Two-Commodity Flow is Equivalent to Linear Programming under Nearly-Linear Time Reductions

Ming Ding
ming.ding@inf.ethz.ch
Department of Computer Science
ETH Zurich

Rasmus Kyng*
kyng@inf.ethz.ch
Department of Computer Science
ETH Zurich

Peng Zhang
pz149@rutgers.edu
Department of Computer Science
Rutgers University

Abstract

We give a nearly-linear time reduction that encodes any linear program as a 2-commodity flow problem with only a polylogarithmic blow-up in size. Our reduction applies to high-accuracy approximation algorithms and exact algorithms. Given an approximate solution to the 2-commodity flow problem, we can extract a solution to the linear program in linear time with only a polynomial factor increase in the error. This implies that any algorithm that solves the 2-commodity flow problem can solve linear programs in essentially the same time. Given a directed graph with edge capacities and two source-sink pairs, the goal of the 2-commodity flow problem is to maximize the sum of the flows routed between the two source-sink pairs subject to edge capacities and flow conservation. A 2-commodity flow problem can be formulated as a linear program, which can be solved to high accuracy in almost the current matrix multiplication time (Cohen-Lee-Song JACM'21). In this paper, we show that linear programs can be approximately solved, to high accuracy, using 2-commodity flow as well. As a corollary, if a 2-commodity flow problem can be approximately solved in time $O(|E|^c \operatorname{polylog}(U |E| \epsilon^{-1}))$, where E is the graph edge set, U is the ratio of maximum to minimum edge capacity, ϵ is the multiplicative error parameter, and c is a constant greater than or equal to 1, then a linear program with integer coefficients and feasible set radius r can be approximately solved in time $O(N^c \operatorname{polylog}((r+1)X\epsilon^{-1}))$, where N is the number of nonzeros and X is the largest magnitude of the coefficients. Thus a solver for 2-commodity flow with running time exponent $c < \omega$, where $\omega < 2.37286$ is the matrix multiplication constant, would improve the running time for solving sparse linear programs.

Our proof follows the outline of Itai's polynomial-time reduction of a linear program to a 2-commodity flow problem (JACM'78). Itai's reduction shows that exactly solving 2-commodity flow and exactly solving linear programming are polynomial-time equivalent. We improve Itai's reduction to preserve the sparsity of all the intermediate steps. In addition, we establish an error bound for approximately solving each intermediate problem in the reduction, and show that the accumulated error is polynomially bounded. We remark that our reduction does not run in strongly polynomial time and that it is open whether 2-commodity flow and linear programming are equivalent in strongly polynomial time.

*The research leading to these results has received funding from grant no. 200021 204787 of the Swiss National Science Foundation.

1 Introduction

Multi-commodity maximum flow is a very well-studied problem, which can be formulated as a linear program. In this paper, we show that general linear programs can be very efficiently encoded as a multi-commodity maximum flow programs. Many variants of multi-commodity flow problems exist. We consider one of the simplest directed variants, 2-commodity maximum through-put flow. Given a directed graph with edge capacities and two source-sink pairs, this problem requires us to maximize the sum of the flows routed between the two source-sink pairs, while satisfying capacity constraints and flow conservation at the remaining nodes. In the rest of the paper, we will simply refer to this as *the 2-commodity flow problem*. We abbreviate this problem as 2CF. Our goal is to relate the hardness of solving 2CF to that of solving linear programs (LPs). 2-commodity flow is easily expressed as a linear program, so it is clearly no harder than solving LPs. We show that the 2-commodity flow problem can encode a linear program with only a polylogarithmic blow-up in size. Our reduction runs in nearly-linear time. Given an approximate solution to the 2-commodity flow problem, we can recover, in linear time, an approximate solution to the linear program with only a polynomial factor increase in the error. Our reduction also shows that an exact solution to the flow problem yields an exact solution to the linear program.

Multi-commodity flow problems are extremely well-studied and have been the subject of numerous surveys [Ken78; AMO93; OMV00; BKV09; Wan18], in part because a large number of problems can be expressed as variants of multi-commodity flow.

Our result shows a very strong form of this observation: In fact, general linear programs can be expressed as 2-commodity flow problems with essentially the same size. Early in the study of these problems, before a polynomial-time algorithm for linear programming was known, it was shown that the *undirected* 2-commodity flow problem can be solved in polynomial time [Hu63]. In fact, it can be reduced to two undirected single commodity maximum flow problems. In contrast, directed 2-commodity flow problems were seemingly harder, despite the discovery of non-trivial algorithms for some special cases [Eva76; Eva78].

Searching for multi-commodity flow solvers. Alon Itai [Ita78] proved a polynomial-time reduction from linear programming to 2-commodity flow, before a polynomial-time algorithm for linear programming was known. For decades, the only major progress on solving multi-commodity flow came from improvements to general linear program solvers [Kha80; Kar84; Ren88; Vai89]. An important development in the research on undirected multi-commodity flow for approximation algorithms was the characterization of the undirected multi-commodity flow-cut gap [LR89; LR99; AR98; LLR95; CSW10]. Leighton et al. [Lei+95] showed that undirected capacitated k-commodity flow in a graph with m edges and n vertices can be approximately solved in $\widetilde{O}(kmn)$ time, completely routing all demands with $1+\epsilon$ times the optimal congestion, albeit with a poor dependence on the error parameter ϵ. This beats solve-times for linear programming in sparse graphs for small k, even with today's LP solvers that run in current matrix multiplication time, albeit with much worse error. This result spurred a number of follow-up works with improvements for low-accuracy algorithms [GK07; Fle00; Mad10]. Later, breakthroughs in achieving almost- and nearly-linear time algorithms for undirected single-commodity maximum flow also lead to faster algorithms for undirected k-commodity flow [Kel+14; She13; Pen16], culminating in Sherman's introduction of area-convexity to build a $\widetilde{O}(mk\epsilon^{-1})$ time algorithm for approximate undirected k-commodity flow [She17].

Solving single-commodity flow problems. Single commodity flow problems have been an area of tremendous success for the development of graph algorithms, starting with an era of algorithms influenced by early results on maximum flow and minimum cut [FF56] and later the development of powerful combinatorial algorithms for maximum flow [Din70; ET75; GR98] with polynomially bounded edge capacities. Later, a breakthrough nearly-linear time algorithm for electrical flows by Spielman and Teng [ST04] lead to the *Laplacian paradigm.* A long line of work explored direct improvements and simplifications of this result [KMP10; KMP11; Kel+13; PS14; KS16; JS21]. This also motivated a new line of research on undirected maximum flow [Chr+11; LRS13; Kel+14; She13], which in turn lead to faster algorithms for directed maximum flow and minimum cost flow [Mad13; Mad16; LS20; KLS20; van+21; GLP21] building on powerful tools using mixed-ℓ_2, ℓ_p-norm minimizing flows [Kyn+19] and inverse-maintenance ideas [Che+20]. Certain developments are particularly relevant to our result: For a graph $G = (V, E)$ these works established high-accuracy algorithms with $\widetilde{O}(|E|)$ running time for computing electrical flow [ST04] and $O(|E|^{4/3})$ running time for unit capacity directed maximum flow [Mad13; KLS20], and $\widetilde{O}(\min(|E|^{1.497}, |E| + |V|^{1.5}))$ running time for directed maximum flow with general capacities [GLP21; van+21].

Solving general linear programs. As described in the previous paragraphs, there has been tremendous success in developing fast algorithms for single-commodity flow problems and undirected multi-commodity flow problems, albeit in the latter case only in the low-accuracy regime (as the algorithm running times depend polynomially on the error parameter). In contrast, the best known algorithms for directed multi-commodity flow simply treat the problem as a general linear program, and use a solver for general linear programs. Recently, there has been significant progress on solvers for general linear programs, but the running time required to solve a linear program with roughly n variables and $\widetilde{O}(n)$ constraints (and polynomially bounded entries and polytope radius) is stuck at the $\widetilde{O}(n^{2.372\cdots})$, the running time provided by LP solvers that run in current matrix multiplication time [CLS21]. To compare these running times with those for single-commodity maximum flow algorithms on a graph with $|V|$ vertices and $|E|$ edges, observe that in a sparse graph with $|E| = \widetilde{O}(|V|)$, by writing the maximum flow problem as a linear program, we can solve it using general linear program solvers and obtain a running time of $\widetilde{O}(|V|^{2.372\cdots})$, while the state-of-the art maximum flow solver obtains a running time of $\widetilde{O}(|V|^{1.497})$ on such a sparse graph. On dense graphs with $|E| = \Theta(|V|^2)$, the gap is smaller but still substantial: The running time is $\widetilde{O}(|V|^2)$ using maximum flow algorithms vs. $\widetilde{O}(|V|^{2.372\cdots})$ using general LP algorithms.

Numerical stability of linear program solvers. Current research on fast algorithms for linear program generally relies on assuming bounds on (1) the size of the entries and (2) bounds on the norm of all feasible solutions. Generally, algorithm running time depends logarithmically on these quantities, and hence to make these factors negligible, entry size and feasible solution norms are assume to be polynomially bounded, for example in [CLS21]. We will refer to a linear program satisfying these assumptions as *polynomially bounded.*

Suppose we are given a polynomially bounded linear program $\max\{\boldsymbol{c}^\top \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ (also referred to as $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$) with polytope radius at most R such that slightly perturbing the entries of $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}$ by at most δ does not change the feasibility of the the linear program (the supreme of such δ was introduced by Renegar as a condition number in [Ren88]). We wish to compute a vector $\boldsymbol{x} \geq \boldsymbol{0}$ with an ϵ additive error to each constraint and to the optimal value. We can reduce this problem for instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ to a polynomially bounded linear program instance with integral input numbers. Specifically, we round the entries of $\boldsymbol{A}$ down to $\tilde{\boldsymbol{A}}$ and those of $\boldsymbol{b}, \boldsymbol{c}$ up to $\tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}$ all by at most $\min\{\frac{\epsilon}{3R}, \frac{\delta}{R}\}$ such that each entry of $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}$ only needs a logarithmic number of bits. Suppose $\tilde{\boldsymbol{x}}$ is a solution to $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}})$ with $\frac{\epsilon}{3}$ additive error to each constraint and to the optimal value. Then,

$$\begin{aligned}\boldsymbol{A}\tilde{\boldsymbol{x}} &= \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} + (\boldsymbol{A} - \tilde{\boldsymbol{A}})\tilde{\boldsymbol{x}} \le \boldsymbol{b} + \epsilon\boldsymbol{1} \\ \boldsymbol{c}^\top\tilde{\boldsymbol{x}} &\ge \tilde{\boldsymbol{c}}^\top\tilde{\boldsymbol{x}}^* - \frac{2\epsilon}{3}\end{aligned}$$

where $\boldsymbol{1}$ is the all-one vector, $\tilde{\boldsymbol{x}}^*$ is an optimal solution to $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}})$. In addition, the optimal value of $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}})$ is greater than or equal to that of $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$. So, $\tilde{\boldsymbol{x}}$ is a desired solution to $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c})$ with ϵ additive error. Since each entry of $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}$ has a logarithmic number of bits, we can scale all of them to polynomially bounded integers without changing the feasible set and the optimal solutions.

The fastest known solvers for general polynomially bounded linear programs are based on interior point methods [Kar84], and in particular central path methods [Ren88].

How hard is it solve multi-commodity flow? The many successes in developing high-accuracy algorithms for single-commodity flow problems highlight an important open question: Can multi-commodity flow be solved to high accuracy faster than general linear programs? We rule out this possibility, by proving that any linear program (with polynomially bounded entries and polytope radius) can be encoded as a multi-commodity flow problem in nearly-linear time. This implies that any improvement in the running time of (high-accuracy) algorithms for sparse multi-commodity flow problems would directly translate to a faster algorithm for solving sparse linear programs to high accuracy, with only a polylogarithmic increase in running time.

Previous work by Kyng and Zhang [KZ20] had shown that fast algorithms for multi-commodity flow were unlikely to arise from combining interior point methods with special-purpose linear equation solvers. Concretely, they showed that the linear equations that arise in interior point methods for multi-commodity flow are as hard to solve as arbitrary linear equations. This ruled out algorithms following the pattern of the known fast algorithms for high-accuracy single-commodity flow problems. However, it left open the broader question that if some other family of algorithms could succeed. We now show that in general, a separation between multi-commodity flow and linear programming is not possible.

1.1 Previous work

Our paper follows the proof by Itai [Ita78] that linear programming is polynomial-time reducible to 2-commodity flow. However, it is also inspired by recent works on hardness for structured linear equations [KZ20] and packing/covering LPs [KWZ20], which focused on obtaining nearly-linear time reductions in somewhat related settings. These works in turn were motivated by the last decade's substantial progress on fine-grained complexity for a range of polynomial time solvable problems, e.g. see [WW18]. Also notable is the result by Musco et al. [Mus+19] on hardness for matrix spectrum approximation.

1.2 Our contributions

In this paper, we explore the hardness of 2-commodity maximum throughput flow, which for brevity we refer to as the 2-commodity flow problem or 2CF. We relate the difficulty of this problem to that of linear programming (LP). We develop a nearly-linear time, sparsity-preserving polynomial reduction from LP to 2CF, and we show that given an approximate 2CF solution, we can obtain an approximate LP solution with only polynomially larger error.

