

The Space Complexity of Sum Labelling

Henning Fernau^{*1} and Kshitij Gajjar^{†2}

¹Universität Trier, FB 4 – Informatikwissenschaften, Trier, Germany

²National University of Singapore, 21 Lower Kent Ridge Rd, Singapore

January 6, 2022

Abstract

A graph is called a *sum graph* if its vertices can be labelled by distinct positive integers such that there is an edge between two vertices if and only if the sum of their labels is the label of another vertex of the graph. Most papers on sum graphs consider combinatorial questions like the minimum number of isolated vertices that need to be added to a given graph to make it a sum graph. In this paper, we initiate the study of sum graphs from the viewpoint of computational complexity. Notice that every n -vertex sum graph can be represented by a sorted list of n positive integers where edge queries can be answered in $\mathcal{O}(\log n)$ time. Therefore, limiting the size of the vertex labels upper-bounds the space complexity of storing the graph in the database.

We show that every n -vertex, m -edge, d -degenerate graph can be made a sum graph by adding at most m isolated vertices to it, such that the size of each vertex label is at most $\mathcal{O}(n^2d)$. This enables us to store the graph using $\mathcal{O}(m \log n)$ bits of memory. For sparse graphs (graphs with $\mathcal{O}(n)$ edges), this matches the trivial lower bound of $\Omega(n \log n)$. As planar graphs and forests have constant degeneracy, our result implies an upper bound of $\mathcal{O}(n^2)$ on their label size. The previously best known upper bound on the label size of general graphs with the minimum number of isolated vertices was $\mathcal{O}(4^n)$, due to Kratochvíl, Miller & Nguyen (2001). Furthermore, their proof was existential, whereas our labelling can be constructed in polynomial time.

^{*}fernau@uni-trier.de, ORCID: [0000-0002-4444-3220]

[†]kshitij@comp.nus.edu.sg, ORCID: [0000-0003-0890-199X]

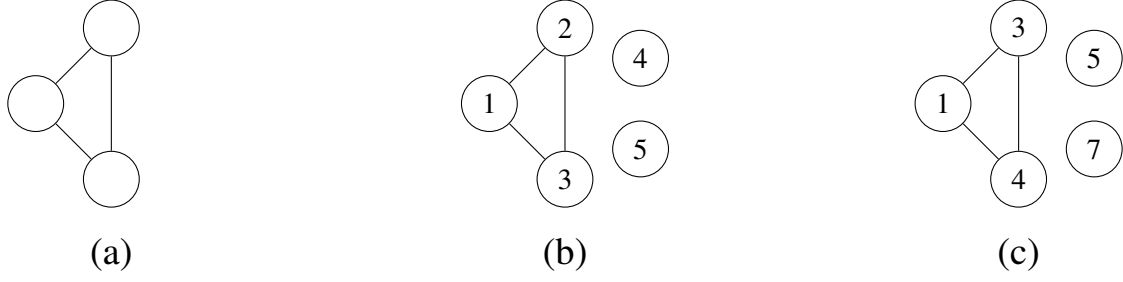


Figure 1: (a) This graph is not a sum graph, as it has no isolated vertices (**Fact 2**); (b) This is an incorrect sum labelling of a sum graph, because the vertices labelled 1 and 4 are not adjacent but there is a vertex labelled $1 + 4 = 5$ in the graph (**Definition 1**); (c) This is a correct sum labelling of a sum graph.

1 Introduction

There is a vast body of literature on graph labelling, testified by an ever-expanding survey on the topic maintained by Gallian [Gal20]. The 553-page survey (as of December 2020) mentions over 3000 papers on different ways of labelling graphs. In this paper, we focus on a type of labelling introduced by Harary [Har90] in 1990, called sum labelling.

Definition 1. A simple, undirected, unweighted graph G is called a sum graph if there exists an injective function $\lambda : V(G) \rightarrow \mathbb{N}$ such that for all vertices $v_1, v_2 \in V(G)$,

$$(v_1, v_2) \in E(G) \iff \exists v_3 \in V(G) \text{ s.t. } \lambda(v_1) + \lambda(v_2) = \lambda(v_3).$$

Then we say that λ is a sum labelling of (the vertices of) G . ◇

Notice that **Definition 1** implies that given only the function λ on the vertex set of a sum graph G , the edge set of G can be obtained. Thus, λ encodes the graph G . **Figure 1** illustrates a helpful example to better understand sum labellings. The following elementary fact about sum graphs is fundamental to almost all research done so far on sum graphs, including ours.

Fact 2. Every sum graph has at least one isolated vertex (a vertex of degree zero).

Proof. We will prove this fact by contradiction. Suppose there exists a sum graph G without an isolated vertex. Let L be the maximum label of a vertex in a sum labelling of G . As L is not the label of an isolated vertex, there is a vertex adjacent to it. Let the label of the adjacent vertex be x . Then there exists a vertex with label $L + x$ in G (**Definition 1**), contradicting the fact that L is the maximum label. □

Gould & Rödl [GR91] showed that every n -vertex graph can be made a sum graph by adding at most n^2 isolated vertices to it. In fact, certain graphs can be encoded much more succinctly with sum labelling than with the more traditional methods of storing a graph (e.g., adjacency matrix, incidence matrix, adjacency list). This makes sum labelling an intriguing concept not just to mathematicians but also to computer scientists. Sum labelling could also be of interest in graph databases [AG08, Ang12, KK15] and in collections of benchmark graphs [LFR08, DSUBGV⁺10, JV13]. However, no systematic study of this question has been undertaken so far. With this paper, we intend to start such a line of research, bringing sum labellings closer to the research in *labelling schemes* [KNR92]. To the best of our knowledge, the only known application of sum labelling before our work is in secret sharing schemes [SSM06].

The idea of using sum labelling to efficiently store graphs was already considered by Sutton [Sut00]. However, Sutton focused on the number of additional isolated vertices needed to store a given graph, whereas our focus is on the number of *bits* needed to store the graph.

In other words, while Sutton’s work attempts to minimize the number of additional vertices, it does not take into account the *size* of the vertex labels required to do so. This is crucial because it is known that there are several graph families for which the size of the vertex labels grows exponentially with the number of vertices. One popular example is the sum labelling scheme for trees presented by Ellingham [Ell93]. Another example is the more esoteric graph family known as the generalised friendship graph [FRS08].

Another parameter associated with sum graphs is the difference between the largest and smallest label, called *spum* (also called *range* in [KMN01]). Interestingly, while the concept of spum was around for quite some time (Gallian’s survey [Gal20] refers to an unpublished manuscript by a group of six students), the first publication that studies spum for various basic classes of graphs is a very recent one [STT21]. Unfortunately, this measure also does not reflect the whole truth about storing graphs, as it neglects the number of additional vertices that need to be stored. Moreover, spum is somewhat dependent on the definition of the sum number (see below for a formal definition), which might be slightly unnatural for the purpose of storing a graph.

In this paper, we also introduce a new graph parameter σ_{store} that takes into account both the number of additional vertices and their label size. We explain this formally in the next section.

2 Definitions and Main Result

Let us now fix some notation in order to formally introduce the concepts in this paper. All our graphs are undirected, unweighted and simple, specified as $G = (V, E)$, where V is the set of vertices and E is the set of edges. If a vertex v is an endpoint of an edge e , then we say that v and e are incident. The number of edges incident to a vertex is called its degree.

2.1 Sums and Spums

As isolated vertices (or simply, isolates) are usually irrelevant in most practical applications, λ (where λ is a sum labelling of a sum graph G) can be also viewed as a description of $G \setminus I$, where I is the set of isolates of G . Then, λ is called the *sum number encoding* of $G \setminus I$. Conversely, given a graph G without isolates, the minimum number of isolates needed to be added in order to turn G into a sum graph is called the *sum number* of G , written $\sigma(G)$, i.e., $G + \overline{K_{\sigma(G)}}$ is a sum graph. Here, $+$ denotes the disjoint union of graphs, \overline{H} denotes the complement of graph H , and K_n is the complete graph on n vertices. Thus, $\overline{K_n}$ is the empty (edgeless) graph on n vertices. The *spum* of G , written $\text{spum}(G)$, is defined as the minimum over all sum labellings of $G + \overline{K_{\sigma(G)}}$ of the difference between the maximum and minimum labels.

2.2 The Size of Sum Number Encodings of Graphs

A labelling function λ can be also seen as operating on edges by the summability condition. $\lambda(e)$ for an edge $e = xy \in E$ is defined as $\lambda(x) + \lambda(y)$. A labelling of a sum graph $G = (V, E)$ is called an *exclusive* sum labelling [MPR⁺05, MRR17, Rya09, TM03] if for every $e \in E$, we have $\lambda(e) = \lambda(i)$ for some isolate $i \in I \subseteq G$. Accordingly, $\varepsilon(G)$ denotes the *exclusive sum number* of G , which is the minimum

number of isolates to be added to G such that $G + \overline{K_{\varepsilon(G)}}$ is a sum graph that allows an exclusive sum labelling. Clearly, $\sigma(G) \leq \varepsilon(G)$.

Are substantial savings possible when considering sum number encodings of graphs? As most research in the area of sum labellings went into studying quite specific families of graphs, some partial answers are possible. For instance, analyzing the expositions in [Pya01, WL01], one sees that for the complete bipartite graph $K_{n,n+1}$, with n vertices in one partition and $n+1$ vertices in the other, $\sigma(K_{n,n+1}) = 2n - 1$. In other words, we need $4n$ numbers in order to represent $K_{n,n+1}$. Ignoring the size of these numbers, this is a clear advantage over any traditional way to store $K_{n,n+1}$, which would need $\mathcal{O}(n^2)$ bits when using adjacency matrices and even $\mathcal{O}(n^2 \log n)$ bits when using adjacency lists. However, after a closer look at the labelling presented in [Pya01], it becomes clear that the numbers needed to label a $K_{m,n}$ are of size $\mathcal{O}(nm)$. Therefore, storing the complete bipartite graph $K_{n,n+1}$ needs only $\mathcal{O}(n \log n)$ bits, using its sum graph encoding. As we will see later, this is in fact storage-optimal, in a certain sense.

