On the NP-Hardness Approximation Curve for Max-2Lin(2)

Björn Martinsson
KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

In the Max-2Lin(2) problem you are given a system of equations on the form $x_i + x_j \equiv b \pmod 2$, and your objective is to find an assignment that satisfies as many equations as possible. Let $c \in [0.5, 1]$ denote the maximum fraction of satisfiable equations. In this paper we construct a curve $s(c)$ such that it is NP-hard to find a solution satisfying at least a fraction s of equations. This curve either matches or improves all of the previously known inapproximability NP-hardness results for Max-2Lin(2). In particular, we show that if $c \geqslant 0.9232$ then $\frac{1-s(c)}{1-c} > 1.48969$, which improves the NP-hardness inapproximability constant for the min deletion version of Max-2Lin(2). Our work complements the work of O'Donnell and Wu that studied the same question assuming the Unique Games Conjecture.

Similar to earlier inapproximability results for Max-2Lin(2), we use a gadget reduction from the $(2^k - 1)$-ary Hadamard predicate. Previous works used k ranging from 2 to 4. Our main result is a procedure for taking a gadget for some fixed k, and use it as a building block to construct better and better gadgets as k tends to infinity. Our method can be used to boost the result of both smaller gadgets created by hand ($k = 3$) or larger gadgets constructed using a computer ($k = 4$).

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Constraint and logic programming; Theory of computation → Graph algorithms analysis

Keywords and phrases Inapproximability, NP-hardness, 2Lin(2), Max-Cut, Gadget

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.11

Category APPROX

Supplementary Material
Software (Source code): `https://github.com/bjorn-martinsson/NP-hardness-of-Max-2Lin-2`
archived at `swh:1:dir:24803e393eb5826edd7af8e837b182cb24282691`

1 Introduction

Maximum constraint satisfaction problems (Max-CSPs) form one of the most fundamental classes of problems studied in computational complexity theory. A Max-CSP is a type of problem where you are given a list of variables and a list of constraints, and your goal is to find an assignment that satisfies as many of the constraints as possible. Some common examples of Max-CSP are Max-Cut and Max-2Sat. Every Max-CSP also has a corresponding Min-CSP-deletion problem where your objective is deleting as few constraints as possible to make all of the remaining constraints satisfiable. The Min-CSP-deletion problem is fundamentally the same optimisation problem as its corresponding Max-CSP, however their objective values are different.

1.1 History of Max-Cut

The Max-Cut problem is arguably both the simplest Max-CSP as well as the simplest NP-hard problem. In the Max-Cut problem you are given an undirected graph, and your objective is to find a cut of the largest possible size. A cut of an undirected graph is a

© Björn Martinsson;
licensed under Creative Commons License CC-BY 4.0

partition of the vertices into two sets and the size of a cut is the fraction of edges that connect the two sets relative to the total number of edges. Solving Max-Cut exactly is difficult, but there are trivial approximation algorithms that get within a factor of $\frac{1}{2}$ of the optimum. One such algorithm is randomly picking the cut by tossing one coin per vertex.

Knowing this, one natural question is, how close can a polynomial time algorithm get to the optimum? Goemans and Williamson partly answered this in a huge breakthrough in 1995 [8] by applying semi-definite programming (SDP) to create a polynomial time algorithm that finds a solution that is within a factor of $\alpha_{\mathrm{GW}} \approx 0.87856$ of the optimum. At the time, there was hope that Goemans and Williamson's algorithm could be improved further to get even better approximation factors than 0.87856, but no such improvements were ever found. Instead, in 2004 Khot et al [11] proved using the Unique Games Conjecture (UGC), that approximating Max-Cut within a factor of $\alpha_{\mathrm{GW}} + \varepsilon$ is NP-hard for any $\varepsilon > 0$. This conjecture had been introduced by Khot two years prior [12]. This was possibly the first result establishing the close connection between UGC and SDP based algorithms.

To this day, UGC remains an open problem, and in particular no one has been able to find an approximation algorithm for Max-Cut with a better approximation ratio than α_{GW}. In 2008 O'Donnell and Wu [14] were able to very precisely describe the tight connection between SDP based approximation algorithms for Max-Cut and UGC. They constructed a curve $\mathrm{Gap}_{\mathrm{SDP}}(c) : [0.5, 1] \to [0.5, 1]$ with the following two properties:

1. It is UGC-hard to find a cut of size $\mathrm{Gap}_{\mathrm{SDP}}(c) + \varepsilon$ given that the optimal cut has size c for any $\varepsilon > 0$. We here use *UGC-hard* as a short hand for "NP-hard under UGC".
2. Within the RPR^2-framework [7, 14], there are polynomial time algorithms that are guaranteed to find a cut of size at least $\mathrm{Gap}_{\mathrm{SDP}}(c) - \varepsilon$ if the optimal cut has size c. The RPR^2-framework is a generalisation of Goemans and Williamson's algorithm.

This means that their work both describe the best known polynomial time approximation algorithms for Max-Cut, and also show that under UGC these approximation algorithms cannot be improved. It is important to note that their algorithmic results do not require UGC. We emphasis that one implication of their result is that giving efficient algorithms with a better performance would disprove UGC.

1.2 NP-hardness inapproximability of Max-2Lin(2)

Max-2Lin(2) is a Max-CSP that is very closely related to Max-Cut. An instance of Max-2Lin(2) is a system of linear equations on the form $x_i + x_j \equiv b \pmod 2$, and the objective is to find an assignment that satisfies as many equations as possible. Max-Cut is the special case where we only allow equations with right hand side equal to 1. This implies that any hardness result for Max-Cut immediately yields the same hardness result for Max-2Lin(2). One example of this is the UGC-hardness of Max-Cut described by the $\mathrm{Gap}_{\mathrm{SDP}}(c)$ curve by O'Donnell and Wu [14].

Furthermore, O'Donnell and Wu's algorithmic results [14] also directly carries over to Max-2Lin(2). This is because the RPR^2-framework that they relied on uses odd rounding functions, and therefore does not differentiate between Max-Cut and Max-2Lin(2).

The conclusion is that the $\mathrm{Gap}_{\mathrm{SDP}}(c)$ describes a tight connection between the UGC-hardness of Max-2Lin(2) as well as the best known polynomial time approximation algorithms for Max-2Lin(2). On the other hand, the NP-hardness inapproximability of Max-2Lin(2) is not well understood. The strongest NP-hardness inapproximability results known for Max-2Lin(2) ([9], [16]) are still far off from the UGC-hardness described by the $\mathrm{Gap}_{\mathrm{SDP}}(c)$ curve.

The aim of this paper is to improve the state of the art NP-hardness inapproximability of Max-2Lin(2) and also to give the full picture of the state of the art NP-hardness inapproximability of Max-2Lin(2). We do this by constructing a curve $s(c) : [0.5, 1] \to [0.5, 1]$ such that it is NP-hard to distinguish between instances where the optimal assignment satisfies a fraction of c of the equations, and instances where all assignments satisfy at most a fraction of $s(c)$ of the equations. Our curve either matches or improves all previously known NP-hardness inapproximability results for Max-2Lin(2). We construct the curve by solving a separate optimisation problem for each value of c, so our result covers the entire spectrum of $c \in [0.5, 1]$.

Our result complements the work by O'Donnell and Wu [14]. Our curve describes the state of the art NP-hardness inapproximability of Max-2Lin(2) while O'Donnell and Wu's $\text{Gap}_{\text{SDP}}(c)$ curve describes the UGC-hardness of Max-2Lin(2). It is worth noting that UGC is still an open problem that over the years has been the subject of much debate. There are results that indicate that UGC might be true, such as the proof of the closely related 2-to-2 Games Conjecture [13]. But on the other hand there are also results that indicate the UGC might be false, such as the existence of subexponential algorithms for Unique Games [2]. Currently there is no consensus for whether UGC is true or not. It is for this reason that it is important to study NP-hardness independent of UGC, especially for fundamental problems such as Max-2Lin(2).

1.3 Gadget reductions

Gadgets are the main tools used to create reductions from one Max-CSP Φ to another Max-CSP Ψ. A gadget is a description of how to translate a specific constraint φ of Φ into one or more constraints of Ψ. For example, if Φ is Max-3Lin(2) and Ψ is Max-Cut, then a gadget from φ to Ψ is a graph. A gadget is allowed to use both the original variables in the constraint φ, which are called *primary variables*, and new variables specific to the gadget, which are called *auxiliary variables*.

The standard technique used to construct gadgets is to follow the "automated gadget" framework of Trevisan et al [15]. This framework describes how to construct a gadget by solving a linear program and also proves that the constructed gadget is optimal. This framework is mainly used to construct gadgets for small and simple Max-CSPs. This is because the number of variables in the gadget scales exponentially with the number of satisfying assignments of φ. Furthermore, the number of constraints in the LP scales exponentially with the number of variables, so it scales double exponentially with the number of satisfying assignment of φ.

As an example let us take the gadget from Max-3Lin(2) to Max-2Lin(2) used by Håstad [9], which was originally constructed by Trevisan et al [15]. A constraint in Max-3Lin(2) has 4 satisfiable assignments. Having 4 satisfiable assignments means that the gadget uses $2^4 = 16$ variables. Furthermore, since Max-2Lin(2) allow negations, half of these variables can be removed because of negations. So the actual number of variables in the gadget is $2^{4-1} = 8$. This in turn implies that the number of constraints in the LP is $2^8 = 256$. This number is small enough that it is feasible for a computer to solve the LP. In this paper we are interested in constructing gadgets from generalisations of Max-3Lin(2), called the Hadamard Max-CSPs. These have significantly more satisfying assignments than Max-3Lin(2). It is easy to see that a simple-minded application of the "automated gadget" framework leads to an LP that is far too large to naively be solved by a computer. This means that we have to deviate from the "automated gadget" framework in order to construct our gadgets.

Gadgets have two important properties, called *soundness* s and *completeness* c. If a gadget is constructed using the "automated gadget" framework, then it is trivial to calculate the completeness of the gadget. On the other hand, calculating the soundness of a gadget from Φ to Ψ involves solving instances of Ψ. In practice, calculating the soundness of a large gadget can be very difficult since Ψ is usually an NP-hard problem.

Gadgets can be constructed with different goals in mind. The case that we are interested in is finding the gadget with the largest soundness for a fixed completeness. This is what allows us to construct our curve $s(c)$. In general there are also other objectives that could be of interest when constructing gadgets. One such case is finding the gadget with the smallest ratio of $\frac{s}{c}$. This corresponds to finding the best lower bound for the approximation ratio of Max-2Lin(2). Another possibility is to maximise $\frac{1-s}{1-c}$. This corresponds to finding the best upper bound for the approximation ratio of Min-2Lin(2)-deletion. It is possible to use the "automated gadget" framework by Trevisan et al [15] to find the optimal gadgets for all of these scenarios.

1.4 The Hadamard Max-CSPs Max-Had$_k$

One of the earliest gadget reductions used to show NP-hardness inapproximability of Max-2Lin(2) is a gadget reduction from Max-3Lin(2) used by Håstad in his classical paper from 1997 [9], which was constructed by Trevisan et al [15]. More recently, NP-hardness inapproximability results for Max-2Lin(2) have used gadget reductions from a generalisation of Max-3Lin(2) called the Hadamard Max-CSPs [10, 16]. The (2^k-1)-ary Hadamard Max-CSP, $k \geqslant 2$, is a constraint satisfaction problem where a clause is satisfied if and only if its literals form the truth table of a linear k-bit Boolean function. The (2^k-1)-ary Hadamard CSP is denoted by Max-Had$_k$. One special case is $k = 2$, where the number of literals of a clause is 3. It turns out that this case coincides with Max-3Lin(2). This means that Max-Had$_k$ can be seen as a generalisation of Max-3Lin(2).

There are mainly two reasons as to why Max-Had$_k$ is useful for gadget reductions. The first reason is that Max-Had$_k$ is a very sparse CSP. It being sparse refers to the number of satisfiable assignments of a clause being few in relation to the total number of possible assignments. The number of satisfying assignments of a clause is just 2^k, one for each linear k-bit Boolean function, while the total number of possible assignments is $2^{(2^k-1)}$.

The second reason is that Max-Had$_k$ is a *useless predicate* for any $k \geqslant 2$, which is an even stronger property than being approximation resistant. This was shown by Chan in 2013 [6]. Max-Had$_k$ being a useless predicate means that if you are given a nearly satisfiable instance of Max-Had$_k$, then it is NP-hard to find an assignment such that the distribution over the (2^k-1) long bit strings given by the literals of the clauses is discernibly different from the uniform distribution.

1.4.1 Historical overview of Had$_k$-to-2Lin(2) gadgets

In 1996, Trevisan et al [15] constructed the optimal gadget from Max-Had$_2$ to Max-2Lin(2). They showed that the Max-Had$_2$ gadget that minimises $\frac{s}{c}$ is the same gadget as the one that maximises $\frac{1-s}{1-c}$. Furthermore, since this gadget is very small, using only 8 variables, they were able to construct it using the "automated gadget" framework.

In 2015, Håstad et al. [10] constructed gadgets from Max-Had$_3$ to Max-2Lin(2). They showed that the Max-Had$_3$ gadget that minimises $\frac{s}{c}$ is equivalent to the Max-Had$_2$ gadget. So using Max-Had$_3$ over Max-Had$_2$ does not give an improved hardness for the approximation ratio of Max-2Lin(2). However, the Max-Had$_3$ gadget that maximises $\frac{1-s}{1-c}$ is notably better than the Max-Had$_2$ gadget. This gadget is relatively small, only using 128 variables. This is too many variables for it to be possible to naively apply the "automated gadget" framework. However, Håstad et al. were still able to construct and analyse the optimal gadget by hand based on ideas from the "automated gadget" framework.

In 2018, Wiman [16] constructed gadgets from Max-Had$_4$ to Max-2Lin(2). Note that Max-Had$_4$ gadgets have 2^{15} variables. Calculating the soundness of a such a gadget requires solving an instance of Max-2Lin(2) with 2^{15} variables, which is infeasible to do by hand or even with a computer. Wiman initially followed the "automated gadget" framework. However, in order to be able to calculate the soundness of the gadget, Wiman relaxed the Max-2Lin(2) problem into a Max-Flow problem. This *relaxed soundness* rs is an upper bound of the true soundness. This relaxation made it possible for Wiman to use a computer to find the gadget that maximises $\frac{1-\text{rs}}{1-c}$. Wiman's relaxation was successful, since by using it he was able to find a Max-Had$_4$ gadget that was better than the optimal Max-Had$_3$ gadget. Note, however, that by using a relaxation, it is uncertain whether Wiman found the optimal Max-Had$_4$ gadget or not.

1.4.2 Our Had$_k$-to-2Lin(2) gadgets

In this paper, we construct gadgets from Max-Had$_k$ to Max-2Lin(2) for k approaching infinity. Recall that a gadget uses 2^{2^k-1} variables, so using a computer to construct gadgets for $k > 5$ is normally impossible. We get around this limitation by introducing a procedure for taking Max-Had$_k$ gadgets and transforming them to Max-Had$_{k'}$ gadgets, for $k' > k$. We refer to this procedure as the *lifting* of a Max-Had$_k$ gadget into a Max-Had$_{k'}$ gadget. Two of the properties of lifting is that the completeness stays the same and the soundness does not decrease.

To show NP-hardness of approximating Max-2Lin(2), we start by constructing Max-Had$_k$ to Max-2Lin(2) gadgets for $k = 4$ using a computer. We then analytically prove an upper bound of Wiman's relaxed soundness of the lifting of these gadgets as $k' \to \infty$.

The method we use to construct our gadgets is by solving an LP. This LP is similar to what Wiman could have used to construct his gadget. The difference is that the LP we use is made to minimise the soundness of the lifted gadget, instead of minimising the soundness of the gadget itself. If done naively, this LP would have roughly $2^{3\cdot(2^k-1)} = 2^{45}$ variables. But by making heavy use of symmetries of the LP, we are able to bring it down to a feasible size.

The main technical work of this paper is proving an upper bound on Wiman's relaxed soundness of a lifted gadget as $k' \to \infty$. Recall that calculating Wiman's relaxed soundness involves solving instances of Max-Flow. As k' tends to infinity, the size of these instances also tend to infinity. In order to lower bound the value of these Max-Flow problems, we introduce the concept of a type of infeasible flows which we call *leaky flows*. A leaky flow is a flow for which the conservation of flows constraint has been relaxed. This allows leaky flows to attain higher values compared to feasible flows. We then show that by randomly overlapping leaky flows onto the large Max-Flow instances, we are able to get closer and closer to a feasible flow as the size of the instances tend to infinity.

1.5 Our results and comparison to previous results

Using a gadget reductions from Max-Had$_k$ to Max-2Lin(2), we are able to construct a curve $s(c) : [0.5, 1] \to [0.5, 1]$ such that it is NP-hardto distinguish between instances of Max-2Lin(2) where the optimal assignment satisfies a fraction of c of the equations and instances where all assignments satisfy at most a fraction of $s(c)$ of the equations. This curve does not have an explicit formula. Instead, each point on the curve is defined as the solution to an LP, which we solve using a computer.

