \in V$. When removing all incoming edges and links of v, we have to go through all its neighbors $u \in N_2[v]$ for which we iterate through $N_2[u]$ to delete the edge to v. With the on-demand technique, we can also reduce the number of computations there, i.e. if the link-neighborhood of a vertex v is not computed in advance, there are potentially fewer links that are deleted in this step.

4.4 Graph Transformation

After we applied all data reduction rules exhaustively, i.e. there is no data reduction rule that can still be applied, we begin the transformation of the reduced graph to the square graph.

To build the square graph of $G = (V, E)$, the set of regular edges E is extended by edges connecting all pairs of vertices that have a common neighbor. For the square graph of the link graph $\mathcal{G} = (G, \mathcal{L})$, we first compute the square graph of G and then extend the set of edges E connecting all linked vertices in $\mathcal{G}$. Note that since $E \cap \mathcal{L} = \emptyset$, we do not add parallel edges. See Figure 3 for an example.

Theorem 1 *Let $G = (V, E)$ be a given graph and $G^2 = (V, E^2)$ the square graph. Then, an optimal solution to the maximum independent set problem on G^2 is an optimal solution for the maximum 2-packing set problem on the original graph G [30].*

4.5 MIS Solver

We have chosen to use the solver KaMIS BnR by Lamm et al. [34] since it is a state-of-the-art exact solver for the independent set problem. However, it is also possible to integrate any other exact solver for the maximum independent set problem. Note that we did not choose the branch and reduce solver for the unweighted problem [33], as it restricts us to smaller graphs. While our focus is on the optimal solution, we also combine our reduction approach with the local search approach OnlineMIS by Dahlum et al. [13], resulting in our heuristic red2pack_heuristic. To

be fully self-contained, we briefly describe the main components of the solver by Lamm et al. [34] in Section 4.5.1 and the solver by Dahlum et al. [13] in Section 4.5.2.

Algorithm 2: Branch-and-reduce algorithm KaMIS BnR for Maximum (Weight) Independent Set problem by Lamm et al. [34]

Data: Graph G, current solution weight c (initially zero), best solution weight $\mathcal{W}$ (initially zero)
Result: M2S

```
1  Procedure SOLVE(G, c, W):
2      (G, c) ← REDUCE(G, c)
3      if W = 0 then W ← c + ILS(G)
4      if c + UpperBound(G) ≤ W then return W
5      if G is empty then return max{W, c}
6      if G is not connected then
7          forall G_i ∈ Components(G) do
8              c ← c + SOLVE(G_i, 0, 0)
9          return max(W, c)
10     (G_1, c_1), (G_2, c_2) ← BRANCH(G, c)
       // Run 1st case, update currently best solution
11     W ← SOLVE(G_1, c_1, W)
       // Use updated W to shrink the search space
12     W ← SOLVE(G_2, c_2, W)
13     return W
```

4.5.1 KaMIS BnR for red2pack_b&r

The solver KaMIS BnR employs a branch and reduce framework. Algorithm 2 gives an overview. Throughout the algorithm, it maintains the current solution weight as well as the best solution weight. The algorithm applies a wide set of reduction rules (Line 2) before branching on a vertex (Line 10). The branch-and-reduce solver KaMIS BnR applies a large set of reduction rules as preprocessing and after each branching step. The reduction rules introduced in this work for the 2-packing set problem are derived from the set of rules used in KaMIS BnR. Note, however, that in KaMIS BnR, there is a larger number and more complex reductions implemented. Therefore, if time and space constraints are not an issue, the 2pack approach combined with the preprocessing of KaMIS BnR can result in smaller instances. However, as we show in our experiments, it is always beneficial to first apply our new reductions.

After the initial reduction phase, a local search algorithm is run on the reduced graph to compute a lower bound on the solution weight (Line 3), which later helps prune the search space. The algorithm then prunes the search by excluding unnecessary parts of the branch-and-bound tree to be explored. If the graph is not connected, each connected component is solved separately. If the graph is connected, the algorithm branches into two cases by applying a branching rule. As for the branching rule, initially, vertices are sorted in non-decreasing order by degree, with ties broken by weight. Throughout the algorithm, the next vertex to be chosen is the highest vertex in the ordering. This way, the algorithm quickly eliminates the largest neighborhoods and makes the problem "simpler". If the algorithm does not finish within a certain time limit, the

currently best solution is improved using a greedy algorithm. More precisely, the algorithm sorts the vertices in decreasing order of their weight and adds vertices in that order if feasible. Note that the pseudocode in Algorithm 2 describes the algorithm such that it outputs the weight of a maximum weight independent set in the graph. However, the algorithm is implemented to output the maximum weight independent set.

4.5.2 OnlineMIS for red2pack_heuristic

In this section, we give a detailed explanation of the OnlineMIS solver from et al. [13]. For this solver, the authors combined the iterated local search algorithm ARW [4] with both exact and inexact data reduction rules. The exact rules reduce the size of the search space without losing solution quality. Especially for large-scale networks, this approach can significantly boost the performance of the algorithm by running the local search on the reduced instance. In the paper, the authors show that applying inexact reductions accelerates the performance while computing results that still compete with the best solutions reported in the literature.

The algorithm OnlineMIS [3] applies a set of simple simplicial vertex removal reductions (for degree zero, one, and two) on the fly. A simplicial vertex v is a vertex where $N(v)$ forms a clique. In that case, the simplicial vertex can be included in the solution. Only using these reductions enables the algorithm to reduce the graph by marking these simplicial vertices and their neighbors as removed during the local search.

This is done by first performing a quick single pass when computing the initial solution for ARW. The algorithm further marks the top 1 % of high-degree vertices as removed during this pass, which is the inexact data reduction. During the local search, whenever a vertex is checked to see if it can be inserted into the solution, the isolated vertex removal reduction is checked for this vertex. If the reduction is applicable, the solution is updated.

5 Experimental Evaluation

Methodology. We implemented our algorithm using C++17. The code is compiled using g++ version 12.2 with full optimizations enabled (-O3). We used a machine equipped with an AMD EPYC 7702P (64 cores) processor and 1 TB RAM running Ubuntu 20.04.1. We conducted all our experiments using four different random seeds and report geometric mean values unless otherwise specified. We set a time limit of 10 hours for all algorithms. If a solver exceeds a memory threshold of 100 GB during execution or is terminated due to the time limit, we report the best solution found until this point. To compare different algorithms, we use performance profiles [15], which depict the relationship between the objective function size or running time of each algorithm and the corresponding values produced or consumed by the competing algorithms.

Let $\mathcal{A}$ be the set of algorithms and I the set of instances that we want to compare. For every $A \in \mathcal{A}$, we define $S^A_{\geq\tau} = \{G \in I \mid A_{\text{sol}}(G) \geq \tau \cdot \max\{B_{\text{sol}}(G) \mid B \in \mathcal{A}\}\}$, where $A_{\text{sol}}(G)$ is the solution algorithm A found on instance G. The set $S^A_{\geq\tau}$ contains instances G where the algorithm A found a solution $A_{\text{sol}}(G)$ that was better or equal to τ times the best solution found for G by any algorithm in $\mathcal{A}$. Similarly, we define the set $T^A_{\leq\tau}$ of instances that are solved within a factor of τ times the fastest running time. Formally, for an algorithm $A \in \mathcal{A}$, this yields the set $T^A_{\leq\tau} = \{G \in I \mid A_{\text{time}}(G) \leq \tau \cdot \min\{B_{\text{time}}(G) \mid B \in \mathcal{A}\}\}$, where $A_{\text{time}}(G)$ is the running time of algorithm A on instance G. For a performance profile comparing solution quality, we plot $|S^A_{\geq\tau}|/|I|$, as a function of τ, decreasing from 1 to 0. The smaller the τ value gets, the

more instances are in the set $S^A_{\geq\tau}$. Each algorithm's performance profile yields a non-decreasing, piecewise constant function.

To compare the running times for each algorithm A, plot $|T^A_{\leq\tau}|/|I|$ as a function of increasing $\tau \geq 1$. We always report the best solution found until the time limit is reached, along with the time it takes to find this solution. If the time limit is reached, this can result in reported solutions from exact solvers not being optimal and smaller than reported heuristic solutions.