Similarly, $\sigma(K_n) = 2n - 3$ is known for $n \geq 4$, i.e., $3n - 3$ numbers are necessary to store the information about the complete graph K_n , while again traditional methods would need $\mathcal{O}(n^2)$ bits at least. As mentioned in [Smy91], this can be obtained by labelling vertex x_i with $4i - 3$, with $1 \leq i \leq n$, leading to isolate labels $4j + 2$ for $1 \leq j \leq 2n - 3$. Hence, the sizes of the labels are in fact linear in n , which is, in a sense, even better than what is known for complete bipartite graphs. We will continue our discussions on storage issues in the next section. It is known that the sum number of general graphs grows with the order of its edges [NMS01]. In fact, this can happen even with sparse graphs [HS95, SM01].

As we have seen so far, neither the sum number of a graph nor the spum of a graph models the storage requirements of storing graphs with the help of sum numberings in a faithful manner. Therefore, we suggest another graph parameter, based on

$$\text{storage}(\lambda, G) = \sum_{v \in V} \lceil \log_2(\lambda(v)) \rceil \leq |V| \cdot \max_{v \in V} \lceil \log_2(\lambda(v)) \rceil \quad (3)$$

for a labelling $\lambda : V \rightarrow \mathbb{N}$ of a sum graph $G = (V, E)$. (Notice that one can store variable-size numbers using at most twice as many bits when compared to Equation 3 with Elias prefix codes [Eli75].) Now, define

$$\text{storage}(G) = \min\{\text{storage}(\lambda, G) \mid \exists \lambda : V \rightarrow \mathbb{N} : \lambda \text{ is a sum labelling of } G\}.$$

Then, for an arbitrary graph $G' = (V', E')$ one could define

$$\sigma_{\text{store}}(G') = \min\{\text{storage}(G) \mid \exists s \in \mathbb{N} : G = G' + \overline{K_s} \text{ is a sum graph}\}.$$

For instance, Ellingham's proof can be used to state: for an n -vertex tree T , Ellingham's construction leads to $\sigma_{\text{store}}(T) \in \mathcal{O}(n^2)$. This should be compared to any standard representation of trees that obviously needs $\mathcal{O}(n \log(n))$ space. However, our results prove that also with sum label representations, this upper bound can be obtained. In our construction, it is crucial that we also consider labellings that do not necessarily lead to a minimum sum number. This is also a difference concerning the definition of spum. As we are mostly interested in upper-bounding $\sigma_{\text{store}}(G')$ in this paper, we mainly discuss

$$\sigma_{\text{store}}^{\max}(G') = \min\{\text{storage}^{\max}(G) \mid \exists s \in \mathbb{N} : G = G' + \overline{K_s} \text{ is a sum graph}\},$$

where for a sum graph $G = (V, E)$,

$$\text{storage}^{\max}(G) = \min\{\text{storage}^{\max}(\lambda, G) \mid \exists \lambda : V \rightarrow \mathbb{N} : \lambda \text{ labels } G\},$$

with

$$\text{storage}^{\max}(\lambda, G) = |V| \cdot \max_{v \in V} \lceil \log_2(\lambda(v)) \rceil = |V| \cdot \lceil \log_2(\max \lambda(V)) \rceil.$$

By [Equation 3](#), $\sigma_{\text{store}}(G') \leq \sigma_{\text{store}}^{\max}(G')$. A reader who likes to get more familiar with these notions is invited to first go through the next section.

However, let us first state the main result of this paper, as we are now ready for it.

Theorem 4 (Main Result). *Let G' be a graph on n vertices and m edges with minimum degree at least one. Then, $\sigma_{\text{store}}^{\max}(G') \in \mathcal{O}(m \cdot \log(n))$. More specifically,*

$$\sigma_{\text{store}}^{\max}(G') \leq 9m(\log_2(n) + 1)$$

for general graphs and

$$\sigma_{\text{store}}^{\max}(G') \leq 3m(2\log_2(n) + \log_2(12d)) < 3dn(2\log_2(n) + \log_2(12d))$$

for d -degenerate graphs. Furthermore, the sum labelling can be computed in polynomial time.

In particular, this means that $\mathcal{O}(n \log(n))$ bits are sufficient to store trees with sum labellings, as they are 1-degenerate graphs. A similar result holds for planar graphs, as they are 5-degenerate. We show that these bounds are optimal for storing graphs, up to constant factors. We also relate to the literature on adjacency labelling schemes (see, e.g., [KNR92, Pel00], or more recently, [BGP20, DEG⁺20]).

To give a flavour of our algorithm, notice that it also works in the streaming or online setting, in which vertices are being given one-by-one by an adversary.

3 Sum Labelling a Disjoint Collection of Edges

This section should be treated as an introductory exercise on sum labelling, and has no bearing on our main result. A reader familiar with sum labelling schemes may skip to the next section, possibly apart from the very last lines of this section.

It is known that trees have sum number 1; according to a remark following Theorem 5.1 in [Eli93], this result translates to forests. However, the label sizes may grow exponentially in these constructions. As a warm-up and to explain the difficulties encountered while designing sum labellings, we present some constructions that label a disjoint collection of edges, or more mathematically speaking, a 1-regular graph, which we denote by M_n (a matching on n vertices, where n is an even number).

3.1 Exponential Solution ([Figure 2 \(a\)](#))

If we have n vertices (hence $n/2$ edges), we label the first edge $(2, 3)$, the second one starts with the sum of the labels of the previous edge followed by its successor, i.e., $(5, 6)$. Then we add up the previous two labels, continue with the successor, and so on. This can be brought into the following sum labelling

scheme for 1-regular graphs.

$$\lambda(n) = \begin{cases} 2 & \text{if } n = 1 \\ \lambda(n-1) + 1 & \text{if } n \text{ is even} \\ \lambda(n-2) + \lambda(n-1) & \text{if } n \text{ is odd and } n > 1 \end{cases} \quad (5)$$

Lemma 6. *For the labelling defined in Equation 5, we have $\lambda(n) \in \Theta(\sqrt{2}^n)$.*

Proof. The Online Encyclopedia of Integer Sequences suggests that this is another variation on Ulam numbers [Slo73] if we think of the starting point to be $\lambda(0) = 1$. Then, $\lambda(n)$ (for $n > 1$) can be seen as the smallest (when n is even) or largest (when n is odd) number larger than $\lambda(n-1)$ that is a unique sum of two distinct earlier terms of the sequence. This connection suggests the following closed form:

$$\lambda(n) = \begin{cases} 3 \cdot 2^{k-1} & \text{if } n \text{ is even, i.e., } n = 2k \\ 3 \cdot 2^k - 1 & \text{if } n \text{ is odd, i.e., } n = 2k + 1 \end{cases}$$

In other words, we have $\lambda(n) \in \Theta(\sqrt{2}^n)$, implying that λ increases exponentially with n . □

Using this lemma, we can also conclude that $\text{storage}(\lambda, M_n) \in \Theta(n^2)$ for this labelling λ .

3.2 Linear Solution (Figure 2 (b))

Consider the following sum labelling scheme for 1-regular graphs M_n on n vertices. (We group endpoint labels of each edge together by parentheses.)

$$(n, 2n-1), (n+1, 2n-2), \dots, \left(\frac{3n}{2}-1, \frac{3n}{2}\right). \quad (7)$$

All edge labels sum up to $3n-1$, which is the label of the isolated vertex. Also, it easy to see that these edges are the only ways in which two of the given n numbers can sum to $3n-1$. Finally, even the sum of the two smallest labels between non-adjacent vertices (i.e., $n + (n+1) = 2n+1$) is larger than the label of any other non-isolated vertex in the graph, proving that this is a valid sum labelling. As each label is in $\Theta(n)$, the overall space requirement of this labelling scheme is $\Theta(n \log(n))$. Moreover, as we can also see with the first labelling scheme, $\sigma(M_n) = 1$. Also, in contrast to the first scheme, this labelling scheme is exclusive. Hence, this approach also shows that $\varepsilon(M_n) = 1$. Finally, as the labels only grow linearly with n with this labelling λ , we can also conclude that $\text{storage}(\lambda, M_n) \in \Theta(n \log(n))$.

3.3 Disjoint Union of Several Identical Components

The previous consideration was quite special to 1-regular graphs. We now develop an argument that can be generalised towards a certain type of graph operation. One can think of M_n as being the disjoint graph union of $n/2$ times M_2 . For simplicity of the exposition, assume $n/2 = 2^d$ in the following. Label the vertices $(v_{1,1}, v_{2,1}), (v_{1,2}, v_{2,2}), \dots, (v_{1,2^d}, v_{2,2^d})$ of M_n as follows, for $j = 1, \dots, 2^d$:

$$\begin{aligned} \lambda(v_{1,j}) &= 1 + 8 \cdot (j-1) + 2^{4+d} \cdot (2^d - j) \\ \lambda(v_{2,j}) &= 2 + 8 \cdot (2^d - j) + 2^{5+d} \cdot (j-1) \end{aligned}$$

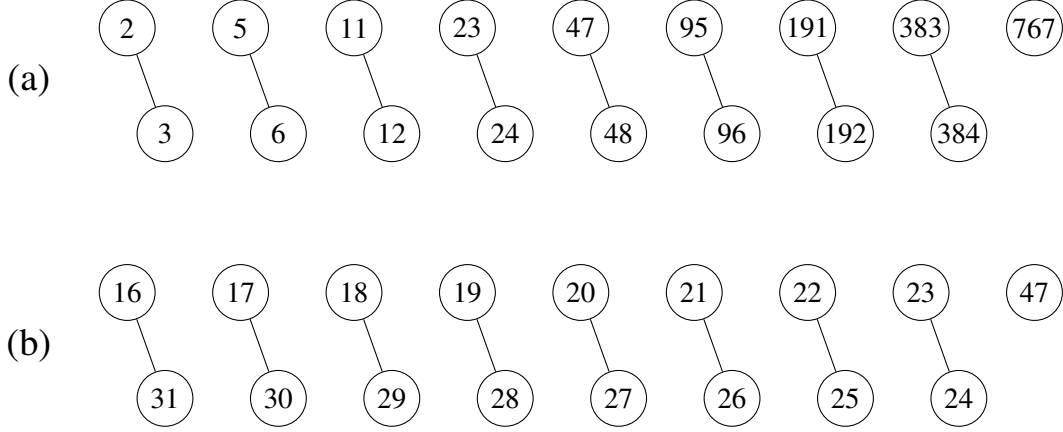


Figure 2: (a) Labelling M_{16} using Equation 5; (b) Labelling M_{16} using Equation 7.

