Research Article

Uroš Čibej* and Jurij Mihelič

Graph automorphisms for compression

https://doi.org/10.1515/comp-2020-0186
Received Feb 28, 2020; accepted May 08, 2020

Abstract: Detecting automorphisms is a natural way to identify redundant information presented in structured data. When such redundancies are detected they can be used for data compression. In this paper we explore two different classes of graphs to capture this intuitive property of automorphisms. Symmetry-compressible graphs are the first class which introduces the basic concepts but use only global symmetries for the compression. In order for this concept to be more practical, we need to use local symmetries. Thus, we extend the basic graph class with Near Symmetry compressible graphs. Furthermore, we develop two algorithms that can be used to compress practical instances and empirically evaluate them on a set of realistic graphs.

Keywords: graph automorphisms, graph classes, compression algorithms

1 Introduction

Due to an enormous increase in data gathering and generating (according to [15] humanity will generate many zettabytes by 2025), data compression is an important research topic to enable good resource exploitation. Another interesting phenomenon is the emergence of different network data as well as new kinds of databases that are more suitable to manipulate such data. These databases are called graph databases [16]. Graphs are a general structure for representing more complex relations between various entities. Graphs originate in discrete mathematics and have been extensively studied for a few centuries. Many concepts have been developed in graph theory that have turned out to be extremely useful in practice, and in this paper, we explore another such concept and demonstrate its practical potential.

Many methods have been developed for lossless graph compression. This topic is approached in a wide spectrum of areas such as succinct data structures [8] where the most memory efficient data structures are considered. The second approach deals with graph compression in specific domains such as graph databases [1], web graphs [2], hierarchical schemes [7] and many others. These compression techniques mostly exploit domain specific knowledge and in many cases established compression techniques are utilized from e.g. text compression. When dealing with compression of pure structural data, i.e., only the graph and no metadata, some approaches utilize graph grammars [13], specific subgraphs, such as cliques [17], stars [12], and some results from extremal combinatorics such as the regularity lemma [14].

For a very extensive survey of different methods please see [3]. But to our knowledge, none of the methods use symmetries as the underlying compression mechanisms, which is the novelty presented in this paper.

People detect symmetries mostly visually and we consider ourselves good at spotting such self-similarities in images. However, in general data, symmetries can be uncovered only when an appropriate view of data has been constructed. Mathematicians, therefore, have formalized the notion of symmetry in graphs to more accurately capture this intuitive notion. Namely, graph symmetries are mathematically defined as graph automorphisms, which are bijective mappings between the vertices of a graph, such that the connectivity between the mapped vertices remains the same as in the original graph. In any graph, there can be many such symmetries.

In this paper, we introduce a representation of graphs that uses one of the graph automorphisms to describe a set of its edges. When such a representation requires less data than the original description then we will call such a graph symmetry-compressible. Unfortunately, realistic graphs rarely exhibit global symmetries, so we need another concept that captures local symmetries. This will enable us to test our concept on a larger class of instances. We will denote such graphs as near symmetry-compressible. A simple example is given in Figure 1.

***Corresponding Author: Uroš Čibej:** University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia; Email: uros.cibej@fri.uni-lj.si
Jurij Mihelič: University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia; Email: jurij.mihelic@fri.uni-lj.si

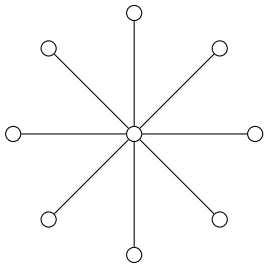


Figure 1: A highly symmetrical graph, describing such a structure naively would be just a set of edges, whereas we could state just one edge and a rotational symmetry, which would then generate the entire graph. We formalize this intuitive notion in this paper.

Because we strive for practical applications of the introduced concepts, we dedicate the second part of the paper to practical algorithms and an empirical evaluation, which shows the potential for compression of real graph data, as well as data that is not directly given as a graph (such as images).

The paper is structured as follows. The next section gives some preliminaries and introduces the concept of symmetry-compressible graphs. The third section extends this concept with near symmetry-compressible graphs. The fourth section starts the practical part of the paper, describing two heuristic algorithms for graph compression. Section 5 describes an empirical evaluation of the proposed algorithm, and Section 6 concludes the paper by summarizing the obtained results and giving some pointers for future work.

2 Symmetry-compressible graphs

This section formalizes the main idea of using symmetries for data compression. We encompass this in a class of graphs that get compressed when using such a representation. We start by defining some basic concepts that are standard in graph theory.

2.1 Preliminaries

A graph is presented as a set of edges, $G \subseteq V \times V$, where V is some set of vertices. We will use mostly the set of vertices $\{1, 2, \ldots, n\}$. Without loss of generality, we assume the graph has no isolated vertices. To explicitly denote the set of vertices of the graph, we will write $V(G)$.