We say a linear program with N non-zero coefficients is *polynomially bounded* if it has coefficients in the range $[-X, X]$ and $\|\boldsymbol{x}\|_1 \leq R$ for all feasible $\boldsymbol{x}$ (i.e. the polytope of feasible solutions has radius of R in ℓ_1 norm), and $X, R \leq O(N^c)$ for some constant c. In fact, if there exists a feasible solution $\boldsymbol{x}$ satisfying $\|\boldsymbol{x}\|_1 \leq R$, then we can add a constraint $\|\boldsymbol{x}\|_1 \leq R$ to the LP (which can be rewritten as linear inequality constraints) so that in the new LP, all feasible solutions have ℓ_1 norm at most R. This only increases the number of nonzeros in the LP by at most a constant factor.

Theorem 1.1 (Main Theorem (Informal))**.** *Consider any polynomially bounded linear program* $\max\{\boldsymbol{c}^\top \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$ *with integer coefficients and N non-zero entries. In nearly-linear time, we can convert this linear program to a 2-commodity flow problem which is feasible if and only if the original program is. The 2-commodity flow problem has $\widetilde{O}(N)$ edges and has polynomially bounded integral edge capacities. Furthermore, any solution to the 2-commodity flow instance with at most ϵ additive error on each constraint and value at most ϵ from the optimum can be converted to a solution to the original linear program with additive error $\widetilde{O}(\mathrm{poly}(N)\epsilon)$ on each constraint and similarly value within $\widetilde{O}(\mathrm{poly}(N)\epsilon)$ of the optimum.*

This implies that, for any constant $a > 1$, if any 2-commodity flow instance with polynomially bounded integer capacities can be solved with ϵ additive error in time $\widetilde{O}\left(|E|^a \cdot \mathrm{poly}\log(1/\epsilon)\right)$, then any polynomially bounded linear program can be solved to ϵ additive error in time $\widetilde{O}\left(N^a \cdot \mathrm{poly}\log(1/\epsilon)\right)$.

We obtain our result by making several improvements to Itai's reduction from LP to 2CF.

Firstly, while Itai produced a 2CF with the number of edges on the order of $\Theta\left(N^2 \log^2 X\right)$, we show that an improved gadget can reduce this to $O\left(N \log X\right)$. Thus, in the case of polynomially bounded linear programs, where $\log X = O(\log N)$, we get an only polylogarithmic blow-in the number of non-zero entries from N to $\widetilde{O}(N)$, whereas Itai had a increase in number of nonzeros by a factor $\widetilde{O}(N)$, from N to $\widetilde{O}(N^2)$.

Secondly, Itai used very large graph edge capacities that require $O\left(\left(N \log X\right)^{1.01}\right)$ many bits *per edge*, letting the capacities grow exponentially given an LP with polynomially bounded entries. We show that when the feasible polytope radius R is bounded, we can ensure capacities remain a polynomial function of the initial parameters N, R, and X. In the important case of polynomially bounded linear programs, this means the capacities stay polynomially bounded.

Thirdly, while Itai only analyzed the chain of reductions under the case with exact solutions, we generalize the analysis to the case with approximate solutions by establishing an error analysis along the chain of reductions. We show that the error only grows polynomially during the reduction. More precisely, if a 2CF can be solved within ϵ additive error, then we can in nearly-linear time convert this to a solution to the original LP with additive error $O(N^7 R X^{3.01} \epsilon)$. Moreover, to simplify our error analysis, we observe that additional structures can be established in many of Itai's reductions. For instance, we propose the notion of a *fixed flow network*, which consists of a subset of edges with equal lower and upper bound of capacity. It is a simplification of Itai's (l, u) network with general capacity (both lower and upper bounds on the amount of flow).

Open problems. Our reductions do not suffice to prove that a strongly polynomial time algorithm for 2-commodity flow would imply a strongly polynomial time algorithm for linear programming. In a similar vein, it is unclear if a more efficient reduction could exist for the case of linear programs that are not polynomially bounded. We leave these as very interesting open problems.

Finally, our reductions do not preserve the "shape" of the linear program, in particular, a dense linear program may be reduced to a sparse 2-commodity flow problem with a similar number of edges as there are non-zero entries in the original program. It would be interesting to convert a dense linear program into a dense 2-commodity flow problem, e.g. to convert a linear program with m constraints and n variables (say, $m \le n$) into a 2-commodity flow problem with $\widetilde{O}(n)$ edges and $\widetilde{O}(m)$ vertices.

1.3 Organization of the remaining paper

In Section 2, we give some general notations and problem definitions. These definitions include those problems that are involved in the reduction from LP to 2CF. In Section 3, we state our main theorem, and overview the proof that reduces an LP instance to a 2CF instance by a chain of efficient reductions. In Section 4, we provide proof details for all the steps along the chain. In each step, we describe a (nearly-)linear-time method of reducing a problem A to a problem B, and a linear-time method of mapping a solution of B to a solution of A. More importantly, we prove that the size of B is nearly linear in that of A, and an approximate solution to B can be mapped back to an approximate solution to A with a polynomial blow-up in error parameters. In Section 5, we prove the main theorem by putting all intermediate steps together.

2 Preliminaries

2.1 Notation

Matrices and vectors We use parentheses to denote entries of a matrix or a vector: Let $\boldsymbol{A}(i,j)$ denote the (i,j)th entry of a matrix $\boldsymbol{A}$, and let $\boldsymbol{x}(i)$ denote the ith entry of a vector $\boldsymbol{x}$. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, we use $\boldsymbol{a}_i^\top$ to denote the ith row of a matrix $\boldsymbol{A}$ and $\mathrm{nnz}(\boldsymbol{A})$ to denote the number of nonzero entries of $\boldsymbol{A}$. Without loss of generality, we assume that $\mathrm{nnz}(\boldsymbol{A}) \ge \max\{m,n\}$. For any vector $\boldsymbol{x} \in \mathbb{R}^n$, we define $\|\boldsymbol{x}\|_{\max} = \max_{i\in[n]} |\boldsymbol{x}(i)|$, $\|\boldsymbol{x}\|_1 = \sum_{i\in[n]} |\boldsymbol{x}(i)|$. For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, we define $\|\boldsymbol{A}\|_{\max} = \max_{i,j} |\boldsymbol{A}(i,j)|$.

We define a function X that takes an arbitrary number of matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{k_1}$, vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{k_2}$, and scalars $K_1, \ldots, K_{k_3}$ as arguments, and returns the maximum of $\|\cdot\|_{\max}$ of all the arguments, i.e.,

$$\begin{aligned} &X(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{k_1}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{k_2}, K_1, \ldots, K_{k_3}) \\ &\qquad = \max\left\{\|\boldsymbol{A}_1\|_{\max}, \ldots, \|\boldsymbol{A}_{k_1}\|_{\max}, \|\boldsymbol{b}_1\|_{\max}, \ldots, \|\boldsymbol{b}_{k_2}\|_{\max}, |K_1|, \ldots, |K_{k_3}|\right\}. \end{aligned}$$

2.2 Problem Definitions

In this section, we formally define the problems that we use in the reduction. These problems fall into two categories: one category is related to linear programming and linear equations, and the other is related to flow problems in graphs. In addition, we give the error notions used for defining approximate solvers of the two problem categories separately. We defer formal definitions for approximately solving these problems to Section 4.

2.2.1 Linear Programming and Linear Equations with Positive Variables

For the convenience of our reduction, we define linear programming as a "decision" problem. We can solve the optimization problem $\max\{\boldsymbol{c}^\top \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}, \boldsymbol{x} \ge \boldsymbol{0}\}$ by binary searching its optimal value via the decision problem.

Since we are interested in linear programs specified using finite precision coefficients, we assume we are given a linear program with integer coefficients. A linear program with rational coefficients can be converted to a linear program with integer coefficients by multiplying all coefficients with an appropriate common scaling factor.

Definition 2.1 (Linear programming (LP))**.** Given a matrix $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$, vectors $\boldsymbol{b} \in \mathbb{Z}^m$ and $\boldsymbol{c} \in \mathbb{Z}^n$, an integer K, and $R \geq \max\{1, \max\{\|\boldsymbol{x}\|_1 : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}\}$, we refer to the LP problem for $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ as the problem of finding a vector $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ satisfying

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \text{ and } \boldsymbol{c}^\top \boldsymbol{x} \geq K$$

if such an $\boldsymbol{x}$ exists and returning "infeasible" otherwise.

We will reduce a linear program to linear equations with nonnegative variables (LEN) and linear equations with nonnegative variables and small integral coefficients (k-LEN).

Definition 2.2 (Linear Equations with Nonnegative Variables (LEN))**.** Given $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Z}^m$, and $R \geq \max\{1, \max\{\|\boldsymbol{x}\|_1 : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}\}$, we refer to the LEN problem for $(\boldsymbol{A}, \boldsymbol{b}, R)$ as the problem of finding a vector $\boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ satisfying $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ if such an $\boldsymbol{x}$ exists and returning "infeasible" otherwise.

Definition 2.3 (k-LEN (k-LEN))**.** The k-LEN problem is an LEN problem $(\boldsymbol{A}, \boldsymbol{b}, R)$ where the entries of $\boldsymbol{A}$ are integers in $[-k, k]$ for some given $k \in \mathbb{Z}_+$.

To define the approximate version of the above problems, we employ the following additive error notion. Note that the nonnegativity constraint $\boldsymbol{x} \geq \boldsymbol{0}$ remains fulfilled in the approximate version.

1. *Error in objective*: $\boldsymbol{c}^\top \boldsymbol{x} \geq K$ is related to $\boldsymbol{c}^\top \boldsymbol{x} \geq K - \epsilon$;
2. *Error in constraint*:
   - For inequality constraint $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$, it is relaxed to $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \epsilon\boldsymbol{1}$, where $\boldsymbol{1}$ is the all-1 vector;
   - For equality constraint $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, it is relaxed to $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_\infty \leq \epsilon$.

2.2.2 Flow Problems

A *flow network* is a directed graph $G = (V, E)$, where V is the set of vertices and $E \subset V \times V$ is the set of edges, together with a vector of edge capacities $\boldsymbol{u} \in \mathbb{Z}^{|E|}_{>0}$ that upper bound the amount of flow passing each edge. A *2-commodity flow network* is a flow network together with two source-sink pairs $s_i, t_i \in V$ for each commodity $i \in \{1, 2\}$.

Given a 2-commodity flow network $(G = (V, E), \boldsymbol{u}, s_1, t_1, s_2, t_2)$, a *feasible 2-commodity flow* is a pair of flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \in \mathbb{R}^{|E|}_{\geq 0}$ that satisfies

1. capacity constraint: $\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e),\ \forall e \in E$, and
2. conservation of flows: $\sum_{u:(u,v)\in E} \boldsymbol{f}_i(u, v) = \sum_{w:(v,w)\in E} \boldsymbol{f}_i(v, w),\ \forall i \in \{1, 2\}, v \in V \setminus \{s_i, t_i\}$[1].

For each commodity flow $\boldsymbol{f}_i$, we let $F_i \boldsymbol{d}_i$ be the demand vector of commodity i, where $\boldsymbol{d}_i \in \mathbb{R}^{|V|}$ such that $\boldsymbol{d}_i(s_i) = -1$, $\boldsymbol{d}_i(t_i) = 1$, and $\boldsymbol{d}_i(v) = 0$ for all $v \in V\backslash\{s_i, t_i\}$.

Similar to the definition of LP, we define 2-commodity flow problem as a decision problem. We can solve a decision problem by solving the corresponding optimization problem.

[1]Note that for commodity i, this constraint includes the case of $v \in \{s_{\bar{i}}, t_{\bar{i}}\}, \bar{i} = \{1, 2\}\backslash i$.

Definition 2.4 (2-Commodity Flow Problem (2CF))**.** Given a 2-commodity flow network $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ together with $R \geq 0$, we refer to the 2CF problem for $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R)$ as the problem of finding a feasible 2-commodity flow $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \boldsymbol{0}$ satisfying

$$F_1 + F_2 \geq R$$

if such flows exist and returning "infeasible" otherwise.

To reduce LP to 2CF, we need a sequence of variants of flow problems.

Definition 2.5 (2-Commodity Flow with Required Flow Amount (2CFR))**.** Given a 2-commodity flow network $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ together with $R_1, R_2 \geq 0$, we refer to the 2CFR for $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R_1, R_2)$ as the problem of finding a feasible 2-commodity flow $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \boldsymbol{0}$ satisfying

$$F_1 \geq R_1, \quad F_2 \geq R_2$$

if such flows exist and returning "infeasible" otherwise.