For instance, for $d = 2$, we get $\lambda(v_{1,1}) = 1 + 8 \cdot 0 + 64 \cdot 3$, $\lambda(v_{2,1}) = 2 + 8 \cdot 3 + 64 \cdot 0$, so that the connecting edge is testified by the isolate label $3 + 8 \cdot 3 + 64 \cdot 3 = 219 = (11011011)_2$. Also, $\lambda(v_{1,2}) = 1 + 8 \cdot 1 + 64 \cdot 2$, $\lambda(v_{2,2}) = 2 + 8 \cdot 2 + 64 \cdot 1$, adding up again to 219. Likewise, $\lambda(v_{1,3}) = 1 + 8 \cdot 2 + 64 \cdot 1$, $\lambda(v_{2,3}) = 2 + 8 \cdot 1 + 64 \cdot 2$, and finally $\lambda(v_{1,4}) = 1 + 8 \cdot 3 + 64 \cdot 0$ and $\lambda(v_{2,4}) = 2 + 8 \cdot 0 + 64 \cdot 3$. By construction, all numbers need at most $2d + 4$ bits for labelling 2^{d+1} vertices. Hence, the overall space requirement for storing M_n is again $\mathcal{O}(n \log(n))$ bits.

The zero bit introduced in the third and sixth binary position in the example ensures that the labels of two non-adjacent vertices cannot add up to the label of another vertex. This technique can be easily generalised to obtain the following result.

Lemma 8. *Let G be a graph. Then, the n -fold disjoint graph union G_n of G with itself obeys $\sigma_{\text{store}}(G_n) \in \mathcal{O}(n \log(n))$. Moreover, $\sigma(G_n) \leq \sigma(G)$.*

4 Storing Graphs using Sum Labelling

Alternative Notions. One of our motivations to return to sum labellings was the idea that one can use them to store graphs space-efficiently. This idea was already expressed in [KMN01]. There, they consider the notion of the *range* of a sum graph G that is realizing $\sigma(G')$, which happens to coincide with the notion called spum later. But following this motivation (to store graphs), let us define the *range* of a labelling λ of a sum graph $G = (V, E)$ as the difference between $\max \lambda(V)$ and $\min \lambda(V)$. The idea behind is that it would suffice to store the numbers $\lambda(v) - \min \lambda(V)$ for all vertices $v \in V$, plus the value of $\min \lambda(V)$ once, instead of storing all values $\lambda(v)$, which could help us save some bits.

The following lemma tells us that this variation in our considerations (which could also lead to variations of our definition of σ_{store} and related notions) is not essential for our current considerations, as we mostly neglect constant factors. In particular, we might consider $|V| \cdot \lceil \log_2(\max \lambda(V) - \min \lambda(V)) \rceil + \lceil \log_2(\min \lambda(V)) \rceil$ as a more appropriate definition of the maximum estimate of the storage requirements of a sum graph $G = (V, E)$ with respect to a sum labelling λ .

Lemma 9. *Let λ be a sum labelling of a non-empty sum graph $G = (V, E)$, and let $\text{range}(\lambda(V)) =$*

$\max \lambda(V) - \min \lambda(V)$. Then,

$$\begin{aligned} \text{range}(\lambda(V)) &> \min \lambda(V); \\ 2 \cdot \text{range}(\lambda(V)) &> \max \lambda(V). \end{aligned}$$

Thus, $\max \lambda(V) \in \Theta(\text{range}(\lambda(V)))$.

Proof. Let $x \in V$ be the vertex carrying the smallest label $\min \lambda(V)$. As x is not an isolate, there must be an edge incident to x that connects to a vertex y such that $\lambda(y) > \lambda(x)$. Hence, there must be a vertex z in V (possibly, an isolate) that carries a label $\lambda(z) = \lambda(x) + \lambda(y) > 2\lambda(x)$. Now, $\max \lambda(V) - \min \lambda(V) \geq \lambda(z) - \lambda(x) > \lambda(x)$. Moreover, $2 \cdot (\max \lambda(V) - \min \lambda(V)) = (\max \lambda(V) - \min \lambda(V)) + (\max \lambda(V) - \min \lambda(V)) > (\max \lambda(V) - \min \lambda(V)) + \min \lambda(V) = \max \lambda(V)$. \square

What is the main purpose of a graph database? Clearly, one has to access the graphs. A basic operation would be to answer the query if there is an edge between two vertices. Now, if $\max \lambda(V)$ of a sum graph is polynomial in the number $n = |V|$ of its vertices, we can answer this query in time $\mathcal{O}(\log(n))$, a property also discussed as *adjacency labelling scheme* by Peleg [Pel00].

Namely, assuming the polynomial bound on the size of the labels, we would need time $\mathcal{O}(\log(n))$ to add the two labels of the vertices, and we also need time $\mathcal{O}(\log(n))$ to search for the sum in the ordered list of numbers, using binary search, because there are only $\mathcal{O}(n^2)$ many numbers needed to describe a graph. If $\max \lambda(V)$ would be super-polynomial, then the additional time $\mathcal{O}(\log(\max \lambda(V)))$ would be quite expensive, which probably makes the idea of storing large graphs as sum graphs in databases unattractive. This motivates in particular also considering $\max \lambda(V)$ of the labelling λ of a sum graph.

We discuss further graph storing schemes that may be thought of efficient in terms of their memory requirements in [Appendix A](#).

Lower Bounds. How many bits are really necessary to store graphs? We will discuss lower and upper bounds in the following, starting with a lower bound.

Lemma 10. *Let G be an n -vertex graph. Then $\sigma_{\text{store}}^{\max}(G) \in \Omega(n \log n)$, and $\sigma_{\text{store}}(G) \in \Omega(n \log n)$.*

Proof. When it comes to storage costs, the most parsimonious labelling $\lambda : V \rightarrow \mathbb{N}$ obeys $\lambda(V) = [n] = \{1, 2, \dots, n\}$ by injectivity. Now,

$$\sum_{v \in V} \lceil \log_2(\lambda(v)) \rceil \geq \sum_{v \in V} \log_2(\lambda(v)) = \sum_{i \in [n]} \log_2(i) = \log_2 \left(\prod_{i \in [n]} i \right) = \log_2(n!).$$

By Stirling's formula [Dut91], there are constants c, d such that

$$\log_2(n!) \geq \log_2(d \cdot (n/c)^n) = (n/c) \log_2(n) + \log_2(d) \in \Omega(n \log n).$$

As $\sum_{v \in V} \lceil \log_2(\lambda(v)) \rceil \leq |V| \max_{v \in V} \lceil \log_2(\lambda(v)) \rceil$, both lower bound claims are true. \square

This lemma shows that a sum labelling with $\mathcal{O}(n \log n)$ bits is storage-optimal, up to constants. This is one of the motivations underlying the discussions in the next section. Moreover, $\Omega(n \log n)$ is the space requirement for storing sparse graphs using traditional graph-storage methods. $\Omega(n \log n)$ bits are needed just to write the names of the vertices, as can be seen by a calculation similar to the proof of [Lemma 10](#).

Upper Bounds. Here, we start our discussion on upper bounds for storing graphs with sum labellings. First, we briefly discuss the number of isolates in this respect. Based on some probabilistic arguments, it is known that the number of isolates is about the number of edges of the graph to be encoded [GR91, NMS01] for nearly all graphs.

Remark 11. *As there are $2^{\Theta(n^2)}$ many graphs on n vertices, we cannot hope for a sum labelling scheme that uses only $n^{2-\varepsilon}$ many isolates and only polynomial-size labels and hence a polynomial range, because we need at least $\Omega(n^2)$ many bits just to write down n -vertex graphs. As an aside, allowing for n^2 many isolates also means always allowing exclusive labellings.* \diamond

Conversely, assuming we can sum-label each n -vertex, m -edge graph with polynomial-sized labels, then we can upper-bound σ_{store} by $\mathcal{O}(m \log(n))$. By our discussions from Lemma 10 and Remark 11, we cannot hope for anything substantially better. Can we reach this bound? Unfortunately, this seems to be an open question that we will answer to some extent below in our main result. In [KMN01], it was shown that each n -vertex graph without isolates can be represented by a sum labelling that uses numbers no larger than 4^n . In other words, one would need at most $2n$ bits to represent each vertex of an n -vertex graph. This also shows that sum graphs have a *constrained 1-labelling scheme* as defined in [KNR92]. Hitherto, it was unknown how to sum-label arbitrary graphs with polynomial-size labels. As our main result, we solve this problem affirmatively, with nice consequences for d -degenerate graphs.

5 A Novel Algorithm for Sum Labelling

We will now prove our main result (Theorem 4), thereby showing that sum labellings can be used to store graphs as efficiently as traditional methods can do. It is easy to see that the two major theorems shown in this section (Theorem 12, Theorem 24) imply Theorem 4.¹

Theorem 12. *Every n -vertex, m -edge graph G of minimum degree at least one can be made a sum graph H by adding at most m isolates to G , such that H admits a sum labelling λ satisfying*

$$\lambda(v) \leq 4 \cdot n^3 \quad \forall v \in V(G); \quad (13)$$

$$\lambda(v) \leq 8 \cdot n^3 \quad \forall v \in V(H). \quad (14)$$

Furthermore, the labelling is an exclusive sum labelling, computable in polynomial time by Algorithm 1.