We work with two related representations of symmetries. The standard representation is given by permutations of the vertex set, i.e., bijective functions $\pi : V(G) \to V(G)$ that preserves connectivity ($(u, v) \in G \implies (\pi(u), \pi(v)) \in G$). All such permutations form a permutation group $Aut(G)$. We also require an alternative representation of an automorphism π, namely, the permutation of the edges. The edge permutation, induced by the vertex permutation π, is defined as $\overline{\pi}((u, v)) = (\pi(u), \pi(v))$.

Permutations are written in standard cycle notation, i.e., as a set of disjoint cycles. This set also contains the cycles with only one element (identities of the permutation). The notation $cyc(\pi)$ denotes the set of all the cycles of the permutation π. The same goes for the edge permutation $\overline{\pi}$. To obtain the cycle which contains a vertex v (or edge e), we use $cyc(\pi, v)$ ($cyc(\overline{\pi}, e)$ for edges). We will also consider cycles as sets, e.g. $c \in cyc(\pi)$, when we will denote a pair that is a part of a cycle.

Since graphs are represented as a set of pairs, to have a comparable representation, the permutations will also be represented as a set of pairs. Trivially, since it is a bijection, a symmetry can be represented by $|V(G)|$ pairs. However, we can remove many redundancies in such a representation. Namely, pairs of the form (x, x) can be omitted, and one pair from each cycle can also be left out. The size (of the representation) is expressed in terms of the cycle sizes as:

$$|\pi| = \sum_{c \in cyc(\pi)} (|c| - 1),$$

where $|c|$ denotes the length of the cycle.

2.2 Definition of the graph class

Knowing a symmetry π of the graph G, we do not have to represent all the edges of the graph since some edges can be generated using π. More specifically, for each cycle of the edge permutation $\overline{\pi}$, only one edge is required. This is the main observation that will be used to compress the graphs.

We denote the required part of the graph (under the symmetry π) as G^π, and define it as

$$G^\pi = \left\{ e \in G \mid e = \min_{f \in cyc(\overline{\pi}, e)} f \right\}$$

and call it a *residual graph*. This definition chooses the minimal element from each cycle of the permutation $\overline{\pi}$. The ordering of the elements can be arbitrary, meaning basically that exactly one element from each cycle is selected.

The pair (π, G^π) is an alternative representation of the graph G since the entire edge set of the graph can be uniquely reconstructed from it. The reconstruction takes every edge of G^π and applies π to both ends of the current edge, until one cycle of $\overline{\pi}$ is obtained.

The size of this alternative representation is in general different from the size of the original graph, depending on both the graph itself as well as on the symmetry in consideration. We define a family of graphs, which possess symmetries for which the alternative representation is smaller than G.

Definition 1 (Symmetry-compressible graphs)**.** *A graph G is symmetry compressible ($G \in \mathcal{SC}$) if there exists*

$$\pi \in Aut(G) \text{ such that } |\pi| + |G^\pi| < |G|.$$

To clarify the introduced concepts, Table 1 shows a simple undirected graph C_4, and three of its representative symmetries. Two of these symmetries increase the size of representation, whereas one symmetry reduces the size by 1, meaning $C_4 \in \mathcal{SC}$.

2.3 Properties

To better understand the newly defined concept, let us look at some basic properties of symmetry-compressible graphs. We will identify a set of theorems that characterize graph families that we know are compressible and some of them which are not.

Let us first explore graphs with more than one connected components, and symmetries that map one component onto the other.

Theorem 1. *Let us assume G is composed of two connected components G_1 and G_2 and there is a symmetry π mapping the component G_1 onto G_2. If $|G_1| > |V(G_1)|$, then $G \in \mathcal{SC}$.*

Proof. Assuming that the size of the symmetry is $|\pi| = |V(G_1)|$, $|G^\pi| = |G_1|$, $|G| = 2|G_1|$, and $|V(G_1)| < |G_1|$, we have the inequality

$$|G_1| + |V(G_1)| < 2|G_1|$$

and making the substitution we get

$$|G^\pi| + |\pi| < |G|,$$

which means $G \in SC$. □

This theorem gives us a criterion (if we know the group $Aut(G)$) for checking if symmetries between connected components can compress a graph. Thus we will mainly focus on determining whether symmetries in connected graphs can compress a graph.

Now let us define a special type of $\mathcal{SC}$ graph and prove a lemma about this type of $\mathcal{SC}$ graph that will be useful in proving that trees are not symmetry-compressible.

Definition 2 (Minimal $\mathcal{SC}$ graph)**.** *A graph $G \in \mathcal{SC}$, π being its compressing symmetry, such that for every $e \in G$ such that $\overline{\pi}(e) = e$, is called a* minimal $\mathcal{SC}$ graph.