Overview/Competing Algorithms. We perform a wide range of experiments. First, we perform experiments to investigate the influence of the data reduction rules in Section 5.1. Therefore, we define three reduction list configurations. The first is called 2PACK and does not include any of our proposed reductions. Then, in MAIN, we only use the clique and domination reduction (see Reduction List 1). For the last variant FAST, we apply the full set of all special case reductions as well as the Domination and Clique Reduction. The order of reductions for this variant is given in Reduction List 2. For this reduction list, we put our special case reductions before the Domination and Clique Reduction in FAST to test efficient reductions first in order to avoid checking easily reducible vertices with computationally expensive reductions. Note that we did not experiment with different orderings for the reductions, as Großmann et al. show in [26] that for the MWIS problem, an intuitive ordering works best, and small changes do not significantly affect solution quality or running time.

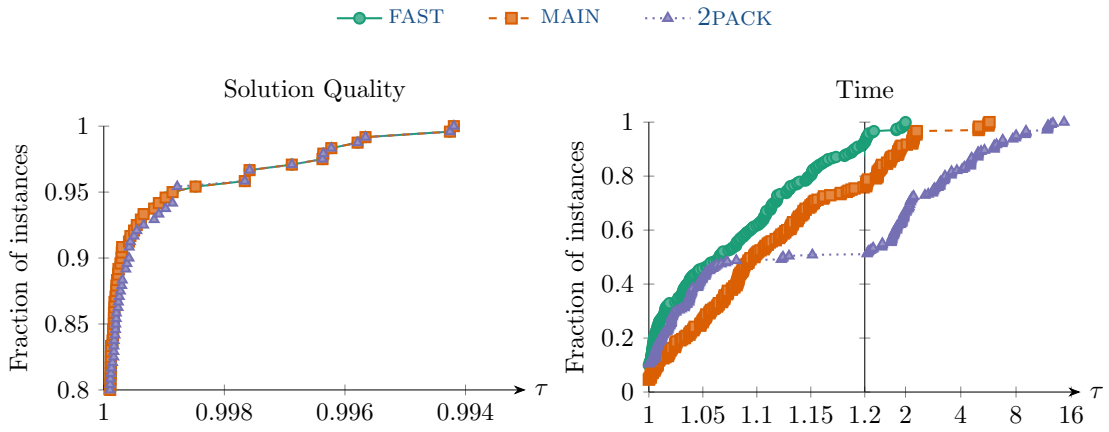
We then compare our algorithms with the state-of-the-art for the problem in Section 5.2. In particular, we compare our approach with the algorithm GEN2PACK by Trejo-Sánchez et al. [45] as well as the APX-2P + IMP2P algorithm by Trejo-Sánchez et al. [49] which only works for planar graphs. We use two configurations of APX-2P + IMP2P. The configurations differ in the parameter h. As h increases, the solution quality improves, but the algorithm's performance slows down. We chose the default configuration with $h = 50$. We added the configuration with $h = 100$ to improve the solution and provide a more fair comparison with our 10-hour time limit e could not perform experiments with MAXIMUM-2-PACK-CACTUS [18] since the code is not available [1] and the data in the paper itself is presented such that a direct comparison is not possible.

To be able to compare against APX-2P + IMP2P, we also include planar graphs from [49]. For an overview of the graph properties see Table 3 in Appendix B.

Table 1: Effect of graph transformation: Arithmetic mean percentage of the number of vertices $\tilde{n} = n(K^2)/n(G)$ and edges $\tilde{m} = m(K^2)/m(G)$ in the reduced square graph K^2. We compare different reduction variants, computed for each graph class and over all instances. Since 2PACK does not apply reductions, this column represents the square graphs G^2. For detailed results, see Table 3 in Appendix B.

	2PACK		MAIN		FAST	
	$\tilde{n}$[%]	$\tilde{m}$[%]	$\tilde{n}$[%]	$\tilde{m}$[%]	$\tilde{n}$[%]	$\tilde{m}$[%]
planar	100	212.11	99.91	211.67	99.91	211.67
social (s)	100	3 154.00	14.48	70.25	14.48	70.25
social (l)	100	4 327.38	17.12	274.56	17.12	274.58
overall	100	2 564.50	43.84	185.49	43.84	185.50

Figure 4: Performance profiles evaluating our reduction configurations with RED2PACK_B&R. We analyse solution quality (left) and running time (right) on *planar* and *social* graphs. The factor τ on the x-axis represents the ratio of the solution quality or running time of the algorithm to the best algorithm. We compare the three configurations 2PACK (direct transformation, no 2-packing reductions), MAIN using Reduction List 1 and FAST using Reduction List 2.



5.1 Impact of Data Reductions

We now investigate the effectiveness of the data reductions. To do so, we use the three reduction configurations for the algorithm RED2PACK_B&R. To evaluate the effectiveness, we do not investigate the influence on *Erdös–Rényi* as well as *Cactus* graphs, as they are already very small. We compare the impact on the size of the squared, reduced instances, as well as solution quality and running time. Details are presented in tables 3 and 4. We summarize these results in Table 1 and Figure 4.

First, we look at the effectiveness of our reductions on the size of the transformed graphs K^2. When applying the graph transformation on the original input, i.e. without applying any reduction (2PACK), the resulting instance has the same amount of vertices and on average $25.65 \cdot m(G)$ edges, whereas MAIN and FAST both yield on average $0.44 \cdot n(G)$ vertices and $1.86 \cdot m(G)$ edges. Thus, our data reductions help to decrease the size of the transformed graph by more than a factor of ten on average. The approaches MAIN and FAST compute overall the same reduced instance sizes. However, on five large social instances, FAST computes smaller reduced graphs, but only with a very small difference. On planar graphs, our reduction rules, as expected, are not working well since they have a mesh-like structure. For mesh graphs, typically, a global perspective is required to decide the solution status of a vertex. Moreover, we suspect that the subset relationship between neighborhoods, a common prerequisite in the reductions, is less likely. The number of vertices in the reduced instance is only reduced by 0.1% compared to the original graph. Altogether, our approach reduces 15 out of 60 instances to an empty instance and thereby solves them solely by our data reductions. Hence, we conclude that the reductions are highly effective in reducing the graph size and especially reduce the size for social networks.

Regarding solution quality overall, our variant FAST is performing only slightly better compared to the other variants, especially on the *small social* and *planar* graphs where no differences between FAST and MAIN can be seen. When considering *large social* graphs, however, we can find no instance

Table 2: Summary comparison of state-of-the-art. Detailed results are presented in tables 6 to 5. Geometric mean over different graph classes of solution size $|S|$ and time t to find it. **Best** results are emphasized in bold. Apx-2P + Im2P with $h = 100$ is omitted as not all planar instances were solved within the time limit.

	gen2pack		Apx-2p + Im2p ($h = 50$)		red2pack b&r		red2pack heuristic	
Class	$\|S\|$	t [s]	$\|S\|$	t [s]	$\|S\|$	t [s]	$\|S\|$	t [s]
cactus	104	1 384 007.11	-		**137**	4.26	**137**	7.30
erdos	8	21 679.67	-		**9**	0.31	**9**	0.53
planar		-	110 009	255.04	92 135	10.41	**110 095**	31 706.65
social (s)		-	-		**159**	11.32	**159**	13.53
social (l)		-	-		30 066	6 442.49	**30 756**	26 377.56

on which 2pack outperforms main or fast. Overall, we achieve an improvement through fast on this graph class of 0.05% compared to 2pack and main. The instance with the largest difference in solution quality is *road_usa*. Here, fast achieves an improvement of 0.93% over the other two strategies. On the instance *amazon-2008* fast performs worse compared to main. On this, the solution quality of main is improved by 0.10% compared to the solution of fast. For all of the six instances, on which fast was outperformed by 2pack, the improvement over fast is always smaller than 0.01%.

Figure 4 also shows that using our different data reduction rules as a preprocessing step (main and fast) especially improves the running time compared to 2pack. Here, we see that, in general, our reductions are improving the performance, and our approach fast works best. In the detailed results in Table 4 in the Appendix, we see that especially for *large social* graphs fast yields a speed up of 2.7 compared to 2pack. On *planar* graphs, where our reductions are not effective in reducing the initial input size, the performance is very similar for all our variants.

Conclusion: We conclude that for all examined criteria, which are the size of the square reduced instance as well as solution quality and running time, our reduction configuration fast performs best. Hence, we choose the fast reduction variant in the following state-of-the-art comparisons for both red2pack_b&r and red2pack_heuristic.