▶ **Theorem 1.1.** *Let $s(c) : [0.5, 1] \to [0.5, 1]$ be the curve defined in Definition 4.1. Then for every sufficiently small $\varepsilon > 0$, it is NP-hard to distinguish between instances of Max-2Lin(2) such that*

Completeness *There exists an assignment that satisfies a fraction at least $c - \varepsilon$ of the constraints.*

Soundness *All assignments satisfy at most a fraction $s(c) + \varepsilon$ of the constraints.*

A notable point on the curve is $c = \frac{590174949}{639271832} \approx 0.9232$ and $s(c) = \frac{141533171}{159817958} \approx 0.8856$. This is the point on the curve that gives the highest NP-hardness inapproximability factor $\frac{1-s}{1-c}$ of Min-2Lin(2)-deletion.

▶ **Corollary 1.2.** *It is NP-hard to approximate Min-2Lin(2)-deletion within a factor of $\frac{73139148}{49096883} + \varepsilon \approx 1.48969 + \varepsilon$.*

In order to be able to compare our curve $s(c)$ to prior results, we plot our curve together with O'Donnell and Wu's $\text{Gap}_{\text{SDP}}(c)$ curve [14], which, as discussed earlier, describes both the UGC-hardness of Max-2Lin(2), as well as the best known polynomial time approximation algorithms of Max-2Lin(2). Additionally, we also include historical NP-hardness inapproximability results as points in the diagram. We have also marked the point (c, s) where Goemans and Williamson's algorithm achieves the approximation ratio of $\frac{s}{c} = \alpha_{\text{GW}} \approx 0.87856$. This point was shown to be UGC-hard by Khot et al. in 2004 [11].

The curve $s(c)$ is plotted in three Figures. All three Figures contain the same exact same data, but the data is plotted in different ways. In Figure 1 the soundness $s(c)$ is on the y-axis and the completeness c is on the x-axis. This plot has the disadvantage that to the eye, it is difficult to distinguish the exact shape of the curve $s(c)$. In the next plot, Figure 2, $\frac{s(c)}{c}$ is on the y-axis and c is on the x-axis. This plot describes the approximation ratio of Max-2Lin(2). The third plot, in Figure 3, has $\frac{1-s(c)}{1-c}$ on the y axis and c on the x-axis. This plot describes the approximation ratio of Min-2Lin(2)-deletion.

It is important to note that the curves in Figure 1 are convex functions since it is possible to take the convex combination of two hard instances using disjoint sets of variables. One implication from this is that it is possible to construct NP-hardness curves using any of the points (c, s) by drawing two lines, one from $(0.5, 0.5)$ to (c, s) and one from (c, s) to $(1, 1)$. This means that all of the historical inapproximability results can also be described using convex curves.

In Figures 1-3 prior inapproximability results for Max-2Lin(2) are marked as dots. Bellare et al [5] was first to give an explicit NP-hardness result, which had $c = 0.72$ and $s = 0.71$. In 2015, Håstad et al [10] used Chan's result [6] to create a gadget reduction from Max-Had_3 which had $c = \frac{7}{8}$ and $s = \frac{53}{64}$. This result became the new record for the upper bound of the approximation ratio of Min-2Lin(2)-deletion, as seen in Figure 3. Three years later, Wiman [16] further improved on this result by using Max-Had_4 instead of Max-Had_3. Wiman's Max-Had_4 gadget has $c = \frac{15}{16}$ and $s = \frac{3308625759}{3640066048} \approx 0.9089$. This further improved the upper bound on the approximation ratio of Min-2Lin(2)-deletion.

Similar to earlier results, the technique we use to construct our curve is also a gadget reduction from Max-Had_k to Max-2Lin(2). But instead of using a gadget reduction from Max-Had_k for a fixed k, we instead let k tend to infinity. This improves the quality of our gadget. One example of such an improvement is our upper bound on the approximation ratio of Min-2Lin(2)-deletion, which can be seen in Figure 3. The ratio $\frac{1-s(c)}{1-c}$ is maximised on our curve at $c = \frac{590174949}{639271832} \approx 0.9232$ and $s = \frac{141533171}{159817958} \approx 0.8856$, which is marked by a blue cross in Figure 3.

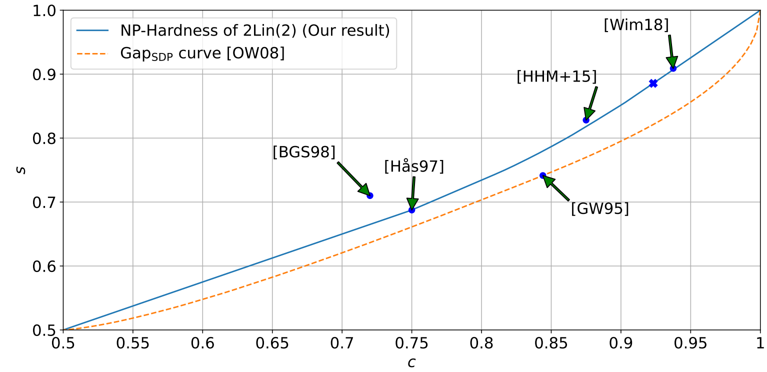

Figure 1 The y-axis shows the soundness s and the x-axis the completeness c. The blue filled curve is our NP-hardness curve $s(c)$. The red dashed curve is the $\text{Gap}_{\text{SDP}}(c)$ by O'Donnell and Wu's [14]. The points marked with arrows are prior inapproximability results of Max-2Lin(2). The blue cross on the curve marks our best inapproximability result for Min-2Lin(2)-deletion, see Figure 3. Note that both of the curves in this figure are convex functions.

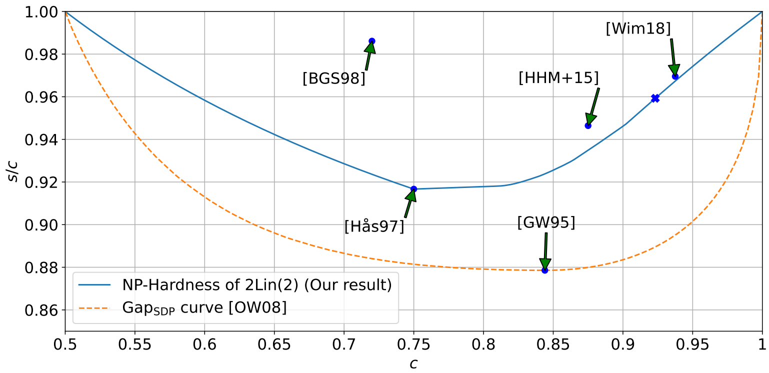

Figure 2 The y-axis shows s/c, which corresponds to the approximation ratio of Max-2Lin(2). The point on the curve $c(s)$ that minimises this ratio is $c = 3/4$ and $s(c) = 11/16$, which exactly matches Håstad's result from 1997 [9].

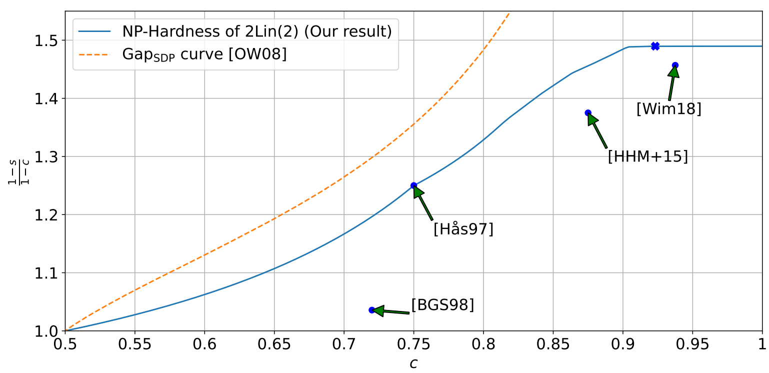

Figure 3 The y-axis shows $(1 - s)/(1 - c)$, which corresponds to the approximation ratio of Min-2Lin(2)-deletion. This ratio reaches its maximum $\frac{1-s(c)}{1-c} = \frac{73139148}{49096883} \approx 1.4896$ at $c = \frac{590174949}{639271832}$, which is marked by a blue cross. The curve stays constant after this point.

1.6 The limitations of Had_k-to-2Lin(2) gadget reductions

In Figure 1, it is possible to see a clear gap between our $s(c)$ curve and O'Donnell and Wu's $\mathrm{Gap}_{\mathrm{SDP}}(c)$ curve [14]. The gap is especially noticeable in Figure 3, since the behaviour of the two curves are completely different when c is close to 1. One natural question is, how close can a Had_k-to-2Lin(2) gadget reduction get to the $\mathrm{Gap}_{\mathrm{SDP}}(c)$ curve?

Håstad et al [10] showed that any gadget reduction from a Hadamard Max-CSP to Max-2Lin(2) can never achieve an approximation ratio for Min-2Lin(2)-deletion better than $\frac{1}{1-e^{-0.5}} \approx 2.54$. In Appendix B we show that any gadget reduction from a Hadamard CSP to Max-2Lin(2) that uses Wiman's soundness relaxation can never achieve an approximation ratio of Min-2Lin(2)-deletion better than 2. Both 2.54 and 2 are fairly large in comparison to the current best value of 1.48969 shown in Figure 3. So it is potentially possible to still improve our results in the future using a Had_k-to-2Lin(2) gadget reduction for some $k \geqslant 4$. However, these limitations means that it is impossible to make $s(c)$ match the behaviour of $\mathrm{Gap}_{\mathrm{SDP}}(c)$ when c is close to 1.

1.7 Outline of proof

Our result is based on Had_k-to-2Lin(2) gadget reduction for arbitrary large values of k. We start from the "automated gadget" framework by Trevisan et al [15]. In this framework, computing the soundness of of a Had_k-to-2Lin(2) gadget involves solving a Max-2Lin(2) problem. Following the work of Wiman [16], we relax the soundness computation to a Max-Flow problem on the 2^k-dimensional hypercube. Using symmetries, it is computationally feasible to construct Had_k-to-2Lin(2) gadgets that are optimal with respect to the relaxed soundness for $k \leqslant 4$.

In order to be able harness the power of arbitrarily large k, we define a procedure of embedding a Had_k-to-2Lin(2) gadget G inside a $\mathrm{Had}_{k'}$-to-2Lin(2) gadget where $k' > k$. By overlapping multiple different embeddings of G, we construct a gadget G' for an arbitrarily large k'.

Recall that the relaxed soundness computation is a Max-Flow problem, which can be expressed as an LP. By carefully relaxing this LP, we are able to create an infeasible flow solution to $\mathrm{rs}(G)$, such that if we lift it, it becomes an almost feasible flow of $\mathrm{rs}(G')$. The underlying idea for this relaxation is based on leaky flows (flows where the flow entering a node can be different than the flow exiting the node). The "leaks" of a leaky flow are signed, so random overlap of leaky flows can result in a feasible flow. We show that this is actually the case for the solution to our relaxed LP using a second order moment analysis.

The final step is to construct the Had_k-to-2Lin(2) gadget G and its corresponding leaky flow for $k = 4$ used in the embedding. This construction is naturally done using a rational LP solver to solve the relaxed LP.

1.8 Organisation of paper

Section 2 contain the preliminaries. It introduces Max-CSPs and the automated gadget framework. Section 3 introduces Wiman's relaxed soundness and the infinity relaxed soundness in terms of an LP. This section also states our main Lemma, Lemma 3.11, relating the infinity relaxed soundness to the relaxed soundness. Appendix A is about Max-Flow, and it proves some general theorems about how symmetries can be used to simplify Max-Flow problems. Appendix B contain an analysis of relaxed soundness, and how it relates to the (true) soundness. Appendix C studies affine maps. These affine maps are used both to analyse the infinity relaxed soundness, and to describe the symmetries of the LPs. Appendix D contains the proof of Lemma 3.11 using the affine maps. Appendix E describes the procedure we use for constructing and verifying the gadgets. Section 4 contains our numerical results. This includes both plots and tables of various Had_k-to-2Lin(2) gadgets. Finally, Appendix F contains a compact description of all gadgets that we construct.

2 Preliminaries

This section is split into three parts. In Subsection 2.1 we introduce some basic concepts and notations for Boolean functions $\mathbb{F}_2^k \to \{1, -1\}$. After that, in Subsection 2.2 we formally define the $(2^k - 1)$-ary Hadamard predicate. The last subsection, Subsection 2.3, introduces the "automated gadget" framework by Trevisan et al [15], and explains the classical result of how to construct reductions from the $(2^k - 1)$-ary Hadamard predicate to Max-2Lin(2).

2.1 Boolean functions

A k-bit Boolean function is a function that takes in k bits and outputs one bit. The k input bits should be thought of as a vector in a k-dimensional vector field over $\mathbb{F}_2$. On the other hand, the output bit is a scalar. For convenience, we denote the vectors as being elements in $\mathbb{F}_2^k$ and the scalars as elements in $\mathbb{R}$, where a scalar bit is represented as 1 (False) or -1 (True).

▶ **Definition 2.1.** *The set of k-bit Boolean functions is denoted by* $\mathcal{F}_k = \left\{ f : \mathbb{F}_2^k \to \{1, -1\} \right\}$.

One special type of Boolean functions that is of great importance is the set of linear Boolean functions. Each linear Boolean function in $\mathbb{F}_2^k$ corresponds to an element $\alpha \in \mathbb{F}_2^k$, and is denoted by χ_α.

▶ **Definition 2.2.** *For $\alpha \in \mathbb{F}_2^k$ let $\chi_\alpha \in \mathcal{F}_k$ be denote the function*

$$\chi_\alpha(x) \quad = \quad (-1)^{(\alpha,x)}$$

where $(\alpha, x) = \sum_{i=1}^k \alpha_i x_i \pmod 2$.

Any Boolean function can be represented as a sum of linear Boolean functions using the Fourier transform.

▶ **Proposition 2.3.** *Given $f \in \mathcal{F}_k$, then*

$$f(x) \quad = \quad \sum_{\alpha \in \mathbb{F}_2^k} \chi_\alpha(x) \hat{f}_\alpha,$$

where $\hat{f}_\alpha$ denotes the Fourier transform of f at α, defined as

$$\hat{f}_\alpha \quad = \quad \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} \chi_\alpha(x) f(x), \qquad \alpha \in \mathbb{F}_2^k.$$

The Fourier transform is used to define the supporting affine subspace of a Boolean function. This also gives a natural definition for the dimension of a Boolean function.

▶ **Definition 2.4.** *Given $f \in \mathcal{F}_k$, its supporting affine sub-space* $\text{affine}(f)$ *is the affine span of* $\left\{ \alpha \in \mathbb{F}_2^k : \hat{f}_\alpha \neq 0 \right\}$.

▶ **Definition 2.5.** *Let* $\dim(f), f \in \mathcal{F}_k$, *denote the dimension of* $\text{affine}(f)$.

▶ Remark 2.6. Affine functions have dimension 0.

The distance between two Boolean function is given by the normalised Hamming distance.

▶ **Definition 2.7.** *Let* $\text{dist} : \mathcal{F}_k \times \mathcal{F}_k \to \mathbb{R}$ *be the normalised Hamming distance between two Boolean functions, i.e.*

$$\text{dist}(f_1, f_2) = \frac{1}{2^k} \sum_{x \in \mathbb{F}_2^k} \frac{1 - f_1(x) f_2(x)}{2}.$$

2.2 Max-CSP

This section introduces Constraint Satisfaction Problems (CSP) and Max-CSP. The framework we use is that CSPs are defined by predicates, which describe which kind of constraints that can appear in the CSP.

▶ **Definition 2.8.** *An* m*-ary predicate is a function* $\phi \in \mathcal{F}_m$*. The predicate is said to be satisfied by* $x \in \mathcal{F}_2^m$ *if* $\phi(x) = -1$*. Otherwise* x *is said to violate* ϕ*. The set of* $x \in \mathcal{F}_2^m$ *that satisfies* ϕ *is denoted by* $\text{Sat}(\phi)$*.*

Given a set of Boolean variables V and a m-ary predicate ϕ, a ϕ*-constraint* $\mathcal{C}$ is a tuple $((x_1, b_1), \ldots, (x_m, b_m))$ where $x_i \in V, i = [m]$, and $b_i \in \mathbb{F}_2$, $i \in [m]$, where all of the x_i's are distinct. The constraint $\mathcal{C}$ is said to be satisfied if

$$\phi(b_1 + x_1, \ldots, b_n + x_m) = -1,$$

where $+$ denotes the xor-operation. In other words, if $b_i = 1$ then x_i is negated.