Definition 2.6 (Fixed flow constraints)**.** Given a set $F \subseteq E$ in a 2-commodity flow network, we say the flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \boldsymbol{0}$ satisfy *fixed flow constraints on* F if

$$\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) = \boldsymbol{u}(e), \ \forall e \in F.$$

Similarly, given a set $F \subseteq E$ in a 1-commodity flow network, we say the flow $\boldsymbol{f} \geq \boldsymbol{0}$ satisfies *fixed flow constraints on* F if

$$\boldsymbol{f}(e) = \boldsymbol{u}(e), \ \forall e \in F.$$

Definition 2.7 (2-Commodity Fixed Flow Problem (2CFF))**.** Given a 2-commodity flow network $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ together with a subset of edges $F \subseteq E$, we refer to the 2CFF problem for the tuple $(G, F, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ as the problem of finding a feasible 2-commodity flow $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \boldsymbol{0}$ which also satisfies the fixed flow constraints on F if such flows exist and returning "infeasible" otherwise.

Definition 2.8 (Selective Fixed Flow Problem (SFF))**.** Given a 2-commodity network $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ together with three edge sets $F, S_1, S_2 \subseteq E$, we refer to the SFF problem for $(G, F, S_1, S_2, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ as the problem of finding a feasible 2-commodity flow $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \boldsymbol{0}$ such that for each $i \in \{1, 2\}$, flow $\boldsymbol{f}_i(e) > 0$ only if $e \in S_i$, and $\boldsymbol{f}_1, \boldsymbol{f}_2$ satisfy the fixed flow constraints on F, if such flows exist, and returning "infeasible" otherwise.

Definition 2.9 (Fixed Homologous Flow Problem (FHF))**.** Given a flow network with a single source-sink pair $(G, \boldsymbol{u}, s, t)$ together with a collection of disjoint subsets of edges $\mathcal{H} = \{H_1, \ldots, H_h\}$ and a subset of edges $F \subseteq E$ such that F is disjoint from all the sets in $\mathcal{H}$, we refer to the FHF problem for $(G, F, \mathcal{H}, \boldsymbol{u}, s, t)$ as the problem of finding a feasible flow $\boldsymbol{f} \geq \boldsymbol{0}$ such that

$$\boldsymbol{f}(e_1) = \boldsymbol{f}(e_2), \quad \forall e_1, e_2 \in H_k, 1 \leq k \leq h,$$

and $\boldsymbol{f}$ satisfies the fixed flow constraints on F, if such flows exist, and returning "infeasible" otherwise.

Definition 2.10 (Fixed Pair Homologous Flow Problem (FPHF))**.** The FPHF is an FHF problem $(G, F, \mathcal{H}, \boldsymbol{u}, s, t)$ where every set in $\mathcal{H}$ has size 2.

Now, we give error notions for the above flow problems. Again, note that the nonnegativity constraint $\boldsymbol{f} \geq \boldsymbol{0}$ remains fulfilled in the approximate version.

1. *Error in congestion*: the capacity constraints are relaxed to:

$$\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \le \boldsymbol{u}(e) + \epsilon, \quad \forall e \in E.$$

It has several variants corresponding to different flow problems.

   - If $e \in F$ is a fixed-flow edge, the fixed-flow constraints are related to

$$\boldsymbol{u}(e) - \epsilon \le \boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \le \boldsymbol{u}(e) + \epsilon, \quad \forall e \in F$$

   - If G is a 1-commodity flow network, we replace $\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e)$ by $\boldsymbol{f}(e)$.

2. *Error in demand*: the conservation of flows is relaxed to

$$\left| \sum_{u:(u,v)\in E} \boldsymbol{f}_i(u,v) - \sum_{w:(v,w)\in E} \boldsymbol{f}_i(v,w) \right| \le \epsilon, \ \forall v \in V \backslash \{s_i, t_i\}, i \in \{1,2\} \tag{1}$$

It has several variants corresponding to different flow problems.

   - If the problem is with flow requirement F_i, then besides Eq. (1), we add demand constraints for s_i and t_i with respect to commodity i:

$$\left| \sum_{w:(s_i,w)\in E} \boldsymbol{f}_i(s_i,w) - F_i \right| \le \epsilon, \qquad \left| \sum_{u:(u,t_i)\in E} \boldsymbol{f}_i(u,t_i) - F_i \right| \le \epsilon, \ i \in \{1,2\} \tag{2}$$

   - If G is a 1-commodity flow network, Eq. (1) can be simplified as

$$\left| \sum_{u:(u,v)\in E} \boldsymbol{f}(u,v) - \sum_{w:(v,w)\in E} \boldsymbol{f}(v,w) \right| \le \epsilon, \ \forall v \in V \backslash \{s, t\}$$

3. *Error in type*: the selective constraints are relaxed to

$$\boldsymbol{f}_{\bar{i}}(e) \le \epsilon, \quad \forall e \in S_i, \ \bar{i} = \{1,2\} \backslash i.$$

4. *Error in (pair) homology*: the (pair) homologous constraints are relaxed to

$$|\boldsymbol{f}(e_1) - \boldsymbol{f}(e_2)| \le \epsilon, \ \forall e_1, e_2 \in H_k, H_k \in \mathcal{H}.$$

3 Main results

Theorem 3.1. *Given an* LPA *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon^{lp})$ *where* $\boldsymbol{A} \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in Z^m, \boldsymbol{c} \in \mathbb{Z}^n, K \in \mathbb{Z}$ *and* $\boldsymbol{A}$ *has* $\mathrm{nnz}(\boldsymbol{A})$ *nonzero entries, we can reduce it to a* 2CFA *instance* $(G = (V,E), \boldsymbol{u}, s_1, t_1, s_2, t_2, R^{2cf}, \epsilon^{2cf})$*, in time* $O(\mathrm{nnz}(\boldsymbol{A}) \log X)$ *where* $X = X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K)$*, such that*

$$\begin{aligned} &|V|, |E| = O(\mathrm{nnz}(\boldsymbol{A}) \log X), \\ &\|\boldsymbol{u}\|_{\max}, R^{2cf} = O(\mathrm{nnz}^3(\boldsymbol{A}) R X^2 \log^2 X), \\ &\epsilon^{2cf} = \Omega\left(\frac{1}{\mathrm{nnz}^7(\boldsymbol{A}) R X^3 \log^6 X}\right) \epsilon^{lp}. \end{aligned}$$

If the LP *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ *has a solution, then the* 2CF *instance* $(G^{2cf}, \boldsymbol{u}^{2cf}, s_1, t_1, s_2, t_2, R^{2cf})$ *has a solution. Furthermore, if* $\boldsymbol{f}^{2cf}$ *is a solution to the* 2CF(A) *instance, then in time* $O(\mathrm{nnz}(\boldsymbol{A}) \log X)$*, we can compute a solution* $\boldsymbol{x}$ *to the* LP(A) *instance, where the exact case holds when* $\epsilon^{2cf} = \epsilon^{lp} = 0$.

Our main theorem immediately implies the following corollary.

Corollary 3.2. *If we can solve any* 2CFA *instance* $(G = (V, E), \boldsymbol{u}, s_1, t_1, s_2, t_2, R^{2cf}, \epsilon)$ *in time* $O\left(|E|^c \operatorname{poly}\log\left(\frac{\|\boldsymbol{u}\|_1}{\epsilon}\right)\right)$ *for some small constant* $c \geq 1$*, then we can solve any* LPA *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon)$ *in time* $O\left(\operatorname{nnz}^c(\boldsymbol{A}) \operatorname{poly}\log\left(\frac{\operatorname{nnz}(\boldsymbol{A}) R X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K)}{\epsilon}\right)\right)$.

3.1 Overview of our proof

We give a summary of notations used in the reduction from LP(A) to 2CF(A), as shown in Table 1.

In this section, we will explain how to reduce an LP instance to a 2-commodity flow (2CF) instance by a chain of efficient reductions between different problems. In each step, we reduce a decision problem A to a decision problem B; we guarantee that: (1) the reduction runs in nearly linear time[2], (2) the size of B is nearly linear in that of A, and (3) that A is feasible implies that B is feasible, and an approximate solution to B can be turned to an approximate solution to A with only a polynomial blow-up in error parameters, in linear time.

We follow the outline of Itai's reduction [Ita78]. Itai first reduced an LP instance to a 1-LEN instance (linear equations with nonnegative variables and ± 1 coefficients). A 1-LEN instance can be represented by a single-commodity flow problem subject to additional homologous constraints and fixed flow constraints (i.e., FHF). Then, Itai dropped these additional constraints step by step, via introducing a second commodity of flow and imposing lower bound requirements on the total amount of flows routed between the source-sink pairs. However, in the worst case, Itai's reduction from 1-LEN to FHF enlarges the problem size quadratically and is thus inefficient. One of our main contributions is to improve this step so that the sparsity is preserved along the reduction chain.

Our second main contribution is an upper bound on the errors accumulated during the process of mapping an approximate solution to the 2CFA instance to an approximate solution to the LPA instance. We show that the error only grows by polynomial factors. Itai only considered exact solutions between these two instances, and showed that exactly solving LP and 2CF are (polynomially) (up to polynomial factors) equivalent. Our analysis on approximate solutions also implies that approximately solving LPA and 2CFA are (up to polylogarithmic factors) equivalent for algorithms whose running time scales as $\operatorname{polylog}(1/\epsilon)$ when computing an ϵ error solution.

We will explain the reductions based on the exact versions of the problems. At the end of this section, we will discuss some intuitions of behind our error analysis.

3.1.1 Reducing Linear Programming to Linear Equations with Nonnegative Variables and ± 1 Coefficients

Given an LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ where $R \geq \max\{1, \max\{\|x\|_1 : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}\}$, we want to compute a vector $\boldsymbol{x} \geq \boldsymbol{0}$ satisfying

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{c}^\top \boldsymbol{x} \geq K$$

or to correctly declare infeasible. We introduce slack variables $\boldsymbol{s}, \alpha \geq \boldsymbol{0}$ and turn the above inequalities to equalities:

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b}, \boldsymbol{c}^\top \boldsymbol{x} - \alpha = K$$

which is an LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$. Comparing to Itai's proof, we need to track two additional parameters: R, the polytope radius, and X, the maximum magnitude of all numbers appearing in the problem instance specification.

[2]Linear in the size of problem A, poly-logarithmic in the maximum magnitude of all the numbers that describe A, the feasible set radius, and the inverse of the error parameter if an approximate solution is allowed.

[3]Error parameters will be defined in Section 4.

Table 1: A summary of notations used in the reduction from LP(A) to 2CF(A). The column "Input" and "Output" are shared for both exact and approximate problems. The column "Error" is only for approximate problems.

Exact problem	Input	Output	Approximate problem	Error [3]
LP (Def. 2.1)	$\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R$	$\boldsymbol{x}$	LPA (Def. 4.2)	ϵ^{lp}
LEN (Def. 2.2)	$\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R}$	$\tilde{\boldsymbol{x}}$	LENA (Def. 4.3)	ϵ^{le}
2-LEN (Def. 2.3)	$\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R}$	$\bar{\boldsymbol{x}}$	2-LENA (Def. 4.6)	ϵ^{2le}
1-LEN (Def. 2.3)	$\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R}$	$\hat{\boldsymbol{x}}$	1-LENA (Def. 4.6)	ϵ^{1le}
FHF (Def. 2.9)	$G^h, F^h, \mathcal{H}^h = \{H_1, \cdots, H_h\}, \boldsymbol{u}^h, s, t$	$\boldsymbol{f}^h$	FHFA (Def. 4.11)	ϵ^h
FPHF (Def. 2.10)	$G^p, F^p, \mathcal{H}^p = \{H_1, \cdots, H_p\}, \boldsymbol{u}^p, s, t$	$\boldsymbol{f}^p$	FPHFA (Def. 4.14)	ϵ^p
SFF (Def. 2.8)	$G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2$	$\boldsymbol{f}^s$	SFFA (Def. 4.17)	ϵ^s
2CFF (Def. 2.7)	$G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2$	$\boldsymbol{f}^f$	2CFFA (Def. 4.20)	ϵ^f
2CFR (Def. 2.5)	$G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2$	$\boldsymbol{f}^r$	2CFRA (Def. 4.23)	ϵ^r
2CF (Def. 2.4)	$G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf}$	$\boldsymbol{f}^{2cf}$	2CFA (Def. 4.27)	ϵ^{2cf}

We then reduce the LEN instance to linear equations with ± 2 coefficients (2-LEN) by bitwise decomposition. For each bit, we need to introduce a carry term, represented as a difference between two nonnegative variables. The following is an example.

$$5x_1 + 3x_2 - 7x_3 = -1$$

$$\Downarrow$$

$$(2^0 + 2^2)x_1 + (2^0 + 2^1)x_2 - (2^0 + 2^1 + 2^2)x_3 = -2^0$$

$$\Downarrow$$

$$(x_1 + x_2 - x_3)2^0 + (x_2 - x_3)2^1 + (x_1 - x_3)2^2 = -1 \cdot 2^0$$

It can be decomposed to 3 linear equations, together with carry term $(c_i - d_i)$, where $c_i, d_i \geq 0$:

$$\begin{aligned} x_1 + x_2 - x_3 - 2(c_0 - d_0) &= -1 \\ x_2 - x_3 + (c_0 - d_0) - 2(c_1 - d_1) &= 0 \\ x_1 - x_3 + (c_1 - d_1) &= 0 \end{aligned}$$

In contrast to Itai's reduction, we impose an upper bound for each carry variable. We show that this upper bound does not change problem feasibility and it guarantees the polytope radius only increases polynomially. Next, we reduce the 2-LEN instance to a 1-LEN instance by replacing each variable with coefficient ± 2 by two new equal-valued variables.