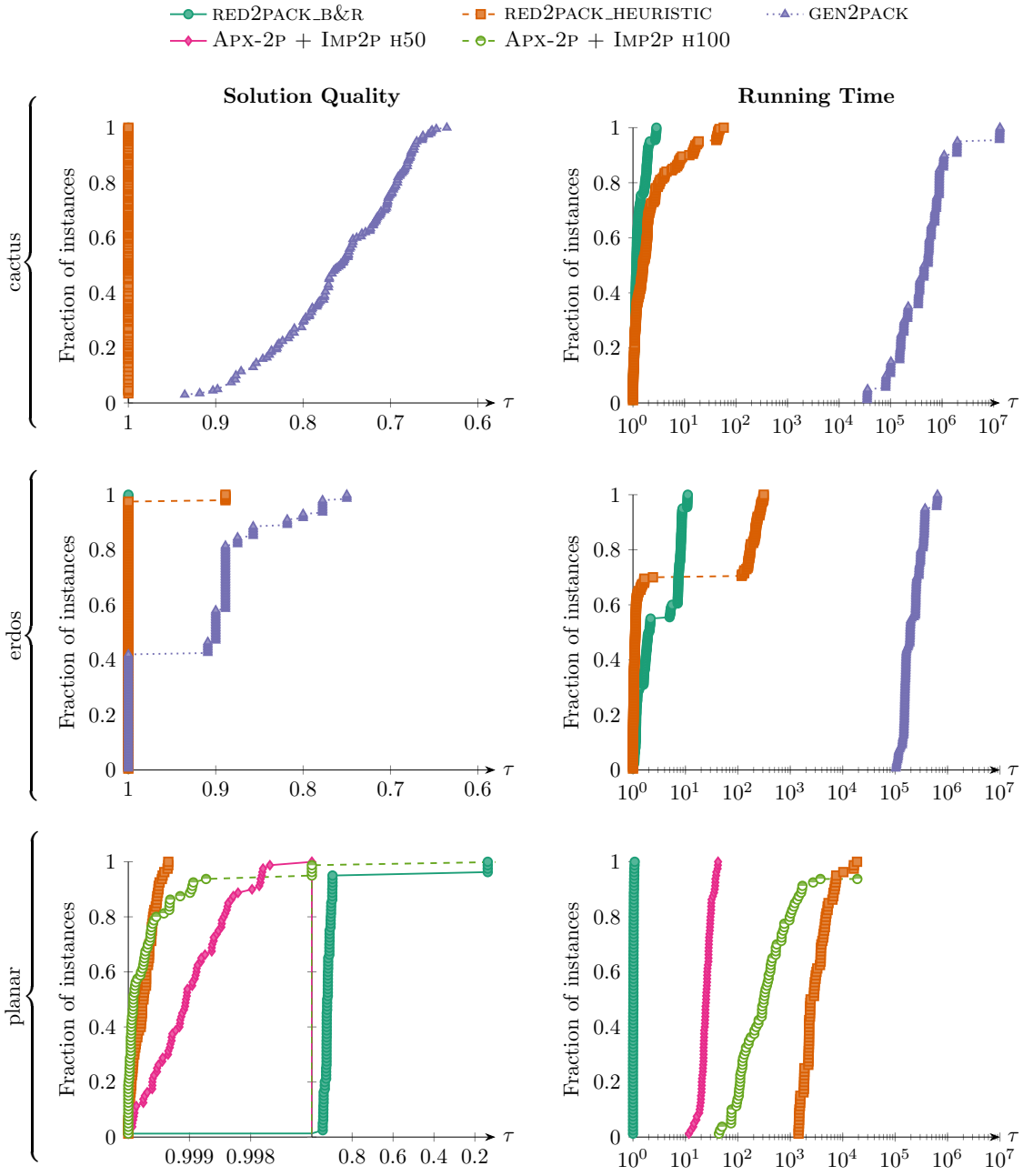
5.2 Comparison against the State of the Art

We now compare red2pack_b&r and red2pack_heuristic using the best reduction variant fast, against the state-of-the-art approaches gen2pack by Trejo-Sánchez et al. [45] as well as two configurations of Apx-2p + Imp2p (h=50 and h=100) by Trejo-Sánchez et al. [49].

gen2pack*:* For the comparison against gen2pack we only use *cactus* graphs and Erdös–Rényi (*erdos*) networks, i.e. the instances used in their paper, as gen2pack is not able to solve any of the other, larger graphs within the given time limit. This can be explained by the initial computations containing matrix multiplication used in gen2pack. These do not finish during the 10-hour limit, so the algorithm does not compute any solution at all. Detailed per-instance results for this comparison can be found in Table 6 in the Appendix.

In Figure 5, we present performance profiles for running time and solution quality. In Table 2, we give the geometric mean running times and solution qualities for these results. Our algorithm

Figure 5: State of the art comparison on solution quality (left) and running time (right) for different graph classes. For planar graphs (bottom) the competitor GEN2PACK is not able to solve the instances. To this end we add APX-2P + IMP2P with $h = 50$ and $h = 100$ as well as RED2PACK_HEURISTIC for comparison.



RED2PACK_B&R as well as RED2PACK_HEURISTIC *find overall the optimal solution in the classes cactus and erdos within a few milliseconds*. Our algorithms outperform GEN2PACK in terms of both solution quality and running time. Especially the differences in running time are very large. On all graphs, our two algorithms are multiple orders of magnitude faster than GEN2PACK. It can only find optimal solutions for 6 out of these 40 graphs, see Table 6. On these two graph classes, *both* of our algorithms always compute the optimum solution quality, resulting in an average solution quality improvement of more than 20% while being a factor of 10^5 times faster than GEN2PACK. Among all instances considered, on *Erdos37-2* GEN2PACK needed the least amount of time to compute an optimum solution. For this instance, we achieve with RED2PACK_B&R a speed up of more than 300 000 and more than 350 000 with RED2PACK_HEURISTIC. The instance on which GEN2PACK needs the most time is *cac1000*. On this instance, RED2PACK_B&R and RED2PACK_HEURISTIC again have similar speedups in the range of 10^5 over GEN2PACK and an improvement in solution quality of roughly 32%. Considering the overall data set, our approach RED2PACK_B&R can solve 63 out of 100 graphs to optimality within less than one second and 71 within the 10-hour time limit and 100 GB restriction. For instances that we could not solve optimally due to experimental restrictions, we present the solution found until this point, shown in tables 4 and 6.

In Table 5, we compare RED2PACK_B&R and RED2PACK_HEURISTIC on social graphs, which are not solvable with the competitors. On these instances, we achieve an average improvement in solution quality of around 1 % with RED2PACK_HEURISTIC compared to RED2PACK_B&R. Especially for large graphs, where our exact solver meets the memory limit, our heuristic is able to outperform RED2PACK_B&R.

APX-2P + IMP2P: Since our reductions do not perform well on planar graphs, we are not able to solve them to optimality with RED2PACK_B&R and exceed the memory threshold quite fast. Overall, the solution quality achieved by RED2PACK_B&R for planar graphs is 84% of the solution quality that the competitor APX-2P + IMP2P ($h = 50$) computes, but it only uses 4% of the time. With RED2PACK_HEURISTIC, on the other hand, we outperform APX-2P + IMP2P ($h = 50$) on all but one instance regarding solution quality, see Table 6. We achieve an average solution quality that is on par in terms of solution quality, i.e. our improvement over the competitor's is 0.08%. Note that the authors experimentally show in [49] that the computed 2-packing set by APX-2P + IMP2P is already at least 99% of the optimum solution. However, on those instances, our algorithms need roughly two orders of magnitude more running time than APX-2P + IMP2P ($h = 50$). APX-2P + IMP2P with $h = 100$ compared to $h = 50$ can improve all but one solution. However, the running time increase is up to multiple orders of magnitude, and some instances were not solved within the 10-hour time limit. RED2PACK_HEURISTIC is able to find better solutions than APX-2P + IMP2P ($h = 100$) on 9 out of 20 instances. Again, the differences in solution quality are very small. Detailed results for these experiments are given in Table 6 and Figure 5 (bottom). The competitor GEN2PACK for general graphs is unable to solve any of these instances, which is why it is omitted in the corresponding tables and performance profiles.

Conclusion: We conclude that on all instances, our approaches outperform GEN2PACK in both solution quality and running time by multiple orders of magnitude. Moreover, we can solve 71 out of 100 instances from our data set to optimality. When comparing against algorithms specialized on planar graphs, we presented two options: one that is at least a factor of 24 times faster with lower solution quality and one that is on par in terms of solution quality but slower, compared to both configurations of the state-of-the-art *specialized* solver for planar graphs.