▶ **Definition 2.9.** *Given a* m*-ary predicate* ϕ*, an instance* $\mathcal{I}$ *of the Max-*ϕ*-CSP is a variable set* V *and a distribution of* ϕ*-constraints over* V*. The Max-*ϕ*-CSP optimisation problem is; given an instance* $\mathcal{I}$*, find the assignment* $A : V \to \mathbb{F}_2$ *that maximises the fraction of satisfied constraints in* $\mathcal{I}$*. The optimum is called the value of* $\mathcal{I}$*.*

The main Max-CSPs of interest in this paper are the Hadamard Had_k Max-CSP, and Max-2Lin(2) and Max-3Lin(2). These have the following predicates.

▶ **Definition 2.10.** *The 2Lin(2) predicate is the function* $f(x, y) = (-1)^{x+y+1}$*. Similarly, the 3Lin(2) predicate is the function* $f(x, y, z) = (-1)^{x+y+z+1}$*.*

▶ **Definition 2.11.** *The Hadamard* Had_k *predicate for* $k \geqslant 2$ *is a* $(2^k - 1)$*-ary predicate. There is one input variable per non-empty subset* $S \subseteq [k]$*. The* Had_k *predicate is satisfied by a binary input string* $\{x_S\}_{\varnothing \neq S \subseteq [k]}$ *if and only if there exists some* $\beta \subseteq [k]$ *such that*

$$\chi_\beta(S) = (-1)^{x_S}$$

for all non-empty subset $S \subseteq [k]$*. I.e. the* Had_k *predicate is satisfied if and only if the input string forms the truth table of a linear function.*

▶ Remark 2.12. The 3Lin(2) predicate and the Had_2 predicate are in fact identical. Thus the family of Hadamard Max-CSPs can be seen as a natural generalisation of Max-3Lin(2).

▶ Remark 2.13. The set $\text{Sat}(\text{Had}_k$ predicate) can be expressed using a 2^k dimensional Hadamard matrix. Let M_k be a $2^k \times (2^k - 1)$ matrix, where the rows are index by subsets $\beta \subseteq [k]$ and the columns are indexed by non-empty subsets $S \subseteq [k]$. Let

$$(M_k)_{\beta,S} = \begin{cases} 0 & \text{if} \quad \chi_\beta(S) = 1, \\ 1 & \text{if} \quad \chi_\beta(S) = -1. \end{cases}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4 The matrix M_k for $k = 3$. It is an 8×7 matrix. Note that prepending a zero column to M_k and switching 0/1 to $1/-1$ would make it into a Hadamard matrix, which is symmetric.

The matrix M_k is the set Sat(Had$_k$ predicate) expressed on the form of a matrix, with one row per element. Note that M_k is almost the 2^k-dimensional Hadamard matrix. M_k can be made into the Hadamard matrix by prepending an all 0 column to it, and then switching out 0/1 to $1/-1$. This connection between Sat(Had$_k$ predicate) and Hadamard matrices is one of the reasons as to why this Max-CSP is called the Hadamard Max-CSP. An example of the matrix M_k can be found in Figure 4.

The Hadamard predicate has been shown to be a *useless predicate*. The concept of useless predicates was first introduced in [3]. This property of Had$_k$ was originally proven by Austrin and Mossel using UGC [4], which relies on the fact that Sat(Had_k) admits a balanced pairwise independent set. Later on Chan was able to show that Had$_k$ is a useless predicate without requiring UGC [6]. To state this result we first need two definitions.

▶ **Definition 2.14.** *Given an instance $\mathcal{I}$ of an m-ary Max-CSP and an assignment A, let $\mathcal{D}(A, \mathcal{I})$ denote the distribution of binary strings $\mathbb{F}_2^m$ generated by sampling $((x_1, b_1), \dots, (x_m, b_m)) \sim \mathcal{I}$ and outputting the binary string $((A(x_1)+b_1), \dots, (A(x_m)+b_m))$.*

▶ **Definition 2.15.** *The total variation distance d_{TV} between two probability measures μ_1 and μ_2 over a finite set Ω is defined as*

$$d_{\mathrm{TV}}(\mu_1, \mu_2) = \frac{1}{2} \sum_{\omega \in \Omega} |\mu_1(\omega) - \mu_2(\omega)|.$$

▶ **Theorem 2.16** ([6]). *For every $\varepsilon > 0$, it is NP-hard to distinguish between instances $\mathcal{I}$ of the Had$_k$ Max-CSP such that*

Completeness *There exists an assignment A such that*

$$d_{\mathrm{TV}}(\mathcal{D}(A, \mathcal{I}), \mathrm{uniform}(\{\mathrm{Sat}\}(Had_k\ predicate))) \leqslant \varepsilon.$$

Soundness *For every assignment A,*

$$d_{\mathrm{TV}}(\mathcal{D}(A, \mathcal{I}), \mathrm{uniform}(\mathbb{F}_2^{2^k-1})) \leqslant \varepsilon.$$

Here $\mathrm{uniform}(\mathrm{Sat}(Had_k\ predicate))$ *denotes the uniform distribution over binary strings that satisfy the Had$_k$ predicate. Similarly,* $\mathrm{uniform}(\mathbb{F}_2^{2^k-1})$ *denotes the uniform distribution over all binary strings of length $2^k - 1$.*

▶ Remark 2.17. The uniform distribution on satisfiable instances in the completeness case is a subtle detail. The result by [6] is not formulated like this. However, it is trivial to take the instances constructed by Chan and modify them to make the completeness case be uniformly distributed over satisfied instances. The first time this was used was by [16]. However, this detail turns out to not actually matter in the end since all of the gadgets that we construct and all of the gadgets that Wiman construct are symmetric. So this uniform randomness assumption is only there because of convenience, and is not actually used in the end.

2.3 The automated gadget framework

The "automated gadget" framework by Trevisan et al [15] describes how to construct optimal gadgets when reducing from one predicate to another. Let us denote the starting predicate as ϕ and the target predicate as ψ. A *ϕ-to-ψ-gadget* is a description for how to reduce a ϕ-constraint to one or more ψ-constraints. As an example, let us take a gadget from 3SAT to 2Lin(2). In this case the gadget describes a system of linear equations that both involve the three original variables from the 3SAT constraint (called *primary variables*, denoted by $\mathbb{X}$) as well as new extra variables (called *auxiliary variables*, denoted by $\mathbb{Y}$).

Gadgets have two important properties, called *completeness* and *soundness*. These properties describe how closely the ψ-constraints are able to mimic the satisfiability of the original ϕ-constraint. The completeness of a gadget is a value between 0 and 1 that describe how many of the ψ-constraints that can be satisfied under the restriction that $\mathbb{X}$ satisfies the original ϕ-constraint. In a similar fashion, the soundness of a gadget is a value between 0 and 1 that describes the case when $\mathbb{X}$ does not satisfy the original ϕ-constraint. When we construct our gadgets, we fix the completeness of the gadget, and then we find the gadget that minimises the soundness for this fixed completeness. A gadget that minimises the soundness for a given completeness is referred to as an *optimal gadget*.

There is no a priori upper bound on how many auxiliary variables that a ϕ-to-ψ-gadget can have. However, the "automated gadget" framework by Trevisan et al [15] proves that, under some reasonable assumptions, the number of variables $|\mathbb{X} \cup \mathbb{Y}|$ in an optimal gadget can be assumed to be at most $2^{|\operatorname{Sat}(\phi)|}$. Furthermore, if ψ allows the negations of variables, then this number drops to $2^{|\operatorname{Sat}(\phi)|-1}$.

In the case of a Had_k-to-2Lin(2) gadget, the number of satisfying assignments of Had_k is 2^k, and 2Lin(2) allow the negation of variables. This means that the total number of variables in the gadget is 2^{2^k-1}. Out of these, $2^k - 1$ variables are in $\mathbb{X}$, and $2^{2^k-1} - \left(2^k - 1\right)$ variables are in $\mathbb{Y}$. Furthermore, the "automated gadget" framework gives a natural way to index these variables in terms of $|\operatorname{Sat}(Had_k)|$-long bitstrings. According to the framework, each primary variable should be indexed by a bitstring describing that variable's assignment to all of the satisfying assignments to Had_k, meaning a column in the matrix shown in Figure 4. The auxiliary variables are indexed by the bitstrings that do not appear in the matrix.

Instead of using 2^k-long bitstrings to index the variables, it is arguably more natural to index the variables using functions in $\mathbb{F}_2^k$. These representations are equivalent since every 2^k long bitstring can be interpreted as a truth table of a function in $\mathcal{F}_k$, and vice versa. By indexing the set of variables using functions in $\mathcal{F}_k$, the set of primary variables are indexed by linear functions $\{\chi_\alpha\}_{\varnothing \subset \alpha \subseteq [k]}$, and the negations of linear functions $\{-\chi_\alpha\}_{\varnothing \subset \alpha \subseteq [k]}$. This gives us the following description of a Had_k-to-2Lin(2) gadget.

▶ **Definition 2.18.** *A Had_k-to-2Lin(2) gadget is given by a tuple $(G, \mathbb{X}_k, \mathbb{Y}_k)$, where G is a probability distribution over $\binom{\mathcal{F}_k}{2}$ where $G(f_1, f_2) = 0$ if $f_1 = -f_2$. $\mathbb{X}_k$ is the set of primary variables and $\mathbb{Y}_k$ is the set of auxiliary variables. The set of variables $\mathbb{X}_k \cup \mathbb{Y}_k$ are indexed by functions in $\mathcal{F}_k$, meaning $\mathbb{X}_k \cup \mathbb{Y}_k = \{x_f : f \in \mathcal{F}_k\}$. A variable x_f is a primary variable if and only if f is a linear function or $-f$ is a linear function.*

The reduction from a Had_k *constraint* $\mathrm{Had}_k\left(b_{\{1\}} + y_{\{1\}}, \ldots, b_{[k]} + y_{[k]}\right)$ *to 2Lin(2) is given by the distribution formed by*

1. *Sampling* $(f_1, f_2) \sim G$,
2. *Outputting the constraint* $T(f_1) = T(f_2)$ *where*

$$T(f) = \begin{cases} x_f & \text{if } x_f \in \mathbb{Y}_k, \\ b_\alpha + y_\alpha & \text{if } f = \chi_\alpha \text{ for some } \alpha \in \mathbb{F}_2^k, \\ b_\alpha + y_\alpha + 1 & \text{if } f = -\chi_\alpha \text{ for some } \alpha \in \mathbb{F}_2^k. \end{cases}$$

Let us now precisely define the soundness and completeness of a Had_k-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$. From Theorem 2.16 it follows that the natural definition of soundness is to uniformly at random assign the primary variables $\mathbb{X}_k$ to $\mathbb{F}_2$, and then assign the rest of the variables in order to satisfy as many of the equations as possible.

▶ **Definition 2.19.** *Given a set of Boolean variables* $\mathbb{X}$. *Let* $\mathcal{F}(\mathbb{X})$ *denote the set of assignments* $\mathbb{X} \to \mathbb{F}_2$. *Let* $\mathcal{F}_{\mathrm{fold}}(\mathbb{X})$ *the set of all folded assignments, i.e. functions* $P : \mathbb{X} \to \mathbb{F}_2$ *such that* $P(1 + x) = 1 + P(x) \forall x \in \mathbb{X}$. *Here* $1 + x$ *denotes the negation of the variable* x.

▶ **Definition 2.20.** *The soundness of* G *is defined as*

$$s(G) = \mathop{\mathbb{E}}_{P \in \mathcal{F}_{\mathrm{fold}}(\mathbb{X}_k)} \max_{\substack{A \in \mathcal{F}_{\mathrm{fold}}(\mathbb{X}_k \cup \mathbb{Y}_k), \\ A(x) = P(x), x \in \mathbb{X}_k}} \mathrm{val}(A, G),$$

where

$$\mathrm{val}(A, G) = \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2)[A(x_{f_1}) = A(x_{f_2})].$$

The completeness of G is defined using dictator cuts. A *dictator cut* δ_y of $y \in \mathbb{F}_2^k$ is an assignment where $(-1)^{\delta_y(x_f)} = f(y)$. From Theorem 2.16 we see that that the natural definition for completeness is the expectation over $\mathrm{val}(\delta_y, G)$, where δ_y is a random dictator cut.

▶ **Definition 2.21.** *The completeness of* G *is defined as*

$$c(G) = \mathop{\mathbb{E}}_{y \in \mathbb{F}_2^k} \mathrm{val}(\delta_y, G) = 1 - \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2) \operatorname{dist}(f_1, f_2).$$

There is a result based on Theorem 2.16 that relates the soundness and completeness of Had_k-to-2Lin(2) gadgets to NP-hardness results for Max-2Lin(2).

▶ **Proposition 2.22** ([10, Proposition 2.17])**.** *Given a* Had_k*-to-2Lin(2) gadget* $(G, \mathbb{X}_k, \mathbb{Y}_k)$ *with* $s = s(G)$ *and* $c = c(G)$, *where* $c > s$. *Then for every sufficiently small* $\varepsilon > 0$, *it is NP-hard to distinguish between instances* $\mathcal{I}$ *of Max-2Lin(2) such that*

(Completeness) There exists an assignment that satisfies a fraction at least $c - \varepsilon$ *of the constraints.*

(Soundness) All assignments satisfy at most a fraction $s + \varepsilon$ *of the constraints.*

One particularly interesting case is the inapproximability of Min-2Lin(2)-deletion. From UGC it follows that it is NP-hard to approximate Min-2Lin(2)-deletion within any constant [14]. The following proposition from Håstad et al. [10] tells us that a Had_k-to-2Lin(2) gadget reduction can never be used to show an inapproximability factor of Min-2Lin(2)-deletion better than 2.54. This means that any NP-hardness result for Min-2Lin(2)-deletion shown using a gadget reduction from Had_k-to-2Lin(2) cannot match results obtained by UGC.

▶ **Proposition 2.23** ([10, Proposition 2.29 and Theorem 6.1]). *For any given Had$_k$-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$. There exists a Had$_k$-to-2Lin(2) gadget $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$ with completeness $1 - 2^{-k}$ such that*

$$\frac{1 - s(G)}{1 - c(G)} \leqslant \frac{1 - s(\tilde{G})}{1 - c(\tilde{G})},$$

and

$$\frac{1 - s(\tilde{G})}{1 - c(\tilde{G})} \leqslant \frac{1}{1 - e^{-0.5}} \approx 2.54.$$

▶ Remark 2.24. A Had$_k$-to-2Lin(2) gadget having completeness $1 - 2^{-k}$ implies that the gadget only have positive weight edges of length $2^{-}k$. So far fewer edges are used compared to the total number of possible edges.

▶ Remark 2.25. The upper limit of 2.54 shown by [10] is much more general than what is stated here. In fact, they show that the bound of 2.54 holds for any gadget reduction from a useless predicate ϕ such that Sat(ϕ) has a balanced pairwise independent subset.

3 Relaxed soundness and infinity relaxed soundness

The main difficulty when designing and analysing gadgets is that the soundness is difficult to compute. In the case of a gadget reduction from Max-Had$_k$ to Max-2Lin(2), computing the soundness of the gadget involves solving an instance of Max-2Lin(2). For $k \leqslant 3$ this is computationally feasible, since the Max-2Lin(2) instance is rather small, but for $k \geqslant 4$ the instances can become so large that, even using a computer, it is practically impossible to solve them.

To get around this issue, Wiman [16] proposed to relax the definition of the soundness by not requiring that the assignment A of the auxiliary variables $\mathbb{Y}_k$ is folded. Note that the assignment A is still required to be folded on the primary variables $\mathbb{X}_k$, meaning $A(x_f) = 1 + A(x_{-f}) \forall x_f \in \mathbb{X}_k$. Removing the requirement that A is folded over $\mathbb{Y}_k$ makes it significantly easier to compute the soundness.

▶ **Definition 3.1** ([16, Definition 3.3]). *Wiman's relaxed soundness*

$$\mathrm{rs}(G) = \mathop{\mathbb{E}}_{P \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \{x_1, x_{-1}\})} \max_{\substack{A \in \mathcal{F}(\mathbb{X}_k \cup \mathbb{Y}_k),\\ A(x) = P(x), x \in \mathbb{X}_k \cup \{x_1, x_{-1}\}}} \mathrm{val}(A, G),$$

where

$$\mathrm{val}(A, G) = \sum_{(f_1, f_2) \in \binom{\mathcal{F}_k}{2}} G(f_1, f_2)[A(x_{f_1}) = A(x_{f_2})].$$

This relaxation fundamentally changes the soundness computation from being a Max-2Lin(2) problem to being an s-t Min-Cut problem. This is because the computation of $1 - \mathrm{rs}(G)$ for a fixed P is a minimisation problem where the goal is to minimise the number of times that $A(x_{f_1}) \neq A(x_{f_2})$, which makes it a s-t Min-Cut problem. According to the Max-Flow Min-Cut Theorem, this also means that $\mathrm{rs}(G)$ can be computed by solving a Max-Flow problem.

The conclusion from this is that $1 - \mathrm{rs}(G)$ can be interpreted as the average max flow on the fully connected 2^k-dimensional hypercube, where the placement of sources and sinks is randomly distributed over nodes labeled by affine functions. The sources correspond to primary variables x_f where $P(x_f) = 1$ and the sink nodes correspond to primary variables x_f where $P(x_f) = 0$. The capacity of an edge $\{x_{f_1}, x_{f_2}\}$ in the fully connected hypercube is given by $G(f_1, f_2)$. Note that the sum over capacities in the graph is equals to 1.