All the above three reduction steps run in nearly linear time, and the problem sizes increase nearly linearly.

3.1.2 Reducing Linear Equations with Nonnegative Variables and ± 1 Coefficients to Fixed Homologous Flow Problem

One of our main contributions is a linear-time reduction from 1-LEN to FHF (single-commodity fixed homologous flow problem). Our reduction is similar to Itai's reduction, but more efficient.

Itai observed that a linear equation $\boldsymbol{a}^\top \boldsymbol{x} = b$ with ± 1 coefficients can be represented as a fixed homologous flow network G. G has a source vertex s, a sink vertex t, and two additional vertices

J^+ and J^-. Each variable $\boldsymbol{x}(i)$ corresponds to an edge: There is an edge from s to J^+ if $\boldsymbol{a}(i) = 1$, and an edge from s to J^- if $\boldsymbol{a}(i) = -1$. The amount of flow passing this edge corresponds to the value of $\boldsymbol{x}(i)$. The difference between the total amount of flow entering J^+ and that entering J^- equals to $\boldsymbol{a}^\top \boldsymbol{x}$. To force $\boldsymbol{a}^\top \boldsymbol{x} = b$, we add two edges e_1, e_2 from J^+ to t and one edge e_3 from J^- to t [4]; we require e_1 and e_3 to be homologous and require e_2 to be a fixed flow with value b.

The above construction can be generalized to a system of linear equations (see Figure 1 in Section 4.4). Specifically, we create a gadget as above for each equation i, and then glue all the source (sink) vertices for each equation together as the source (sink, respectively) of the graph. To force the consistency of the variable values, we require the edges corresponding to the same variable in different equations to be homologous. The number of the vertices is linear in the number of equations; the number of the edges and the total size of the homologous sets are both linear in the number of nonzero coefficients of the linear equation system.

3.1.3 Dropping the Homologous and Fixed Flow Constraints

To reduce FHF to 2CF (2-commodity flow problem), we need to drop the homologous and fixed flow constraints. The reduction has three main steps.

Reducing FHF to SFF. Given an FHF instance, we can reduce it to a fixed homologous flow instance in which each homologous edge set has size 2 (i.e., FPHF). To drop the homologous requirement in FPHF, we introduce a second commodity of flow with source-sink pair (s_2, t_2), and for each edge, we carefully select the type(s) of flow that can pass through this edge. Specifically, given two homologous edges (v, w) and (y, z), we construct a constant-sized gadget (see Figure 4 in Section 4.6): We introduce new vertices vw, vw', yz, yz', construct a directed path $P : s_2 \to vw \to vw' \to yz \to yz' \to t_2$, and add edges $(v, vw), (vw', w)$ and $(y, yz), (yz', z)$. Now, there is a directed path $P_{vw} : v \to vw \to vw' \to w$ and a directed path $P_{yz} : y \to yz \to yz' \to z$. Paths P and P_{vw} (P_{yz}) share an edge $e_{vw} = (vw, vw')$ ($e_{yz} = (yz, yz')$, respectively). We select e_{vw} and e_{yz} for both flow $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, select the rest of the edges along P for only $\boldsymbol{f}_2$, and select the rest of the edges along P_{vw}, P_{yz} for only $\boldsymbol{f}_1$. By this construction, in this gadget, we have $\boldsymbol{f}_2(e_{vw}) = \boldsymbol{f}_2(e_{yz})$ being the amount of flow routed in P, $\boldsymbol{f}_1(e_{vw})$ and $\boldsymbol{f}_1(e_{yz})$ being the amount of flow routed in P_{vw} and P_{yz}, respectively. Next, we choose e_{vw} and e_{yz} to be fixed flow edges with equal capacity; this guarantees the same amount of $\boldsymbol{f}_1$ is routed through P_{vw} and P_{yz}. The new graph is an SFF instance.

Reducing SFF to 2CFF. Next, we will drop the selective requirement of the SFF instance. For each edge (x, y) selected for flow i, we construct a constant-sized gadget (see Figure 5 in Section 4.7): We introduce two vertices xy, xy', construct a direct path $s_i \to xy' \to xy \to t_i$, and add edge (x, xy) and (xy', y). This gadget simulates a directed path from x to y for flow $\boldsymbol{f}_i$, and guarantees no directed path from x to y for flow $\boldsymbol{f}_{\bar{i}}$ so that $\boldsymbol{f}_{\bar{i}}$ cannot be routed from x to y. We get a 2CFF instance.

Reducing 2CFF to 2CF. It remains to drop the fixed flow constraints. The gadget we will use is similar to that used in the last step. We first introduce new sources $\bar{s}_1, \bar{s}_2$ and sinks $\bar{t}_1, \bar{t}_2$. Then, for each edge (x, y) with capacity u, we construct a constant-sized gadget (see Figure 6 in Section 4.8). We introduce two vertices xy, xy', add edges $(\bar{s}_1, xy'), (\bar{s}_2, xy'), (xy, \bar{t}_1), (xy, \bar{t}_2), (xy', xy)$, and $(x, xy), (xy', y)$. This simulates a directed path from x to y that both flow $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ can pass

[4] We remark that e_1, e_2 are multi-edges, but after the next reduction step, we will get a simple graph.

through. We let (xy', xy) have capacity u if (x, y) is a fixed flow edge and $2u$ otherwise; we let all the other edges have capacity u. Assume all the edges incident to the sources and the sinks are saturated, then the total amount of flows routed from x to y in this gadget must be u if (x, y) is a fixed flow edge and no larger than u otherwise. Moreover, since the original sources and sinks are no longer sources and sinks now, we have to satisfy the conservation of flows at these vertices. For each $i \in \{1, 2\}$, we create a similar gadget involving $\bar{s}_i, \bar{t}_i$ to simulate a directed path from t_i to s_i (the original sink and source), and let the edges incident to $\bar{s}_i, \bar{t}_i$ have capacity M, the sum of all the edge capacities in the 2CFF instance. This gadget guarantees that assuming the edges incident to $\bar{s}_i$ and $\bar{t}_i$ are saturated, the amount of flow routed from t_i to s_i through this gadget can be any number at most M. To force the above edge-saturation assumptions to hold, we require the amount of flow $\boldsymbol{f}_i$ routed from $\bar{s}_i$ to $\bar{t}_i$ to be no less than $2M$ for each $i \in \{1, 2\}$.

Now, this instance is close to a 2CF instance except that we require a lower bound for each flow value instead of a lower bound for the sum of two flow values. To handle this, we introduce new sources $\bar{\bar{s}}_1, \bar{\bar{s}}_2$ and for each $i \in \{1, 2\}$, we add an edge $(\bar{\bar{s}}_i, \bar{s}_i)$ with capacity $2M$, the lower bound required for the value of $\boldsymbol{f}_i$.

One can check that in each reduction step, the reduction time is nearly linear and the problem size increases nearly linearly. In addition, given a solution to the 2CF instance, one can construct a solution to the LP instance in nearly linear time.

We also establish an error bound for mapping an approximate solution to 2CFA to an approximate solution to LPA. We focus on the intuition of error analysis of flow problems. Even though we only maintain one error parameter per problem, we keep track with multiple types of error separately, (e,g., error in congestion, demand, selective types, and homology), depending on the problem settings. And we set the error parameter to be the error notion with the largest value.

Suppose we reduce problem A to problem B with a certain gadget, then when we map a solution to B back to a solution to A, it is observed that each error notion of A is an additive accumulation of multiple error notions of B. It is because we have to map the flows of B passing through a gadget including multiple edges back to a flow of A passing a single edge. Each time we remove an edge, various errors related to this edge and incident vertices get transferred to its neighbors. Thus, the total error accumulation by the solution mapping can be polynomially bounded by the number of edges. So, the final error only increases polynomially.

Contents

4 Proof details

The chain of reductions from LP(A) to 2CF(A) consists of nine steps. In each step, we analyze the reduction from some problem A to some problem B in both the exact case and the approximate case. In the exact case, we start with describing a nearly-linear-time method of reducing A to B, and a nearly-linear-time method of mapping a solution of B back to a solution of A. Then, we prove the correctness of the reduction method in the forward direction, that is if A has a solution then B has a solution. In addition, we provide a fine-grained analysis of the size of B given the size of A.

In the approximate case, we formally define approximately solving each problem in the first place, specifying bounds on various different types of error. We always use the same reduction method from problem A to problem B in the exact and approximate cases. Thus the conclusion that problem B has a feasible solution when problem A has a feasible solution also applies in the approximate case.

We also always use a solution map back for the approximate case that agrees with the exact case when there is no error. Crucially, we conduct an error analysis. That is given an approximate solution to B and its error parameters (by abusing notations, we use ϵ^B to denote), we map it back to an approximate solution to A and measure its error τ^A with respect to ϵ^B. In other words, we can reduce an approximate version of A with error parameters $\epsilon^A \geq \tau^A$ to an approximate version of B with error parameters ϵ^B. Note that the correctness of the reduction method in the backward direction for the exact case follows from the approximate case analysis by setting all error parameters to zero, which completes the proof of correctness.

4.1 LP(A) to LEN(A)

4.1.1 LP to LEN

We show the reduction from an LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ to an LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$. The LP instance has the following form:

$$\begin{aligned} \boldsymbol{c}^\top \boldsymbol{x} &\geq K \\ \boldsymbol{A}\boldsymbol{x} &\leq \boldsymbol{b} \\ \boldsymbol{x} &\geq \boldsymbol{0} \end{aligned} \tag{3}$$

To reduce it to an LEN instance, we introduce slack variables α and $\boldsymbol{s}$:

$$\begin{aligned} \begin{pmatrix} \boldsymbol{c}^\top & \boldsymbol{0} & -1 \\ \boldsymbol{A} & \boldsymbol{I} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \alpha \end{pmatrix} &= \begin{pmatrix} K \\ \boldsymbol{b} \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \alpha \end{pmatrix} &\geq \boldsymbol{0} \end{aligned} \tag{4}$$

Setting

$$\tilde{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{c}^\top & \boldsymbol{0} & -1 \\ \boldsymbol{A} & \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}, \qquad \tilde{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{s} \\ \alpha \end{pmatrix}, \qquad \tilde{\boldsymbol{b}} = \begin{pmatrix} K \\ \boldsymbol{b} \end{pmatrix},$$

we get an LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$ where $\tilde{R} = \max\{1, \max\{\|\tilde{\boldsymbol{x}}\|_1 : \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{x}} \geq \boldsymbol{0}\}\}$.

If an LEN solver returns $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, \boldsymbol{s}^\top, \alpha)^\top$ for the LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$, then we return $\boldsymbol{x}$ for the LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$; if the LEN solver returns "infeasible" for the LEN instance, then we return "infeasible" for the LP instance.

Lemma 4.1 (LP to LEN)**.** *Given an* LP *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ *where* $\boldsymbol{A} \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n, K \in \mathbb{Z}$*, we can construct, in* $O(\mathrm{nnz}(\boldsymbol{A}))$ *time, an* LEN *instance* $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$ *where* $\tilde{\boldsymbol{A}} \in \mathbb{Z}^{\tilde{m}\times\tilde{n}}, \tilde{\boldsymbol{b}} \in \mathbb{Z}^{\tilde{m}}$ *such that*

$$\tilde{n} = n + m + 1, \quad \tilde{m} = m + 1, \quad \mathrm{nnz}(\tilde{\boldsymbol{A}}) \leq 4\,\mathrm{nnz}(\boldsymbol{A}),$$

$$\tilde{R} = 5mRX(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K), \quad X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}) = X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K) \geq 1.$$

If the LP *instance has a solution, then the* LEN *instance has a solution.*

Remark. In the backward direction, if the LEN instance has a solution, then in time $O(n)$, we can map this solution back to a solution to the LP instance. It is proven in Lemma 4.4 by setting $\epsilon^{lp} = \epsilon^{le} = 0$.

This solution mapping conclusion is also true for all the rest steps in the chain of reductions.

Proof. Based on the reduction method described above, if $\boldsymbol{x}$ is a solution to the LP instance as shown in Eq. (3), we can derive a solution $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, \boldsymbol{s}^\top, \alpha)^\top$ to the LEN instance as shown in Eq. (4), by setting

$$\boldsymbol{s} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}, \qquad \alpha = \boldsymbol{c}^\top \boldsymbol{x} - K.$$

Thus, if the LP instance has a solution, then the LEN instance has a solution.