In the following analysis of $s\mathcal{SC}$ graphs we can focus mainly on graphs that are minimal $\mathcal{SC}$ graphs because of the following lemma.

Lemma 1. *All $G \in \mathcal{SC}$ contain a minimal $\mathcal{SC}$ subgraph.*

Proof. Let $G \in \mathcal{SC}$, π its compressing symmetry, and $F \subseteq G$ such that for all $e \in F$ such that $\overline{\pi}(e) = e$, in particular, the edges are fixed by symmetry π. We can show that the minimal $\mathcal{SC}$ subgraph is simply $H = G \setminus F$. Notice that by removing edges that are fixed by the symmetry π the graph $G \setminus F$ remains symmetric under the symmetry π. Notice also that the edges not moved by π are also present in G^π and that $G^\pi \setminus F = (G \setminus F)^\pi$. So since we assume $G \in \mathcal{SC}$ then

$$|\pi| + |G^\pi| < |G|,$$

by removing the edges that remain fixed under the symmetry we get

$$|\pi| + |G^\pi \setminus F| < |G \setminus F|,$$

which means that $G \setminus F \in \mathcal{SC}$. □

The second lemma will show a result for trees that are candidates for being minimal $\mathcal{SC}$ trees.

Lemma 2. *In any tree T where a symmetry $\pi \in Aut(T)$ moves all the edges, exactly one vertex is fixed.*

Proof. This follows from the fact that any tree has one or two centers [10]. Since any automorphism preserves distances, the case of two centers is not possible since the two centers would either be mapped to each other or would stay fixed. In both cases the edge between them would stay fixed thus contradicting the assumption that all edges must be moved. So the only option is that there is only one center, which cannot be mapped to any other vertex, so it is fixed by any automorphism. □

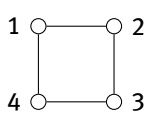
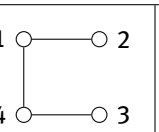
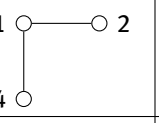
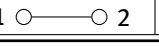
We now move on to the main result about trees.

Theorem 2. *If T is a tree, then $T \notin \mathcal{SC}$.*

Proof. By definition, we have to show for all $\pi \in Aut(T)$, $|\pi| + |T^\pi| < |T|$.

By Lemma 1, if a tree would be symmetry-compressible, then it would contain a subgraph (i.e., a forest) that is a minimal $\mathcal{SC}$ graph. But from Theorem 1 we know that we can focus only on connected subgraphs

Table 1: Three representative symmetries π ($\overline{\pi}$ also shown for easier interpretation of results) of the 4-cycle, the corresponding compressed graph G^π, and the size of (π, G^π). The symmetry shown in row 2 results in a smaller representation than the original graph, making $C_4 \in \mathcal{SC}$.

G	π	$\overline{\pi}$	G^π	$\lvert\pi\rvert + \lvert G^\pi\rvert$
	(12)(43)	$(\langle 1,4\rangle\langle 2,3\rangle)$		$2 + 3 \not< \lvert G\rvert$
	(13)	$(\langle 1,2\rangle\langle 2,3\rangle)(\langle 1,4\rangle\langle 3,4\rangle)$		$1 + 2 < \lvert G\rvert$
	(1234)	$(\langle 1,4\rangle\langle 1,2\rangle\langle 2,3\rangle\langle 3,4\rangle)$		$3 + 1 \not< \lvert G\rvert$

since any symmetry mapping one tree onto another tree cannot compress the graph since the number of edges is less than the number of vertices in a component.

By Lemma 2 we focus only on symmetries where only one vertex is fixed and is therefore the pivot of the symmetry. Based on this pivot (let us say it has k neighbours), we can partition the entire tree into subtrees $T_1, \ldots, T_k$, where these subtrees are trees originating from the pivot (all of them contain the pivot). Any symmetry that moves all the vertices has to map pairs of these trees onto each other. Without loss of generality we can thus assume that we have only two trees, T_1 and T_2.

The two trees are edge disjoint, thus $|T_1 \cup T_2| = |T_1| + |T_2| = 2|T_1|$. There are $|T_1| + 1$ vertices in $|T_1|$, but since there is one vertex that remains fixed under π, we need $|T_1|$ pairs to describe the symmetry. The size of the representation is thus $|(T_1 \cup T_2)^\pi| + |\pi| = |T_1| + |T_1| = 2|T_1|$, which is the same size as $|T_1 \cup T_2|$. □

We now explore the compressibility of cycle graphs.