6 Conclusion and Future Work

This work introduces novel data reduction rules to solve the maximum 2-packing set problem as well as proposes a new exact algorithm RED2PACK_B&R that uses these reductions to solve the maximum 2-packing set problem on large-scale arbitrary graphs. Additionally, a new heuristic RED2PACK_HEURISTIC is introduced that works similarly. Both of the algorithms RED2PACK_B&R and RED2PACK_HEURISTIC work in three phases. First the new data reduction rules are applied to the given input resulting in a reduced instance. Following the reduction phase, the resulting graph is transformed, such that a solution on the transformed graph for the maximum independent set problem corresponds to a solution of the maximum 2-packing set problem for the original graph. The third phase of the algorithms consists of solving the maximum independent set problem on the transformed graph. Our tests indicate that our algorithms outperform the previous algorithm for *arbitrary* graphs both in terms of solution quality and running time on all instances evaluated. For instance, we can compute optimal solutions for 63% of our graphs in under a second, whereas the competing method for arbitrary graphs achieves this only for 5% of the graphs even with a 10-hour time frame. Furthermore, our method successfully solves many large instances that remained unsolved before. Lastly, our approach can compete with a specialized solver on planar instances. Our code is publicly available at https://github.com/KarlsruheMIS/red2pack.

In future work, we want to find more reduction rules, especially for mesh-like graphs. We are also interested in the weighted 2-packing set problem as well as the k-packing set problem for larger values of k and to find independent motifs in graphs via hypergraphs matching algorithms.

References

[1] Personal Communication with A. Lamas. 2023.

[2] F. N. Abu-Khzam, S. Lamm, M. Mnich, A. Noe, C. Schulz, and D. Strash. Recent advances in practical data reduction. In *Collection of Algorithms for Big Data: DFG Priority Program 1736*, pages 97–133. Springer Nature Switzerland, Cham, 2022. doi:10.1007/978-3-031-21534-6_6.

[3] T. Akiba and Y. Iwata. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theoretical Computer Science*, 609, Part 1:211–225, 2016. doi:10.1016/j.tcs.2015.09.023.

[4] D. V. Andrade, M. G. Resende, and R. F. Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics*, 18(4):525–547, 2012. doi:10.1007/s10732-012-9196-4.

[5] D. Bacciu, A. Conte, and F. Landolfi. Generalizing downsampling from regular data to graphs, 2022. arXiv:2208.03523, doi:10.1609/aaai.v37i6.25824.

[6] B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *ACM Journal*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

[7] S. Cai. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 747–753. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/111.

[8] S. Cai, W. Hou, J. Lin, and Y. Li. Improving local search for minimum weight vertex cover by dynamic strategies. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1412–1418. ijcai.org, 2018. doi:10.24963/ijcai.2018/196.

[9] L. Chang. Efficient maximum clique computation and enumeration over large sparse graphs. *VLDB Journal*, 29(5):999–1022, 2020. doi:10.1007/s00778-020-00602-z.

[10] L. Chang, W. Li, and W. Zhang. Computing a near-maximum independent set in linear time by reducing-peeling. In *Proceedings of the ACM International Conference on Management of Data*, pages 1181–1196. ACM, 2017. doi:10.1145/3035918.3035939.

[11] A. Conte, D. Firmani, M. Patrignani, and R. Torlone. A meta-algorithm for finding large k-plexes. *Knowledge and Information Systems*, 63(7):1745–1769, 2021. doi:10.1007/s10115-021-01570-8.

[12] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[13] J. Dahlum, S. Lamm, P. Sanders, C. Schulz, D. Strash, and R. F. Werneck. Accelerating local search for the maximum independent set problem. In *Proceedings of the 15th International Symposium on Experimental Algorithms (SEA)*, volume 9685 of *Lecture Notes in Computer Science*, pages 118–133. Springer, 2016. doi:10.1007/978-3-319-38851-9_9.

[14] Y. Ding, J. Z. Wang, and P. K. Srimani. Self-stabilizing algorithm for maximal 2-packing with safe convergence in an arbitrary graph. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshops*, pages 747–754. IEEE, 2014. doi:10.1109/IPDPSW.2014.86.

[15] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. doi:10.1007/s101070100263.

[16] Y. Dong, A. V. Goldberg, A. Noe, N. Parotsidis, M. G. Resende, and Q. Spaen. A Metaheuristic Algorithm for Large Maximum Weight Independent Set Problems. *Networks*, 85(1):91–112, 2024. doi:10.1002/net.22247.

[17] D. Eppstein. Simple recognition of halin graphs and their generalizations. *Journal of Graph Algorithms and Applications*, 20(2):323–346, 2016. doi:10.7155/jgaa.00395.

[18] A. Flores-Lamas, J. A. Fernández-Zepeda, and J. A. Trejo-Sánchez. Algorithm to find a maximum 2-packing set in a cactus. *Theoretical Computer Science*, 725:31–51, 2018. doi:10.1016/j.tcs.2017.11.030.

[19] A. Flores-Lamas, J. A. Fernández-Zepeda, and J. A. Trejo-Sánchez. A distributed algorithm for a maximal 2-packing set in Halin graphs. *Journal of Parallel and Distributed Computing*, 142:62–76, 2020. doi:10.1016/j.jpdc.2020.03.016.

[20] M. Gairing, R. M. Geist, S. T. Hedetniemi, and P. Kristiansen. A self-stabilizing algorithm for maximal 2-packing. *Nordic Journal of Computing*, 11:1–11, 2004. URL: https://dl.acm.org/doi/abs/10.5555/1029407.1029408, doi:10.5555/1029407.1029408.

[21] M. Gairing, W. Goddard, S. T. Hedetniemi, P. Kristiansen, and A. A. McRae. Distance-two information in self-stabilizing algorithms. *Parallel Processing Letters*, 14(03n04):387–398, 2004. doi:10.1142/S0129626404001970.

[22] M. Gairing, S. T. Hedetniemi, P. Kristiansen, and A. A. McRae. Self-stabilizing algorithms for $\{k\}$-domination. In *Proceedings of Self-Stabilizing Systems*, pages 49–60. Springer, 2003. doi:10.1007/3-540-45032-7_4.

[23] W. Gao, T. Friedrich, T. Kötzing, and F. Neumann. Scaling up local search for minimum vertex cover in large graphs by parallel kernelization. In *Proceedings of the 30th Australasian Joint Conference Advances in Artificial Intelligence*, volume 10400 of *Lecture Notes in Computer Science*, pages 131–143. Springer, 2017. doi:10.1007/978-3-319-63004-5_11.

[24] A. Gellner, S. Lamm, C. Schulz, D. Strash, and B. Zaválnij. Boosting data reduction for the maximum weight independent set problem using increasing transformations. In *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 128–142. SIAM, 2021. doi:10.1137/1.9781611976472.10.

[25] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2009. doi:10.1145/1412228.1412236.

[26] E. Großmann, S. Lamm, C. Schulz, and D. Strash. Finding Near-Optimal Weight Independent Sets at Scale. *Journal of Graph Algorithms and Applications*, 28(1):439–473, 2024. doi:10.7155/jgaa.v28i1.2997.

[27] J. Gu, W. Zheng, Y. Cai, and P. Peng. Towards computing a near-maximum weighted independent set on massive graphs. In *Proceedings of the 27th Conference on Knowledge Discovery & Data Mining*, pages 467–477, 2021. doi:10.1145/3447548.3467232.

[28] J. Gómez Soto, J. Leaños, L. Ríos-Castro, and L. Rivera. The packing number of the double vertex graph of the path graph. *Discrete Applied Mathematics*, 247:327–340, 2018. doi:10.1016/j.dam.2018.03.085.

[29] W. Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980. doi:10.1109/PROC.1980.11899.

[30] M. M. Halldórsson, J. Kratochvíl, and J. A. Telle. Independent sets with domination constraints. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 1998. doi:10.1007/BFb0055051.

[31] D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985. doi:10.1287/moor.10.2.180.

[32] H. Jiang, D. Zhu, Z. Xie, S. Yao, and Z.-H. Fu. A new upper bound based on vertex partitioning for the maximum k-plex problem. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1689–1696, 2021. doi:10.24963/ijcai.2021/233.

[33] S. Lamm, P. Sanders, C. Schulz, D. Strash, and R. F. Werneck. Finding near-optimal independent sets at scale. *Journal of Heuristics*, 23(4):207–229, 2017. doi:10.1007/s10732-017-9337-x.