Given the size of the LP instance with n variables, m linear constraints, and $\mathrm{nnz}(\boldsymbol{A})$ nonzero entries, we observe the size of the reduced LEN instance as following:

1. $\tilde{n}$ variables, where $\tilde{n} = n + m + 1$.
2. $\tilde{m}$ linear constraints, where $\tilde{m} = m + 1$.
3. $\mathrm{nnz}(\tilde{\boldsymbol{A}})$ nonzeros, where

$$\mathrm{nnz}(\tilde{\boldsymbol{A}}) = \mathrm{nnz}(\boldsymbol{A}) + \mathrm{nnz}(\boldsymbol{c}) + m + 1 \leq 4\,\mathrm{nnz}(\boldsymbol{A}), \tag{5}$$

   where we use $\mathrm{nnz}(\boldsymbol{A}) \geq m, n \geq 1$, and $\mathrm{nnz}(\boldsymbol{c}) \leq n$.

4. $\tilde{R} = \max\{1, \max\{\|\tilde{\boldsymbol{x}}\|_1 : \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{x}} \geq \boldsymbol{0}\}\}$, the radius of polytope in ℓ_1 norm. Our goal is to upper bound $\|\tilde{\boldsymbol{x}}\|_1$ for every feasible solution to the LEN instance. By definition and the triangle inequality,

$$\|\tilde{\boldsymbol{x}}\|_1 \leq \|\boldsymbol{x}\|_1 + \alpha + \|\boldsymbol{s}\|_1.$$

   Note $\|\boldsymbol{x}\|_1 \leq R$, the polytope radius in the LP instance. In addition,

$$\alpha = \boldsymbol{c}^\top\boldsymbol{x} - K \leq \|\boldsymbol{c}\|_{\max} \|\boldsymbol{x}\|_1 + |K| \leq \|\boldsymbol{c}\|_{\max} R + |K|,$$

$$\|\boldsymbol{s}\|_1 = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1 \leq \|\boldsymbol{A}\boldsymbol{x}\|_1 + \|\boldsymbol{b}\|_1 \leq m\,\|\boldsymbol{A}\|_{\max} \|\boldsymbol{x}\|_1 + \|\boldsymbol{b}\|_1 \leq m(\|\boldsymbol{A}\|_{\max} R + \|\boldsymbol{b}\|_{\max}).$$

   Therefore, we have

$$\begin{aligned} \|\tilde{\boldsymbol{x}}\|_1 &\leq R + |K| + \|\boldsymbol{c}\|_{\max} R + m(\|\boldsymbol{A}\|_{\max} R + \|\boldsymbol{b}\|_{\max}) \\ &\leq mR(1 + |K| + \|\boldsymbol{c}\|_{\max} + \|\boldsymbol{A}\|_{\max} + \|\boldsymbol{b}\|_{\max}) && \text{Because } R \geq 1 \\ &\leq 5mRX(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K). \end{aligned}$$

   Hence, it suffices to set

$$\tilde{R} = 5mRX(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K).$$

5. $X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}) = X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K)$ because

$$\left\|\tilde{\boldsymbol{A}}\right\|_{\max} = \max\left\{\|\boldsymbol{A}\|_{\max}, \|\boldsymbol{c}\|_{\max}, 1\right\} = \max\left\{\|\boldsymbol{A}\|_{\max}, \|\boldsymbol{c}\|_{\max}\right\},$$

$$\left\|\tilde{\boldsymbol{b}}\right\|_{\max} = \max\left\{|K|, \|\boldsymbol{b}\|_{\max}\right\}.$$

To estimate the reduction time, as it takes $O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}})\right)$ time to construct $\tilde{\boldsymbol{A}}$, and $O\left(\mathrm{nnz}(\tilde{\boldsymbol{b}})\right)$ time to construct $\tilde{\boldsymbol{b}}$, thus the reduction takes time

$$\begin{aligned} O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}}) + \mathrm{nnz}(\tilde{\boldsymbol{b}})\right) &= O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}})\right) && \text{Because } \mathrm{nnz}(\tilde{\boldsymbol{b}}) \le m \le \mathrm{nnz}(\tilde{\boldsymbol{A}}) \\ &= O\left(\mathrm{nnz}(\boldsymbol{A})\right). && \text{By Eq. (5)} \end{aligned}$$

□

4.1.2 LPA to LENA

The above lemma shows the reduction between exactly solving an LP instance and exactly solving an LEN instance. Next, we generalize the case with exact solutions to the case that allows approximate solutions. First of all, we give a definition of the approximate version of LP and LEN.

Definition 4.2 (LP Approximate Problem (LPA))**.** An LPA instance is given by an LP instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon)$. We say an algorithm solves the LPA problem, if, given any LPA instance, it returns a vector $\boldsymbol{x} \ge \boldsymbol{0}$ such that

$$\begin{aligned} \boldsymbol{c}^\top \boldsymbol{x} &\ge K - \epsilon \\ \boldsymbol{A}\boldsymbol{x} &\le \boldsymbol{b} + \epsilon \boldsymbol{1} \end{aligned}$$

where $\boldsymbol{1}$ is the all-1 vector, or it correctly declares that the associated LP instance is infeasible.

Remark. Note that our definition of LPA does not require the algorithm to provide a certificate of infeasibility – but our notion of an *algorithm* for LPA requires the algorithm never incorrectly asserts infeasibility. Also note that when the LP instance is infeasible, the algorithm is still allowed to return an approximately feasible solution, if it finds one. We use the same approach to defining all our approximate decision problems.

Definition 4.3 (LEN Approximate Problem (LENA))**.** An LENA instance is given by an LEN instance $(\boldsymbol{A}, \boldsymbol{b}, R)$ as in Definition 2.2 and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(\boldsymbol{A}, \boldsymbol{b}, R, \epsilon)$. We say an algorithm solves the LENA problem, if, given any LENA instance, it returns a vector $\boldsymbol{x} \ge \boldsymbol{0}$ such that

$$|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}| \le \epsilon \boldsymbol{1},$$

where $|\cdot|$ is entrywise absolute value and $\boldsymbol{1}$ is the all-1 vector, or it correctly declares that the associated LEN instance is infeasible.

We use the same reduction method in the exact case to reduce an LPA instance to an LENA instance. Furthermore, if an LENA solver returns $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, \boldsymbol{s}^\top, \alpha)^\top$ for the LENA instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R}, \epsilon^{le})$, then we return $\boldsymbol{x}$ for the LPA instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon^{lp})$; if the LENA solver returns "infeasible" for the LENA instance, then we return "infeasible" for the LPA instance.

Hence, the conclusions in the exact case (Lemma 4.1) also apply here, including the reduction time, problem size, and that the LEN instance has a feasible solution when the LP instance has one. In the approximate case, it remains to show the solution mapping time, as well as how the problem error changes by mapping an approximate solution to the LENA instance back to an approximate solution to the LPA instance.

Lemma 4.4 (LPA to LENA)**.** *Given an* LPA *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon^{lp})$ *where* $\boldsymbol{A} \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in \mathbb{Z}^m, \boldsymbol{c} \in \mathbb{Z}^n, K \in \mathbb{Z}$, *if we use Lemma 4.1 to construct an* LEN *instance* $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$ *from the* LP *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$*, and let*

$$\epsilon^{le} = \epsilon^{lp},$$

then the LENA *instance* $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R}, \epsilon^{le})$ *satisfies.*

Furthermore, if $\tilde{\boldsymbol{x}}$ *is a solution to the* LEN(A) *instance, then in time* $O(n)$*, we can compute a solution* $\boldsymbol{x}$ *to the* LP(A) *instance, where the exact case holds when* $\epsilon^{le} = \epsilon^{lp} = 0$.

Proof. Based on the solution mapping method described above, given a solution $\tilde{\boldsymbol{x}}$, we discard those entries of $\boldsymbol{s}$ and α, and map back trivially for those entries of $\boldsymbol{x}$. As it takes constant time to set the value of each entry of $\boldsymbol{x}$ by mapping back trivially, and the size of $\boldsymbol{x}$ is n, thus the solution mapping takes $O(n)$ time.

If $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, \boldsymbol{s}^\top, \alpha)^\top$ is a solution to the LENA instance, then by Definition 4.3, $\tilde{\boldsymbol{x}}$ satisfies

$$\begin{aligned} \left|\boldsymbol{c}^\top \boldsymbol{x} - \alpha - K\right| &\le \epsilon^{le}, \\ |\boldsymbol{A}\boldsymbol{x} + \boldsymbol{s} - \boldsymbol{b}| &\le \epsilon^{le}\mathbf{1}. \end{aligned}$$

Taking one direction of the absolution value, we obtain

$$\begin{aligned} \boldsymbol{c}^\top \boldsymbol{x} &\ge \alpha + K - \epsilon^{le} \ge K - \epsilon^{le}, \\ \boldsymbol{A}\boldsymbol{x} &\le -\boldsymbol{s} + \boldsymbol{b} + \epsilon^{le}\mathbf{1} \le \boldsymbol{b} + \epsilon^{le}\mathbf{1}, \end{aligned}$$

As we set in the reduction that $\epsilon^{le} = \epsilon^{lp}$, then we have

$$\begin{aligned} \boldsymbol{c}^\top \boldsymbol{x} &\ge K - \epsilon^{lp}, \\ \boldsymbol{A}\boldsymbol{x} &\le \boldsymbol{b} + \epsilon^{lp}\mathbf{1}, \end{aligned}$$

which indicates that $\boldsymbol{x}$ is a solution to the LPA instance by Definition 4.2.

□

4.2 LEN(A) to 2-LEN(A)

4.2.1 LEN to 2-LEN

We show the reduction from an LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$ to a 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$. The LEN instance has the form of $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$, where $\tilde{\boldsymbol{A}} \in \mathbb{Z}^{\tilde{m}\times\tilde{n}}, \tilde{\boldsymbol{b}} \in \mathbb{Z}^{\tilde{m}}$. To reduce it to a 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$ in which the coefficients of $\bar{\boldsymbol{A}}$ are in $\{\pm 1, \pm 2\}$, we do *bitwise decomposition.* Algorithm 1 describes how to obtain a binary representation of an integer. The algorithm takes $z \in \mathbb{Z}$ as an input and output a list L consisting of all the powers such that $z = s(z)\sum_{l\in L} 2^l$ where $s(z)$ is the sign of z. For example, $z = -5$, then $L = \{2, 0\}$ and $s(z) = -1$.

We will reduce each linear equation in LEN to a linear equation in 2-LEN. For an arbitrary linear equation q of LEN: $\tilde{\boldsymbol{a}}_q^\top \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}(q), q \in [\tilde{m}]$, we describe the reduction by the following 4 steps.

Algorithm 1: BinaryRepresentation

Input: $z \in \mathbb{Z}$

Output: L is a list of powers of 2 such that $z = s(z) \sum_{l \in L} 2^l$, where $s(z)$ returns the sign of z.

$r \leftarrow |z|$;
$L \leftarrow []$;
for $r > 0$ **do**
    L.append($\lfloor \log_2 r \rfloor$);
    $r \leftarrow r - 2^{\lfloor \log_2 r \rfloor}$;
end

1. We run Algorithm 1 for each nonzero entry of $\tilde{\boldsymbol{a}}_q$ and $\tilde{\boldsymbol{b}}(q)$ so that each nonzero entry has a binary representation. To simplify notations, we denote the sign of $\tilde{\boldsymbol{a}}_q(i)$ as s_q^i and the list returned by BinaryRepresentation($\tilde{\boldsymbol{a}}_q(i)$) as L_q^i, and the sign of $\tilde{\boldsymbol{b}}(q)$ as s_q and the list returned by BinaryRepresentation($\tilde{\boldsymbol{b}}(q)$) as L_q. Thus, the qth linear equation of LEN can be rewritten as

$$\sum_{i \in [\tilde{n}]} \underbrace{\left(s_q^i \sum_{l \in L_q^i} 2^l \right)}_{\tilde{\boldsymbol{a}}_q(i)} \tilde{\boldsymbol{x}}(i) = \underbrace{s_q \sum_{l \in L_q} 2^l}_{\tilde{\boldsymbol{b}}(q)} . \tag{6}$$

2. Letting N_q denote the maximum element of L_q and $L_q^i, i \in [\tilde{n}]$, i.e.,

$$N_q = \left\lfloor \log_2 \max \left\{ \|\tilde{\boldsymbol{a}}_q\|_{\max}, \left| \tilde{\boldsymbol{b}}(q) \right| \right\} \right\rfloor ,$$

then we can rearrange the left hand side of Eq. (6) by gathering those terms located at the same bit (i.e., with the same weight of power of 2), and obtain

$$\sum_{l=0}^{N_q} \left(\sum_{i \in [\tilde{n}]} \mathbb{1}_{[l \in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) \right) 2^l = s_q \sum_{l \in L_q} 2^l, \tag{7}$$

where $\mathbb{1}$ is an indicator function such that

$$\mathbb{1}_{[l \in L_q^i]} = \begin{cases} 1 & \text{if } l \in L_q^i, \\ 0 & \text{otherwise.} \end{cases}$$

3. We decompose Eq. (7) into $N_q + 1$ linear equations such that each one representing a bit, by matching its left hand side and right hand side of Eq. (7) that are located at the same bit. We will also introduce a carry term for each bit, which passes the carry from the equation corresponding to that bit to the equation corresponding to the next bit. Without carry terms, the new system may be infeasible even if the old one is. Moreover, since a carry can be any real number, we represent each carry as a difference of two nonnegative variables $\boldsymbol{c}_q(i) - \boldsymbol{d}_q(i), \boldsymbol{c}_q(i), \boldsymbol{d}_q(i) \geq 0$. Starting from the lowest bit, the followings are $N_q + 1$ linear

equations after decomposition.