There are some significant benefits to using the relaxed soundness. Firstly, it is significantly simpler to solve a Max-Flow problem compared to a Max-2Lin(2) problem. The implication from this is that it is computationally simple to compute the relaxed soundness of Had$_4$-to-2Lin(2) gadgets and even possible to compute relaxed soundness of Had$_5$-to-2Lin(2) gadgets if given enough computational resources. Furthermore, the relaxed soundness allows us to analyse Had$_k$-to-2Lin(2) gadgets even in the case where k is very large. The disadvantage to using relaxed soundness is that it is not guaranteed to be close to the true soundness.

3.1 Relaxed soundness described as an LP

Recall that $1 - \mathrm{rs}(G)$ can be expressed as the average max flow on a fully connected 2^k-dimensional hypercube with randomised source/sink placements. This means that $\mathrm{rs}(G)$ can be stated as an LP. One reason for why it is preferable to express this Max-Flow problem as an LP is because it is possible to move the capacities (i.e. the "gadget variables") of the Max-Flow problem to the variable side of the LP. So the same LP can be used both to calculate the the relaxed soundness of a specific gadget and to construct new gadgets.

One additional step we use in the formulation of this LP is to use a function $g \in \mathcal{F}_k$ to describe the source/sink placement instead of using the assignment P. A node $v_{\chi_\alpha}, \alpha \in \mathbb{F}_2^k$ is a sink node if $g(\alpha) = 1$, and a source node if $g(\alpha) = -1$. These two representations of the source/sink placement are equivalent, but using a Boolean function g is more helpful for understanding the symmetries of the LP, as done in Appendix C. The following is the LP reformulation of the relaxed soundness $\mathrm{rs}(G)$.

▶ **Definition 3.2.** *A flow w of a Had$_k$-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$ is a function $\mathcal{F}_k^3 \to \mathbb{R}_{\geqslant 0}$. The flow w is said to be feasible if and only if*

$$w(f_1, f_2, g) + w(f_2, f_1, g) \leqslant G(f_1, f_2) \quad \forall f_1, f_2, g \in \mathcal{F}_k, \tag{1}$$

$$\mathrm{out}_w(f, g) = \mathrm{in}_w(f, g) \quad \forall f, g \in \mathcal{F}_k, \dim(f) \geqslant 1. \tag{2}$$

where $\mathrm{out}_w(f, g) = \sum_{f_2 \in \mathcal{F}_k} w(f, f_2, g)$ *and* $\mathrm{in}_w(f, g) = \sum_{f_2 \in \mathcal{F}_k} w(f_2, f, g)$*. The value of w for at a source/sink placement $g \in \mathcal{F}_k$ is defined as*

$$\mathrm{val}_g(w) = \sum_{\alpha \in \mathbb{F}_2^k} \mathrm{out}_w(g(\alpha)\chi_\alpha, g) - \mathrm{in}_w(g(\alpha)\chi_\alpha, g).$$

▶ **Definition 3.3.** *The relaxed soundness LP for a Had$_k$-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$, denoted by* $\mathrm{rsLP}(G)$*, is the following LP*

$$\mathrm{rs}(G) = 1 - \max_w \mathbb{E}_{g \in \mathcal{F}_k} \mathrm{val}_g(w),$$

where the maximum is taken over feasible flows w of G.

▶ Remark 3.4. Recall that $1 - \mathrm{rs}(G)$ is the average of 2^{2^k} independent Max-Flow problems. The different Max-Flow problems are indexed by the function $g \in \mathcal{F}_k$, which describes the placements of sinks and sources. The nodes in each Max-Flow problem are indexed by functions in $\mathcal{F}_k$. The sink nodes in the g-th Max-Flow problem are the nodes $v_{g(\alpha)\chi_\alpha}, \alpha \in \mathbb{F}_2^k$, and the source nodes are the nodes $v_{-g(\alpha) \cdot \chi_\alpha}, \alpha \in \mathbb{F}_2^k$. The flow from $v_{f_1} \to v_{f_2}$ is $w(f_1, f_2, g)$, and the capacity of the undirected edge $\{v_{f_1}, v_{f_2}\}$ is $G(f_1, f_2)$.

▶ Remark 3.5. Note that it is possible to modify the rsLP(G) to include the capacities of the graph (i.e. gadget G) as variables. The implications of this is that the optimisation problem of finding a Had_k-to-2Lin(2) gadget with the maximum relaxed soundness for a fixed completeness can also be expressed as an LP.

▶ Remark 3.6. Note that the rsLP(G) has roughly $|\mathcal{F}_k|^3 = 2^{3\cdot 2^k}$ variables. This is a very large number, even for small values of k. So in order to be able to solve this LP, we have to make use of the symmetries of the LP in order to reduce the number of variables.

3.2 Introduction of infinity relaxed soundness

One natural question is, how small can one make the relaxed soundness if we fix the completeness of a Had_k-to-2Lin(2) gadget and let $k \to \infty$? In practice, even just finding the gadget minimising the relaxed soundness when $k = 5$ is a very daunting task, so cannot hope to calculate this limit directly from the rsLP(G).

Our method to handle large values of k is to create a Had_k-to-2Lin(2) gadget for some small value of k, for example $k = 4$, and then introduce the concept of embedding a Had_k-to-2Lin(2) gadget G into a $\text{Had}_{k'}$-to-2Lin(2) gadget G', where $k' > k$. It is also possible to embed the flow of the rsLP(G) onto the rsLP(G'). This embedding has the property that the completeness and the soundness of both gadgets are the same.

The key insight is that by using multiple overlapping embeddings of G, we can improve the soundness of G' without affecting its completeness. Our argument for why multiple overlapping embeddings improve the relaxed soundness is based on leaky flows. Note that the leaks of a leaky flow have signs. This means that it is possible that overlapping embeddings of leaky flows could become a feasible flow, since the overlap of the embeddings could cause the signed leaks to sum to 0. We use this type of argument to show an upper bound on rs(G') based on a leaky flow solution to the rsLP(G).

The exact procedure for the embeddings is defined in Appendix C and analysed in detail in Appendix D using second moment analysis. The conclusion from that analysis is that the following relaxation of the rsLP(G), which we call the *infinity relaxed soundness LP*, denoted by $\text{rs}_\infty\text{LP}(G)$, has the following two important properties. Firstly, the solution of $\text{rs}_\infty\text{LP}(G)$ is a leaky flow of rsLP(G), and secondly, overlapping embeddings of this leaky flow tends to a feasible flow of rsLP(G') as $k' \to \infty$.

▶ **Definition 3.7.** *A flow $\tilde{w}$ of a Had_k-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$ is said to be a infinity relaxed flow if constraint* (1) *is satisfied and*

$$\sum_{g'} \text{out}_w(f, g') = \sum_{g'} \text{in}_w(f, g') \quad \forall g, f \in \mathcal{F}_k : \dim(f) \geqslant 1, \tag{3}$$

where the sums are over functions $g' \in \mathcal{F}'_k$ such that $g'|_{\text{affine}(f)} = g|_{\text{affine}(f)}$. The (signed) leak at (f, g), where $f, g \in \mathcal{F}_k, \dim(f) \geqslant 1$, is defined as $\text{leak}_{\tilde{w}}(f, g) = \text{in}_w(f, g) - \text{out}_w(f, g)$.

▶ **Definition 3.8.** *The infinity relaxed soundness of G, denoted by $\text{rs}_\infty(G)$, is the solution to the* $\text{rs}_\infty\text{LP}(G)$

$$\text{rs}_\infty(G) = 1 - \max_{\tilde{w}} \mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(\tilde{w}),$$

where the maximum is taken over all infinity relaxed flows $\tilde{w}$ of G.

▶ Remark 3.9. The $\text{rs}_\infty\text{LP}(G)$ is a constraint relaxation of the rsLP(G) where constraint (2) has been relaxed to constraint (3). So a solution of the $\text{rs}_\infty\text{LP}(G)$ is a leaky flow in the rsLP(G).

▶ Remark 3.10. Constraint (3) is used for a proof in Appendix D of Lemma D.5, which is a 2nd order moment analysis of the overlap of leaks from random embeddings. In the proof, constraint (3) is used to show that if w is an infinity relaxed flow then $\forall g, f \in \mathcal{F}_k, \dim(f) \geqslant 1$: $\sum_{g'} \text{leak}_w(f, g') = 0$, where the sum is over $g' \in \mathcal{F}_k$ such that $g'|_{\text{affine}(f)} = g|_{\text{affine}(f)}$.

The following lemma describes a relationship between the rsLP(G) and the rs$_\infty$LP(G). This is the key Lemma, which is proven in Appendix D.

▶ **Lemma 3.11.** *Let $(G, \mathbb{X}_k, \mathbb{Y}_k)$ be a Had$_k$-to-2Lin(2) gadget. For any $\varepsilon > 0$ there exists a Had$_{k'}$-to-2Lin(2) gadget $(G', \mathbb{X}_{k'}, \mathbb{Y}_{k'})$ for some $k' > k$ such that $c(G) = c(G')$ and $\text{rs}(G') \leqslant \text{rs}_\infty(G) + \varepsilon$.*

From this Lemma, it follows that $\text{rs}_\infty(G) + \varepsilon$ is the upper bound of $\text{rs}(G')$ for some gadget G', which in turn is an upper bound of $s(G')$. This means that the NP-hardness result of Max-2Lin(2) stated in Proposition 2.22 for rs(G) also holds for $\text{rs}_\infty(G)$.This gives us our main result.

▶ **Theorem 3.12.** *Let $(G, \mathbb{X}_k, \mathbb{Y}_k)$ be a Had$_k$-to-2Lin(2) gadget with $s = \text{rs}_\infty(G)$ and $c = c(G)$, where $c > s$. Then for every sufficiently small $\varepsilon > 0$, it is NP-hard to distinguish between instances of Max-2Lin(2) such that*

Completeness *There exists an assignment that satisfies a fraction at least $c - \varepsilon$ of the constraints.*

Soundness *All assignments satisfy at most a fraction $s + \varepsilon$ of the constraints.*

4 Numerical results

This section contains a presentation of constructed Had$_k$-to-2Lin(2) gadgets. Recall that there are three different ways to measure the soundness of a Had$_k$-to-2Lin(2) gadget. There is the true soundness of a gadget, which can be used to show NP-hardness results for Max-2Lin(2), see Proposition 2.22. Then there is the relaxed soundness, denoted by rs. This is an upper bound of the true soundness, see Proposition B.1. Finally there is the infinity relaxed soundness, denoted by rs$_\infty$, which according to our main result, Theorem 3.12, also imply NP-hardness results for Max-2Lin(2).

We compute gadgets for $k = 2, 3, 4$, optimised either for rs or rs$_\infty$. The short rundown of the process of constructing a gadget is to first decide on the completeness of the gadget, and then call an LP-solver to find the gadget with that completeness that either minimises rs or rs$_\infty$, depending on which measure of soundness we want to optimise the gadget for.

4.1 Edges used/unused in constructed gadgets

The capacity G of a Had$_k$-to-2Lin(2) gadget $(G, \mathbb{X}_k, \mathbb{Y}_k)$ is a probability distribution over (undirected) edges. Every gadget that we construct is symmetrical under the mappings of $\mathcal{M}_{k \to k}$, so edges from the same edge orbit share the same capacity. Tables 7–9 in Appendix F list all edge orbits that have non-zero weight in at least one of our constructed gadgets for $k = 2, 3, 4$. Note that as discussed in Appendix E.2.1, in the case of $k = 4$ it is possible that the gadgets we construct are sub-optimal if $c < 1 - 2^{-k}$. This means that it is possible that the Table for $k = 4$, Table 9, could look slightly different had we constructed optimal gadgets.

Table 1 The curve $s(c)$ as shown in Figure 1. The values of $s(c)$ in this table are rounded up to 4 decimals. This table has the same format as the table describing the $\text{Gap}_{\text{SDP}}(c)$ curve, found in Appendix E of [14].

c	$s(c)$	c	$s(c)$	c	$s(c)$	c	$s(c)$	c	$s(c)$
0.500	0.5000	0.600	0.5750	0.700	0.6500	0.800	0.7343	0.900	0.8516
0.505	0.5038	0.605	0.5788	0.705	0.6538	0.805	0.7390	0.905	0.8586
0.510	0.5075	0.610	0.5825	0.710	0.6575	0.810	0.7437	0.910	0.8661
0.515	0.5113	0.615	0.5863	0.715	0.6613	0.815	0.7485	0.915	0.8735
0.520	0.5150	0.620	0.5900	0.720	0.6650	0.820	0.7535	0.920	0.8809
0.525	0.5188	0.625	0.5938	0.725	0.6688	0.825	0.7588	0.925	0.8884
0.530	0.5225	0.630	0.5975	0.730	0.6725	0.830	0.7642	0.930	0.8958
0.535	0.5263	0.635	0.6013	0.735	0.6763	0.835	0.7696	0.935	0.9032
0.540	0.5300	0.640	0.6050	0.740	0.6800	0.840	0.7752	0.940	0.9107
0.545	0.5338	0.645	0.6088	0.745	0.6838	0.845	0.7809	0.945	0.9181
0.550	0.5375	0.650	0.6125	0.750	0.6875	0.850	0.7868	0.950	0.9256
0.555	0.5413	0.655	0.6163	0.755	0.6922	0.855	0.7927	0.955	0.9330
0.560	0.5450	0.660	0.6200	0.760	0.6969	0.860	0.7988	0.960	0.9405
0.565	0.5488	0.665	0.6238	0.765	0.7016	0.865	0.8050	0.965	0.9479
0.570	0.5525	0.670	0.6275	0.770	0.7063	0.870	0.8115	0.970	0.9554
0.575	0.5563	0.675	0.6313	0.775	0.7109	0.875	0.8181	0.975	0.9628
0.580	0.5600	0.680	0.6350	0.780	0.7156	0.880	0.8247	0.980	0.9703
0.585	0.5638	0.685	0.6388	0.785	0.7203	0.885	0.8313	0.985	0.9777
0.590	0.5675	0.690	0.6425	0.790	0.7250	0.890	0.8380	0.990	0.9852
0.595	0.5713	0.695	0.6463	0.795	0.7297	0.895	0.8448	0.995	0.9926

4.2 Lists and plots of gadgets

Figures 5, 6 and 7 show Had_k-to-2Lin(2) gadgets with completeness on the x-axis, and either maximal $\frac{1-\text{rs}(G)}{1-c(G)}$ or maximal $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ on the y-axis. To create this plot, we construct one gadget for each completeness value from 0.5 to $1-2^{-k}$ (inclusive), with a spacing of 2^{-9}. The curve is constructed using interpolation by taking convex combinations of pairs of neighbouring gadgets.

4.2.1 The curve $s(c)$

The curve $s(c)$ describes the infinity relaxed soundness of Had_4-to-2Lin(2) gadgets as a function of completeness, shown as the upper curve in Figure 7, as well as in Figures 1, 2 and 3. The data for this curve can be found in Table 1. It has the following formal definition.

▶ **Definition 4.1.** *The curve* $s(c) : [0.5, 1] \to [0.5, 1]$*,* $k = 4$*, is for* $c \in [0.5, 1-2^{-k}]$ *defined as the solution to the restricted compressed* $\text{rs}_\infty\text{LP}$*. For* $c \geqslant 1-2^{-k}$ *the curve is defined as* $s(c) = 1 + 2^k(s(1-2^{-k}) - 1)(1-c)$*, meaning* $\frac{1-s(c)}{1-c}$ *is constant for all* $c \geqslant 1-2^{-k}$*.*

Proof of Theorem 1.1. For $c \in [0.5, 1-2^{-k}]$, the NP-hardness result follows directly from Theorem 3.12 since the solution of the restricted compressed $\text{rs}_\infty\text{LP}(G)$ is an upper bound of the (non-restricted) $\text{rs}_\infty\text{LP}(G)$. For $c \geqslant 1-2^{-k}$ the NP-hardness result follows from taking the convex combination of $(c, s) = (1-2^{-k}, s(1-2^{-k}))$ and $(c, s) = (1, 1)$. Since it is possible to create a hard instance by taking the convex combination of two hard instances using separate variables. ◀

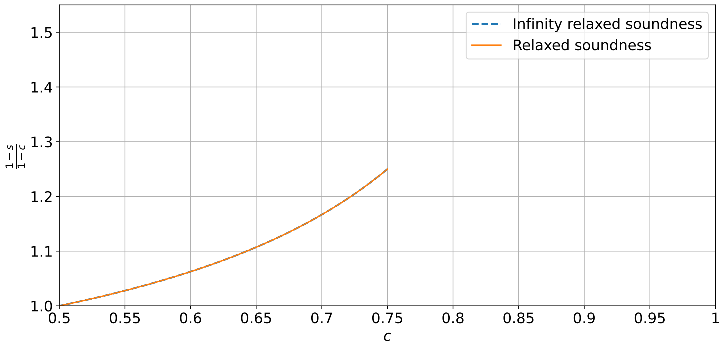

Figure 5 This plot shows two types of Had_2-to-2Lin(2) gadgets. The filled curve describes the minimisation of rs and the striped curve describes the minimisation of rs_∞. The completeness value is on the x-axis, and either $\frac{1-\mathrm{rs}(G)}{1-c(G)}$ or $\frac{1-\mathrm{rs}_\infty(G)}{1-c(G)}$ on the y-axis. In this particular case, the case of $k=2$, it turns out that these two curves are identical.