Theorem 3. *For cycle graphs C_k the following holds: if k is even, $C_k \in \mathcal{SC}$, if k is odd $C_k \notin \mathcal{SC}$.*

Proof. If k is even, then the reflexive symmetry which leaves two vertices fixed compresses the graph. Namely, the other $k-2$ are mapped onto each other, i.e., $|\pi| = \frac{k-2}{2}$. The graph G^π constitutes exactly half of the edges of the original graph, i.e., $\frac{k}{2}$. The entire size of the representation is

$$|\pi| + |G^\pi| = \frac{k-2}{2} + \frac{k}{2} = k - 1 < k = |G|.$$

Even cycles are therefore symmetry-compressible.

If k is odd, we have to consider both types of symmetries that cycles have: rotational and reflective symmetries.

Let us first look at the reflective symmetry, where only one scenario is possible, namely when the cycle is reflected over one node and one edge (both remain fixed by the symmetry). Thus $|\pi| = \frac{k-1}{2}$. This symmetry maps one edge to itself, all the other edges are mapped to a different edge. This makes the size of $|G^\pi| = \frac{k-1}{2} + 1$. The entire size of the representation is

$$|\pi| + |G^\pi| = \frac{k-1}{2} + \frac{k-1}{2} + 1 = k = |C_k|.$$

And finally we investigate the rotational symmetries of an odd cycle. It is easy to see that the rotational symmetry does not compress any C_k. Let us suppose we have a rotation for m positions. Let us denote the length of any cycle as l (all cycles have the same length). Then the number of the cycles of the permutation is equal to $\frac{k}{l}$. From this is follows that

$$|\pi| + |G^\pi| = \frac{k}{l}(l-1) + \frac{k}{l} = k - \frac{k}{l} + \frac{k}{l} = k = C_k,$$

which shows that odd cycles are not symmetry compressible.

□

The following theorem demonstrates the relation between π and $\overline{\pi}$ for symmetry-compressible graphs.

Theorem 4. *A graph G is symmetry-compressible, if and only if there exists $\pi \in Aut(G)$, such that $|\pi| < |\bar{\pi}|$.*

Proof. ($\Longrightarrow$) Assume $G \in \mathcal{SC}$, which means there exists a $\pi \in Aut(G)$ such that $|\pi| < |G| - |G^\pi|$. Knowing that any edge permutation $\overline{\pi}$ in cycle notation contains exactly all the edges of G, i.e.,

$$|G| = \sum_{c \in cyc(\overline{\pi})} |c|,$$

and that G^π contains exactly one edge from each cycle of $\overline{\pi}$, i.e.,

$$|G^\pi| = \sum_{c \in cyc(\overline{\pi})} 1,$$

we can write

$$|G| - |G^\pi| = \sum_{c \in cyc(\overline{\pi})} |c| - \sum_{c \in cyc(\overline{\pi})} 1 = \sum_{c \in cyc(\overline{\pi})} |c| - 1$$

which is exactly the definition of $|\overline{\pi}|$. Thus $|\pi| < |\overline{\pi}|$.

($\Longleftarrow$) Assuming there exists $\pi \in Aut(G)$ such that $|\pi| < |\overline{\pi}|$ and knowing that $|G| - |G^\pi| = |\overline{\pi}|$ we get

$$|\pi| < |G| - |G^\pi|$$

and then simply

$$|\pi| + |G^\pi| < |G|,$$

which is the definition of $G \in \mathcal{SC}$. □

3 Near Symmetry-compressible Graphs

Graphs arising in practice rarely exhibit (non-trivial) symmetries, making them non-compressible using the representation (π, G^π). In this section we present an extension to the class of graphs $\mathcal{SC}$, which includes also many graphs without global symmetries, making it a more viable possibility for compression of realistic graphs.

A graph G might not exhibit any significant symmetries, but with a few modifications many new symmetries might arise, resulting in a symmetry-compressible graph H. Let us first give the representation of the allowed transformations of G. We restrict the modifications to adding and removing edges. Both adding and removing an edge can be described by simply specifying the edge; if the edge is present in the graph it is a deletion, otherwise it is an addition of the edge. The entire set of transformations from G to H can be described as the symmetric difference of both graphs, i.e., $G \oplus H$.

Of course, any graph can be transformed into a symmetry-compressible graph (e.g. a clique), but we are interested in graphs where the transformation is worthwhile, in the sense that it can result in a smaller representation.

Definition 3 (Near Symmetry-Compressible graph)**.** *A graph is near symmetry-compressible ($G \in \mathcal{NSC}$) if there exist H and a $\pi \in Aut(H)$ such that*

$$|\pi| + |H^\pi| + |G \oplus H| < |G|.$$

A simple example of an $\mathcal{NSC}$ graph is shown in Figure 2.