$$
\begin{aligned}
\sum_{i\in[\tilde{n}]} \mathbb{1}_{[0\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) - 2[\boldsymbol{c}_q(0) - \boldsymbol{d}_q(0)] &= s_q \mathbb{1}_{[0\in L_q]} \\
\sum_{i\in[\tilde{n}]} \mathbb{1}_{[1\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) + [\boldsymbol{c}_q(0) - \boldsymbol{d}_q(0)] - 2[\boldsymbol{c}_q(1) - \boldsymbol{d}_q(1)] &= s_q \mathbb{1}_{[1\in L_q]} \\
&\vdots \\
\sum_{i\in[\tilde{n}]} \mathbb{1}_{[N_q\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) + [\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)] &= s_q \mathbb{1}_{[N_q\in L_q]}
\end{aligned} \tag{8}
$$

4. We add an additional constraint for each carry variable $\boldsymbol{c}_q(i), \boldsymbol{d}_q(i)$[5]:

$$\boldsymbol{c}_q(i) \le 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}, \;\; \boldsymbol{d}_q(i) \le 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}. \tag{9}$$

   These constraints guarantee that the polytope radius of the reduced 2-LEN instance cannot be too large [6]. In our proofs, we will show that these additional constraints do not affect the problem feasibility. We then add slack variables $\boldsymbol{s}_q^c(i), \boldsymbol{s}_q^d(i) \ge 0$ for each carry term and turn Eq. (9) to

$$\boldsymbol{c}_q(i) + \boldsymbol{s}_q^c(i) = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}, \quad \boldsymbol{d}_q(i) + \boldsymbol{s}_q^d(i) = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}, \qquad 0 \le i \le N_q - 1. \tag{10}$$

Repeating the above process for $\tilde{m}$ times, we get a 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$, where $\bar{\boldsymbol{A}}$ is the coefficient matrix, $\bar{\boldsymbol{b}}$ is the right hand side vector, and $\bar{R}$ is the polytope radius.

If a 2-LEN solver returns $\bar{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}^\top, \boldsymbol{c}^\top, \boldsymbol{d}^\top, \boldsymbol{s}^{c\top}, \boldsymbol{s}^{d\top})^\top$ for the 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$, then we return $\tilde{\boldsymbol{x}}$ for the LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$; if the 2-LEN solver returns "infeasible" for the 2-LEN instance, then we return "infeasible" for the LEN instance.

Lemma 4.5 (LEN to 2-LEN). *Given an LEN instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$ where $\tilde{\boldsymbol{A}} \in \mathbb{Z}^{\tilde{m}\times\tilde{n}}, \tilde{\boldsymbol{b}} \in \mathbb{Z}^{\tilde{m}}$, we can construct, in $O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}}) \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right)$ time, a 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$ where $\bar{\boldsymbol{A}} \in \mathbb{Z}^{\bar{m}\times\bar{n}}, \bar{\boldsymbol{b}} \in \mathbb{Z}^{\bar{m}}$ such that*

$$\bar{n} \le \tilde{n} + 4\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right), \;\; \bar{m} \le 3\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right), \;\; \mathrm{nnz}(\bar{\boldsymbol{A}}) \le 17\,\mathrm{nnz}(\tilde{\boldsymbol{A}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right),$$

$$\bar{R} = 8\tilde{m}\tilde{R}X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right), \quad X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}.$$

If the LEN instance has a solution, then the 2-LEN instance has a solution.

Proof. Based on the reduction method described above, from any solution $\tilde{\boldsymbol{x}}$ to the LEN instance such that $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$, we can derive a solution $\bar{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}^\top, \boldsymbol{c}^\top, \boldsymbol{d}^\top, \boldsymbol{s}^{c\top}, \boldsymbol{s}^{d\top})^\top$ to the 2-LEN instance. Concretely, for any linear equation q in LEN, $q \in [\tilde{m}]$, with its decomposed equations as shown in Eq. (8), we can set the value of $\boldsymbol{c}_q, \boldsymbol{d}_q$ from the highest bit as

$$\boldsymbol{c}_q(N_q - 1) = \max\left\{0, \;\; s_q \mathbb{1}_{[N_q\in L_q]} - \sum_{i\in[\tilde{n}]} \mathbb{1}_{[N_q\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i)\right\},$$

[5]We remark that Itai's reduction does not have these upper bound on carry variables. We need these constraints in our reduction to guarantee that the polytope radius is always well bounded.

[6]Note that without these additional constraints the radius of polytope can be unbounded.

$$\boldsymbol{d}_q(N_q-1)=\max\left\{0,\quad \sum_{i\in[\tilde{n}]}\mathbb{1}_{[N_q\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)-s_q\mathbb{1}_{[N_q\in L_q]}\right\}.$$

And then using backward substitution, we can set the value for the rest entries of $\boldsymbol{c}_q$ and $\boldsymbol{d}_q$ similarly.

$$\begin{aligned}\boldsymbol{c}_q(N_q-2)&=\max\left\{0,\quad s_q\mathbb{1}_{[(N_q-1)\in L_q]}+2[\boldsymbol{c}_q(N_q-1)-\boldsymbol{d}_q(N_q-1)]-\sum_{i\in[\tilde{n}]}\mathbb{1}_{[(N_q-1)\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)\right\}\\&=\max\left\{0,\quad \sum_{l=\{N_q-1,N_q\}}\left(s_q\mathbb{1}_{[l\in L_q]}-\sum_{i\in[\tilde{n}]}\mathbb{1}_{[l\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)\right)2^{l-(N_q-1)}\right\},\\ \boldsymbol{d}_q(N_q-2)&=\max\left\{0,\quad \sum_{l=\{N_q-1,N_q\}}\left(\sum_{i\in[\tilde{n}]}\mathbb{1}_{[l\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)-s_q\mathbb{1}_{[l\in L_q]}\right)2^{l-(N_q-1)}\right\};\\&\vdots\\ \boldsymbol{c}_q(0)&=\max\left\{0,\quad \sum_{l=1}^{N_q}\left(s_q\mathbb{1}_{[l\in L_q]}-\sum_{i\in[\tilde{n}]}\mathbb{1}_{[l\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)\right)2^{l-1}\right\},\\ \boldsymbol{d}_q(0)&=\max\left\{0,\quad \sum_{l=1}^{N_q}\left(\sum_{i\in[\tilde{n}]}\mathbb{1}_{[l\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)-s_q\mathbb{1}_{[l\in L_q]}\right)2^{l-1}\right\}.\end{aligned}$$

Substituting $\boldsymbol{c}_q(0),\boldsymbol{d}_q(0)$ back to the equation of the lowest bit, we will get Eq. (7), the rearranged binary representation of equation q: $\tilde{\boldsymbol{a}}_q^\top\tilde{\boldsymbol{x}}=\tilde{\boldsymbol{b}}(q)$. By our setting of the carry terms, we have

$$\begin{aligned}\boldsymbol{c}_q(i),\boldsymbol{d}_q(i)&\le\left|\sum_{l=i+1}^{N_q}\left(\sum_{i\in[\tilde{n}]}\mathbb{1}_{[l\in L_q^i]}\cdot s_q^i\tilde{\boldsymbol{x}}(i)-s_q\mathbb{1}_{[l\in L_q]}\right)2^{l-i-1}\right|\\&\le 2^{N_q}(\|\tilde{\boldsymbol{x}}\|_1+1)\le 2\max\left\{\left\|\tilde{\boldsymbol{A}}\right\|_{\max},\left\|\tilde{\boldsymbol{b}}\right\|_{\max}\right\}\tilde{R}=2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\tilde{R}.\end{aligned}$$

By setting slacks

$$\boldsymbol{s}_q^c(i)=2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\tilde{R}-\boldsymbol{c}_q(i),\quad \boldsymbol{s}_q^d(i)=2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\tilde{R}-\boldsymbol{d}_q(i),$$

we satisfy the equations (10) in the 2-LEN instance. Repeating the above process for all $q\in[\tilde{m}]$, we get a feasible solution $\bar{\boldsymbol{x}}=(\tilde{\boldsymbol{x}}^\top,\boldsymbol{c}^\top,\boldsymbol{d}^\top,\boldsymbol{s}^{c\top},\boldsymbol{s}^{d\top})^\top$ to the 2-LEN instance.

Now, we track the change of problem size after reduction. Based on the reduction method, each linear equation in LEN can be decomposed into at most N linear equations, where

$$N=1+\max_{q\in[\tilde{m}]}N_q=1+\left\lfloor\log_2\left\|\tilde{\boldsymbol{A}}\right\|_{\max}\right\rfloor\le 1+\left\lfloor\log X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\right\rfloor. \tag{11}$$

Thus, given an LEN instance with $\tilde{n}$ variables, $\tilde{m}$ linear equations, and $\mathrm{nnz}(\tilde{\boldsymbol{A}})$ nonzero entries, we compute the size of the reduced 2-LEN instance as follows.

1. $\bar{n}$ variables. First, all $\tilde{n}$ variables in LEN are maintained. Then, for each of the $\tilde{m}$ equations in LEN, it is decomposed into at most N equations, where at most a pair of carry variables are introduced for each newly added equation. Finally, we introduce a slack variable for each carry variable. Thus, we have
$$\bar{n}\le\tilde{n}+2\tilde{m}N+2\tilde{m}N\le\tilde{n}+4\tilde{m}\left(1+\log X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\right).$$

2. $\bar{m}$ linear constraints. First, each equation in LEN is decomposed into at most N equations in 2-LEN. Next, we add a new constraint for each carry variable. Thus, we have

$$\bar{m} \leq \tilde{m}N + 2\tilde{m}N \leq 3\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right). \tag{12}$$

3. $\mathrm{nnz}(\bar{\boldsymbol{A}})$ nonzeros. To bound it, first, for each nonzero entry in $\tilde{\boldsymbol{A}}$, it will be decomposed into at most N bits, thus becomes at most N nonzero entries in $\bar{\boldsymbol{A}}$. Then, each equation in 2-LEN involves at most 4 carry variables. Furthermore, there are $2\tilde{m}N$ new constraints for carry variables, and each constraint involves a carry variable and a slack variable. In total, we have

$$\begin{aligned}\mathrm{nnz}(\bar{\boldsymbol{A}}) &\leq \mathrm{nnz}(\tilde{\boldsymbol{A}})N + 4\bar{m} + 4\tilde{m}N \\ &\overset{(1)}{\leq} \mathrm{nnz}(\tilde{\boldsymbol{A}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) + 12\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) + 4\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) \\ &\overset{(2)}{\leq} 17\,\mathrm{nnz}(\tilde{\boldsymbol{A}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right),\end{aligned} \tag{13}$$

where in step (1), we utilize Eq. (12) for the bound of $\bar{m}$; and in step (2), we use $\tilde{m} \leq \mathrm{nnz}(\tilde{\boldsymbol{A}})$.

4. $\bar{R}$, the radius of the polytope in ℓ_1 norm. We want to upper bound $\bar{\boldsymbol{x}}_1$ for every feasible solution to the 2-LEN instance. By definition and the triangle inequality,

$$\|\bar{\boldsymbol{x}}\|_1 \leq \|\tilde{\boldsymbol{x}}\|_1 + \|\boldsymbol{c}\|_1 + \|\boldsymbol{d}\|_1 + \|\boldsymbol{s}^c\|_1 + \left\|\boldsymbol{s}^d\right\|_1$$

Note $\|\tilde{\boldsymbol{x}}\|_1 \leq \tilde{R}$ and the maximum magnitude of the entries of $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{s}^c, \boldsymbol{s}^d$ is at most $2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}$ by Eq. (10). Also note the dimensions of $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{s}^c, \boldsymbol{s}^d$ are $\tilde{m}N$. Thus,

$$\|\bar{\boldsymbol{x}}\|_1 \leq \tilde{R} + 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R} \cdot 4\tilde{m}N \leq 8\tilde{m}\tilde{R}X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right).$$

Hence, it suffices to set

$$\bar{R} = 8\tilde{m}\tilde{R}X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right).$$

5. $X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R}$ because by construction,

$$\left\|\bar{\boldsymbol{A}}\right\|_{\max} = 2, \qquad \left\|\bar{\boldsymbol{b}}\right\|_{\max} = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R} \geq 2.$$

To estimate the reduction time, it is noticed that it takes $O(N)$ time to run Algorithm 1 for each nonzero entry of $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{b}}$. And there are at most $\mathrm{nnz}(\tilde{\boldsymbol{A}}) + \mathrm{nnz}(\tilde{\boldsymbol{b}}) = O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}})\right)$ entries to be decomposed. In addition, it takes $O\left(\mathrm{nnz}(\bar{\boldsymbol{A}}) + \mathrm{nnz}(\bar{\boldsymbol{b}})\right) = O\left(\mathrm{nnz}(\bar{\boldsymbol{A}})\right)$ to construct $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{b}}$. By Eq. (13), performing such a reduction takes time in total

$$O\left(N\,\mathrm{nnz}(\tilde{\boldsymbol{A}}) + \mathrm{nnz}(\bar{\boldsymbol{A}})\right) = O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}})\log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right).$$

$\square$

4.2.2 LENA to 2-LENA

Definition 4.6 (k-LEN Approximate Problem (k-LENA)). A k-LENA instance is given by a k-LEN instance $(\boldsymbol{A}, \boldsymbol{b}, R)$ as in Definition 2.3 and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(\boldsymbol{A}, \boldsymbol{b}, R, \epsilon)$. We say an algorithm solves the k-LENA problem, if, given any k-LENA instance, it returns a vector $\boldsymbol{x} \geq \boldsymbol{0}$ such that

$$|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}| \leq \epsilon\boldsymbol{1},$$

where $|\cdot|$ is entrywise absolute value and $\boldsymbol{1}$ is the all-1 vector, or it correctly declares that the associated k-LEN instance is infeasible.