Our proof of Theorem 12 is constructive and algorithmic in nature, described formally by Algorithm 1. The proof itself explains in words the working of this algorithm, its correctness, and provides upper bounds on the sizes of the vertex labels.

The Algorithm. Algorithm 1 takes a non-empty graph G as input, and outputs a sum graph H and a labelling λ of H such that $H = G + \overline{K_c}$ (for some $c \leq m$) is a sum graph with sum labelling λ . Algorithm 1 uses Algorithm 2 as a subroutine. Algorithm 2 takes a graph H and a labelling λ of H as input, and outputs TRUE if λ is a sum labelling of H and FALSE otherwise.²

¹For the sake of simplicity, in our proofs we assume that the given graphs have no isolates. It is easy to see that the same bounds hold (up to constant additive terms) even when the given graphs have isolates.

²For technical reasons, Algorithm 2 allows different vertices to have the same label as long as they are isolates. This is not allowed in sum labelling; in the end, Algorithm 1 fixes this by eliminating isolates with duplicate labels.

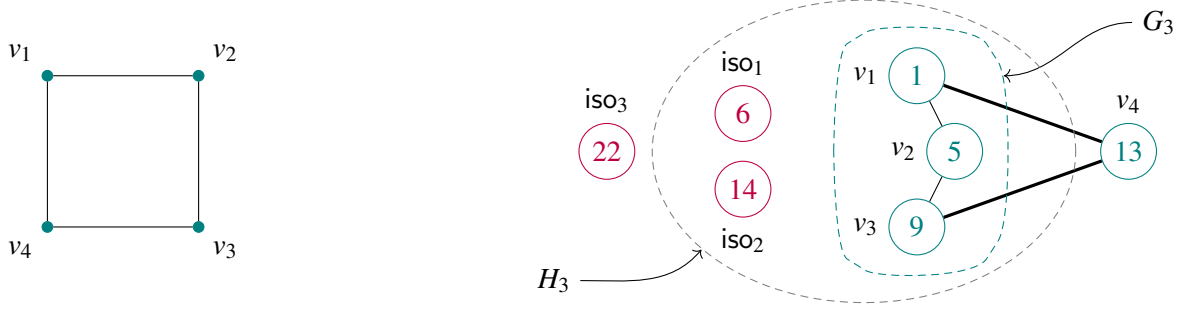


Figure 3: (Left) The graph G with the vertex ordering $\{v_1, v_2, v_3, v_4\}$ is provided as input to [Algorithm 1](#). (Right) G_3 is the induced subgraph of G on the vertex set $\{v_1, v_2, v_3\}$ and H_3 is its corresponding sum graph, constructed by the first three iterations of the algorithm, along with its labelling. At the fourth iteration, v_4 is added to G_3 to obtain G_4 , and a new isolate iso_3 is also added to H_3 to obtain H_4 . Notice that the vertices of G_4 are labelled $1 \bmod 4$ and the isolates are labelled $2 \bmod 4$; this labelling scheme is described in the proof of [Theorem 12](#). This specific example is also explained in more detail in [Example 22](#).

Proof of Theorem 12. It is helpful to follow [Figure 3](#) while reading this proof. Notice that [Equation 13](#) implies [Equation 14](#), as isolate labels are sums of labels of $V(G)$. So we will focus on showing [Equation 13](#) in this proof. Let the vertices of G be $\{v_1, v_2, \dots, v_n\}$. Let G_i be the induced subgraph on the first i vertices of G , that is,

$$V(G_i) = \{v_1, v_2, \dots, v_i\}.$$

For each G_i ($2 \leq i \leq n$), we will show that there is a sum graph H_i which can be obtained by adding $r_i \leq \binom{i}{2}$ isolates to G_i (since G_i has at most $\binom{i}{2}$ edges), satisfying $\lambda(v) \leq 4 \cdot i^3$ for each $v \in V(G_i)$. Moreover, all vertices of G_i will carry labels that equal 1 modulo 4, and all isolates in H_i will carry labels that equal 2 modulo 4. This modulo condition ensures that our labelling is exclusive. Our proof is by induction on i , yielding an algorithm explicitly described by [Algorithm 1](#).

Although the statement of the theorem makes sense only from $n \geq 2$ onward to meet the minimum-degree requirement, it is convenient for our inductive proof to start with $i = 1$:

Base case ($i = 1$): We set $\lambda(v_1) = 1$. Notice that $\lambda(v_1) = 1^3$. Set $r_1 = 0$.

Induction hypothesis: There is a sum graph H_i for G_i such that H_i has r_i isolates (in other words, $H_i = G_i \cup \{\text{iso}_1, \text{iso}_2, \dots, \text{iso}_{r_i}\}$), where $r_i \leq \binom{i}{2}$, and $\lambda(v) \leq 4 \cdot i^3$ for each $v \in V(G_i)$. Moreover, all vertices of G_i carry labels that equal 1 modulo 4, and all isolates in H_i carry labels that equal 2 modulo 4.

Induction step: We add the vertex v_{i+1} to the graph H_i and connect it to its neighbours in G_i . Suppose v_{i+1} has t_i neighbours $\{v_{j_1}, v_{j_2}, \dots, v_{j_{t_i}}\}$ in G_i . Then add t_i isolates $\{\text{iso}_{r_i+1}, \text{iso}_{r_i+2}, \dots, \text{iso}_{r_i+t_i}\}$ to H_i , giving the graph H_{i+1} . Thus,

$$H_{i+1} = G_{i+1} \cup \{\text{iso}_1, \text{iso}_2, \dots, \text{iso}_{r_i+t_i}\}.$$

We define $r_{i+1} = r_i + t_i$. Next, we set the labels of the newly added vertices. If λ is not a valid sum labelling for H_{i+1} , then we will change the λ -values of the newly added vertices. We will show that their

λ -values need to be changed less than i^3 times until we reach a valid sum labelling for H_{i+1} .

$$\lambda(v_{i+1}) = 5; \quad (15)$$

$$\lambda(\text{iso}_{r_i+k}) = \lambda(v_{i+1}) + \lambda(v_{j_k}) \quad \forall k \in \{1, 2, \dots, t_i\}. \quad (16)$$

Claim 17. λ is a valid sum labelling of H_{i+1} if and only if it has none of the following violations.

- (i) A violating pair: an ordered set of two vertices (u, w) from G_i such that $\lambda(u) = \lambda(w)$.
- (ii) A violating triple: an ordered set of three vertices (u, w, y) such that $\lambda(u) < \lambda(w) < \lambda(y)$ and $\lambda(u) + \lambda(w) = \lambda(y)$ and $(u, w) \notin E(H_{i+1})$.

Remark 18. Notice that it could happen that some of the ‘new’ isolates in H_{i+1} carry labels that are already labels of isolates from H_i . In that case, we implicitly delete the extra isolates (t_i is decreased accordingly), automatically avoiding violating pairs among them. Also, the modulo 4 arithmetics prevent vertices of G_i from pairing up with the isolates to form a violating pair. \diamond

Proof of Claim 17. It is easy to see that if H_{i+1} has any of the above violations, then λ is not a valid sum labelling of H_{i+1} . Now we will prove the other direction: if λ is not a valid sum labelling of H_{i+1} , then it either has a violating pair or a violating triple.

Notice that H_{i+1} has $i + r_i + t_i + 1 = (i + 1) + r_{i+1}$ many vertices, each with its corresponding λ -value. If two of the vertices have the same λ -value, then it is a type (i) violation, and we are done. So, we assume that all the λ -values are distinct. Given these $(i + 1) + r_{i+1}$ distinct numbers, we construct their corresponding sum graph H'_{i+1} on $(i + 1) + r_{i+1}$ vertices using the sum labelling property.

Both H_{i+1} and H'_{i+1} have the same set of vertices and the same labelling scheme λ . However, since λ is a valid labelling scheme for H'_{i+1} but not for H_{i+1} , they cannot have the same set of edges. Furthermore, H_{i+1} is a subgraph of H'_{i+1} . This is because every edge $e = (u, w)$ of H_{i+1} is either an edge that was also present in H_i (in which case there is a vertex labelled $u + w$ in H_{i+1} and H'_{i+1} , since H_i is a sum graph by the induction hypothesis), or it is one of the t_i new edges added (in which case one of the t_i new isolates $\{\text{iso}_{r_i+1}, \text{iso}_{r_i+2}, \dots, \text{iso}_{r_{i+1}}\}$ is labelled $u + w$ by Equation 16).

Due to Remark 18, the only way for the edge sets of H_{i+1} and H'_{i+1} to differ is if there is an edge $e = (u, w)$ such that $e \in E(H'_{i+1})$ and $e \notin E(H_{i+1})$. This means there are three vertices (u, w, y) in H'_{i+1} (and so also in H_{i+1}) such that $\lambda(u) + \lambda(w) = \lambda(y)$, a type (ii) violation. \square

Now, if H_{i+1} is a sum graph with the labelling scheme derived from Equation 15 and Equation 16, then we are done. Otherwise, we (slightly) modify these labels to obtain a new labelling, as follows.

$$\lambda(v_{i+1}) \leftarrow \lambda(v_{i+1}) + 4; \quad (19)$$

$$\lambda(\text{iso}_{r_i+k}) \leftarrow \lambda(\text{iso}_{r_i+k}) + 4. \quad (20)$$

We again check if with these new labels, H_{i+1} is a sum graph. If not, we increment these values by 4 again. We keep doing this until H_{i+1} becomes a sum graph. The crucial point to note is that each time we increment by 4, at least one of the violations disappears, never to occur again.