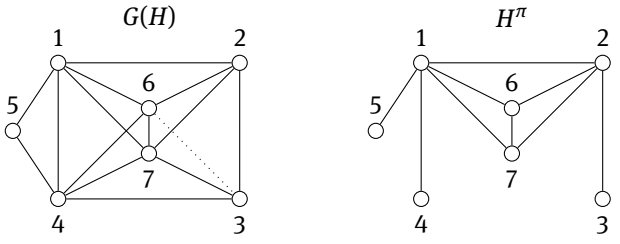


Figure 2: A graph G (full lines) which has no symmetries, but is $\mathcal{NSC}$. The dotted line shows the edge that, when added, induces new symmetries in the graph, one of them, namely $(14)(23)$, reduces the representation size by 2, since $|G| = 14$, but $|\pi| + |H^\pi| + |G \oplus H| = 2 + 9 + 1 = 12$.

The following theorem establishes a relation between the classes $\mathcal{SC}$ and $\mathcal{NSC}$.

Theorem 5. *For $H \subseteq G$, $H \in \mathcal{SC} \Longrightarrow G \in \mathcal{NSC}$.*

Proof. We know that there exists a $\pi : |\pi| + |H^\pi| < |H|$ (since $H \in \mathcal{SC}$). We can show that $G \in \mathcal{NSC}$ by noticing that $|G \oplus H| = |G| - |H|$, from which it follows that:

$$|\pi| + |H^\pi| + |G \oplus H| = |\pi| + |H^\pi| + (|G| - |H|) < < |H| + (|G| - |H|) = |G|,$$

which is exactly the definition of near symmetry-compressibility. □

4 Algorithms

In this section, we focus on practical compression algorithms, based on the notions of $\mathcal{SC}$ and $\mathcal{NSC}$. The first algorithm searches for small compressible patterns inside the graph, whereas the second algorithm is more general, finding arbitrary large compressible patterns of a particular type.

In order to compare the quality of the algorithms, we must find a suitable measure of how good a compression is. Since a graph can have more than one symmetry with the above property, we also define a natural optimization problem. Let us first define two measures of "compressibility", which will enable us to compare different symmetries. The first measure is the absolute efficiency of the symmetry

on G

$$\Delta(\pi, G) = |G| - |\pi| - |G^{\pi}|.$$

In order to compare the efficiencies of the symmetry compression on different graphs, we also consider the relative efficiency of the compression, i.e.,

$$\Delta^r(\pi, G) = \frac{\Delta(\pi, G)}{|G|}.$$

For symmetry-compressible graphs $\Delta^r(\pi, G) \in (0, 1)$.

4.1 Graphlet based compression

In Theorem 5, we showed that graphs containing $\mathcal{SC}$ subgraphs are $\mathcal{NSC}$. This fact is used here to obtain a simple and practical algorithm. Namely, we focus on small graphs (a.k.a. graphlets). To do this we first find the symmetry-compressible subset of graphlets with 4 to 9 vertices. Table 2 summarizes the results, showing that a significant number of graphlets can be compressed. To give an idea of how much these graphlets can be compressed, we also computed the average and the maximum relative compression efficiency on these sets.

The algorithm uses a list of these graphlets as the basis for the compression. For each graphlet G' in this list, a maximal set of edge-disjoint subgraphs $H \subseteq G$ that are isomorphic to G' are found. All such subgraphs H are removed from G and their symmetry representation (π, H^{π}) is used instead. Algorithm 1 gives a more detailed description of the described procedure. To achieve better results (a higher compression ratio), a heuristic rule is used. Namely, the list of graphlets is initially sorted by decreasing relative efficiency. The intuition behind this heuristic is that we need to find the most compressible patterns first, otherwise the removal of subgraphs makes such patterns less likely to occur.

Table 2: Symmetry compressible graphlets. We generated all connected graphs of size 4-9, the number of such graphlets is given in column 2. For each of these graphlets, we checked if it is symmetry-compressible. The number of such graphlets is given in column 3. Column 4 gives the average relative efficiency and column 5 gives the maximum relative efficiency.

#	all	$\mathcal{SC}$	avg. Δ^r	max. Δ^r
4	6	3	0.53	0.67
5	21	13	0.49	0.75
6	112	87	0.43	0.8
7	853	649	0.39	0.86
8	11117	7254	0.34	0.88
9	261080	126221	0.29	0.9

Algorithm 1 Compressing the graph with symmetric graphlets.

```
function GraphletCompress(G)
    Comp = []
    for all (π, G') ∈ Graphlets do
        while ∃H ⊆ G : H ≅ G' do
            let π_H be the automorphism for H yielded by π
            Comp = Comp + (π_H, H^{π_H})
            G = G \ H
    return Comp + G
```

The presented algorithm uses the search for subgraphs extensively. Even though, in theory, this is a computationally intractable problem, many practical approaches exist [5, 6]. These algorithms work well also on large instances of graphs, especially when pattern graphs are small. And this is exactly the context we are using it in our algorithm, so the subgraph isomorphism is tractable for our application.