[34] S. Lamm, C. Schulz, D. Strash, R. Williger, and H. Zhang. Exactly solving the maximum weight independent set problem on large real-world graphs. In *Proceedings of the 21st Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 144–158. SIAM, 2019. doi:10.1137/1.9781611975499.12.

[35] K. Langedal, J. Langguth, F. Manne, and D. T. Schroeder. Efficient minimum weight vertex cover heuristics using graph neural networks. In *Proceedings of the 20th International Symposium on Experimental Algorithms (SEA)*, volume 233 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SEA.2022.12.

[36] R. Li, S. Hu, S. Cai, J. Gao, Y. Wang, and M. Yin. Numwvc: A novel local search for minimum weighted vertex cover problem. *Journal of the Operational Research Society*, 71(9):1498–1509, 2020. doi:10.1080/01605682.2019.1621218.

[37] Y. Li, S. Cai, and W. Hou. An efficient local search algorithm for minimum weighted vertex cover on massive graphs. In *Proceedings of the 11th International Conference on Simulated Evolution and Learning (SEAL)*, volume 10593 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2017. doi:10.1007/978-3-319-68759-9_13.

[38] J. Lin, S. Cai, C. Luo, and K. Su. A reduction based method for coloring very large graphs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 517–523, 2017. doi:10.24963/ijcai.2017/73.

[39] F. Manne and M. Mjelde. A memory efficient self-stabilizing algorithm for maximal k-packing. In *Proceeding of the 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 428–439. Springer, 2006. doi:10.1007/978-3-540-49823-0_30.

[40] M. Mjelde. k-packing and k-domination on tree graphs. Master's thesis, University of Bergen, 2004. URL: https://bora.uib.no/bora-xmlui/handle/1956/1467.

[41] B. C. S. Nogueira, R. G. S. Pinheiro, and A. Subramanian. A hybrid iterated local search heuristic for the maximum weight independent set problem. *Optimimization Letters*, 12(3):567–583, 2018. doi:10.1007/s11590-017-1128-7.

[42] Z. Shi. A self-stabilizing algorithm to maximal 2-packing with improved complexity. *Information Processing Letters*, 112(13):525–531, 2012. doi:10.1016/j.ipl.2012.03.018.

[43] D. Strash. On the power of simple reductions for the maximum independent set problem. In *Proceedings of the International Computing and Combinatorics Conference (CCC)*, pages 345–356. Springer, 2016. doi:10.1007/978-3-319-42634-1_28.

[44] D. Strash and L. Thompson. Effective data reduction for the vertex clique cover problem. In *Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 41–53. SIAM, 2022. doi:10.1137/1.9781611977042.4.

[45] J. A. Trejo-Sánchez, D. Fajardo-Delgado, and J. O. Gutierrez-Garcia. A genetic algorithm for the maximum 2-packing set problem. *International Journal of Applied Mathematics and Computer Science*, 30(1):173–184, 2020. doi:10.34768/amcs-2020-0014.

[46] J. A. Trejo-Sánchez and J. A. Fernández-Zepeda. A self-stabilizing algorithm for the maximal 2-packing in a cactus graph. In *Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 863–871. IEEE, 2012. doi:10.1109/IPDPSW.2012.106.

[47] J. A. Trejo-Sánchez and J. A. Fernández-Zepeda. Distributed algorithm for the maximal 2-packing in geometric outerplanar graphs. *Journal of Parallel and Distributed Computing*, 74(3):2193–2202, 2014. doi:10.1016/j.jpdc.2013.12.002.

[48] J. A. Trejo-Sánchez, J. A. Fernández-Zepeda, and J. C. Ramírez-Pacheco. A self-stabilizing algorithm for a maximal 2-packing in a cactus graph under any scheduler. *International Journal of Foundations of Computer Science*, 28(08):1021–1045, 2017. doi:10.1142/S012905411750037X.

[49] J. A. Trejo-Sánchez, F. A. Madera-Ramírez, J. A. Fernández-Zepeda, J. L. u. López-Martínez, and A. Flores-Lamas. A fast approximation algorithm for the maximum 2-packing set problem on planar graphs. *Optimization Letters*, 14:1435–1454, 2023. doi:10.1007/s11590-022-01939-w.

[50] V. Turau. Efficient transformation of distance-2 self-stabilizing algorithms. *Journal of Parallel and Distributed Computing*, 72(4):603–612, 2012. doi:10.1016/j.jpdc.2011.12.008.

[51] A. Verma, A. Buchanan, and S. Butenko. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, 27(1):164–177, 2015. doi:10.1287/ijoc.2014.0618.

A Reduction and Transformation Details

Table 3: Properties of the input instance (vertices n, edges m, average degree d) and percentage of remaining vertices and edges for different reduction variants after transformation. Bold numbers indicate the **best** results, gray background marks fully reduced graphs.