To fully understand this last sentence, we need to refine our analysis of potential conflicts that might occur when running our algorithm. Namely, following up on the proof of the previous lemma, consider three vertices $\{u, w, y\}$ in H_{i+1} such that (incorrectly) $\lambda(u) + \lambda(w) = \lambda(y)$ in the labelling λ of H_{i+1} .

First observe that not all vertices from $\{u, w, y\}$ can be isolates, as the isolates carry labels that are 2 modulo 4.

As we know that λ , restricted to the vertices of H_i , turns H_i into a sum graph, not all of the vertices $\{u, w, y\}$ belong to H_i . If y is one of the isolates of H_i , then its labelling will not change when updating λ according to [Equation 20](#). As one of the vertices u, w does not belong to H_i , we have, w.l.o.g., $u \in V(H_i)$ and $w = v_{i+1}$, because if w would be among the isolates, the sum of the labels of u and w would equal 0 modulo 4, but all isolates carry labels that are 2 modulo 4. This means that out of the three labels of u, w, y , exactly one will change according to [Equation 19](#) and as it will also be the only one that might increase in further modifications, a violation will never re-appear in the triple (u, w, y) .

Assume now that y is one of the new isolates, say, $y = \text{iso}_{r_{i+1}}$. If exactly one of the two other vertices, say, u , already belongs to H_i , then the other one, w , must be v_{i+1} . As $\lambda(u) + \lambda(w) = \lambda(y) = \lambda(\text{iso}_{r_{i+1}})$, we must have $u = v_{j_1}$, as we have no violating pairs. However, this means that the edge (u, w) belongs to both H_{i+1} and to H'_{i+1} , contradicting our assumption. Therefore, if y is one of the new isolates, then both u and w must belong to H_i . This means that the labellings of u and of w will never change by the re-labellings described in [Equation 19](#) and [Equation 20](#), while the labelling of y will only (further) increase, so that indeed a violation will never re-appear in the triple (u, w, y) .

How often might we have to update a labelling when moving from H_i to a valid sum graph H_{i+1} ? Our previous analysis shows that the following are the only two scenarios that could possibly be encountered for a violating triple (u, w, y) :

1. y is an isolate of H_i and exactly one of $\{u, w\}$ belongs to $V(G_i)$, while the other is v_{i+1} . There are at most $i \cdot r_i$ many cases when this might occur.
2. y is an isolate of H_{i+1} and $\{u, w\} \subseteq V(H_i)$. There are at most $t_i \cdot \binom{i}{2} = t_i \cdot i(i-1)/2$ many cases when this might occur.

Recall that r_i isolates are contained in the sum graph H_i and $t_i = r_{i+1} - r_i$ isolates are newly added to yield H_{i+1} . Our analysis shows that after at most $s_i = i \cdot r_i + t_i \cdot i(i-1)/2$ many steps, a valid sum labelling of H_{i+1} was found. By observing that r_i cannot be larger than the number $\binom{i}{2} = i(i-1)/2$ of hypothetical edges in H_i , and t_i is upper-bounded by the number i of vertices in H_i , we can furthermore estimate:

$$s_i \leq i \cdot i(i-1)/2 + i \cdot i(i-1)/2 = i^3 - i^2.$$

By induction hypothesis, we know that for each of the i vertices v in H_i , we have $\lambda(v) \leq i^3$. As H_i contains only i vertices that are labelled with number that are equal to 1 modulo 4, within at most $i^3 - i^2$ increment steps, we will find a label for v_{i+1} that is no larger than $4 \cdot (i^3 - i^2) + 1 \leq (\sqrt[3]{4} (i+1))^3$, basically using the pigeonhole principle. As all labels of isolates are sums of labels of vertices from G_i , their sizes are upper-bounded by $4i^3 + 4(i-1)^3 < 8 \cdot i^3$. \square

This gives an upper bound of $(n+m)(\log(8n^3))$ on the total number of bits required to store H . Since every vertex in G has degree at least one, we have $n \leq 2m$. Substituting, we get an upper bound of $3m(\log(8n^3)) \leq 3m(3\log n + 3) = 9m(\log n + 1)$, as required by [Theorem 4](#).

Some Concrete Examples. We now look at how [Algorithm 1](#) performs on some small graphs.

Example 21. Let $\{v_1, v_2, v_3, v_4\}$ be the vertices of K_4 . We label $\lambda(v_1) = 1$, $\lambda(v_2) = 5$ and introduce the isolate iso_1 with $\lambda(\text{iso}_1) = 6$. Then, we label $\lambda(v_3) = 9$, and introduce the isolates $\text{iso}_2, \text{iso}_3$ with $\lambda(\text{iso}_2) = 10$, $\lambda(\text{iso}_3) = 14$. Next, we label $\lambda(v_4) = 13$. In principle, we would now introduce three

Algorithm 1 SUMLABEL(G)

```
1:  $V(H) \leftarrow v_1$  ▷ Initialising  $H$ 
2:  $\lambda(v_1) \leftarrow 1$ 
3:  $E(H) \leftarrow \emptyset$ 
4:  $c \leftarrow 0$  ▷ Counter for the number of isolates
5:  $i \leftarrow 2$  ▷ The current vertex being processed
6: while  $i \leq n$  do ▷ The ordering  $V(G) = \{v_1, v_2, \dots, v_n\}$  is part of the input
7:    $V(H) \leftarrow V(H) \cup \{v_i\}$ 
8:    $\lambda(v_i) \leftarrow 5$  ▷ See Equation 15
9:    $c_{\text{before}} \leftarrow c$ 
10:  for each  $j$  such that  $1 \leq j \leq i-1$  and  $v_i v_j \in E(G)$  do
11:     $V(H) \leftarrow V(H) \cup \{\text{iso}_{c+1}\}$ 
12:     $E(H) \leftarrow E(H) \cup \{v_i v_j\}$ 
13:     $\lambda(\text{iso}_{c+1}) \leftarrow \lambda(v_i) + \lambda(v_j)$  ▷ See Equation 16
14:     $c \leftarrow c + 1$ 
15:  end for
16:   $c_{\text{after}} \leftarrow c$ 
17:  while CHECKVALIDSUMGRAPH( $H, \lambda$ ) = FALSE do
18:     $\lambda(v_i) \leftarrow \lambda(v_i) + 4$  ▷ See Equation 19
19:    for each  $\ell$  such that  $1 + c_{\text{before}} \leq \ell \leq c_{\text{after}}$  do
20:       $\lambda(\text{iso}_\ell) \leftarrow \lambda(\text{iso}_\ell) + 4$  ▷ See Equation 20
21:    end for
22:  end while
23: end while
24: for each  $(\text{iso}_i, \text{iso}_j) \in V(H) \times V(H)$  such that  $i < j$  do
25:   if  $\lambda(\text{iso}_i) = \lambda(\text{iso}_j)$  then
26:      $V(H) \leftarrow V(H) \setminus \{\text{iso}_j\}$  ▷ Remove isolates with duplicate labels
27:   end if
28: end for
29: return  $(H, \lambda)$ 
```

Algorithm 2 CHECKVALIDSUMGRAPH(H, λ)

```
1:  $f \leftarrow \text{TRUE}$  ▷  $f = \text{TRUE} \Leftrightarrow H$  is a sum graph with sum labelling  $\lambda$ 
2: for each  $(v_1, v_2) \in V(H) \times V(H)$  such that  $v_1 \neq v_2$  do
3:    $s \leftarrow \text{FALSE}$  ▷  $s = \text{TRUE} \Leftrightarrow v_1 v_2$  is an edge as per the sum labelling  $\lambda$ 
4:   if  $(\deg(v_1) \neq 0 \vee \deg(v_2) \neq 0) \wedge (\lambda(v_1) = \lambda(v_2))$  then
5:      $f \leftarrow \text{FALSE}$  ▷ Two vertices cannot have the same label, unless they are both isolates
6:   end if
7:   for each  $v_3 \in V(H)$  do
8:     if  $\lambda(v_3) = \lambda(v_1) + \lambda(v_2)$  then
9:        $s \leftarrow \text{TRUE}$  ▷ The sum labelling says that  $v_1 v_2$  is an edge
10:    end if
11:  end for
12:  if  $(v_1 v_2 \in E(H) \wedge s = \text{FALSE}) \vee (v_1 v_2 \notin E(H) \wedge s = \text{TRUE})$  then
13:     $f \leftarrow \text{FALSE}$  ▷ The sum labelling  $\lambda$  does not concur with the graph  $H$ 
14:  end if
15: end for
16: return  $f$ 
```

isolates with labels $13 + 1$, $13 + 5$, $13 + 9$. But, as the label 14 is already present for iso_3 , we need only two new isolates with labels 18, 22. In this way, our labelling scheme even finds the optimal sum labelling for K_n in general. Incidentally, this labelling scheme also gives a labelling of minimum spum (range). \diamond

Example 22. For labelling a C_4 whose vertices are (v_1, v_2, v_3, v_4) in cyclic order (see [Figure 3](#)), the first two steps are the same (i.e., $\lambda(v_1) = 1, \lambda(v_2) = 5, \lambda(\text{iso}_1) = 6$) as in [Example 21](#), but after setting $\lambda(v_3) = 9$, the second isolate iso_2 is labelled $\lambda(\text{iso}_2) = 14$. This describes the edges v_1v_2 and v_2v_3 . Now v_4 enters the scene, with edges to v_1 and to v_3 . When using $\lambda(v_4) = 13$, the edge v_1v_4 is already properly labelled by iso_2 . With a third isolate iso_3 labelled $\lambda(\text{iso}_3) = 22$, we again find an optimal sum labelling, since we know that $\sigma(C_4) = 3$.

However, this was a bit lucky: if the cyclic order was (v_1, v_2, v_4, v_3) , then we would have $\lambda(v_3) = 9$ and $\lambda(\text{iso}_2) = 10$. Now, $\lambda(v_4) = 13$ would lead to isolates labelled $\lambda(\text{iso}_3) = 18$ and $\lambda(\text{iso}_4) = 22$, so we would actually need four isolates in this case. \diamond

As we always start with setting the label of the first vertex to 1, the obtained labelling uses the number 1 as a label. Notice that this is related to the (to the best of our knowledge, still open) question whether every graph G (without isolates) can be embedded into a sum graph H with $\sigma(G)$ many isolates such that there is a sum labelling λ of H with $\lambda(v) = 1$ for some vertex $v \in V(H)$, see [\[MRS98, KKN⁺18\]](#).