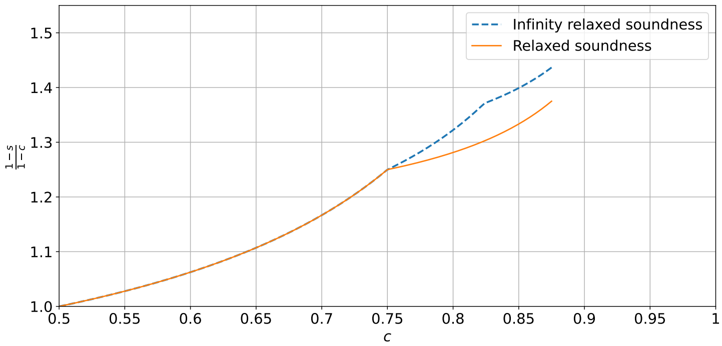

Figure 6 This plot shows two types of Had_3-to-2Lin(2) gadgets. The filled curve describes the minimisation of rs and the striped curve describes the minimisation of rs_∞. The completeness value is on the x-axis, and either $\frac{1-\mathrm{rs}(G)}{1-c(G)}$ or $\frac{1-\mathrm{rs}_\infty(G)}{1-c(G)}$ on the y-axis.

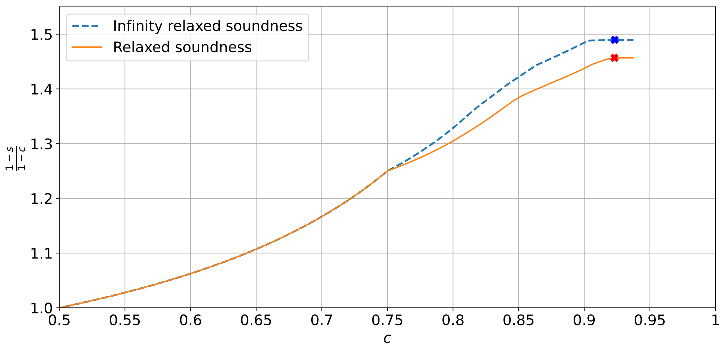

Figure 7 This plot shows two types of Had_4-to-2Lin(2) gadgets. The filled curve describes the minimisation of rs and the striped curve describes the minimisation of rs_∞. The completeness value is on the x-axis, and either $\frac{1-\mathrm{rs}(G)}{1-c(G)}$ or $\frac{1-\mathrm{rs}_\infty(G)}{1-c(G)}$ on the y-axis. The top part of both of these curves are perfectly flat, which is not the case in Figure 5 and Figure 6. The gadgets that mark the point where the curves become flat can be found in Tables 2 and 3, and are marked by crosses in the plot.

Table 2 The Had_4-to-2Lin(2) gadget G with minimal completeness among those that minimise $\frac{1-\text{rs}(G)}{1-c(G)}$. The completeness of G is $c(G) = 9939/10768$ and relaxed soundness is $\text{rs}(G) = 2623643487/2955083776$. The right most column tells how much of the total capacity is contained in each edge orbit. This column sums up to 100%.

f_1	f_2	length	$G(f_1, f_2)$	% of total
1100000000000000	1110000000000000	1	5461/969636864	30.3
1110000000000000	1111000000000000	1	17007/1616061440	18.9
1110000000000000	1110100000000000	1	437/404015360	23.2
1110100000000000	1110100010000000	1	19/92346368	4.4
0000000000000000	1100000000000000	2	13/215360	23.2

Table 3 The Had_4-to-2Lin(2) gadget G with minimal completeness among those that minimise $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$. The completeness of G is $c(G) = 590174949/639271832$ and the infinity relaxed soundness is $\text{rs}_\infty(G) = 141533171/159817958$. The right most column tells how much of the total capacity is contained in each edge orbit. This column sums up to 100%.

f_1	f_2	length	$G(f_1, f_2)$	% of total
1100000000000000	1110000000000000	1	4899/799089790	33.0
1110000000000000	1111000000000000	1	11843/799089790	26.5
1110000000000000	1110100000000000	1	1427/1917815496	16.0
1110100000000000	1110100010000000	1	1427/19178154960	1.60
0000000000000000	1100000000000000	2	6094929/102283493120	22.9

4.3 Notable gadgets

There are two gadgets that are of particular interest. These are the gadgets with minimal completeness among those that maximises either $\frac{1-\text{rs}(G)}{1-c(G)}$ or $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$. These gadgets are marked by crosses in Figure 7. The gadget with minimal completeness that maximises $\frac{1-\text{rs}(G)}{1-c(G)}$ can be found in Table 2. The gadget with minimal completeness that maximises $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ can be found in Table 3, and is also marked by a cross on the curve $s(c)$ in Figures 1-3. The method used to construct such minimal completeness gadgets is slightly different compared to the construction of gadgets with fixed completeness. Propositions 2.23 and B.1 guarantees that gadgets with completeness $1 - 2^{-k}$ can be used to maximise $\frac{1-\text{rs}(G)}{1-c(G)}$ and $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$. This means that the maximum values of $\frac{1-\text{rs}(G)}{1-c(G)}$ and $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$ can be computed by fixing the completeness to $c(G) = 1 - 2^{-k}$. Using these maximums, it is possible to slightly modify the objective of the LP such that its solution is the gadget with minimal completeness that maximises either $\frac{1-\text{rs}(G)}{1-c(G)}$ or $\frac{1-\text{rs}_\infty(G)}{1-c(G)}$.

5 Conclusions

In this work, we have introduced a procedure called lifting for taking a Had_k-to-2Lin(2) gadget for a fixed k and using that gadget to construct better and better $\text{Had}_{k'}$-to-2Lin(2) gadgets, as k' tends to infinity. In order to be able to analyse this, both numerically and analytically, we made use of a relaxation of the (true) soundness, first introduced by Wiman [16] in their analysis of the Had_4-to-2Lin(2) gadget. This procedure allowed us to show new inapproximability results of Max-2Lin(2), and most notably using $k = 4$, we have shown that Min-2Lin(2)-deletion has an inapproximability factor of $\frac{73139148}{49096883} \approx 1.48969$.

Some open problems still remain. The most obvious one is that it is likely within reach to carry out the analysis we did for $k = 4$ also for $k = 5$. The main bottleneck is to find or write a very efficient LP solver that is able to handle large instances and give consistent and stable results. The solvers available to us were not quite able to get trustworthy results. This being said, without substantial new ideas we do not see how to attack the $k = 6$ case.

Another open problem is to understand the best possible gadget reduction from Had_k-to-2Lin(2) as $k \to \infty$. More specifically, which is the best possible inapproximability factor of Min-2Lin(2)-deletion attainable using such a gadget reduction? We were able to show an inapproximability factor of $\frac{73139148}{49096883} \approx 1.48969$ using relaxed soundness. We have also shown that by using relaxed soundness, it is impossible to go above 2 (see Proposition B.1). Furthermore, it is known from a previous work [10, Theorem 6.1] that by using (non-relaxed) soundness, $\frac{1}{1-e^{-0.5}} \approx 2.54$ is an upper bound. This leaves us with a fairly large gap. So it would be of interest to close this gap.

In comparison, by assuming the Unique Games Conjecture (UGC), it is possible to show that the inapproximability factor of Min-2Lin(2)-deletion can be made arbitrarily large. The main open problem here is to show this without assuming UGC. This however, is not possible to do using a gadget reduction from Had_k-to-2Lin(2), and would instead require a completely new approach.

Finally, as a concluding remark, it would be interesting to see if our ideas of lifting small gadgets and analysing them using a relaxed version of the (true) soundness, could be used in other applications. Maybe there are other gadgets out there that could be improved using a similar procedure?

References

1 David Applegate, William Cook, Sanjeeb Dash, and Daniel Espinoza. Qsopt_ex rational lp solver. `https://www.math.uwaterloo.ca/~bico/qsopt/ex/`. Accessed: 2023-09-20.

2 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, volume 62, pages 563–572, October 2010. `doi:10.1109/FOCS.2010.59`.

3 Per Austrin and Johan Håstad. On the usefulness of predicates. In *2012 IEEE 27th Conference on Computational Complexity*, pages 53–63, 2012. `doi:10.1109/CCC.2012.18`.

4 Per Austrin and Elchanan Mossel. Approximation resistant predicates from pairwise independence. *CoRR*, abs/0802.2300, 2008. `arXiv:0802.2300`.

5 Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. `doi:10.1137/S0097539796302531`.

6 Siu On Chan. Approximation resistance from pairwise independent subgroups. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 447–456, New York, NY, USA, 2013. Association for Computing Machinery. `doi:10.1145/2488608.2488665`.

7 Uriel Feige and Michael Langberg. The rpr2 rounding technique for semidefinite programs. *Journal of Algorithms*, 60(1):1–23, 2006. `doi:10.1016/j.jalgor.2004.11.003`.

8 Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, November 1995. `doi:10.1145/227683.227684`.

9 Johan Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 1–10, New York, NY, USA, 1997. Association for Computing Machinery. `doi:10.1145/258533.258536`.

10 Johan Håstad, Sangxia Huang, Rajsekar Manokaran, Ryan O'Donnell, and John Wright. Improved NP-Inapproximability for 2-Variable Linear Equations. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 341–360, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.341.

11 S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 146–154, 2004. doi:10.1109/FOCS.2004.49.

12 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/509907.510017.

13 Subhash Khot, Dor Minzer, and Muli Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601, October 2018. doi:10.1109/FOCS.2018.00062.

14 Ryan O'Donnell and Yi Wu. An optimal sdp algorithm for max-cut, and equally optimal long code tests. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 335–344, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374425.

15 L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 617–626, 1996. doi:10.1109/SFCS.1996.548521.

16 Mårten Wiman. Improved inapproximability of max-cut through min-cut. Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2018.

A Max-Flow and symmetries

This section introduces the concepts of feasible flows and leaky flows, and show how to make use symmetries in a graph to more efficiently solve the Max-Flow problem. These Max-Flow techniques and concepts are used during the construction of gadgets. These techniques are very general, and become easier to explain without involving the intricacies of gadgets. Let us start by defining the Max-Flow problem as an LP.

▶ **Definition A.1.** *A flow graph is a tuple* $G = (V, C, S, T)$*, where* $C(u, v) = C(v, u) \geqslant 0$ *is the capacity of edge* $(u, v) \in V \times V$*, and* $S \subset V$ *is a set of sources and* $T \subset V$ *is a set of sinks, and* $S \cap T = \varnothing$*.*

▶ **Definition A.2.** *A flow* w *of a flow graph* $G = (V, C, S, T)$ *is a function* $V \times V \to \mathbb{R}_{\geqslant 0}$*. The flow* w *is said to be feasible if and only if*

$$w(v,u) + w(u,v) \leqslant C(u,v) \quad \forall v,u \in V, \tag{4}$$

$$\text{out}_w(v) = \text{in}_w(v) \quad \forall v \in V \setminus (S \cup T). \tag{5}$$

where

$$\text{out}_w(v) = \sum_{u \in V} w(v,u),$$

$$\text{in}_w(v) = \sum_{u \in V} w(u,v).$$

The value of a flow is defined as

$$\text{val}(w) = \sum_{s \in S} \text{out}_w(s) - \text{in}_w(s).$$

The value of the maximum flow of a flow graph G is denoted by $\text{max\_flow}(G)$.

A.1 Feasible flows and leaky flows

When solving a Max-Flow problem we normally require the flow to be conserved (constraint (5) above), meaning that the incoming flow into a node is equal to the outgoing flow. This is the definition of a feasible flow. However, to find an approximate solution to a Max-Flow problem, it can be helpful to relax the conservation of flows constraint, allowing for "leaks". A flow that does not fulfil the conservation of flow constraint is called a *leaky flow*. This section aims to analyse the relation between leaky flows and feasible flows, with the goal of showing that if the leaks of a leaky flow are small, then there is a feasible flow with almost the same value as the leaky flow.

▶ **Definition A.3.** *A flow $\tilde{w}$ is said to be a leaky flow if constraint (4) is satisfied. The (signed) leak at node v be defined as $\text{leak}_{\tilde{w}}(v) = \text{in}_{\tilde{w}}(v) - \text{out}_{\tilde{w}}(v)$ for $v \in V \setminus (S \cup T)$.*

▶ Remark A.4. Note that a leaky flow $\tilde{w}$ is also a feasible flow if and only if $\text{leak}_{\tilde{w}}(v) = 0$ for all $v \in V \setminus (S \cup T)$.

The following theorem tells us that if the sum of absolute values of the leaks are small, then there is a feasible flow having almost the same value as the leaky flow. The implications from this is that we can use leaky flows to get an approximation of the true Max-Flow.

▶ **Theorem A.5.** *Given a leaky flow $\tilde{w}$ of a flow graph $G = (V, C, S, T)$, there exists a feasible flow w of G such that*

$$\text{val}(w) \geqslant \text{val}(\tilde{w}) - \sum_{v \in V \setminus (S \cup T)} |\,\text{leak}_{\tilde{w}}(v)|.$$

Proof. Create a new graph $\tilde{G} = (V \cup \{\tilde{s}, \tilde{t}\}, \tilde{C}, S \cup \{\tilde{s}\}, T \cup \{\tilde{t}\})$ with an additional new source node $\tilde{s}$ and sink node $\tilde{t}$. We construct $\tilde{C}$ using C. Firstly let $\tilde{C}(u, v) = C(u, v)$ for all nodes $u, v \in V$. Secondly, for every $v \in V \setminus (S \cup T)$ such that $\text{leak}_{w'}(v) > 0$, let $\tilde{C}(u, \tilde{t}) = \text{leak}_{w'}(v)$, and for every $v \in V \setminus (S \cup T)$ such that $\text{leak}_{w'}(v) < 0$ let $\tilde{C}(u, \tilde{s}) = -\,\text{leak}_{\tilde{w}}(v)$. Finally let $\tilde{C}$ be 0 in all other cases.

Note that for this new graph $\tilde{G}$, the leaky flow $\tilde{w}$ can be extended into a feasible flow since all of the leaks can be routed to either $\tilde{s}$ or $\tilde{t}$ depending on the sign of the leakage. Furthermore, if we can show that

$$\text{max\_flow}(\tilde{G}) \leqslant \text{max\_flow}(G) + \sum_{v \in V \setminus (S \cup T)} |\,\text{leak}_{\tilde{w}}(v)|, \tag{6}$$

then that would imply the the Theorem.

To show (6) we use the Max-Flow Min-Cut Theorem. Note that any S-T cut in $\tilde{G}$ has a corresponding S-T cut in G and vice versa since G and $\tilde{G}$ share the same non-source/sink nodes. Additionally, note that the value of a S-T cut in $\tilde{G}$ can be bounded from above by the value of the corresponding cut in G plus the extra capacities in $\tilde{G}$. The conclusion from this is that

$$\begin{aligned}\text{max\_flow}(\tilde{G}) &= \text{min\_cut}(\tilde{G}) \\ &\leqslant \text{min\_cut}(G) + \sum_{v \in V \setminus (S \cup T)} |\,\text{leak}_{\tilde{w}}(v)| \\ &= \text{max\_flow}(G) + \sum_{v \in V \setminus (S \cup T)} |\,\text{leak}_{\tilde{w}}(v)|.\end{aligned}$$ ◀

A.2 Symmetries of Max-Flow graphs

If a flow graph $G = (V, C, S, T)$ has some kind of symmetry, then we can use them to more efficiently solve the Max-Flow problem. In our setting, the symmetries are described by a group H acting on V with the property that the capacities are invariant under the group action, meaning $C(u, v) = C(h \cdot u, h \cdot v)$ for all $h \in H$ and $u, v \in V$. Here $h \cdot u$ denotes the group action of h on u.

▶ **Definition A.6.** *Given a flow graph $G = (V, C, S, T)$ and a group H acting on V, then H is said to be a symmetry group of G if and only if $\forall h \in H$:*

1. $h \cdot s \in S \, \forall s \in S$,
2. $h \cdot t \in T \, \forall t \in T$,
3. $C(u, v) = C(h \cdot u, h \cdot v) \forall h \in H$ *and* $\forall u, v \in V$.

Using G and the group H acting on V, we can create a new flow graph where the set of vertices is the quotient space V/H. This "compresses" the graph G into one vertex per orbit. Let the capacities between two orbits $A, B \in V/H$ be the sum capacities over all pairs in $A \times B$.