We can use the same reduction method in the exact case to reduce an LENA instance to a 2-LENA instance. Furthermore, if a 2-LENA solver returns $\bar{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}^\top, \boldsymbol{c}^\top, \boldsymbol{d}^\top, \boldsymbol{s}^{c\top}, \boldsymbol{s}^{d\top})^\top$ for the 2-LENA instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R}, \epsilon^{2le})$, then we return $\tilde{\boldsymbol{x}}$ for the LENA instance $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R}, \epsilon^{le})$; if the 2-LENA solver returns "infeasible" for the 2-LENA instance, then we return "infeasible" for the LENA instance.

Lemma 4.7 (LENA to 2-LENA)**.** *Given an* LENA *instance* $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R}, \epsilon^{le})$, *where* $\tilde{\boldsymbol{A}} \in \mathbb{Z}^{\tilde{m}\times\tilde{n}}$, $\tilde{\boldsymbol{b}} \in \mathbb{Z}^{\tilde{m}}$, *if we use Lemma 4.5 to construct a* 2-LEN *instance* $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$ *from the* LEN *instance* $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{R})$, *and let*

$$\epsilon^{2le} = \frac{\epsilon^{le}}{2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})},$$

then the 2-LENA *instance* $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R}, \epsilon^{2le})$ *satisfies.*

Furthermore, if $\bar{\boldsymbol{x}}$ *is a solution to the* 2-LEN(A) *instance, then in time* $O(\tilde{n})$, *we can compute a solution* $\tilde{\boldsymbol{x}}$ *to the* LEN(A) *instance, where the exact case holds when* $\epsilon^{2le} = \epsilon^{le} = 0$.

Proof. Based on the solution mapping method described above, given a solution $\bar{\boldsymbol{x}}$, we discard those entries of carry variables and slack variables, and map back trivially for those entries of $\tilde{\boldsymbol{x}}$. As it takes constant time to set the value of each entry of $\tilde{\boldsymbol{x}}$ by mapping back trivially, and the size of $\tilde{\boldsymbol{x}}$ is $\tilde{n}$, thus the solution mapping takes $O(\tilde{n})$ time.

Now, we conduct an error analysis. By Definition 4.6, the error of each linear equation in 2-LENA can be bounded by ϵ^{2le}. In particular, the equation in the 2-LENA instance that corresponds to the highest bit of the qth equation in the LENA instance satisfies

$$\left|\sum_{i\in[\tilde{n}]} \mathbb{1}_{[N_q\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) + [\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)] - s_q \mathbb{1}_{[N_q\in L_q]}\right| \le \epsilon^{2le},$$

which can be rearranged as

$$-\epsilon^{2le} - [\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)] \le \sum_{i\in[\tilde{n}]} \mathbb{1}_{[N_q\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) - s_q \mathbb{1}_{[N_q\in L_q]} \le \epsilon^{2le} - [\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)]. \quad (14)$$

For the second highest bit, we have

$$\begin{aligned}
&-\epsilon^{2le} + 2[\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)] - [\boldsymbol{c}_q(N_q-2) - \boldsymbol{d}_q(N_q-2)] \\
&\qquad \le \sum_{i\in[\tilde{n}]} \mathbb{1}_{[(N_q-1)\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) - s_q \mathbb{1}_{[(N_q-1)\in L_q]} \\
&\qquad \le \epsilon^{2le} + 2[\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)] - [\boldsymbol{c}_q(N_q-2) - \boldsymbol{d}_q(N_q-2)]. \quad (15)
\end{aligned}$$

We can eliminate the pair of carry $[\boldsymbol{c}_q(N_q-1) - \boldsymbol{d}_q(N_q-1)]$ by computing $2\times$Eq. (14)$+$Eq. (15), and obtain

$$\begin{aligned}
&-(2^0+2^1)\epsilon^{2le} - [\boldsymbol{c}_q(N_q-2) - \boldsymbol{d}_q(N_q-2)] \\
&\qquad \le \sum_{l=\{N_q-1,N_q\}} \left(\sum_{i\in[\tilde{n}]} \mathbb{1}_{[l\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) - s_q \mathbb{1}_{[l\in L_q]}\right) 2^{l-(N_q-1)} \\
&\qquad \le (2^0+2^1)\epsilon^{2le} - [\boldsymbol{c}_q(N_q-2) - \boldsymbol{d}_q(N_q-2)]. \quad (16)
\end{aligned}$$

By repeating the process until the equation of the lowest bit, we can eliminate all pairs of carry variables and obtain

$$-(2^0+\cdots+2^{N_q})\epsilon^{2le} \le \underbrace{\sum_{l=0}^{N_q} \left(\sum_{i\in[\tilde{n}]} \mathbb{1}_{[l\in L_q^i]} \cdot s_q^i \tilde{\boldsymbol{x}}(i) - s_q \mathbb{1}_{[l\in L_q]}\right) 2^l}_{=\tilde{\boldsymbol{a}}_q^\top \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}(q) \quad \text{by Eq. (7)}} \le (2^0+\cdots+2^{N_q})\epsilon^{2le}. \quad (17)$$

Hence, we can bound the qth linear equation in LEN by

$$\left|\tilde{\boldsymbol{a}}_q^\top \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}(q)\right| \leq (2^0 + \cdots + 2^{N_q})\epsilon^{2le} \leq 2^{N_q+1}\epsilon^{2le},$$

which implies that the error of the qth equation of LENA is accumulated as a weighted sum of at most N_q equations in 2-LENA, where the weight is in the form of power of 2.

To bound all the linear equations in LENA uniformly, we have

$$\begin{aligned}
\tau^{le} &= \max_{q\in[\tilde{m}]} \left|\tilde{\boldsymbol{a}}_q^\top \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}(q)\right| \\
&\leq \max_{q\in[\tilde{m}]} 2^{N_q+1}\epsilon^{2le} \\
&\leq 2^{N}\epsilon^{2le} && \text{Because } N = 1 + \max_{q\in\tilde{m}} N_q \text{ as in Eq. (11)} \\
&\leq 2^{1+\log X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})}\epsilon^{2le} && \text{Because } N = 1 + \left\lfloor \log X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\right\rfloor \text{ as in Eq. (11)} \\
&\leq 2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\epsilon^{2le}
\end{aligned}$$

As we set in the reduction that $\epsilon^{2le} = \frac{\epsilon^{le}}{2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})}$, then we have

$$\tau^{le} \leq 2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})\frac{\epsilon^{le}}{2X(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}})} = \epsilon^{le},$$

which indicates that $\tilde{\boldsymbol{x}}$ is a solution to the LENA instance. □

4.3 2-LEN(A) to 1-LEN(A)

4.3.1 2-LEN to 1-LEN

We show the reduction from a 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$ to a 1-LEN instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$. The 2-LEN instance has the form of $\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}$, where entries of $\bar{\boldsymbol{A}}$ are integers between $[-2, 2]$. To reduce it to a 1-LEN instance, for each variable $\bar{\boldsymbol{x}}(j)$ that has a ± 2 coefficient, we introduce a new variable $\bar{\boldsymbol{x}}'(j)$, replace every $\pm 2\bar{\boldsymbol{x}}(j)$ with $\pm(\bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j))$, and add an additional equation $\bar{\boldsymbol{x}}(j) - \bar{\boldsymbol{x}}'(j) = 0$.

If a 1-LEN solver returns $\hat{\boldsymbol{x}} = (\bar{\boldsymbol{x}}^\top, (\bar{\boldsymbol{x}}')^\top)^\top$ for the 1-LEN instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$, then we return $\bar{\boldsymbol{x}}$ for the 2-LEN instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$; if the 1-LEN returns "infeasible" for the 1-LEN instance, then we return "infeasible" for the 2-LEN instance.

Lemma 4.8 (2-LEN to 1-LEN)**.** *Given a* 2-LEN *instance* $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$ *where* $\bar{\boldsymbol{A}} \in \mathbb{Z}^{\bar{m}\times\bar{n}}, \bar{\boldsymbol{b}} \in \mathbb{Z}^{\bar{m}}$*, we can construct, in* $O(\mathrm{nnz}(\bar{\boldsymbol{A}}))$ *time, a* 1-LEN *instance* $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$ *where* $\hat{\boldsymbol{A}} \in \mathbb{Z}^{\hat{m}\times\hat{n}}, \hat{\boldsymbol{b}} \in \mathbb{Z}^{\hat{m}}$ *such that*

$$\hat{n} \leq 2\bar{n}, \quad \hat{m} \leq \bar{m} + \bar{n}, \quad \mathrm{nnz}(\hat{\boldsymbol{A}}) \leq 4\,\mathrm{nnz}(\bar{\boldsymbol{A}}), \quad \hat{R} = 2\bar{R}, \quad X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}) = X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}).$$

If the 2-LEN *instance has a solution, then the* 1-LEN *instance has a solution.*

Proof. Based on the reduction described above, from any solution $\bar{\boldsymbol{x}}$ to the 2-LEN instance such that $\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}$, we can derive a solution $\hat{\boldsymbol{x}} = (\bar{\boldsymbol{x}}^\top, (\bar{\boldsymbol{x}}')^\top)^\top$ to the 1-LEN instance. Concretely, for each $\bar{\boldsymbol{x}}(j)$ having a ± 2 coefficient in $\bar{\boldsymbol{A}}$, we set $\bar{\boldsymbol{x}}'(j) = \bar{\boldsymbol{x}}(j)$, where $\bar{\boldsymbol{x}}'(j)$ is the entry that we use to replace $\pm 2\bar{\boldsymbol{x}}(j)$ with $\pm(\bar{\boldsymbol{x}}_j + \bar{\boldsymbol{x}}'_j)$ in the reduction. We can check that $\hat{\boldsymbol{x}}$ is a solution to the 1-LEN instance.

Now, we track the change of problem size after reduction. Given a 2-LEN instance with $\bar{n}$ variables, $\bar{m}$ linear equations, and $\mathrm{nnz}(\bar{\boldsymbol{A}})$ nonzero entries, we can compute the size of the reduced 1-LEN instance as follows.

1. $\hat{n}$ variables, where $\hat{n} \le 2\bar{n}$. It is because each variable $\bar{\boldsymbol{x}}(j)$ in 2-LEN is replaced by at most 2 variables $\bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j)$ in 1-LEN.

2. $\hat{m}$ linear constraints. In addition to the original $\bar{m}$ linear equations, each variable $\bar{\boldsymbol{x}}(j)$ with ± 2 coefficient in 2-LEN will introduce a new equation $\bar{\boldsymbol{x}}(j) - \bar{\boldsymbol{x}}'(j) = 0$ in 1-LEN. Thus, $\hat{m} \le \bar{m} + \bar{n}$.

3. $\mathrm{nnz}(\hat{\boldsymbol{A}})$ nonzeros. To bound it, first, each nonzero entry in $\bar{\boldsymbol{A}}$ becomes at most two nonzero entries in $\hat{\boldsymbol{A}}$ because of the replacement of $\pm 2\bar{\boldsymbol{x}}(j)$ by $\pm(\bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j))$. Next, at most $2\bar{n}$ new nonzero entries are generated because of the newly added equation $\bar{\boldsymbol{x}}(j) - \bar{\boldsymbol{x}}'(j) = 0$. Thus,

$$\mathrm{nnz}(\hat{\boldsymbol{A}}) \le 2(\mathrm{nnz}(\bar{\boldsymbol{A}}) + \bar{n}) \le 4\,\mathrm{nnz}(\bar{\boldsymbol{A}}).$$

4. $\hat{R}$ radius of polytope in ℓ_1 norm. We have

$$\|\hat{\boldsymbol{x}}\|_1 = \|\bar{\boldsymbol{x}}\|_1 + \left\|\bar{\boldsymbol{x}}'\right\|_1 \le 2\,\|\bar{\boldsymbol{x}}\|_1 \le 2\bar{R}.$$

   Hence, it suffices to set $\hat{R} = 2\bar{R}$.