Modifications of our Algorithm. Notice that we are creating new isolates only when necessary. This has the nice consequence that we can use the same isolate for various edges. Due to this, our algorithm recovers the optimal sum labelling of the complete graph K_n , for example.

However, there are circumstances when this kind of optimization is not really wanted. For instance, when we store graphs that behave more dynamically, we might want to have the possibility to quickly delete edges. In that case, it is beneficial to use exactly m distinct edge labels (i.e., isolates) to help with these updates, as then, no further changes or re-computations of vertex labels are necessary, as only the respective isolates have to be deleted. Similarly, vertex deletions can be incorporated efficiently.

We can modify our algorithm to ensure that (new) vertex labels are changed (according to [Equation 19](#) and [Equation 20](#)) until this uniqueness condition concerning edge labels is satisfied. By using the same pigeonhole argument, the overall argument of the algorithm is not changed, so that we can even meet that upper bounds on label sizes promised in [Theorem 12](#) and [Theorem 24](#) for this modification.

As our labelling algorithm can be thought of building up the graph vertex-by-vertex, also adding vertices to an existing, labelled graph is not that difficult, because we can simply run our algorithm one step further, this way processing the new vertex (and its incident edges).

It is not that clear if we can further modify our algorithm to also cope with edge additions, as this might require re-labelling the vertices. All these discussions respond to the question if and how sum labellings can be used for storing and accessing possibly dynamically evolving graphs.

A further natural modification of our algorithm would be a randomized variation thereof.³ At first thought, one might think that by selecting a random number in a certain interval, there is a good chance to pick a number that produces the required edges and avoids any unwanted ones. However, our thoughts in this direction revealed that this interval should be a range of numbers in $\{1, 2, \dots, n^6\}$ or a similar polynomial upper bound. This is obviously worse than what we could achieve with our deterministic algorithm. Yet, further improvements of a randomized algorithm might be possible and could then lead to some ideas of storing graphs that are better suited for update operations on graphs.

³This idea was given by Jaikumar Radhakrishnan during a (virtual) talk on this work at TIFR in October, 2021.

6 Labelling Sparse Graphs

We will now look into specific classes of sparse graphs. We consider graph degeneracy as our primary measure of sparseness. Notice that sparse graphs are often considered as modelling real-world networks more faithfully than general graphs that could be arbitrarily dense. In fact, as we will see, this restriction does give us some advantage when storing graphs with sum-labelling schemes.

Definition 23. A graph is called d -degenerate if every subgraph of the graph has a vertex of degree at most d . The degeneracy of a graph is the minimum d for which it is d -degenerate. \diamond

It is easy to see that the vertex set of a d -degenerate graph $G = (V, E)$ can be ordered as $V = \{v_1, v_2, \dots, v_n\}$ in polynomial time such that the vertex v_i has degree at most d in the graph G_i induced by the vertices $V_i = \{v_1, v_2, \dots, v_i\}$. We call such an ordering a d -degenerate vertex ordering. We will use this concept in the proof of the following theorem.

Theorem 24. Every d -degenerate, n -vertex, m -edge graph G of minimum degree at least one can be made a sum graph H by adding at most m isolates to G , such that H admits a sum labelling λ satisfying

$$\lambda(v) \leq 6d \cdot n^2 \quad \forall v \in V(G); \quad (25)$$

$$\lambda(v) \leq 12d \cdot n^2 \quad \forall v \in V(H). \quad (26)$$

This sum labelling is an exclusive labelling, computable in polynomial time.

Proof. We will only point to the changes needed to make the analysis of [Theorem 12](#) work in this special case. Recall that [Theorem 12](#) was proved by induction on an arbitrary ordering of its vertex set V . However, in this proof, since G is d -degenerate, we pick a d -degenerate vertex ordering $V = \{v_1, v_2, \dots, v_n\}$ of G . Recall that $G_i = G[\{v_1, v_2, \dots, v_i\}]$. For G_i , a sum graph H_i was constructed by adding r_i isolates. We add the following assertions that we are going to prove inductively about H_i :

- H_i contains $r_i \leq d \cdot (i - 1)$ many isolates that are not vertices of G_i .
- For labelling vertices of G_i , labels no larger than $6d \cdot i^2$ are used.
- For labelling isolates of H_i , labels no larger than $12d \cdot i^2$ are used.

Moreover, the vertex v_{i+1} added to G_i in order to obtain G_{i+1} has $t_i \leq d$ many neighbours in $V(G_i)$, as guaranteed by a d -degenerate vertex ordering. Now, in the analysis of the induction step, the main point was to discuss two cases of a violating triple (u, w, y) .

- y is an isolate of H_i and exactly one of $\{u, w\}$ belongs to $V(G_i)$, while the other is v_{i+1} . There are at most $i \cdot r_i \leq d \cdot i \cdot (i - 1)$ many cases when this might occur.
- y is an isolate of H_{i+1} and $\{u, w\} \subseteq V(G_i)$. There are at most $t_i \cdot i(i - 1)/2 \leq d \cdot i \cdot (i - 1)/2$ many cases when this might occur.

This proves that after at most $s_i = \frac{3}{2}d \cdot i \cdot (i - 1)$ many increment steps, v_{i+1} will have a label no larger than $6d \cdot i^2$. This also proves the claimed bound on the label size for the isolates. \square

This gives an upper bound of $(n + m)(\log(12dn^2))$ on the total number of bits required to store H . Since every vertex in the graph G has degree at least one, we have $n \leq 2m$. Substituting, we get an upper bound of $3m(\log(12dn^2)) \leq 3m(2\log n + \log 12d)$, as required by [Theorem 4](#).

Labelling Planar Graphs. Since planar graphs are 5-degenerate [LW70], our sum labelling needs labels with $2\log_2(n) + \mathcal{O}(1)$ bits for storing planar graphs (by taking logarithms in Equation 26), improving on previous published bounds for implicit representations of planar graphs [Sch89, Sch90, BGH⁺06, GL07, KNR92, KW95, MR01], except the very last proposal [BGP20] (see also [DEG⁺20]).

In adjacency labelling, the labels of two vertices alone are enough to decide whether the vertices are adjacent or not; for sum labelling, one needs to additionally check the labels of all the other vertices. Thus, sum labelling is not an adjacency labelling. However, our approach generalises to graphs of arbitrary fixed degeneracy, which is unclear for other approaches from the literature on adjacency labelling schemes.

In a recent breakthrough [DEG⁺20], it was shown that for every n , there is a “universal graph” U_n on $n^{1+o(1)}$ vertices such that every n -vertex planar graph is an induced subgraph of U_n . Analogously, Theorem 24 implies that every n -vertex planar graph can be represented by a subset of $[60n^2]$ (as planar graphs are 5-degenerate and our upper bound is $12dn^2$). Is it possible to arrive at sum labelling representations for planar graphs that only need numbers from $[c \cdot n^{1+o(1)}]$ instead, for some constant c ? This open question is a bridge to the final section, where we also discuss several lines of future research in this area.

7 Discussion

It is an interesting question how bad the labelling produced by our algorithm could get if it comes to determining the exclusive sum number of a graph. To give another example, when labelling the complete bipartite graph $K_{|P|,|Q|}$, with its vertex set V split into two independent sets P, Q , the ordering that first lists P and then Q will actually produce the optimal exclusive sum labelling as suggested in [MPR⁺05, Rya09]. Also by presenting the vertices of P and Q alternatingly to our algorithm, one can produce a labelling that realizes the exclusive sum number $|P| + |Q| - 1$ of $K_{|P|,|Q|}$, but then the range is nearly twice as large.

This brings us to the following interesting question: is there always a vertex ordering such that our algorithm yields an optimal exclusive sum labelling?

Proposition 27. *There exists a family of graphs (G_n) such that, if our algorithm is presented with a certain ordering of $V(G_n)$, where $|V(G_n)| = n \geq 3$, then it will produce a labelling λ_n matching $\varepsilon(G_n)$, but if presented with a different ordering, it will yield a labelling λ'_n requiring $|E(G_n)|$ many isolates. The ratio between the number of isolates produced by λ'_n and $\varepsilon(G_n)$ grows beyond any limit.*

Proof. The mentioned family of graphs is the family of paths. The exclusive sum number of paths equals two. Let us check this first with a small example: let us discuss $1 - 2 - 3 - 4 - 5$ as a P_5 . However, given the ordering $1, 2, 3, 4, 5$ of the vertices, our algorithm would produce the labelling $\lambda(1) = 1, \lambda(2) = 5, \lambda(3) = 9, \lambda(4) = 17, \lambda(5) = 29$, with the isolates labelled $6, 14, 26, 46$. In general, presenting the vertices in such a sequence would require $n - 1$ isolates for an n -vertex path, which is as bad as it could be in terms of the number of isolates. Yet, the ordering $1, 3, 5, 4, 2$ gives $\lambda(1) = 1, \lambda(3) = 5, \lambda(5) = 9, \lambda(4) = 13, \lambda(2) = 17$, with only two isolates (which is optimal), 18 and 22 . This is also true in general: if the vertices $1 - 2 - \dots - n$ of a P_n are presented as $1, 3, \dots, n, n - 1, n - 3, \dots, 2$ (if n is odd) or as $1, 3, \dots, n - 1, n, n - 2, \dots, 2$ (if n is even), then an optimal exclusive sum labelling is achieved, with the isolates labelled $4n - 2$ and $4n + 2$ (if n is odd) or $4n + 2$ and $4n + 6$ (if n is even). \square

As shown in this proof, the family of paths on n vertices gives such a graph family. The labelling

that is optimal with respect to the exclusive sum number is different from the one proposed in [MPR⁺05, Rya09].