		input instance properties			2PACK		MAIN		FAST	
Class	Graphs	n	m	d	$n(K^2)/n$	$m(K^2)/m$	$n(K^2)/n$	$m(K^2)/m$	$n(K^2)/n$	$m(K^2)/m$
social (s)	*PGPgiantcompo*	10 680	24 316	4.55	100.00	873.91	**0.00**	**0.00**	**0.00**	**0.00**
	adjnoun	112	425	7.59	100.00	725.18	**0.00**	**0.00**	**0.00**	**0.00**
	as-22july06	22 963	48 436	4.22	100.00	22 941.92	**0.00**	**0.00**	**0.00**	**0.00**
	celegans_metabolic	453	2 025	8.94	100.00	2 239.56	**0.00**	**0.00**	**0.00**	**0.00**
	celegansneural	297	2 148	14.46	100.00	1 123.00	**30.64**	**98.46**	**30.64**	**98.46**
	chesapeake	39	170	8.72	100.00	407.65	**0.00**	**0.00**	**0.00**	**0.00**
	cond-mat	16 726	47 594	5.69	100.00	678.06	**0.00**	**0.00**	**0.00**	**0.00**
	dolphins	62	159	5.13	100.00	381.76	**0.00**	**0.00**	**0.00**	**0.00**
	email	1 133	5 451	9.62	100.00	1 114.57	**0.00**	**0.00**	**0.00**	**0.00**
	email-EuAll	16 805	60 260	7.17	100.00	16 882.03	**0.14**	**0.07**	**0.14**	**0.07**
	football	115	613	10.66	**100.00**	**476.18**	**100.00**	**476.18**	**100.00**	**476.18**
	hep-th	8 361	15 751	3.77	100.00	535.64	**0.00**	**0.00**	**0.00**	**0.00**
	jazz	198	2 742	27.70	100.00	488.48	**6.57**	**1.79**	**6.57**	**1.79**
	lesmis	77	254	6.60	100.00	491.73	**0.00**	**0.00**	**0.00**	**0.00**
	netscience	1 589	2 742	3.45	100.00	245.15	**0.00**	**0.00**	**0.00**	**0.00**
	p2p-Gnutella04	6 405	29 215	9.12	100.00	1 227.55	**74.85**	**638.90**	**74.85**	**638.90**
	polbooks	105	441	8.40	100.00	453.97	**50.48**	**105.67**	**50.48**	**105.67**
	power	4 941	6 594	2.67	100.00	343.18	**2.83**	**4.88**	**2.83**	**4.88**
	soc-Slashdot0902	28 550	379 445	26.58	100.00	8 678.73	**24.12**	**79.04**	**24.12**	79.07
	wordassociation-2011	10 617	63 788	12.02	100.00	2 771.82	**0.00**	**0.00**	**0.00**	**0.00**
social (l)	*G_n_pin_pout*	100 000	501 198	10.02	100.00	1 088.81	**99.35**	**1 075.38**	**99.35**	**1 075.38**
	amazon-2008	735 323	3 523 472	9.58	100.00	901.89	**11.27**	**35.64**	**11.27**	35.66
	astro-ph	16 706	121 251	14.52	100.00	1 469.69	**0.02**	**0.00**	**0.02**	**0.00**
	caidaRouterLevel	192 244	609 066	6.34	100.00	1 843.47	**0.67**	**0.67**	**0.67**	**0.67**
	citationCiteseer	268 495	1 156 647	8.62	100.00	2 980.56	**0.08**	**0.03**	**0.08**	**0.03**
	cnr-2000	325 557	2 738 969	16.83	100.00	20 072.23	**4.32**	**378.64**	4.34	378.94
	coAuthorsCiteseer	227 320	814 134	7.16	100.00	1 005.95	**0.00**	**0.00**	**0.00**	**0.00**
	coAuthorsDBLP	299 067	977 676	6.54	100.00	1 262.80	**0.01**	**0.01**	**0.01**	**0.01**
	coPapersCiteseer	434 102	16 036 720	73.88	100.00	823.64	**0.01**	**0.00**	**0.01**	**0.00**
	coPapersDBLP	540 486	15 245 729	56.41	100.00	1 572.96	**0.01**	**0.00**	**0.01**	**0.00**
	cond-mat-2003	31 163	120 029	7.70	100.00	1 149.60	**0.03**	**0.02**	**0.03**	**0.02**
	cond-mat-2005	40 421	175 691	8.69	100.00	1 467.49	**0.00**	**0.00**	**0.00**	**0.00**
	enron	69 244	254 449	7.35	100.00	9 684.16	**0.00**	**0.00**	**0.00**	**0.00**
	loc-brightkite_edges	56 739	212 945	7.51	100.00	4 026.75	**0.00**	**0.00**	**0.00**	**0.00**
	loc-gowalla_edges	196 591	950 327	9.67	100.00	25 177.85	**0.03**	**0.01**	**0.03**	**0.01**
	preferentialAttachment	100 000	499 985	10.00	**100.00**	**3 386.36**	**100.00**	**3 386.36**	**100.00**	**3 386.36**
	road_central	14 081 816	16 933 413	2.40	100.00	261.10	**13.55**	**35.31**	**13.55**	**35.31**
	road_usa	23 947 347	28 854 312	2.41	100.00	261.17	**13.46**	**35.13**	**13.46**	**35.13**
	smallworld	100 000	499 998	10.00	100.00	550.88	**99.20**	**543.05**	**99.20**	**543.05**
	web-Google	356 648	2 093 324	11.74	100.00	7 560.20	**0.41**	**0.87**	0.42	0.94
planar	*outP500_1*	62 320	65 138	2.09	100.00	213.51	**99.94**	**213.20**	**99.94**	**213.20**
	outP500_2	73 959	76 976	2.08	100.00	212.22	**99.93**	**211.87**	**99.93**	**211.87**
	outP1000_1	148 564	154 609	2.08	100.00	212.19	**99.85**	**211.53**	**99.85**	**211.53**
	outP1000_2	151 091	157 100	2.08	100.00	211.92	**99.87**	**211.29**	**99.87**	**211.29**
	outP1500_1	227 107	236 077	2.08	100.00	211.86	**99.95**	**211.60**	**99.95**	**211.60**
	outP1500_2	226 090	235 100	2.08	100.00	211.97	**99.94**	**211.68**	**99.94**	**211.68**
	outP2000_1	301 431	313 433	2.08	100.00	211.93	**99.93**	**211.58**	**99.93**	**211.58**
	outP2000_2	301 692	313 729	2.08	100.00	211.97	**99.91**	**211.53**	**99.91**	**211.53**
	outP2500_1	375 728	390 874	2.08	100.00	212.10	**99.86**	**211.45**	**99.86**	**211.45**
	outP2500_2	373 931	389 003	2.08	100.00	212.14	**99.91**	**211.72**	**99.91**	**211.72**
	outP3000_1	448 689	466 782	2.08	100.00	212.12	**99.89**	**211.62**	**99.89**	**211.62**
	outP3000_2	451 224	469 413	2.08	100.00	212.11	**99.88**	**211.56**	**99.88**	**211.56**
	outP3500_1	523 959	545 122	2.08	100.00	212.13	**99.88**	**211.55**	**99.88**	**211.55**
	outP3500_2	529 022	550 144	2.08	100.00	212.00	**99.91**	**211.58**	**99.91**	**211.58**
	outP4000_1	600 173	624 188	2.08	100.00	212.01	**99.90**	**211.55**	**99.90**	**211.55**
	outP4000_2	600 288	624 264	2.08	100.00	211.99	**99.92**	**211.61**	**99.92**	**211.61**
	outP4500_1	675 339	702 423	2.08	100.00	212.04	**99.88**	**211.50**	**99.88**	**211.50**
	outP4500_2	677 075	704 222	2.08	100.00	212.05	**99.92**	**211.69**	**99.92**	**211.69**
	outP5000_1	748 383	778 411	2.08	100.00	212.06	**99.91**	**211.64**	**99.91**	**211.64**
	outP5000_2	750 308	780 191	2.08	100.00	211.96	**99.92**	**211.56**	**99.92**	**211.56**
overall	*planar* and *social*				100.00	2 564.50	**43.84**	**185.49**	**43.84**	185.50

Table 4: Solution size $|S|$ and time t needed to find it. The time t_p is the time needed to prove the optimality. Bold numbers indicate **best** results, gray background optimally solved instances.