▶ **Definition A.7.** *Given a flow graph $G = (V, C, S, T)$and a symmetry group H of G. Let the quotient flow graph $G/H = (V/H, C/H, S/H, T/H)$ where V/H is the set of all orbits of V under the action of H, and similarly S/H is the set of orbits of S and T/H is the set of orbits of T. Let C/H be defined as a function $V/H \times V/H \to \mathbb{R}$ such that*

$$(C/H)(A, B) = \sum_{u \in A} \sum_{v \in B} C(u, v)$$

for all $A, B \in V/H$.

What remains to show is that the original graph G and the compressed graph G/H has the same Max-Flow.

▶ **Theorem A.8.** *Given a flow graph $G = (V, C, S, T)$and a symmetry group H of G. Then* $\text{max\_flow}(G) = \text{max\_flow}(G/H)$.

Proof. First let us show that $\text{max\_flow}(G) \leqslant \text{max\_flow}(G/H)$. Let w be the max-flow of G. Now define w/H as a function from $V/H \times V/H \to \mathbb{R}$ such that

$$(w/H)(A, B) = \sum_{a \in A} \sum_{b \in B} w(a, b).$$

What remains to show is that that w/H is a feasible flow of G/H and that $\text{val}(w) = \text{val}(w/G)$ since those two properties would imply that $\text{max\_flow}(G) \leqslant \text{max\_flow}(G/H)$. Firstly, note that w/H fulfills (4) and (5) from Definition A.2 for the graph G/H since the constraints are linear. For example take constraint (4),

$$\begin{aligned}(w/H)(A, B) + (w/H)(B, A) &= \sum_{a \in A} \sum_{b \in B} w(a, b) + w(b, a) \\ &\leqslant \sum_{a \in A} \sum_{b \in B} C(a, b) \\ &= (C/H)(A, B).\end{aligned}$$

So w/H is a feasible flow of G/H. Secondly note that the value of w is the same as the value of w/H since

$$\begin{aligned}\operatorname{val}(w/H) &= \sum_{A\in S/H} \operatorname{out}_{w/H}(A) - \operatorname{in}_{w/H}(A)\\ &= \sum_{A\in S/H}\sum_{s\in A} \operatorname{out}_w(s) - \operatorname{in}_w(s)\\ &= \sum_{s\in S} \operatorname{out}_w(s) - \operatorname{in}_w(s)\\ &= \operatorname{val}(w).\end{aligned}$$

It remains to show that max_flow$(G) \geqslant$ max_flow(G/H). Let w' be a max-flow of G/H. Now define $w : V \times V \to \mathbb{R}$ such that

$$w(a, b) = w'(H\cdot a, H\cdot b)\frac{C(a, b)}{(C/H)(H\cdot a, H\cdot b)}$$

where $a, b \in V$ and $H\cdot a$ is the orbit of a and $H\cdot b$ is the orbit of b. What remains to show is that $w(a, b)$ is a feasible flow of G and that the value of w is the same as the value of w'. Firstly, note that w/H fulfill constraints (4) and (5) from Definition A.2 for the graph G since the constraints are linear. For example take constraint (4),

$$\begin{aligned}w(a, b) + w(b, a) &= (w'(H\cdot a, H\cdot b) + w'(H\cdot b, H\cdot a))\frac{C(a, b)}{(C/H)(H\cdot a, H\cdot b)}\\ &\leqslant (C/H)(H\cdot a, H\cdot b)\frac{C(a, b)}{(C/H)(H\cdot a, H\cdot b)}\\ &= C(a, b).\end{aligned}$$

Secondly note that the value of w is the same as w' since

$$\begin{aligned}\operatorname{val}(w') &= \sum_{A\in S/H} \operatorname{out}_{w'}(A) - \operatorname{in}_{w'}(A)\\ &= \sum_{A\in S/H}\sum_{B\in V/H} w'(A, B) - w'(B, A)\\ &= \sum_{A\in S/H}\sum_{B\in V/H} (w'(A, B) - w'(B, A))\left(\sum_{a\in A}\sum_{b\in B}\frac{C(a, b)}{(C/H)(A, B)}\right)\\ &= \sum_{A\in S/H}\sum_{B\in V/H}\sum_{a\in A}\sum_{b\in B}(w'(A, B) - w'(B, A))\frac{C(a, b)}{(C/H)(A, B)}\\ &= \sum_{A\in S/H}\sum_{B\in V/H}\sum_{a\in A}\sum_{b\in B} w(a, b) - w(b, a)\\ &= \sum_{a\in S}\sum_{b\in V} w(a, b) - w(b, a)\\ &= \sum_{a\in S} \operatorname{out}_w(a) - \operatorname{in}_w(a)\\ &= \operatorname{val}(w).\end{aligned}$$

So w is a feasible flow of G and $\operatorname{val}(w) = \operatorname{val}(w')$, so max_flow$(G) \geqslant$ max_flow(G/H). ◀

B Properties of relaxed soundness

The relaxed soundness share many similarities with the (true) soundness. One example is the following Proposition, which is an analogue to Proposition 2.23 but for relaxed soundness.

▶ **Proposition B.1.** *For any Had$_k$-to-2Lin(2) gadget* $(G, \mathbb{X}_k, \mathbb{Y}_k)$

(a)

$$s(G) \leqslant \mathrm{rs}(G).$$

(b) *There exists a Had$_k$-to-2Lin(2) gadget* $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$ *with completeness* $1 - 2^{-k}$ *such that*

$$\frac{1-\mathrm{rs}(G)}{1-c(G)} \leqslant \frac{1-\mathrm{rs}(\tilde{G})}{1-c(\tilde{G})},$$

(c) *and for any Had$_k$-to-2Lin(2) gadget* $(\tilde{G}, \mathbb{X}_k, \mathbb{Y}_k)$ *with completeness* $1 - 2^{-k}$

$$\frac{1-\mathrm{rs}(\tilde{G})}{1-c(\tilde{G})} \leqslant 2.$$

Proof.

(a) Note that interpreting x_1 and x_{-1} as being primary variables do not affect soundness, i.e.

$$s(G) = \mathop{\mathbb{E}}_{P \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \{x_1, x_{-1}\})} \max_{\substack{A \in \mathcal{F}_{\text{fold}}(\mathbb{X}_k \cup \mathbb{Y}_k), \\ A(x) = P(x), x \in \mathbb{X}_k \cup \{x_1, x_{-1}\}}} \mathrm{val}(A, G). \tag{7}$$

The reason for this is that there exists a degree of freedom in the choice of A since for any A, $\mathrm{val}(A, G) = \mathrm{val}(1 + A, G)$. This means for example that we can add one extra constraint like $A(x_1) = 1 + A(x_{-1}) = 1$ to the definition of $s(G)$ without affecting its value.
Comparing (7) and the definition of relaxed soundness, we can clearly see that $s(G) \leqslant \mathrm{rs}(G)$ since the relaxed soundness is a less constrained maximisation problem compared to the right hand side of (7).

(b) This proof is analogous to the proof of [10, Proposition 2.29]. Note that by definition $1 - c(\tilde{G})$ is the average length of edges (f_1, f_2) of the gadget $\tilde{G}$, weighted by $\tilde{G}(f_1, f_2)$. For $\tilde{G}$ to have completeness $1 - 2^{-k}$, the edges in $\tilde{G}$ need to have an average length of 2^{-k}. Since there are no edges shorter than 2^{-k}, $\tilde{G}$ can only put non-zero capacity on edges of length exactly 2^{-k}.
Construct $\tilde{G}$ using the following procedure. Start with G. Split up each edge (f_1, f_2) in G into an arbitrary path starting at f_1, ending at f_2, with edges of length 2^{-k}, where the sum of lengths of edges in the path should be equal to the length of the original edge (f_1, f_2). Remove the capacity of edge (f_1, f_2) and give each edge in the path the same capacity as the capacity of the original edge (f_1, f_2). This will increase the total capacity of the graph by a factor of $(1 - c(G))/2^k$. As a final step, normalize the capacity by dividing the capacity of all edges by $(1 - c(G))/2^k$. Let the resulting graph be $\tilde{G}$. Note that $\tilde{G}$ is a Had$_k$-to-2Lin(2) consisting only of edges of length 2^{-k}, so its completeness is $1 - 2^{-k}$.

Recall that $1 - \mathrm{rs}(G)$ can be interpreted as the expected value of a Max-Flow problem on a fully connected 2^k-dimensional hypercube, where the placements of sources and sinks have been randomised. Note that any feasible flow ω of G, when scaled down by a factor of $(1 - c(G))/2^{-k}$, corresponds to a feasible flow of $\tilde{G}$. This implies that $(1 - \mathrm{rs}(G)) \leqslant (1 - \mathrm{rs}(\tilde{G}))(1 - c(G))/2^{-k}$.
The conclusion from this is that

$$\frac{1 - \mathrm{rs}(\tilde{G})}{1 - c(\tilde{G})} = \frac{1 - \mathrm{rs}(\tilde{G})}{2^{-k}} \geqslant \frac{1 - \mathrm{rs}(G)}{1 - c(G)}.$$

(c) Let $\tilde{G}$ be the gadget from b). Recall that $1 - \mathrm{rs}(\tilde{G})$ can be interpreted as the expected value of a Max-Flow problem on a fully connected 2^k-dimensional hypercube, where the placements of sources and sinks have been randomised. The capacities of this flow graph sum to 1.
Note that the sources and sinks correspond to affine functions, which have a normalised Hamming distance of at least $1/2$. Furthermore, since all edges in $\tilde{G}$ has length 2^{-k}, any path in $\tilde{G}$ between a source and a sink must contain at least 2^{k-1} edges.
For any flow graph, if all paths between sources and sinks contain at least 2^{k-1} edges, and the sum of capacity over all edges in the graph is 1, then the maximum flow is at most 2^{1-k}. So $1 - \mathrm{rs}(\tilde{G}) \leqslant 2^{1-k}$, which implies that

$$\frac{1 - \mathrm{rs}(\tilde{G})}{1 - c(\tilde{G})} \leqslant \frac{2^{1-k}}{2^{-k}} = 2.$$ ◀

▶ Remark B.2. Since the relaxed soundness is an upper bound of the true soundness, it follows that the NP-hardness result of Max-2Lin(2) as stated in Proposition 2.22 also holds for $s = \mathrm{rs}(G)$.

C Affine maps and lifts

Recall that the $\mathrm{rsLP}(G)$ can be interpreted as the expected value of a Max-Flow problem with a randomised source/sink placement over a fully connected 2^k-dimensional hypercube, where the nodes are indexed by Boolean functions $f \in \mathcal{F}_k$. The source/sink nodes are indexed by affine Boolean functions. In order to be able to describe the symmetries of these graphs, we want to study mappings $M : \mathcal{F}_k \to \mathcal{F}_k$ with the following properties:

1. Source and sink nodes map to source and sink nodes, i.e. if f is an affine function then $M(f)$ is also an affine function.
2. The length of all edges $\{v_{f_1}, v_{f_2}\}$ are preserved by the mapping, i.e. $\mathrm{dist}(M(f_1), M(f_2)) = \mathrm{dist}(f_1, f_2)$.

There is a natural choice of mappings from $\mathcal{F}_k \to \mathcal{F}_k$ for which Property 1 and 2 hold. Additionally as a bonus, the same natural choice of mappings can also be extended to construct mappings from $\mathcal{F}_k \to \mathcal{F}_{k'}, k \leqslant k'$, and still have that both Property 1 and 2 hold. This can then be used to embed the 2^k-dimensional hypercube in the $2^{k'}$-dimensional hypercube.

▶ **Definition C.1.** *Let $M_{A,b,\beta,c} : \mathcal{F}_k \to \mathcal{F}_{k'}$ be defined as*

$$M_{A,b,\beta,c}(f)(y) = f(Ay + b)(-1)^c \chi_\beta(y),$$

where $k, k' \in \mathbb{Z}_{>0}$, $k \leqslant k'$, $y \in {\mathbb{F}_2}^{k'}$, $A \in {\mathbb{F}_2}^{k \times k'}$ is a full rank matrix, $b \in \mathbb{F}_2^k$, $c \in \mathbb{F}_2$ and $\beta \in {\mathbb{F}_2}^{k'}$. Let $\mathcal{M}_{k \to k'}$ denote the set of all maps $M_{A,b,\beta,c}$ from $\mathcal{F}_k \to \mathcal{F}_{k'}$. For convenience, we often denote $M_{A,b,\beta,c}$ by M, where the A, b, β, c are all implicit.

Since these mappings are reminiscent of affine maps from linear algebra, we call them *affine maps*. However, they are not affine maps in the classical sense.

The function $M(f) \in \mathcal{F}_{k'}$ is called the *M-lift of f*. It is not hard to see that the M-lift of an affine function is an affine function. More generally, M-lifts always preserve the dimension of Boolean functions.

▶ **Proposition C.2.** *Given $f \in \mathcal{F}_k$ and $M \in \mathcal{M}_{k\to k'}$, $k \leqslant k'$, then* $\dim(M(f)) = \dim(f)$.

Proof. It follows from a direct calculation that

$$M_{A,b,\beta,c}(f)(y) = (-1)^c \sum_{\alpha\in\{0,1\}^k} \chi_{A^T\alpha+\beta}(y)\hat{f}_\alpha \chi_b(\alpha).$$

This shows that the affine mapping M moves affine(f) to affine$(M(f)) = \{A^T\alpha + \beta : \alpha \in$ affine$(f)\}$. Furthermore, since A is a full rank matrix, $\dim(f) = \dim(M(f))$. ◀

Affine maps also preserve the (normalised Hamming) distance of affine functions.

▶ **Proposition C.3.** *Given $f_1, f_2 \in \mathcal{F}_k$ and $M \in \mathcal{M}_{k\to k'}$, $k \leqslant k'$, then* $\mathrm{dist}(M(f_1), M(f_2)) = \mathrm{dist}(f_1, f_2)$.

Proof. Let $M = M_{A,b,\beta,c}$. Note that $\mathrm{dist}(M(f_1), M(f_2))$ only depends on A and b since

$$\begin{aligned}\mathrm{dist}(M(f_1), M(f_2)) &= \frac{1}{2^{k'}} \sum_{y\in\mathbb{F}_2{}^{k'}} \frac{1 - M(f_1)(y)M(f_2)(y)}{2}\\ &= \frac{1}{2^{k'}} \sum_{y\in\mathbb{F}_2{}^{k'}} \frac{1 - f_1(Ay+b)f_2(Ay+b)}{2}.\end{aligned}$$

Furthermore, since A is a full rank $k \times k'$ Boolean matrix, the kernel of A has dimension $k' - k$ and size $2^{k'-k}$, so

$$\sum_{y\in\{0,1\}^{k'}} f_1(Ay+b)f_2(Ay+b) = 2^{k'-k} \sum_{x\in\{0,1\}^k} f_1(x)f_2(x).$$

This shows that $\mathrm{dist}(M(f_1), M(f_2)) = \mathrm{dist}(f_1, f_2)$. ◀

The last notable property of the affine maps is that they form a group under composition. This property is needed to be able to apply the techniques from Appendix A.2 to the rsLP(G) and to the rs$_\infty$LP(G) in order to "compress" them.

▶ **Proposition C.4.** *$\mathcal{M}_{k\to k}$ under composition forms a group.*

Proof. The composition of two affine maps $M_{A',b',\beta',c'} \circ M_{A,b,\beta,c}$, is an affine map $M_{A'',b'',\beta'',c''}$, where

$$\begin{aligned}A'' &= AA',\\ b'' &= Ab' + b,\\ \beta'' &= (A')^T\beta + \beta',\\ c'' &= (b', \beta) + c' + c.\end{aligned}$$

Furthermore, the left and right inverse of an affine map $M_{A,b,\beta,c}$ is given by $M_{A',b',\beta',c'}$ where

$$A' = A^{-1},$$
$$b' = A^{-1}b,$$
$$\beta' = (A^{-1})^T\beta,$$
$$c' = c + (A^{-1}b, \beta).$$

This shows that $\mathcal{M}_{k\to k}$ forms a group under composition. ◀

C.1 M-lifts of sink and sources

Recall that the source/sink placements of the rsLP(G) and the rs$_\infty$LP(G) are described using a Boolean function $g \in \mathcal{F}_k$,

$$g(\alpha) \quad = \quad \begin{cases} 1 & \text{iff } v_{\chi_\alpha} \text{ is a sink,} \\ -1 & \text{iff } v_{\chi_\alpha} \text{ is a source.} \end{cases}$$

Note that M-lifts move the sink and source nodes. If $k = k'$, then the M-lift permutes the sink and source nodes. If $k < k'$, then the M-lift "lifts" the sink and source nodes onto a higher dimensional hypercube. This means that there exists multiple different source/sink placements $g' \in \mathcal{F}_{k'}$ that all match the lifted positions of the sinks and sources. The condition for when an M-lift of a source/sink placement $g \in \mathcal{F}_k$ is described by a source/sink placement $g' \in \mathcal{F}_{k'}$ is given by the following proposition.