4.2 Compression with complete bipartite graphs

The second algorithm builds on the observation that complete bipartite graphs are very compressible, i.e., have a large relative efficiency. However, in real graphs, it is difficult to find large complete bipartite graphs. Instead of searching for complete subgraphs, the algorithm strives to find dense bipartite graphs and adds edges to them until they are complete. In order for this to work, we must investigate when such addition is possible so that we do not increase the size of the final representation.

Let us first give a few basic definitions. We denote complete bipartite graphs on two vertex sets U, V as $K(U, V)$. Let $|U| = u$ and $|V| = v$. The size of the complete bipartite graph $|K(U, V)| = uv$. We will also use the notation $G(U, V)$ to signify the bipartite subgraph of G on vertex sets U and V. There are many symmetries in $K(U, V)$, we will use only one, which works well for any choice of u and v. The symmetry used henceforth is $\pi = (n_1 n_2 n_3 \dots n_u)$, where $n_i \in U$, therefore $|\pi| = u - 1$. The graph $K(U, V)^{\pi}$ consists of all the edges from one vertex of U to all vertices of V, i.e., $|K(U, V)^{\pi}| = v$. This results in a representation of size $v + (u - 1)$, which (for all $u, v \geq 2$) is smaller than uv, i.e., the size of the original graph.

As mentioned earlier, graphs do not necessarily contain (large) complete bipartite subgraphs. However, they often contain large dense subgraphs, which can be transformed into $K(U, V)$ with only a few additional edges. For

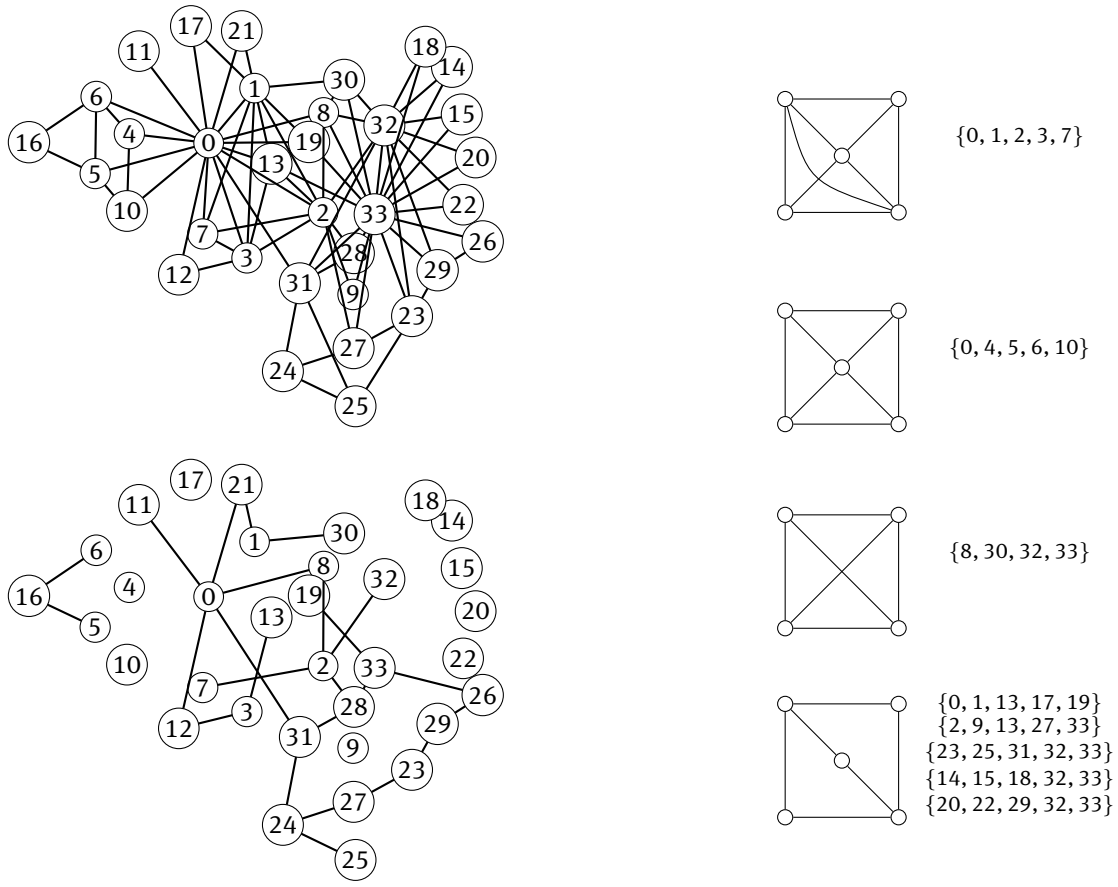


Figure 3: On the left, the original graph is shown on the top, and the residual graph is shown on the bottom. The graph was compressed using only the graphlets of sizes up to 5. The four graphlets that construct the final graph are shown on the right, with the corresponding vertex set. The last graphlet appears 5 times. The size of the residual graph is 25 edges, the size of the first graphlet and the symmetry is $3 + 2$, for the second graphlet is $2 + 3$, the third graphlet $2 + 2$, and the fourth graphlet $5(3 + 1)$. The cumulative size of the representation is 59 pairs, whereas the original graph has 78 edges.

such almost complete subgraphs, we first explore how dense such subgraph must be in order to still be $\mathcal{NSC}$.