		2PACK			MAIN-RED2PACK_B&R			FAST-RED2PACK_B&R		
Class	Graphs	$\|S\|$	t [s]	t_p [s]	$\|S\|$	t [s]	t_p [s]	$\|S\|$	t [s]	t_p [s]
social (s)	*PGPgiantcompo*	**2 708**	0.02	0.02	**2 708**	0.01	0.01	**2 708**	**0.01**	**0.01**
	adjnoun	**18**	<0.01	<0.01	**18**	**<0.01**	**<0.01**	**18**	<0.01	<0.01
	as-22july06	**2 026**	4.26	4.26	**2 026**	**0.95**	**0.96**	**2 026**	1.74	1.74
	celegans_metabolic	**29**	0.01	0.01	**29**	**<0.01**	**<0.01**	**29**	<0.01	<0.01
	celegansneural	**14**	0.06	0.06	**14**	**0.05**	**0.06**	**14**	0.06	0.06
	chesapeake	**3**	<0.01	<0.01	**3**	**<0.01**	**<0.01**	**3**	<0.01	<0.01
	cond-mat	**3 391**	0.03	0.03	**3 391**	0.02	0.02	**3 391**	**0.02**	**0.02**
	dolphins	**13**	<0.01	<0.01	**13**	<0.01	<0.01	**13**	**<0.01**	**<0.01**
	email	**209**	0.01	0.01	**209**	0.01	0.01	**209**	**<0.01**	**<0.01**
	email-EuAll	**696**	2.74	2.74	**696**	1.83	1.83	**696**	**1.56**	**1.56**
	football	**7**	1.36	1.39	**7**	1.36	1.39	**7**	**1.35**	**1.39**
	hep-th	**2 611**	0.01	0.01	**2 611**	0.01	0.01	**2 611**	**<0.01**	**0.01**
	jazz	**13**	<0.01	<0.01	**13**	**<0.01**	**<0.01**	**13**	<0.01	<0.01
	lesmis	**10**	<0.01	<0.01	**10**	<0.01	<0.01	**10**	**<0.01**	**<0.01**
	netscience	**477**	<0.01	<0.01	**477**	**<0.01**	**<0.01**	**477**	<0.01	<0.01
	p2p-Gnutella04	**825**	0.35	m.o.	**825**	0.35	m.o.	**825**	**0.33**	m.o.
	polbooks	**12**	**<0.01**	**<0.01**	**12**	<0.01	<0.01	**12**	<0.01	<0.01
	power	**1 465**	**<0.01**	<0.01	**1 465**	<0.01	<0.01	**1 465**	<0.01	**<0.01**
	soc-Slashdot0902	3 280	27.43	m.o.	**3 282**	19.77	m.o.	**3 282**	**3.53**	m.o.
	wordassociation-2011	**2 473**	0.28	0.28	**2 473**	0.04	0.04	**2 473**	**0.02**	**0.02**
overall	*social (s)*	**159**	0.02	-	**159**	0.01	-	**159**	**0.01**	-
social (l)	*G_n_pin_pout*	**7 116**	**8.18**	m.o.	**7 116**	8.46	m.o.	**7 116**	8.64	m.o.
	amazon-2008	106 533	71.17	m.o.	**106 558**	**68.58**	m.o.	106 556	70.55	m.o.
	astro-ph	**2 926**	0.28	0.28	**2 926**	0.09	0.09	**2 926**	**0.05**	**0.05**
	caidaRouterLevel	**40 138**	3.11	3.21	**40 138**	1.49	1.51	**40 138**	**0.73**	**0.74**
	citationCiteseer	**43 238**	10.39	10.39	**43 238**	2.51	2.51	**43 238**	**1.72**	**1.72**
	cnr-2000	**21 897**	1 031.59	m.o.	21 896	986.12	m.o.	**21 897**	**1 030.06**	m.o.
	coAuthorsCiteseer	**33 167**	1.22	1.22	**33 167**	0.56	0.56	**33 167**	**0.40**	**0.40**
	coAuthorsDBLP	**43 960**	2.37	2.37	**43 960**	0.73	0.73	**43 960**	**0.67**	**0.67**
	coPapersCiteseer	**26 001**	47.88	47.88	**26 001**	33.65	33.65	**26 001**	**14.70**	**14.70**
	coPapersDBLP	**35 529**	121.92	121.92	**35 529**	90.89	90.89	**35 529**	**18.21**	**18.21**
	cond-mat-2003	**5 374**	0.17	0.17	**5 374**	0.08	0.08	**5 374**	**0.06**	**0.06**
	cond-mat-2005	**6 505**	0.39	0.39	**6 505**	0.16	0.16	**6 505**	**0.10**	**0.10**
	enron	**4 090**	11.21	11.21	**4 090**	1.42	1.43	**4 090**	**1.25**	**1.26**
	loc-brightkite_edges	**12 940**	2.18	2.18	**12 940**	0.40	0.41	**12 940**	**0.18**	**0.19**
	loc-gowalla_edges	**41 590**	350.55	351.65	**41 590**	76.80	76.82	**41 590**	**70.81**	**70.83**
	preferentialAttachment	**6 397**	**15.35**	m.o.	**6 397**	15.54	m.o.	**6 397**	15.95	m.o.
	road_central	4 289 510	2 566.67	m.o.	4 289 578	2 299.59	m.o.	**4 289 639**	**2 441.79**	m.o.
	road_usa	7 296 706	3 697.55	m.o.	7 296 913	3 043.52	m.o.	**7 297 028**	**3 733.19**	m.o.
	smallworld	**6 872**	**5.19**	m.o.	**6 872**	5.36	m.o.	**6 872**	5.38	m.o.
	web-Google	**30 296**	62.47	63.40	**30 296**	**34.77**	**34.83**	**30 296**	68.16	68.22
overall	*social (l)*	30 065	16.58	-	**30 066**	8.38	-	**30 066**	**6.44**	-
planar	*outP500_1*	**19 140**	**1.64**	m.o.	**19 140**	1.75	m.o.	**19 140**	1.77	m.o.
	outP500_2	**21 894**	**2.16**	m.o.	**21 894**	2.27	m.o.	**21 894**	2.25	m.o.
	outP1000_1	**43 855**	**4.28**	m.o.	**43 855**	4.73	m.o.	**43 855**	4.76	m.o.
	outP1000_2	**45 418**	**4.10**	m.o.	**45 418**	4.53	m.o.	**45 418**	4.65	m.o.
	outP1500_1	**69 368**	**6.33**	m.o.	**69 368**	6.59	m.o.	**69 368**	6.71	m.o.
	outP1500_2	**68 571**	**6.45**	m.o.	**68 571**	6.69	m.o.	**68 571**	6.84	m.o.
	outP2000_1	**90 550**	**8.26**	m.o.	**90 550**	8.60	m.o.	**90 550**	8.76	m.o.
	outP2000_2	**90 507**	**8.44**	m.o.	90 505	9.08	m.o.	90 505	9.18	m.o.
	outP2500_1	**113 643**	**10.83**	m.o.	113 635	11.72	m.o.	113 635	11.99	m.o.
	outP2500_2	**113 073**	**10.73**	m.o.	113 066	11.30	m.o.	113 066	11.55	m.o.
	outP3000_1	135 658	13.22	m.o.	**135 659**	**14.57**	m.o.	**135 659**	14.95	m.o.
	outP3000_2	**136 101**	**13.45**	m.o.	**136 101**	15.11	m.o.	**136 101**	15.67	m.o.
	outP3500_1	161 706	13.85	m.o.	**161 709**	**15.14**	m.o.	**161 709**	15.42	m.o.
	outP3500_2	**162 533**	**14.23**	m.o.	162 528	15.73	m.o.	162 528	16.17	m.o.
	outP4000_1	178 655	17.55	m.o.	**178 671**	**18.77**	m.o.	**178 671**	19.36	m.o.
	outP4000_2	**178 862**	**17.64**	m.o.	178 858	19.12	m.o.	178 858	19.69	m.o.
	outP4500_1	198 172	21.96	m.o.	**198 176**	**23.59**	m.o.	**198 176**	24.17	m.o.
	outP4500_2	**198 440**	**21.81**	m.o.	**198 440**	22.78	m.o.	**198 440**	23.26	m.o.
	outP5000_1	**226 396**	**21.77**	m.o.	226 395	24.31	m.o.	226 395	25.15	m.o.
	outP5000_2	**33 663**	**13.39**	m.o.	**33 663**	15.23	m.o.	**33 663**	15.84	m.o.
overall	*planar*	**92 135**	**9.43**	-	**92 135**	10.19	-	**92 135**	10.41	-
overall	*social* and *planar*	**7 604**	1.50	-	**7 604**	1.04	-	**7 604**	**0.91**	-

B Detailed State-of-the-Art Comparison

Table 5: Solution size $|S|$ and time t needed to find it. The time t_p is the time needed to prove the optimality. Bold numbers indicate **best** results in comparison. Gray background marks optimally solved instances. No competitor is able to solve these instances.

		RED2PACK B&R			RED2PACK HEURISTIC	
Class	Graphs	$\|S\|$	t [ms]	t_p [ms]	$\|S\|$	t [ms]
social (s)	*PGPgiantcompo*	**2 708**	8.34	**8.74**	**2 708**	**8.31**
	adjnoun	**18**	**0.43**	**0.48**	**18**	**0.43**
	as-22july06	**2 026**	**1 737.58**	**1 739.08**	**2 026**	1 774.51
	celegans_metabolic	**29**	**4.77**	**4.84**	**29**	4.81
	celegansneural	**14**	**55.94**	**58.29**	**14**	57.74
	chesapeake	**3**	**0.13**	**0.18**	**3**	**0.13**
	cond-mat	**3 391**	**16.42**	**17.09**	**3 391**	16.92
	dolphins	**13**	0.12	**0.16**	**13**	**0.11**
	email	**209**	**3.44**	**3.58**	**209**	3.49
	email-EuAll	**696**	1 561.44	**1 561.44**	**696**	**1 463.69**
	football	**7**	1 351.83	**1 388.30**	**7**	**223.50**
	hep-th	**2 611**	**4.77**	**5.07**	**2 611**	4.85
	jazz	**13**	**2.18**	**2.18**	**13**	7.12
	lesmis	**10**	**0.16**	**0.20**	**10**	**0.16**
	netscience	**477**	**1.24**	**1.38**	**477**	1.31
	p2p-Gnutella04	825	332.37	m.o.	**837**	**1 695.12**
	polbooks	**12**	**1.97**	**2.62**	**12**	5.57
	power	**1 465**	**3.67**	**3.92**	**1 465**	17.06
	soc-Slashdot0902	3 282	3 532.80	m.o.	**3 288**	**3 314.43**
	wordassociation-2011	**2 473**	**22.04**	**22.50**	**2 473**	22.43
overall	*social (s)*	**159**	**11.32**	-	**159**	13.53
social (l)	*G_n_pin_pout*	7 116	8 640.57	m.o.	**7 970**	**1 706 124.41**
	amazon-2008	106 556	70 549.59	m.o.	**107 165**	**15 503 659.26**
	astro-ph	**2 926**	51.34	**51.34**	**2 926**	**50.86**
	caidaRouterLevel	**40 138**	**732.05**	**742.47**	**40 138**	799.62
	citationCiteseer	**43 238**	1 719.69	**1 719.69**	**43 238**	**1 658.55**
	cnr-2000	21 897	1 030 063.00	m.o.	**21 898**	**931 990.73**
	coAuthorsCiteseer	**33 167**	402.64	**402.64**	**33 167**	**370.94**
	coAuthorsDBLP	**43 960**	672.32	**672.32**	**43 960**	**630.44**
	coPapersCiteseer	**26 001**	**14 702.18**	**14 702.18**	**26 001**	14 979.70
	coPapersDBLP	**35 529**	18 210.66	**18 210.66**	**35 529**	**17 866.85**
	cond-mat-2003	**5 374**	**59.71**	**59.71**	**5 374**	61.45
	cond-mat-2005	**6 505**	**99.69**	**101.85**	**6 505**	100.22
	enron	**4 090**	1 254.90	**1 258.90**	**4 090**	**1 233.50**
	loc-brightkite_edges	**12 940**	182.96	**185.44**	**12 940**	**182.93**
	loc-gowalla_edges	**41 590**	**70 812.67**	**70 825.69**	**41 590**	71 102.93
	preferentialAttachment	6 397	15 947.59	m.o.	**7 034**	**2 608 316.99**
	road_central	4 289 639	2 441 793.42	m.o.	**4 499 839**	**35 841 231.91**
	road_usa	7 297 028	3 733 186.22	m.o.	**7 647 882**	**35 970 721.50**
	smallworld	6 872	5 379.45	m.o.	**7 946**	**11 329 895.89**
	web-Google	**30 296**	68 160.24	**68 222.93**	**30 296**	**67 864.36**
overall	*social (l)*	30 066	6 442.49	-	**30 756**	**26 377.56**
overall		2 184	270.05	-	**2 210**	**597.37**

Table 6: Solution size $|S|$ and time t needed to find it. The time t_p is the time needed to prove the optimality. Bold numbers indicate **best** results in comparison. Gray background marks optimally solved instances. If a solver reached the time limit of 10 hours before computing a solution, we mark it with t.o..