▶ **Proposition C.5.** *An M-lift will map sink nodes in $\mathcal{F}_k$ onto sink nodes of $\mathcal{F}_{k'}$ and source nodes in $\mathcal{F}_k$ onto source nodes in $\mathcal{F}_{k'}$ if and only if*

$$M_{A^T,\beta,b,c}(g') \quad = \quad g.$$

Proof.

Note that the $M_{A,b,\beta,c}$-lift of χ_α is $M_{A,b,\beta,c}(\chi_\alpha) = \chi_{A^T\alpha+\beta}(x)(-1)^c\chi_b(\alpha)$. Using the source/sink placement g' we can tell whether a node v_{χ_α} is lifted onto a sink node or a source node,

$$g'(A^T\alpha + \beta)(-1)^c\chi_b(\alpha) \quad = \quad \begin{cases} 1 & \text{iff } v_{M_{A,b,\beta,c}(\chi_\alpha)} \text{ is a sink according to } g', \\ -1 & \text{iff } v_{M_{A,b,\beta,c}(\chi_\alpha)} \text{ is a source according to } g'. \end{cases}$$

This implies that the sufficient and necessary condition to make all sinks in $\mathcal{F}_k$ to be $M_{A,b,\beta,c}$-lifted to sinks in $\mathcal{F}_{k'}$ and all sources in $\mathcal{F}_k$ to be $M_{A,b,\beta,c}$-lifted to sources in $\mathcal{F}_{k'}$, is that

$$g(\alpha) \quad = \quad g'(A^T\alpha + \beta)(-1)^c\chi_b(\alpha)$$

for all $\alpha \in \mathbb{F}_2^k$. This is identical to requiring that $M_{A^T,\beta,b,c}(g') = g$. ◀

▶ **Definition C.6.** *The operator $M_{A^T,\beta,b,c}$ is denoted by $M^{\#}_{A,b,\beta,c}$.*

C.2 Lifting gadgets and flows

It is possible to extend the definition of M-lifting to Had$_k$-to-2Lin(2) gadgets G by defining $M \cdot G$ as

$$(M \cdot G)(f_1', f_2') = \sum_{\substack{f_1 \in M^{-1}(f_1'),\\ f_2 \in M^{-1}(f_2')}} G(f_1, f_2).$$

This moves the capacity $G(f_1, f_2)$ of edge $\{v_{f_1}, v_{f_2}\}$ onto edge $\{v_{M(f_1)}, v_{M(f_2)}\}$. Furthermore, let the *full $k \to k'$ lift* of G be defined as the average of all possible M-lifts, i.e.

$$\text{lift}_{k\to k'}(G) \quad = \quad \frac{1}{|\,\mathcal{M}_{k\to k'}\,|} \sum_{M\in\mathcal{M}_{k\to k'}} (M \cdot G).$$

Completely analogue to the definition of M-lifts of gadgets, let the M-lift of a flow w of the $\text{rs}(G)$ LP be defined as

$$(M \cdot w)(f_1', f_2', g') \quad = \sum_{\substack{f_1 \in M^{-1}(f_1'),\\ f_2 \in M^{-1}(f_2')}} w(f_1, f_2, M^{\#}(g')),$$

and let the full $k \to k'$ lift of w be defined as

$$\text{lift}_{k\to k'}(w) \quad = \quad \frac{1}{|\,\mathcal{M}_{k\to k'}\,|} \sum_{M\in\mathcal{M}_{k\to k'}} (M \cdot w).$$

By connecting these two concepts of lifting gadgets and flows, we can show the following proposition.

▶ **Proposition C.7.** *The full lift of G is a Had_k-to-2Lin(2) gadget G' where $c(G') = c(G)$ and $\text{rs}(G') \leqslant \text{rs}(G)$.*

Proof. Let w be a feasible flow of G and let $w' = \text{lift}_{k\to k'}(w)$. Note that w' is a feasible flow of G' since the capacity of G is lifted together with the flow w. So constraints (1) and (2) are satisfied by w'. Additionally,

$$\mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(w) \quad = \quad \mathbb{E}_{g'\in\mathcal{F}_{k'}} \text{val}_{g'}(w').$$

since any lift preserves the amount of flow going in and out of sink nodes and source nodes. ◀

The final Proposition that we need for Appendix D is that the full lift of a leaky flow w of the $\text{rsLP}(G)$ is a leaky flow of the $\text{rsLP}(G')$, and that the full lift does not affect the value of the flow. This is a fundamental property of lifts that is used in Appendix D to upper bound $\text{rs}(G')$ when $k' \to \infty$.

▶ **Proposition C.8.** *Let G' be the full lift of G, and let w' be the full lift of a leaky flow w of the* $\text{rsLP}(G)$*. Then w' is a leaky flow of the* $\text{rsLP}(G')$*, and* $\mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g'\in\mathcal{F}_{k'}} \text{val}_{g'}(w')$.

Proof. Let w be a leaky flow of G and let $w' = \text{lift}_{k\to k'}(w)$. Note that constraint (1) is satisfied by w' since the capacity of G is lifted together with the flow w. So w' is a leaky flow. Additionally,

$$\mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(w) \quad = \quad \mathbb{E}_{g'\in\mathcal{F}_{k'}} \text{val}_{g'}(w').$$

since any lift preserves the amount of flow going in and out of sink nodes and source nodes. ◀

D Proving that $\text{rs}_\infty(G)$ can be attained in the limit

The goal of this section is to prove Lemma 3.11, which relates the infinity relaxed soundness to the relaxed soundness. Let G be the Had_k-to-2Lin(2) gadget in Lemma 3.11 and let w be the optimal flow of the $\text{rs}_\infty\text{LP}(G)$, which implies that $\text{rs}_\infty(G) = 1 - \mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(w)$. Let k' be some integer greater than k and define $G' = \text{lift}_{k\to k'}(G)$ and $w' = \text{lift}_{k\to k'}(w)$. According to Proposition C.7 G' is a Had_k-to-2Lin(2) with $c(G') = c(G)$ and according to Proposition C.8 w' is a leaky flow of the $\text{rsLP}(G')$ and $\mathbb{E}_{g\in\mathcal{F}_k} \text{val}_g(w) = \mathbb{E}_{g'\in\mathcal{F}_{k'}} \text{val}_{g'}(w')$. We prove that as k' tends to infinity the total leakage of G' converges to 0. After we have established this, Lemma 3.11 follows from Theorem A.5.

D.1 Total leakage approaches 0 as $k' \to \infty$

Let us start by formally defining the leaks of w and w', where w is a leaky flow of the rsLP(G) and w' is a leaky flow of the rsLP(G'). Recall that the rsLP(G') describe the expectation of the maximum flow of a graph with a random source/sink placement $g' \in \mathcal{F}_{k'}$. It is for this reason that the total leakage of w' is defined as an expectation over $g' \in \mathcal{F}_{k'}$ of the total leakage of the graph with source/sink placement given by g'.

▶ **Definition D.1.** *Let $L_{k'}$ denote the total leakage of w',*

$$L_{k'} = \mathbb{E}_{g' \in \mathcal{F}_{k'}} \left(\sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t. \dim(f') > 0}} |\operatorname{leak}_{w'}(f', g')| \right),$$

where

$$\begin{aligned}\operatorname{leak}_{w'}(f', g') &= \operatorname{out}_{w'}(f', g') - \operatorname{in}_{w'}(f', g') \\ &= \frac{1}{|\mathcal{M}_{k \to k'}|} \sum_{M \in \mathcal{M}_{k \to k'}} \left(\sum_{\substack{f \in \mathcal{F}_k \\ s.t. M(f) = f'}} \operatorname{leak}_w(f, M^{\#}(g')) \right).\end{aligned}$$

The aim of this subsection is to prove that $L_{k'} \to 0$ as $k' \to \infty$. We do this by proving the following upper bound on $L_{k'}$ through a second order moment analysis.

▶ **Proposition D.2.**

$$L_{k'} \leqslant \frac{2^{2^k + k}}{\sqrt{2^{k'} - 2^k}}.$$

The proof of Proposition D.2 relies on the following Proposition describing the relationship between random pairs of affine maps $M_1, M_2 \in \mathcal{M}_{k \to k'}$ such that $M_1(f) = M_2(f)$ for some fixed $f \in \mathcal{F}_k$.

▶ **Definition D.3.** *Given $M_{A,b,\beta,c} \in \mathcal{M}_{k \to k'}$, let $T_M : \mathbb{F}_2^k \to \mathbb{F}_2{}^{k'}$ denote the affine map $T_M(x) = A^T x + \beta$. Furthermore, let $\operatorname{affine}(M_{A,b,\beta,c})$ denote the affine subspace $\{T_M(x) : x \in \mathbb{F}_2^k\} \subseteq \mathbb{R}^{k'}$.*

▶ **Proposition D.4.** *Given $f \in \mathcal{F}_k$ and $f' \in \mathcal{F}_k$ with $\dim(f) = \dim(f') = d$. Then*

$$|\{(M_1, M_2) \in \mathcal{N}_{f \to f'} \times \mathcal{N}_{f \to f'} : \dim(\operatorname{affine}(M_1) \cap \operatorname{affine}(M_2)) > d\}| \leqslant |\mathcal{N}_{f \to f'}|^2 \frac{\left(2^k - 2^d\right)^2}{2^{k'} - 2^d},$$

where $\mathcal{N}_{f \to f'} = \{M \in \mathcal{M}_{k \to k'} : M(f) = f'\}$ denotes the set of affine maps in $\mathcal{M}_{k \to k'}$ that lifts f to f'.

Proof. Note that for any $M_1, M_2 \in \mathcal{N}_{f \to f'}$, the dimension of $\operatorname{affine}(M_1) \cap \operatorname{affine}(M_2)$ is at least d, since according to the proof of Proposition C.2 both T_{M_1} and T_{M_2} must map $\operatorname{affine}(f)$ onto $\operatorname{affine}(f')$, so $\dim(\operatorname{affine}(M_1) \cap \operatorname{affine}(M_2) \cap \operatorname{affine}(f')) = d$. However, the two maps T_{M_1} and T_{M_2} can map the complement of $\operatorname{affine}(f)$ in different ways since there is no restriction to how they map the complement of $\operatorname{affine}(f)$.

Fix M_1 and uniformly at random pick M_2 from $\mathcal{N}_{f\to f'}$. Given any fix $x \notin \text{affine}(f)$, the probability that $T_{M_2}(x) \in \text{affine}(M_1)$ is $(2^k - 2^d)/(2^{k'} - 2^d)$ since $|\text{affine}(M_1) \setminus \text{affine}(f')| = 2^k - 2^d$ and $T_{M_2}(x)$ is uniformly distributed over the complement of $\text{affine}(f')$. Taking a union bound over all $x \notin \text{affine}(f)$ shows that

$$P_{M_2\in\mathcal{N}_{f\to f'}}[\dim(\text{affine}(M_1)\cap\text{affine}(M_2)) > d] \leqslant \frac{\left(2^k - 2^d\right)^2}{2^{k'} - 2^d}.$$

Proposition D.4 follows directly from this inequality. ◀

The takeaway from Proposition D.4 is that if M_1 and M_2 are two random affine maps such that $M_1(f) = M_2(f)$ for some fixed $f \in \mathcal{F}_k$, then with high probability $\text{affine}(M_1) \cap \text{affine}(M_2) = \text{affine}(f)$. This allows us to create a bound on the second order moment of the terms that define $L_{k'}$.

▶ **Lemma D.5.** *Given* $f \in \mathcal{F}_k$, $f' \in \mathcal{F}_{k'}$ *and* $g' \in \mathcal{F}_{k'}$, *where* $\dim(f) = \dim(f') = d > 0$, *then*

$$\mathbb{E}_{g'\in\mathcal{F}_{k'}}\left(\left|\sum_{M\in\mathcal{N}_{f\to f'}} \text{leak}_w(f, M^{\#}(g'))\right|^2\right) \leqslant |\mathcal{N}_{f\to f'}|^2 \frac{\left(2^k - 2^d\right)^2}{2^{k'} - 2^d}.$$

Proof. Expanding the square we need to prove that,

$$\sum_{M_1, M_2\in\mathcal{N}_{f\to f'}} \mathbb{E}_{g'\in\mathcal{F}_{k'}}(\text{leak}_w(f, M_1^{\#}(g'))\,\text{leak}_w(f, M_2^{\#}(g'))) \leqslant |\mathcal{N}_{f\to f'}|^2 \frac{\left(2^k - 2^d\right)^2}{2^{k'} - 2^d}.$$

Split the terms up into two cases, either $\dim(\text{affine}(M_1)\cap\text{affine}(M_2)) > d$ or $\dim(\text{affine}(M_1)\cap \text{affine}(M_2)) = d$. By Proposition D.4 the number of terms of the first type is at most $|\mathcal{N}_{f\to f'}|^2 \left(2^k - 2^d\right)^2/(2^{k'} - 2^d)$. Each term is bounded by one since the sum of capacities in the $\text{rs}(G)$ LP is equal to 1, so the absolute value of a leak is always smaller than or equal to 1 at any node and for any source/sink placement.

In the other case, when $\dim(\text{affine}(M_1)\cap\text{affine}(M_2)) = d$, then the two random functions $M_1^{\#}(g')$ and $M_2^{\#}(g')$ are equal on $\text{affine}(f)$, and independently uniformly random $\{1, -1\}$ on the complement of $\text{affine}(f)$. This allows us to rewrite the expectation over g' as

$$\begin{aligned}
&\mathbb{E}_{g'\in\mathcal{F}_{k'}}(\text{leak}_w(f, M_1^{\#}(g'))\,\text{leak}_w(f, M_2^{\#}(g')))\\
&= \mathbb{E}_{g'\in\mathcal{F}_{k'}}\left(\text{leak}_w(f, M_1^{\#}(g'))\,\mathbb{E}_{\substack{g_2'\in\mathcal{F}_{k'}\\ s.t. M_2^{\#}(g_2')|_{\text{affine}(f)} = M_1^{\#}(g')|_{\text{affine}(f)}}} \text{leak}_w(f, M_2^{\#}(g_2'))\right)\\
&= \mathbb{E}_{g\in\mathcal{F}_k}\left(\text{leak}_w(f, g)\,\mathbb{E}_{\substack{g_2\in\mathcal{F}_k\\ s.t. g_2|_{\text{affine}(f)} = g|_{\text{affine}(f)}}} \text{leak}_w(f, g_2)\right).
\end{aligned}$$

This is equal to 0, since for any infinity relaxed flow w (see Definition 3.7) the expectation of $\text{leak}_w(f, g_2)$ over g_2 given g is 0. ◀

We are now at the point where we can prove Proposition D.2 using Lemma D.5.

Proof of Proposition D.2. A trivial upper bound of $L_{k'}$ using the triangle inequality is

$$L_{k'} \leqslant \frac{1}{|\mathcal{M}_{k\to k'}|} \sum_{f\in\mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t.\dim(f')>0}} \mathbb{E}_{g'\in\mathcal{F}_{k'}} \left(\left| \sum_{M\in\mathcal{N}_{f\to f'}} \mathrm{leak}_w(f, M^{\#}(g')) \right| \right).$$

Applying Jensen's inequality to the expectation over $g' \in \mathcal{F}_{k'}$ gives

$$\mathbb{E}_{g'\in\mathcal{F}_{k'}} \left(\left| \sum_{M\in\mathcal{N}_{f\to f'}} \mathrm{leak}_w(f, M^{\#}(g')) \right| \right) \leqslant \sqrt{\mathbb{E}_{g'\in\mathcal{F}_{k'}} \left(\left| \sum_{M\in\mathcal{N}_{f\to f'}} \mathrm{leak}_w(f, M^{\#}(g')) \right|^2 \right)},$$

which according to to Lemma D.5 can be further upper bounded by

$$\begin{aligned} \sqrt{\mathbb{E}_{g'\in\mathcal{F}_{k'}} \left(\left| \sum_{M\in\mathcal{N}_{f\to f'}} \mathrm{leak}_w(f, M^{\#}(g')) \right|^2 \right)} &\leqslant \frac{2^k - 2^{\dim(f)}}{\sqrt{2^{k'} - 2^{\dim(f)}}} |\mathcal{N}_{f\to f'}| \\ &\leqslant \frac{2^k}{\sqrt{2^{k'} - 2^k}} |\mathcal{N}_{f\to f'}|. \end{aligned}$$

We have so far shown that

$$L_{k'} \leqslant \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f\in\mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t.\dim(f')>0}} \frac{|\mathcal{N}_{f\to f'}|}{|\mathcal{M}_{k\to k'}|}.$$

Finally, note that $\sum_{f'\in\mathcal{F}_{k'}} |\mathcal{N}_{f\to f'}| = |\mathcal{M}_{k\to k'}|$ since $\mathcal{N}_{f\to f'}$ are disjoint subsets of $\mathcal{M}_{k\to k'}$ for different $f' \in \mathcal{F}_{k'}$ and their union over $f' \in \mathcal{F}_{k'}$ is equal to $\mathcal{M}_{k\to k'}$. So

$$L_{k'} \leqslant \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f\in\mathcal{F}_k} \sum_{\substack{f' \in \mathcal{F}_{k'} \\ s.t.\dim(f')>0}} \frac{|\mathcal{N}_{f\to f'}|}{|\mathcal{M}_{k\to k'}|} \leqslant \frac{2^k}{\sqrt{2^{k'} - 2^k}} \sum_{f\in\mathcal{F}_k} 1 \leqslant \frac{2^{2^k+k}}{\sqrt{2^{k'} - 2^k}}. \blacktriangleleft$$

D.2 The proof of Lemma 3.11

All that remains is to tie up the loose ends by proving Lemma 3.11 using Proposition D.2 combined with Theorem A.5.