Theorem 6. *A bipartite graph G' on two vertex sets U, V where $|U| = u, |V| = v$ is $\mathcal{NSC}$ if $|G'| > \frac{v+u-1+uv}{2}$.*

Proof. Assuming $|G'| > \frac{v+u-1+uv}{2}$
we get the inequality

$$v + (u - 1) + uv < 2|G'|$$

and putting one $|G'|$ on the other side of the inequality gives us

$$v + (u - 1) + uv - |G'| < |G'|.$$

And knowing that

$$|K(U, V)^{\pi}| = v, \quad |\pi| = u - 1,$$

and

$$|G' \oplus K(U, V)| = uv - |G'|,$$

gives us

$$|K(U, V)^{\pi}| + |\pi| + |G' \oplus K(U, V)| < |G'|,$$

which shows that $G' \in \mathcal{NSC}$.

□

The general idea of the algorithm is to repeatedly find bipartite subgraphs having the above property, remove them, and represent them with $(K(U, V)^{\pi}, \pi, G(U, V) \oplus K(U, V))$. To find such a bipartite subgraph, the algorithm searches only for specific bipartite graphs, obtained in the following way. For each vertex $v \in G$, a bipartite graph is extracted. Its set U is simply the set of neighbours of v, whereas the set V is the set of the neighbours of vertices in U (which are not already in U). The extracted graph is then greedily optimized, to obtain a bipartite graph with a better relative efficiency, which for such graph is defined as

$$\Delta^r(G, U, V) = \frac{2|G(U, V)| - |V| - |U| + 1 - |U||V|}{|G(U, V)|}.$$

The greedy optimization checks each vertex $v \in G(U, V)$ and computes Δ^r when v is present in the graph and when v (with all adjacent edges) is removed from the graph. If Δ^r is better in the latter case, v is removed from $G(U, v)$. This process is repeated until no further improvements can be made. Among all extracted subgraphs (for all vertices of G), the one with the largest relative efficiency is selected and removed from G. This process is repeated until the best relative efficiency becomes non-positive. Algorithm 2 shows this process in more detail.

Algorithm 2 Compressing the graph with dense bipartite graphs.

function BIPARTITECOMPRESS(G)
  $Comp = []$
  $B = \{\text{EXTRACTBIPART}(G, v) \mid v \in V(G)\}$
  $U, V = \arg\max_{(U,V)\in B} \Delta^r(G, U, V)$
  while $\Delta^r(G, U, V) > 0$ **do**
    $H = G(U, V)$
    $Comp = Comp + (H, \pi_{U,V}, K(U, V) \oplus H)$
    $G = G \setminus H$
    $B = \{\text{EXTRACTBIPART}(G, v) \mid v \in V(G)\}$
    $U, V = \arg\max_{(U,V)\in B} \Delta^r(G, U, V)$
  return $Comp + G$
function EXTRACTBIPART(G, v)
  $U = N_G(v)$
  $V = \bigcup_{u\in U} N_g(u) \setminus U$
  while $\Delta^r(G, U, V)$ is improving **do**
    for all $v \in U$ and $v \in V$ **do**
      if $\Delta^r(G, U, V) < \Delta^r(G, U\setminus\{v\}, V)$ or $< (\Delta^r(G, U, V\setminus\{v\}))$ **then**
        $U = U \setminus \{v\}$ or $V = V \setminus \{v\}$
  return (U, V)

Table 3: The empirical evaluation of the two presented algorithms. The second column gives the size of the graph, the third and fourth column give the sizes obtained with the graphlets and bipartite algorithms respectively, and the last column gives the best relative efficiency obtained for the given testset.

dataset	$\lvert G\rvert$	graphlets	bipartite	best Δ^r
karate	78	54	55	31 %
yeast	6650	4969	5372	25%
powergrid	6594	6046	6261	8%
jazz	2742	1246	1306	55%
lena	1339	678	705	49%
facebook	88234	44559	40457	54%

5 Empirical evaluation

To demonstrate the practical potential of the presented concepts and algorithms, we devised an empirical evaluation on a set of real graphs. Most of these graphs are well-known and used extensively in the various studies of network sciences. To also demonstrate the potential in image compression, we generated one example ourselves. The five well-known graphs are Zachary's karate club graph [19] (karate), yeast protein interaction network [4] (yeast), jazz musicians network [9] (jazz), US electrical power-grid network [18] (powergrid), and the Facebook ego network [11] (facebook). We generated one graph from a binary image of Lena (lena), which is one of the most well-known images in computer science. The transformation of this image to an undirected graph is rather trivial, we viewed the binary image as an adjacency matrix of a graph, but suitably shifted to obtain an undirected graph. All the graphs are undirected and were selected because they come from a variety of different real applications and are also structurally diverse.