		GEN2PACK		RED2PACK B&R			RED2PACK HEURISTIC	
Class	Graphs	$\|S\|$	t [ms]	$\|S\|$	t [ms]	t_p [ms]	$\|S\|$	t [ms]
cactus	*cac50*	15	25 291.12	**17**	**0.05**	**0.09**	**17**	**0.05**
	cac100	28	85 674.26	**31**	**0.09**	**0.14**	**31**	**0.09**
	cac150	42	181 973.07	**49**	**0.31**	**0.31**	**49**	13.49
	cac200	55	313 878.10	**65**	**10.27**	**10.70**	**65**	14.47
	cac250	68	484 267.30	**82**	0.69	**0.79**	**82**	**0.25**
	cac300	80	712 988.40	**100**	**0.83**	**0.83**	**100**	13.07
	cac350	92	963 595.66	**116**	**12.89**	**13.35**	**116**	13.66
	cac400	103	1 230 210.54	**133**	**14.58**	**15.26**	**133**	30.00
	cac450	114	1 574 182.46	**148**	**12.22**	**12.66**	**148**	13.70
	cac500	126	1 987 312.28	**166**	23.73	**25.24**	**166**	**17.65**
	cac550	136	2 389 870.11	**179**	**15.61**	**15.98**	**179**	18.59
	cac600	146	2 779 427.76	**199**	0.24	**0.30**	**199**	**0.23**
	cac650	158	3 328 099.16	**214**	**28.90**	**29.19**	**214**	58.65
	cac700	165	3 859 839.95	**232**	**7.68**	**7.72**	**232**	8.82
	cac750	172	4 477 170.66	**250**	**11.58**	**11.58**	**250**	19.19
	cac800	184	5 088 716.64	**264**	**18.26**	**18.87**	**264**	19.84
	cac850	196	5 648 560.45	**282**	26.46	**27.04**	**282**	**26.27**
	cac900	205	6 490 942.35	**300**	**10.92**	**11.35**	**300**	29.14
	cac950	214	7 281 058.31	**315**	**10.18**	**10.18**	**315**	17.17
	cac1000	223	8 084 805.96	**332**	**17.02**	**17.46**	**332**	58.03
overall	*cactus*	104	1 384 007.11	**137**	**4.26**	**4.66**	**137**	7.30
erdos	*Erdos37-2*	**9**	21 118.31	**9**	0.07	**0.11**	**9**	**0.06**
	Erdos37-16	8	21 119.42	**9**	**1.13**	**1.25**	**9**	2.31
	Erdos37-23	9	21 104.45	**10**	**0.09**	**0.13**	**10**	**0.09**
	Erdos37-44	**7**	21 318.79	**7**	0.99	**1.19**	**7**	**0.14**
	Erdos37-45	10	21 328.35	**11**	0.09	**0.13**	**11**	**0.08**
	Erdos38-2	**9**	21 706.27	**9**	**0.07**	**0.11**	**9**	**0.07**
	Erdos38-14	8	21 915.50	**9**	**0.59**	**0.68**	**9**	2.50
	Erdos38-18	6	21 969.36	**7**	**1.25**	**1.35**	**7**	4.59
	Erdos38-46	8	21 927.82	**9**	**1.05**	**1.19**	**9**	14.31
	Erdos38-48	8	21 643.77	**9**	**0.96**	**1.17**	**9**	1.30
	Erdos39-14	**9**	22 997.65	**9**	0.16	**0.16**	**9**	**0.08**
	Erdos39-22	10	22 626.53	**11**	**0.06**	**0.10**	**11**	**0.06**
	Erdos39-25	**8**	23 005.90	**8**	1.30	**1.69**	**8**	**0.48**
	Erdos39-29	9	22 800.22	**10**	**0.07**	**0.12**	**10**	**0.07**
	Erdos39-44	8	22 758.11	**9**	0.11	**0.15**	**9**	**0.10**
	Erdos40-0	9	23 138.60	**10**	**0.18**	**0.18**	**10**	1.79
	Erdos40-4	8	19 244.44	**9**	**0.26**	**0.26**	**9**	1.21
	Erdos40-8	7	19 161.29	**8**	**1.87**	**2.02**	**8**	12.74
	Erdos40-10	**10**	23 474.02	**10**	**0.22**	**0.22**	**10**	1.04
	Erdos40-43	8	19 918.83	**9**	**1.07**	**1.18**	**9**	3.51
overall	*erdos*	8	21 679.67	**9**	**0.31**	**0.38**	**9**	0.53

		APX-2P + IM2P ($h = 100$)		APX-2P + IM2P ($h = 50$)		RED2PACK B&R		RED2PACK HEURISTIC	
Class	Graphs	$\|S\|$	t [s]	$\|S\|$	t [s]	$\|S\|$	t [s]	$\|S\|$	t [s]
planar	*outP500_1*	**20 333**	**97.51**	20 327	22.99	19 140	1.77	20 332	26 166.03
	outP500_2	**24 187**	**1 978.02**	24 144	58.42	21 894	2.25	24 181	12 631.72
	outP1000_1	**48 575**	**1 066.37**	48 472	145.79	43 855	4.76	48 567	29 215.74
	outP1000_2	49 429	692.06	**49 430**	**111.35**	45 418	4.65	49 416	28 336.12
	outP1500_1	74 232	794.47	74 170	166.89	69 368	6.71	**74 292**	**31 577.14**
	outP1500_2	73 901	2 638.12	73 813	161.62	68 571	6.84	**73 922**	**31 015.92**
	outP2000_1	-	t.o.	98 519	333.63	90 550	8.76	**98 555**	**32 735.67**
	outP2000_2	98 643	2 062.41	98 558	251.90	90 505	9.18	**98 669**	**33 103.01**
	outP2500_1	122 826	1 092.08	122 761	233.20	113 635	11.99	**122 831**	**34 384.01**
	outP2500_2	**122 322**	**4 897.24**	122 275	319.30	113 066	11.55	122 282	34 018.53
	outP3000_1	**146 748**	**1 739.13**	146 622	258.05	135 659	14.95	146 696	35 371.22
	outP3000_2	**147 544**	**1 633.60**	147 360	341.67	136 101	15.67	147 522	35 483.06
	outP3500_1	**171 303**	**14 025.36**	171 127	490.08	161 709	15.42	171 264	34 414.78
	outP3500_2	-	t.o.	172 720	351.27	162 528	16.17	**172 956**	**35 585.91**
	outP4000_1	-	t.o.	196 172	490.55	178 671	19.36	**196 189**	**35 262.82**
	outP4000_2	196 218	19 284.45	196 076	540.26	178 858	19.69	**196 243**	**35 367.19**
	outP4500_1	**220 799**	**4 932.77**	220 677	505.56	198 176	24.17	220 737	35 650.42
	outP4500_2	**221 406**	**11 015.21**	221 283	503.34	198 440	23.26	221 345	35 643.68
	outP5000_1	**244 641**	**12 413.34**	244 350	595.94	226 395	25.15	244 548	35 605.20
	outP5000_2	245 031	6 309.16	245 038	603.84	33 663	15.84	**245 223**	**35 789.86**
overall	*planar*	-	-	110 009	255.04	92 135	10.41	**110 095**	**31 706.65**