Proof of Lemma 3.11. Since w' is a leaky flow of the rsLP(G'), it follows from Theorem A.5 that there exists a feasible flow $\tilde{w}'$ of the rsLP(G') such that

$$\mathbb{E}_{g'\in\mathcal{F}_{k'}} \mathrm{val}_{g'}(\tilde{w}') + L_{k'} \geqslant \mathbb{E}_{g'\in\mathcal{F}_{k'}} \mathrm{val}_{g'}(w').$$

Note that $\mathrm{rs}(G') \geqslant 1 - \mathbb{E}_{g'\in\mathcal{F}_{k'}} \mathrm{val}_{g'}(\tilde{w}')$ since $\tilde{w}'$ is a feasible flow of the rsLP(G'). Furthermore, recall that $\mathrm{rs}_\infty(G) = 1 - \mathbb{E}_{g'\in\mathcal{F}_{k'}} \mathrm{val}_{g'}(w')$. So

$$\mathrm{rs}(G') - L_{k'} \leqslant \mathrm{rs}_\infty(G).$$

Proposition D.2 implies that $L_{k'} \to 0$ as $k' \to \infty$, which proves that $\forall \varepsilon > 0$ there exists a gadget G' with $c(G') = c(G)$ such that $\mathrm{rs}(G') - \varepsilon \leqslant \mathrm{rs}_\infty(G)$. ◀

E Gadget construction and verification

This section contains the details for how to practically compute Had_k-to-2Lin(2) gadgets using the $\text{rsLP}(G)$ and the $\text{rs}_\infty\text{LP}(G)$. These LPs have far too many variables and constraints to directly be solved by a computer when $k \geqslant 4$. The solution is to make use of the symmetries of the LP:s to construct smaller LP:s with the same optimum. This is done in two steps. Step 1 is to use Proposition C.7 to argue that best gadgets are the symmetrical gadgets. This means that we only need to take into account symmetrical gadgets when solving the $\text{rsLP}(G)$ and the $\text{rs}_\infty\text{LP}(G)$. Step 2 is to use the fact that if G is symmetrical, then Theorem A.8 allows us to compress the LP, merging a huge number of variables into a single variable.

E.1 Symmetrical Had_k-to-2Lin(2) gadgets are optimal

The meaning of a Had_k-to-2Lin(2) gadget $(G, \mathbb{X}, \mathbb{Y})$ being *optimal* is that there exists no Had_k-to-2Lin(2) gadget $(\tilde{G}, \mathbb{X}, \mathbb{Y})$ such that $c(G) = c(\tilde{G})$ and $\text{rs}(G) > \text{rs}(\tilde{G})$. The following Proposition states that symmetric gadgets are optimal. By symmetric, we refer to the property that the gadget G is invariant under M-lifts.

▶ **Proposition E.1.** *Given any Had_k-to-2Lin(2) gadget $(G, \mathbb{X}, \mathbb{Y})$, there exists a symmetric Had_k-to-2Lin(2) gadget $(\tilde{G}, \mathbb{X}, \mathbb{Y})$ such that $c(G) = c(\tilde{G})$ and $\text{rs}(G) \geqslant \text{rs}(\tilde{G})$.*

Proof. Let $\tilde{G} = \text{lift}_{k\to k}(G)$. According to Proposition C.7, $c(G) = c(\tilde{G})$ and $\text{rs}(G) \geqslant \text{rs}(\tilde{G})$. Furthermore, $\tilde{G}$ is a symmetric gadget since for any $f_1, f_2 \in \mathcal{F}_k$ and $M \in \mathcal{M}_{k\to k}$,

$$\begin{aligned}(M \cdot \tilde{G})(f_1, f_2) &= \frac{1}{|\mathcal{M}_{k\to k}|} \sum_{M_2 \in \mathcal{M}_{k\to k}} ((M \circ M_2) \cdot \tilde{G})(f_1, f_2) \\ &= \frac{1}{|\mathcal{M}_{k\to k}|} \sum_{M_2 \in M \circ \mathcal{M}_{k\to k}} (M_2 \cdot \tilde{G})(f_1, f_2).\end{aligned}$$

According to Proposition C.4, $\mathcal{M}_{k\to k}$ forms a group, so $M \circ \mathcal{M}_{k\to k} = \mathcal{M}_{k\to k}$. We have shown that $M \cdot \tilde{G} = \tilde{G}$ and thus $\tilde{G}$ is a symmetric gadget. ◀

E.2 Compressing the $\text{rsLP}(G)$ and $\text{rs}_\infty\text{LP}(G)$

As discussed earlier, both the $\text{rsLP}(G)$ and the $\text{rs}_\infty\text{LP}(G)$ can be interpreted as Max-Flow problems. Furthermore, if G is symmetric under M-lifts, then $\mathcal{M}_{k\to k}$ is a symmetry group for both of these Max-Flow problems. This means that we can apply Theorem A.8 to compress the Max-Flow problems, giving us the *compressed* $\text{rsLP}(G)$ and the *compressed* $\text{rs}_\infty\text{LP}(G)$.

One of the symmetries that the compression is able to capture is that many different source/sink placements are equivalent. In a sense, the source/sink placements of the compressed LPs consist of one representative source/sink placement from each set of equivalent source/sink placements. This symmetry turns out to be the main contributor as to why the compressed LP is significantly smaller than the original LP.

Without the compression, the LPs each have $2^{3\cdot 2^k}$ variables, which for $k \geqslant 4$ is computationally infeasible. However, even with the compression, for $k = 4$ the LPs are still large enough that it is computationally challenging to solve them.

E.2.1 Further restricting the compressed LPs

To further restrict the size of the LPs in the case of $k = 4$, we heuristically identify a list of beneficial gadget variables by solving the compressed LPs with floating point numbers

Table 4 Sizes of the $\text{rsLP}(G)$ and $\text{rs}_\infty\text{LP}(G)$ for Had_2-to-2Lin(2) gadgets G. The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints. All variables have the implicit constraint of being non-negative.

	rsLP(G)			$\text{rs}_\infty\text{LP}(G)$		
Original	163	343	534	163	343	534
Compressed	23	38	106	23	38	106

Table 5 Sizes of the $\text{rsLP}(G)$ and $\text{rs}_\infty\text{LP}(G)$ for Had_3-to-2Lin(2) gadgets G. The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints. All variables have the implicit constraint of being non-negative.

	rsLP(G)			$\text{rs}_\infty\text{LP}(G)$		
Original	$8 \cdot 10^6$	$2 \cdot 10^7$	$5 \cdot 10^7$	$8 \cdot 10^6$	$2 \cdot 10^7$	$5 \cdot 10^7$
Compressed	298	546	2330	243	462	1987

using Gurobi. Any gadget variable that is given non-zero weight in at least one floating point solution is added to the list. Using this list, we define the *restricted compressed LP* as the compressed LP but with all other gadget variables that are not on the list, removed. The list we use can be found in Table 9 in Appendix F. Note that one possible drawback to restricting the LPs like this is that the restriction could lead to construction of sub-optimal gadgets.

Tables 4–6 show the sizes of the LPs depending on if compression or restriction is being applied. Note that the restricted and compressed LP:s have significantly fewer variables than the original LP:s.

There is a special case where we do not need the restrictions. If the completeness of a gadget is $1 - 2^{-k}$, then the gadget only has non-zero weight on edges of length 2^{-k}. There are comparatively relatively few edges of length 2^{-k}. This allows us to directly construct the gadget by solving the non-restricted LP. So in the case of completeness $1 - 2^{-k}$, the gadgets we construct are guaranteed to be optimal since we do not make use of any restrictions.

E.3 Implementation details

The compressed $\text{rsLP}(G)$ and compressed $\text{rs}_\infty\text{LP}(G)$ are constructed using a Python script where all of the calculations are done using integer arithmetic. The script makes use of affine maps to efficiently compute the symmetries of the two LPs, in order to compress them. The time and memory complexities of the script are roughly $O(2^{2 \cdot 2^k})$, so the script is able to handle $k = 2$, 3 and 4. In theory it would be possible to also make the script support $k = 5$, but that would require both more powerful hardware, as well as improving the time complexity to roughly $O(2^{2^k})$ time.

Table 6 Sizes of the $\text{rsLP}(G)$ and $\text{rs}_\infty\text{LP}(G)$ for Had_4-to-2Lin(2). The three numbers are the number of linear constraints, number of variables and number of non-zero entries in the constraints.

	rsLP(G)			$\text{rs}_\infty\text{LP}(G)$		
Original	$1 \cdot 10^{14}$	$3 \cdot 10^{14}$	$4 \cdot 10^{14}$	$1 \cdot 10^{14}$	$3 \cdot 10^{14}$	$4 \cdot 10^{14}$
Restricted	$2 \cdot 10^{11}$	$4 \cdot 10^{11}$	$6 \cdot 10^{11}$	$2 \cdot 10^{11}$	$4 \cdot 10^{11}$	$6 \cdot 10^{11}$
Compressed	$4 \cdot 10^5$	$7 \cdot 10^5$	$1 \cdot 10^7$	$3 \cdot 10^5$	$6 \cdot 10^5$	$9 \cdot 10^6$
Restricted & compressed	$3 \cdot 10^4$	$6 \cdot 10^4$	$2 \cdot 10^5$	$3 \cdot 10^4$	$5 \cdot 10^4$	$2 \cdot 10^5$

After having computed the compressed $\mathrm{rsLP}(G)$ and compressed $\mathrm{rs}_\infty\mathrm{LP}(G)$, the list of beneficial gadget variables found in Section 4.1 are used to construct the restricted compressed LPs. In order to solve the compressed LP we use the exact rational number LP solver QSopt_ex[1]. This results in a gadget described only using rational numbers, as well as an accompanying compressed flow, also described only using rational numbers.

E.4 Verification of $\mathrm{rs}(G)$ and $\mathrm{rs}_\infty(G)$

It is significantly simpler to verify the relaxed soundness and the infinity relaxed soundness of a gadget than it is to construct the gadget. The verification can be done almost directly on the original LPs, without needing the restricted compressed LPs or the compressed LPs.

The input to the verification program is a gadget $G : \binom{\mathcal{F}_k}{2} \to [0,1]$ together with a flow $w_g : \mathcal{F}_k \times \mathcal{F}_k \to \mathbb{R}$, for each source/sink placement equivalence class representative g. The flow acts as a witness for the relaxed soundness / infinity relaxed soundness of the gadget. In order to avoid floating point errors, we require both G and the w_g to be rational.

The verification process is done in five steps.

1. For each source/sink placement representative g, verify that the flow w_g satisfies the capacity constraints of the $\mathrm{rs}(G)$ LP / $\mathrm{rs}_\infty(G)$ LP, i.e. that $w_g(f_1, f_2) + w_g(f_2, f_1) \leqslant G(f_1, f_2)$ for all $f_1, f_2 \in \mathcal{F}_k$.
2. Verify that the gadget G is symmetric under action by $M \in \mathcal{M}_{k\to k}$, meaning that for all functions $f_1, f_2 \in \mathcal{F}_k$ and affine maps $M \in \mathcal{M}_{k\to k}$, it holds that $G(f_1, f_2) = G(M(f_1), M(f_2))$.
3. For each source/sink placement representative g and each function $f \in \mathcal{F}_k$, compute $\mathrm{in}(f, g)$ and $\mathrm{out}(f, g)$. Now extend in and out to be defined for all f and g in $\mathcal{F}_k$. For any source/sink placements $\tilde{g} \in \mathcal{F}_k$ that is not a representative, pick a map $M \in \mathcal{M}_{k\to k}$ and representative g such that $g = M^{\#}(\tilde{g})$, and define $\mathrm{in}(f, \tilde{g})$ as $\mathrm{in}(M^{-1}(f), g)$ and $\mathrm{out}(f, \tilde{g})$ as $\mathrm{out}(M^{-1}(f), g)$.
4. Verify the conservation of flow constraint in the $\mathrm{rsLP}(G)$ / $\mathrm{rs}_\infty\mathrm{LP}(G')$ by iterating over all $(f, g) \in \mathcal{F}_k \times \mathcal{F}_k$ that are not sinks or sources. For the $\mathrm{rsLP}(G)$ this just involves checking that $\mathrm{in}(f, g) = \mathrm{out}(f, g)$. For the $\mathrm{rs}_\infty\mathrm{LP}(G)$ this involves checking that $\sum_{g'} \mathrm{in}(f, g') = \sum_{g'} \mathrm{out}(f, g')$, where the sum is over all g' such that $g'|_{\mathrm{affine}(f)} = g|_{\mathrm{affine}(f)}$.
5. Compute and output the completeness and rs / rs_∞ of the gadget using the extended inflow and outflow as a witness.

Note that the first step verifies the capacity constraints only for representatives of equivalent source/sink placements. The second step checks that the gadget G is symmetric, which combined with the first step implies that any extension of the flow to an arbitrary source/sink placement will fulfil the capacity constraints. The fourth step checks that the conservation of flow constraint is fulfilled, which in the case of the $\mathrm{rs}_\infty\mathrm{LP}(G)$ involves computing the affine support of all possible source/sink placements.

The LP's we use and the gadgets we present in this paper can be found at `https://github.com/bjorn-martinsson/NP-hardness-of-Max-2Lin-2`, as well as a stand alone implementation of a verification script written in Python. As described in the verification process above, the verification requires a flow w_g as input. So on the Github, there is also a script used to generate this witness flow. This is done by solving the restricted compressed $\mathrm{rsLP}(G)$ / $\mathrm{rs}_\infty\mathrm{LP}(G)$ using an integral Max-Flow solver, and then uncompressing the result.

F Edges used/unused in constructed gadgets

During the numerical analysis, we solve LPs to construct the gadgets. A gadget can be interpreted as a probability distribution over (undirected) edges. Tables 7–9 list all edges that have been given non-zero weight in at least one solution to an LP, for $k = 2, 3, 4$. Recall that every gadget that we construct is symmetrical under the mappings of $\mathcal{M}_{k\to k}$, so edges from the same edge orbit share the same capacity. More specifically, the tables contain a list of all edge orbits that are used in at least one constructed gadget.

Table 7 The relevant edge orbits for Had_2-to-2Lin(2) gadgets. The edges of a Had_2-to-2Lin(2) gadget has a total of 4 edge orbits, but only two are ever used in our constructed gadgets. The rest of the edges were always given capacity 0 by the (rational) LP-solver.

f_1	f_2	Ham.dist.	size
0000	1000	1	32
0000	1100	2	24

Table 8 The relevant edge orbits for Had_3-to-2Lin(2) gadget. The edges of a Had_3-to-2Lin(2) gadget has a total of 26 edge orbits, but only four are ever used in our constructed gadgets. The rest of the edges were always given capacity 0 by the (rational) LP-solver.

f_1	f_2	Ham.dist.	Size
00000000	10000000	1	128
10000000	11000000	1	896
00000000	11000000	2	448
00000000	11110000	4	112

Table 9 The relevant edge orbits for Had_4-to-2Lin(2) gadget. The edges of a Had_4-to-2Lin(2) gadget has a total of 1061 edge orbits, but only 21 are ever used in our constructed gadgets. Note that as discussed in Appendix E.2.1, this list of edges was identified using the Gurobi LP-solver, and not using a rational LP solver. See Appendix E.2.1 for more information.

f_1	f_2	Ham.dist.	Size
0000000000000000	1000000000000000	1	512
1000000000000000	1100000000000000	1	7680
1100000000000000	1110000000000000	1	53760
1110000000000000	1111000000000000	1	17920
1110000000000000	1110100000000000	1	215040
1110100000000000	1110100010000000	1	215040
0000000000000000	1100000000000000	2	3840
1100000000000000	1111000000000000	2	26880
1100000000000000	1110100000000000	2	322560
1110000000000000	1111100000000000	2	107520
1110000000000000	1110110000000000	2	161280
1111000000000000	1110100000000000	2	107520
1110100000000000	1110100011000000	2	322560
0000000000000000	1110000000000000	3	17920
1100000000000000	1111100000000000	3	322560
1100000000000000	1110101000000000	3	215040
1110000000000000	1110100010001000	3	860160
0000000000000000	1111000000000000	4	4480
0000000000000000	1110100000000000	4	53760
0000000000000000	1111100000000000	5	53760
0000000000000000	1111111100000000	8	480