The two algorithms were run on all instances, and Table 3 summarizes the obtained results. We compared the sizes of the uncompressed graph $|G|$, with the sizes of the obtained representations by the two algorithms (graphlets and bipartite). We can first see that the two algorithms have a comparable efficiency, but for practical applications, the bipartite compression is favorable, since its running time is significantly better. The obtained results divide the tests into two groups. The first three graphs (karate, yeast, powergrid) are less dense graphs, thus their reduced size is not that significant (31%, 25%, 8% respectively). The power grid is very close to a tree graph, and we showed that trees are not $\mathcal{SC}$, so this result is not sur-

Figure 4: Binary version of the famous Lena image.

prising. The second group of graphs (jazz, lena, facebook) shows a much better reduction in size (55%, 49%, 54% respectively). For an example of the compression, Figure 3 shows the karate club graph compressed with graphlets.

6 Conclusions

Symmetries in graphs give a compact representation of self-similarities in graphs, which can be used to reduce redundant information in the classical representations of this type of data. This article formalizes this intuition. We introduce a new class of graphs, which we call symmetry-compressible graphs. Since graphs arising in real applications do not exhibit (m)any symmetries, we defined an extended class, near symmetry-compressible graphs, which are much more suitable for use in practice.

Since this is a new concept, many open questions remain. From a theoretical point of view, new properties of $\mathcal{SC}$ graphs can be discovered, and especially a deeper correlation between $\mathcal{SC}$ and $\mathcal{NSC}$ graphs must be established. Furthermore, the computational complexity of determining whether a graph is in $\mathcal{SC}$ remains unanswered. This problem is related to the graph isomorphism problem, that is why we conjecture this problem to be *GI*-hard, but the exact hardness must be determined by correlating it to some of the many similar problems in computational group theory.

From a practical point of view, the usability of the presented concepts must be explored more thoroughly. The global nature of the graph compression has the potential of improving the existing data compression algorithms, either as a preprocessing step or as the central part of the compression. In our empirical evaluation we demonstrated how much a binary image can be compressed by using a graph representation, we believe that by combining these results with a more traditional image compression technique, better compression can be obtained.

References

[1] Sandra Álvarez, Nieves R Brisaboa, Susana Ladra, and Óscar Pedreira. A compact representation of graph databases. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 18–25, 2010.

[2] Vo Ngoc Anh and Alistair Moffat. Local modeling for webgraph compression. In *2010 Data Compression Conference*, pages 519–519. IEEE, 2010.

[3] Maciej Besta and Torsten Hoefler. Survey and taxonomy of lossless graph compression and space-efficient graph representations. *arXiv preprint arXiv:1806.01799*, 2018.

[4] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, et al. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research*, 31(9):2443–2450, 2003.

[5] Uroš Čibej and Jurij Mihelič. Improvements to Ullmann's algorithm for the subgraph isomorphism problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(07), 2015.

[6] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub) graph isomorphism algorithm for matching large graphs. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1367–1372, 2004.

[7] Olivier Cure, Hubert Naacke, Tendry Randriamalala, and Bernd Amann. Litemat: a scalable, cost-efficient inference encoding scheme for large rdf graphs. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1823–1830. IEEE, 2015.

[8] Arash Farzan and J Ian Munro. Succinct encoding of arbitrary graphs. *Theoretical Computer Science*, 513:38–52, 2013.

[9] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[10] Camille Jordan. Sur les assemblages de lignes. *Journal für die reine und angewandte Mathematik*, 1869(70):185–190, 1869.

[11] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[12] Faming Li, Zhaonian Zou, Jianzhong Li, and Yingshu Li. Graph compression with stars. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 449–461. Springer, 2019.

[13] Sebastian Maneth and Fabian Peternek. Grammar-based graph compression. *Information Systems*, 76:19–45, 2018.

[14] Francesco Pelosin. Graph compression using the regularity method. *arXiv preprint arXiv:1810.07275*, 2018.

[15] David Reinsel, John Gantz, and John Rydning. Data age 2025: the digitization of the world from edge to core. *Seagate, https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf*, 2018.

[16] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data*. O'Reilly Media, Inc., 2015.

[17] Ryan A Rossi and Rong Zhou. Graphzip: a clique-based sparse graph compression method. *Journal of Big Data*, 5(1):10, 2018.

[18] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442, 1998.

[19] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.