Moreover, the following computational complexity questions are of interest, in particular, if one wants to apply sum labellings for storing real-world graphs. Are there polynomial-time algorithms for (any of) the following questions, given a graph G without isolates?

- Determine the sum number $\sigma(G)$ and find a corresponding sum labelling.
- Determine the exclusive sum number $\varepsilon(G)$ and find a corresponding exclusive sum labelling.
- Find a sum labelling minimizing the range of the labels.
- Find a sum labelling minimizing the storage needs $\sigma_{\text{store}}^{\max}(G)$ or $\sigma_{\text{store}}(G)$.

In particular, if a question of the suggested form would be NP-hard, it would be interesting to know if there are good heuristics that order the vertices of a graph in a way that our algorithm produces a provable approximation to the best graph parameter value. As the proof of [Proposition 27](#) shows, for instance the strategy behind the proof of [Theorem 24](#) would actually produce a worst-case labelling in a sense, i.e., even labellings that have some good properties can be really bad with respect to another criterion. If it comes to giving an NP-hardness proof for any of these questions, one of the difficulties is that the graph parameters related to sum labelling have a non-local flavour in the sense that local modifications of a graph could have tremendous effect on the graph parameters. It seems important to further study different typical graph operations with respect to these parameters. Here, more results like [Lemma 8](#) are needed [KPR06].

Among the hundreds of different graph labellings presented in [Gal20], the following are closest to sum labellings and could lead to considerations similar to the ones of this paper.

- Integral sum labellings [Har94], where also negative numbers are allowed to be used for labelling;
- Modulo (mod) sum labellings [Har90, SM99, SMRS99, Sut00], where addition modulo k is used as operation on natural numbers;
- Product labellings [BHJ⁺92], where the product operation on natural numbers is used instead of the summation.

These labellings can also be used to store graphs. Hence, questions similar to the ones raised and partially answered in this paper for sum labellings could be also considered for other graph labellings. Notice that although sum and product graphs coincide [BHJ⁺92], the sizes of the labels are quite different, and therefore different labellings might have their own pros and cons if it comes to storing graphs. Seen from a computer science perspective, it would even make sense to look at further labelling schemes not (yet) considered in the graph theory literature, for instance, mapping vertices to bit-vectors and then storing edges by means of bit-vectors obtained by, say, a bitwise OR-operation or AND-operation, because such operations can be implemented quite efficiently, similar to addition, and better than, say, multiplication, which is likely to be the least interesting number operation in our context anyways.

All these questions could open up quite new and challenging lines of research, possibly also further bridging to adjacency labellings [KNR92, Pel00].

Finally, recall that the sum labelling for trees proposed in [Eil93] introduces labels of exponential size. This means that, although only one isolate is added (proving that trees have sum number one), at worst $\Omega(n^2)$ many bits might be needed to encode trees in this way, while our approach needs only

$\mathcal{O}(n \log(n))$ bits to store trees, using at worst $n - 1$ many isolates, as shown in the proof of [Proposition 27](#) for the case of paths. It is an open question if there is a sum labelling of any n -vertex tree T that uses $\mathcal{O}(n \log(n))$ bits and still certifies that $\sigma(T) = 1$. A similar question can be asked concerning exclusive labellings, aiming at matching $\varepsilon(T)$. However, to the best of our knowledge, no general formula is known for $\varepsilon(T)$. The most interesting fact in this direction was proved in [\[Rya09\]](#) for caterpillar graphs: here, the exclusive sum number matches the maximum degree. Also, the given labelling only uses labels of polynomial size.

One of our motivations to return towards sum labellings of graphs was the possibility to store graphs in a database. We already discussed above that the question if an edge is present or not can be efficiently answered with sum label representations. We have discussed several operations (like accessing adjacency information, and adding or removing vertices or edges) above in the context of our algorithm. In particular, an ever-expanding database that is gradually built up can be efficiently implemented and then accessed using our sum-labelling scheme.

Acknowledgements. The authors are grateful to the organisers of GRAPHMASTERS 2020 [\[GKR20\]](#) for providing the virtual environment that initiated this research. A part of this work was done when the second author was a postdoctoral researcher at Technion, Israel. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 682203-ERC-[Inf-Speed-Tradeoff]. An extended abstract of this paper appeared in [\[FG21\]](#).

References

- [AG08] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.
- [Ang12] R. Angles. A comparison of current graph database models. In *2012 IEEE 28th International Conference on Data Engineering Workshops*, pages 171–177. IEEE, 2012.
- [BGH⁺06] N. Bonichon, C. Gavoille, N. Hanusse, D. Poulalhon, and G. Schaeffer. Planar graphs, via well-orderly maps and trees. *Graphs and Combinatorics*, 22(2):185–202, 2006.
- [BGP20] M. Bonamy, C. Gavoille, and M. Pilipczuk. Shorter labeling schemes for planar graphs. In S. Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 446–462. SIAM, 2020.
- [BHJ⁺92] D. Bergstrand, K. Hodges, G. Jennings, L. Kuklinski, J. Wiener, and F. Harary. Product graphs are sum graphs. *Mathematics Magazine*, 65(4):262–264, 1992.
- [DEG⁺20] V. Dujmovic, L. Esperet, C. Gavoille, G. Joret, P. Micek, and P. Morin. Adjacency labelling for planar graphs (and beyond). In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 577–588. IEEE, 2020.
- [DSUBGV⁺10] D Dominguez-Sal, P Urbón-Bayes, A Giménez-Vanó, S Gómez-Villamor, N Martínez-Bazan, and J-Ll Larriba-Pey. Survey of graph database performance on the HPC scalable graph analysis benchmark. In *International Conference on Web-Age Information Management*, pages 37–48. Springer, 2010. See also: <http://graphanalysis.org/index.html>.

- [Dut91] J. Dutka. The early history of the factorial function. *Archive for History of Exact Sciences*, 43(3):225–249, 1991.
- [Eli75] P. Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.
- [Ell93] M. N. Ellingham. Sum graphs from trees. *Ars Combinatoria*, 35:335–349, 1993.
- [FG21] H. Fernau and K. Gajjar. The space complexity of sum labelling. In E. Bampis and A. Pagourtzis, editors, *Fundamentals of Computation Theory - 23rd International Symposium, FCT*, volume 12867 of *LNCS*, pages 230–244. Springer, 2021.
- [FRS08] H. Fernau, J. F. Ryan, and K. A. Sugeng. A sum labelling for the generalised friendship graph. *Discrete Mathematics*, 308:734–740, 2008.
- [Gal20] J. A. Gallian. A dynamic survey of graph labeling, version 23. *The Electronic Journal of Combinatorics*, DS 6, 2020. <https://www.combinatorics.org/ojs/index.php/eljc/article/view/DS6/pdf>.
- [GKR20] L. Gąsieniec, R. Klasing, and T. Radzik. *Combinatorial Algorithms: 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8-10, 2020, Proceedings*, volume 12126. Springer Nature, 2020.
- [GL07] C. Gavaille and A. Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In L. Arge, M. Hoffmann, and E. Welzl, editors, *Algorithms - ESA 2007, 15th Annual European Symposium*, volume 4698 of *LNCS*, pages 582–593. Springer, 2007.
- [GR91] R. J. Gould and V. Rödl. Bounds on the number of isolated vertices in sum graphs. In Y. Alavi, G. Chartrand, O. R. Ollermann, and A. J. Schwenk, editors, *Graph Theory, Combinatorics, and Applications, 1988. Two Volume Set*, pages 553–562. John Wiley and Sons, 1991.
- [Har90] F. Harary. Sum graphs and difference graphs. *Congressus Numerantium*, 72:101–108, 1990.
- [Har94] F. Harary. Sum graphs over all the integers. *Discrete Mathematics*, 124(1-3):99–105, 1994.
- [HS95] N. Hartsfield and W. F. Smyth. A family of sparse graphs of large sum number. *Discrete Mathematics*, 141(1-3):163–171, 1995.
- [JV13] S. Jouili and V. Vansteenberghe. An empirical comparison of graph databases. In *2013 International Conference on Social Computing*, pages 708–715. IEEE, 2013.
- [KK15] R. Kumar Kaliyar. Graph databases: A survey. In *International Conference on Computing, Communication & Automation*, pages 785–790. IEEE, 2015.
- [KKN⁺18] M. Konečný, S. Kučera, J. Novotná, J. Pekárek, Š. Šimsa, and M. Töpfer. Minimal sum labeling of graphs. *Journal of Discrete Algorithms*, 52-53:29–37, 2018.

- [KMN01] J. Kratochvíl, M. Miller, and H. M. Nguyen. Sum graph labels – an upper bound and related problems. In *12th Australasian Workshop on Combinatorial Algorithms, AWOCA*, pages 126–131. Institut Teknologi Bandung, Indonesia, 2001.
- [KNR92] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.
- [KPR06] A. Korman, D. Peleg, and Y. Rodeh. Constructing labeling schemes through universal matrices. In T. Asano, editor, *Algorithms and Computation, 17th International Symposium, ISAAC*, volume 4288 of *LNCS*, pages 409–418. Springer, 2006.
- [KW95] K. Keeler and J. R. Westbrook. Short encodings of planar graphs and maps. *Discrete Applied Mathematics*, 58(3):239–252, 1995.
- [LFR08] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.
- [LW70] D. R. Lick and A. T. White. k -degenerate graphs. *Canadian Journal of Mathematics*, 22(5):1082–1096, 1970.
- [MPR⁺05] M. Miller, D. Patel, J. Ryan, K. A. Sugeng, Slamin, and M. Tuga. Exclusive sum labeling of graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 55:137–148, 2005.
- [MR01] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.
- [MRR17] M. Miller, J. F. Ryan, and Z. Ryjáček. Characterisation of graphs with exclusive sum labelling. *Electronic Notes on Discrete Mathematics*, 60:83–90, 2017.
- [MRS98] M. Miller, J. Ryan, and W. F. Smith. The sum number of the cocktail party graph. *Bulletin of the Institute of Combinatorics and its Applications*, 22:79–90, 1998.
- [NMS01] H. Nagamochi, M. Miller, and Slamin. On the number of isolates in graph labeling. *Discrete Mathematics*, 243:175–185, 2001.
- [Pel00] D. Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33(3):167–176, 2000.
- [Pya01] A. V. Pyatkin. New formula for the sum number for the complete bipartite graphs. *Discrete Mathematics*, 239(1-3):155–160, 2001.
- [Rya09] J. Ryan. Exclusive sum labeling of graphs: A survey. *AKCE International Journal of Graphs and Combinatorics*, 6(1):113–136, 2009.
- [Sch89] W. Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.
- [Sch90] W. Schnyder. Embedding planar graphs on the grid. In D. S. Johnson, editor, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 138–148. SIAM, 1990.

- [Slo73] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, OEIS Foundation Inc. (2021), 1973. <https://oeis.org/A002858>.
- [SM99] M. Sutton and M. Miller. Mod sum graph labelling of $H_{m,n}$ and K_n . *Australasian Journal of Combinatorics*, 20:233–240, 1999.
- [SM01] M. Sutton and M. Miller. On the sum number of wheels. *Discrete Mathematics*, 232:185–188, 2001.
- [SMRS99] M. Sutton, M. Miller, J. Ryan, and Slamin. Connected graphs which are not mod sum graphs. *Discrete Mathematics*, 195(1):287–293, 1999.
- [Smy91] W. F. Smyth. Sum graphs of small sum number. *Colloquia Mathematica Societatis János Bolyai*, 60:669–678, 1991.
- [SSM06] S. Slamet, K. A. Sugeng, and M. Miller. Sum graph based access structure in a secret sharing scheme. *Journal of Prime Research in Mathematics*, 2:113–119, 2006.
- [STT21] S. Singla, A. Tiwari, and A. Tripathi. Some results on the spum and the integral spum of graphs. *Discrete Mathematics*, 344(5):112311, 2021.
- [Sut00] M. Sutton. *Summable Graph Labellings and Their Applications*. PhD thesis, Department of Computer Science, University of Newcastle, Australia, 2000.
- [TM03] M. Tuga and M. Miller. Delta-optimum exclusive sum labeling of certain graphs with radius one. In J. Akiyama, E. T. Baskoro, and M. Kano, editors, *Combinatorial Geometry and Graph Theory, Indonesia-Japan Joint Conference, IJCCGGT*, volume 3330 of *LNCS*, pages 216–225. Springer, 2003.
- [WL01] Y. Wang and B. Liu. The sum number and integral sum number of complete bipartite graphs. *Discrete Mathematics*, 239(1-3):69–82, 2001.

A More Ways of Using Sum Labellings for Storing Graphs

In [KMN01], the authors started out with optimal sum labellings and showed that a sum labelling of a sum graph $H = G + \overline{K_{\sigma(G)}}$ is possible with label sizes at most $4^{|V(G)| + \sigma(G)}$. This is not what we do in our theorems. Hence, there might be a trade-off between label sizes and number of isolates. We further on this in the next remark. But we must make it clear that our theorems (unfortunately) do not solve Problem 1 as formulated in [KMN01] where the question was asked if one can prove a bound of $o(4^n)$ for the size of labels needed to realize $\sigma(G)$ for every n -vertex sum graph G .

In the following, we discuss an alternative approach for sum labelling arbitrary graphs. As we will see, this might also lead to graph representations that need $\Omega(m \log(n))$ many bits. However, it is not clear if fast graph queries are possible with this representation.

Theorem 28. *Every graph on n vertices without isolates can be embedded into a sum graph with at most $\frac{1}{2}n^2$ many isolates, so that each of the labels (also for the isolates) does not need more than $n + 2$ many bits. Hence, the numbers involved in labelling n -vertex graphs grow with $\mathcal{O}(2^n)$. In addition, the resulting labelling is exclusive.*

Proof. Let G be an n -vertex, m -edge graph. Consider its $n \times m$ incidence matrix I_G . Each column gives a bit-vector for an edge. Now, first add two rows to this matrix I_G on top, an all-ones row, followed by an all-zeroes row; this gives the matrix I'_G , with $(n+2)$ rows and m columns. Then, consider the $n \times n$ identity matrix I . Append two rows to on top: an all-zeros row, followed by an all-ones row, and call this matrix I' , giving an $(n+2) \times n$ matrix. Now, concatenate these matrices: first put the columns of I' , followed by the columns of I'_G , to get an $(n+2) \times (n+m)$ matrix S_G . The columns of S_G are treated as binary numbers that label the vertices of G (in the part I') and that store the labels of the edges of G (as we aim at an exclusive labelling). Notice the importance of the leading two bits (the two rows that are added on top): when we add two bit-vectors denoting edges, we will create a number that is not in the list of vertex labels (as twice the highest bit is set). Also, when adding a bit-vector denoting an edge and a vertex, we create a bit-vector that starts with 11, which is not found among the vertex labels. Finally, by construction, all labels needed for the isolates designating edges in G are present, as we started with the incidence matrix of G . \square

When it comes to storing graphs, the $n+m$ many bit-vectors in the proof of [Theorem 28](#) can be stored more efficiently than using $(n+2)(n+m)$ many bits, because the column vectors contain at most three 1-bits each. Hence, one would need

$$\begin{aligned} & 2\lceil \log_2(n) \rceil + 1 + 2\lceil \log_2(m) \rceil + 1 + n\lceil \log_2(n) \rceil + 2m\lceil \log_2(n) \rceil \\ &= (n+2m+2)\lceil \log_2(n) \rceil + 2\lceil \log_2(m) \rceil + 2 \in \mathcal{O}(m \log(n)) \end{aligned} \quad (29)$$

many bits to store first the number of vertices (in the format $1^{\lceil \log_2(n) \rceil} 0(n)_2$, where $(x)_2$ refers to a binary string for the number x), then the number of edges (in the same format) and finally n pointers signalling the 1-bit of the vertex labels and $2m$ pointers signalling the two 1-bits of the vertex labels. This is much better than the $\mathcal{O}(mn)$, i.e.,

$$(n+m)(n+2) = n^2 + nm + 2n + 2m,$$

bound for the number of bits in the original form of the previous theorem. However, it is not that clear if one can query graph edges as efficiently (as discussed earlier for sum labelling formats).

One might argue that the term m looks bad, but at least on average m is about the sum number of a graph, see [\[NMS01\]](#). So, only for special cases (as the K_n shows), $\sigma(G)$ is way smaller, and then the result [\[KMN01\]](#) that uses at most

$$(n + \sigma(G)) \cdot \log_2(4^{n+\sigma(G)}) = 2(n + \sigma(G))^2$$

many bits is superior to ours. More precisely, for storing K_n , the algebraic method of Kratochvíl, Miller and Nguyen uses at most $18n^2$ many bits, while our original method would need $(n + n^2/2)(n+2)$ many bits, while our improved method needs about $(n + n^2 + 2)\log_2(n) + 4n$ many bits according to [Equation 29](#).

In fact, this calculation shows that (up to logarithmic factors), that our compressed encoding is at least as good as the encoding offered by [\[KMN01\]](#). Compared to traditional ways of storing graphs, observe that with an adjacency matrix, one would need n^2 many bits, while with an adjacency list, one needs (again) $\mathcal{O}(m \log(n))$ many bits, which is comparable with [Equation 29](#).

A Novel Variant of Sum Labelling. We already discussed exclusive sum labellings as a stricter variant of sum labelling. We are now proposing a relaxation of the notion. Namely, another way of using the

concept of sum labelling for storing graphs is when the newly added vertices are not constrained to be isolates, leading to the notion of *supersum labelling*. That is, given a graph G , let $\varsigma(G)$ be the minimum number of (not necessarily isolated) vertices that need to be added to G to make it a sum graph. In other words, we are looking for the smallest sum graph H that is a supergraph of a given graph G and G is an induced graph of H . Clearly, $\varsigma(G) \leq \sigma(G)$. In fact, there are examples where $\varsigma(G) < \sigma(G)$. For instance, we know that for the 4-cycle, $\sigma(C_4) = 3$. We can show that $\varsigma(C_4) \leq 2$ by labelling its vertices $(1, 2, 3, 5)$ in cyclic order, and the two additional vertices 6 and 8. (This is not allowed in sum labelling since $(2, 6, 8)$ would be a violating triple.) We are going to explore this new graph parameter in a subsequent paper.

As a final note, one could also define a supersum variation of exclusive sum labelling, leading to the graph parameter ς_ε , meaning that no vertex in the encoded graph G is a working vertex of the labelling. However, this would not be an interesting field of future research, as our next (and final) theorem proves.

Theorem 30. *Let G be a graph without isolates. Then, $\varepsilon(G) = \varsigma_\varepsilon(G)$.*

Proof. By definition, $\varepsilon(G) \geq \varsigma_\varepsilon(G)$. Let λ be an exclusive labelling of G that certifies $\varsigma_\varepsilon(G)$. Now, consider the labelling λ' that labels $v \in V(G)$ by $4 \cdot \lambda(v) + 1$. Let H be the sum graph realizing G and λ , and also G and λ' . For the vertices v in $V(H) = V(H')$ that are not in $V(G)$, we have $\lambda'(v) = 4 \cdot \lambda(v) + 2$. By modulo-4 arithmetics, it is rather obvious that λ' is an exclusive sum labelling such that there are no edges in the sum graph H' (realizing G and λ') between vertices not from G . Hence, $\varepsilon(G) \leq \varsigma_\varepsilon(G)$. \square

Hence, at best two bits per vertex and edge could be saved with a ς_ε -sum labelling, compared to the classical exclusive labelling. Also, one would need an algorithm different from the one proposed in this paper to exploit the fact that supersum labellings could be more parsimonious than sum labellings.