5. $X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}) = X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}})$ because by construction,

$$\left\|\hat{\boldsymbol{A}}\right\|_{\max} \le \|\bar{\boldsymbol{A}}\|_{\max}, \qquad \left\|\hat{\boldsymbol{b}}\right\|_{\max} = \|\bar{\boldsymbol{b}}\|_{\max}.$$

To estimate the reduction time, it takes constant time to deal with each $\bar{\boldsymbol{A}}(i,j)$ being ± 2, and there are at most $\mathrm{nnz}(\bar{\boldsymbol{A}})$ occurrences of ± 2 to be dealt with. Hence, it takes $O(\mathrm{nnz}(\bar{\boldsymbol{A}}))$ time to eliminate all the occurrence of ± 2. Moreover, copy the rest coefficients also takes $O(\mathrm{nnz}(\bar{\boldsymbol{A}}))$ time. Thus, the reduction of this step takes $O(\mathrm{nnz}(\bar{\boldsymbol{A}}))$ time.

□

4.3.2 2-LENA to 1-LENA

We can use the same reduction method in the exact case to reduce a 2-LENA instance to a 1-LENA instance. We can also use the same solution mapping method, but we make a slight adjustment in the approximate case for the simplicity of the following error analysis. More specifically, if a 1-LENA solver returns $\hat{\boldsymbol{x}} = (\boldsymbol{x}^\top, (\boldsymbol{x}')^\top)^\top$ for the 1-LENA instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R}, \epsilon^{1le})$, instead of returning $\boldsymbol{x}$ directly as a solution to the 2-LENA instance $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R}, \epsilon^{2le})$, we set $\bar{\boldsymbol{x}}(i) = \frac{1}{2}(\boldsymbol{x}(i) + \boldsymbol{x}'(i))$ if $\boldsymbol{x}(i)$ has a coefficient ± 2 in the 2-LEN instance, and set $\bar{\boldsymbol{x}}(i) = \boldsymbol{x}(i)$ otherwise. In addition, if the 1-LENA solver returns "infeasible" for the 1-LENA instance, then we return "infeasible" for the 2-LENA instance.

Lemma 4.9 (2-LENA to 1-LENA)**.** *Given a* 2-LENA *instance* $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R}, \epsilon^{2le})$ *where* $\bar{\boldsymbol{A}} \in \mathbb{Z}^{\bar{m}\times\bar{n}}$, $\bar{\boldsymbol{b}} \in \mathbb{Z}^{\bar{m}}$*, if we use Lemma 4.8 to construct a* 1-LEN *instance* $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$ *from the* 2-LEN *instance* $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}, \bar{R})$*, and let*

$$\epsilon^{1le} = \frac{\epsilon^{2le}}{\bar{n}+1},$$

then the 1-LENA *instance* $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R}, \epsilon^{1le})$ *satisfies.*

Furthermore, if $\hat{\boldsymbol{x}}$ *is a solution to the* 1-LEN(A) *instance, then in time* $O(\bar{n})$*, we can compute a solution* $\bar{\boldsymbol{x}}$ *to the* 2-LEN(A) *instance, where the exact case holds when* $\epsilon^{1le} = \epsilon^{2le} = 0$.

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each entry of $\bar{\boldsymbol{x}}$ by computing an averaging or mapping back trivially, and the size of $\bar{\boldsymbol{x}}$ is $\bar{n}$, thus the solution mapping takes $O(\bar{n})$ time.

Now, we conduct an error analysis. By Definition 4.6, the error of each linear equation in 1-LENA can be bounded by ϵ^{1le}. For a single occurrence of $\bar{\boldsymbol{A}}(i,j) = \pm 2$, we first bound the error of the equation $\bar{\boldsymbol{x}}(j) - \bar{\boldsymbol{x}}'(j) = 0$, and obtain

$$|\bar{\boldsymbol{x}}(j) - \bar{\boldsymbol{x}}'(j)| \le \epsilon^{1le},$$

hence, we have

$$-\epsilon^{1le} + 2\bar{\boldsymbol{x}}(j) \le \bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j) \le \epsilon^{1le} + 2\bar{\boldsymbol{x}}(j). \tag{18}$$

We first consider the case that, in the equation $\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}(i)$, there is only one entry j such that $\bar{\boldsymbol{a}}_i(j) = \pm 2$. By separating this term, we can write

$$\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} = \sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k) \pm 2\bar{\boldsymbol{x}}(j), \qquad \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} = \sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k) \pm (\bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j)).$$

Adding $\sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k)$ to Eq, (18), we have

$$-\epsilon^{1le} \underbrace{\pm 2\bar{\boldsymbol{x}}(j) + \sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k)}_{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}}} \le \underbrace{\sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k) \pm (\bar{\boldsymbol{x}}(j) + \bar{\boldsymbol{x}}'(j))}_{\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}} \le \epsilon^{1le} \underbrace{\pm 2\bar{\boldsymbol{x}}(j) + \sum_{k \ne j} \bar{\boldsymbol{A}}(i,k)\bar{\boldsymbol{x}}(k)}_{\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}}}.$$

And since $\hat{\boldsymbol{b}}(i) = \bar{\boldsymbol{b}}(i)$, we can subtract it from all parts and get

$$-\epsilon^{1le} + \bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i) \le \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i) \le \epsilon^{1le} + \bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i).$$

which can be further transformed to

$$-\epsilon^{1le} + \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i) \le \bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i) \le \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i) + \epsilon^{1le}. \tag{19}$$

If there are k_i occurrence of $\bar{\boldsymbol{A}}(i,j) = \pm 2$, then we can generalize Eq. (19) to

$$-k_i\epsilon^{1le} + \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i) \le \bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i) \le \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i) + k_i\epsilon^{1le}, \tag{20}$$

hence, we can bound

$$\left|\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i)\right| \le k_i\epsilon^{1le} + \left|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i)\right| \le (k_i + 1)\epsilon^{1le}, \tag{21}$$

where the last inequality is because $\left|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i)\right| \le \epsilon^{1le}$ by applying the error of the 1-LENA instance.

To bound all the linear equations in 2-LENA uniformly, we have

$$\begin{aligned}
\tau^{2le} &= \max_{i \in [\bar{m}]} \left|\bar{\boldsymbol{a}}_i^\top \bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}(i)\right| \\
&\le \max_{i \in [\bar{m}]} (k_i + 1)\epsilon^{1le} && \text{Because of Eq. (21)} \\
&\le (\bar{n} + 1)\epsilon^{1le} && \text{Because } k_i \le \bar{n}
\end{aligned}$$

As we set in the reduction that $\epsilon^{1le} = \frac{\epsilon^{2le}}{\bar{n}+1}$, then we have

$$\tau^{2le} \le (\bar{n} + 1)\frac{\epsilon^{2le}}{\bar{n} + 1} \le \epsilon^{2le},$$

which indicates that $\bar{\boldsymbol{x}}$ is a solution to the 2-LENA instance.

□

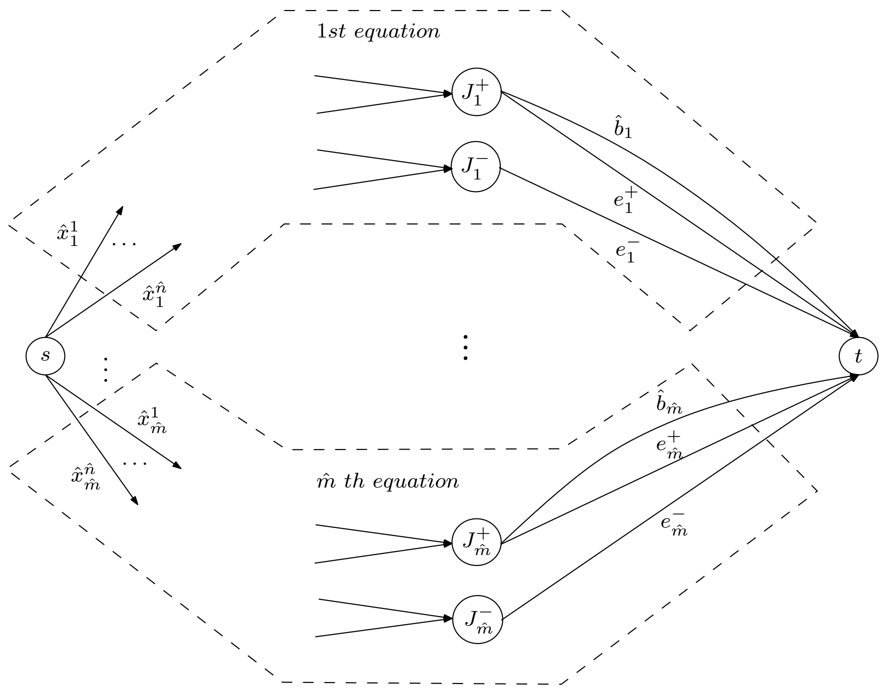

Figure 1: The reduction from 1-LEN to FHF.

4.4 1-LEN(A) to FHF(A)

4.4.1 1-LEN to FHF

The following is the approach to reduce a 1-LEN instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$ to an FHF instance $(G^h, F^h, \boldsymbol{u}^h, \mathcal{H}^h, s, t)$. The 1-LEN has the form of $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$, where $\hat{\boldsymbol{A}} \in \mathbb{Z}^{\hat{m}\times\hat{n}}$, $\hat{\boldsymbol{b}} \in \mathbb{Z}^{\hat{m}}$. For an arbitrary equation i in the 1-LEN instance, $\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}(i), i \in [\hat{m}]$, let $J_i^+ = \{j|\hat{\boldsymbol{a}}_i(j) = 1\}$ and $J_i^- = \{j|\hat{\boldsymbol{a}}_i(j) = -1\}$ denote the set of indices of variables with coefficients being 1 and -1 in equation i, respectively. Then, each equation can be rewritten as a difference of the sum of variables with coefficient 1 and -1:

$$\sum_{j\in J_i^+} \hat{\boldsymbol{x}}(j) - \sum_{j\in J_i^-} \hat{\boldsymbol{x}}(j) = \hat{\boldsymbol{b}}(i), \quad i \in [\hat{m}]. \tag{22}$$

We claim that $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ can be represented by a graph that is composed of a number of homologous edges and fixed flow edges, as shown in Figure 1.

More specifically, the fixed homologous flow network consists of a source s, a sink t, and $\hat{m}$ sections such that each section i represents the ith linear equation in 1-LEN, as shown in Eq. (22).

Inside each section i, there are 2 vertices $\{J_i^-, J_i^+\}$ and a number of edges:

- For the incoming edges of $\{J_i^-, J_i^+\}$,
  - if $\hat{\boldsymbol{a}}_i(j) = 1$, then s is connected to J_i^+ by edge $\hat{x}_i^j$ with capacity $\hat{R}$;
  - if $\hat{\boldsymbol{a}}_i(j) = -1$, then s is connected to J_i^- by edge $\hat{x}_i^j$ with capacity $\hat{R}$;
  - if $\hat{\boldsymbol{a}}_i(j) = 0$, no edge is needed.

  Note that the problem sparsity is preserved in the graph construction. The amount of flow routed in these incoming edges equals the value of the corresponding variables. To ensure the consistency of the value of variables over $\hat{m}$ equations, those incoming edges that correspond to the same variable are forced to route the same amount of flow, i.e., $(\hat{x}_1^j, \cdots, \hat{x}_{\hat{m}}^j), j \in [\hat{n}]$ constitute a homologous edge set that corresponds to the variable $\hat{\boldsymbol{x}}(j)$. Note that the size of such a homologous edge set is at most $\hat{m}$.

- For the outgoing edges of $\{J_i^-, J_i^+\}$,
  - J_i^+ is connected to t by a fixed flow edge $\hat{b}_i$ that routes $\hat{\boldsymbol{b}}(i)$ units of flow;
  - J_i^+ and J_i^- are connected to t by a pair of homologous edges e_i^+, e_i^- with capacity $\hat{R}$.

If an FHF solver returns $\boldsymbol{f}^h$ for the FHF instance $(G^h, F^h, \boldsymbol{u}^h, \mathcal{H}^h, s, t)$, then we return $\hat{\boldsymbol{x}}$ for the 1-LEN instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$, by setting for every $j \in [\hat{n}]$,

$$\hat{\boldsymbol{x}}(j) = \boldsymbol{f}^h(\hat{x}_i^j), \text{ for an arbitrary } i \in [\hat{m}]$$

If the FHF solver returns "infeasible" for the FHF instance, then we return "infeasible" for the 1-LEN instance.

Lemma 4.10 (1-LEN to FHF)**.** *Given a* 1-LEN *instance* $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$ *where* $\hat{\boldsymbol{A}} \in \mathbb{Z}^{\hat{m}\times\hat{n}}, \hat{\boldsymbol{b}} \in \mathbb{Z}^{\hat{m}}$, *we can construct, in time* $O(\mathrm{nnz}(\hat{\boldsymbol{A}}))$*, an* FHF *instance* $(G^h, F^h, \boldsymbol{u}^h, \mathcal{H}^h = (H_1, \cdots, H_h), s, t)$ *such that*

$$|V^h| = 2\hat{m} + 2, \quad |E^h| \le 4\,\mathrm{nnz}(\hat{\boldsymbol{A}}), \quad |F^h| = \hat{m}, \quad h = \hat{n} + \hat{m}, \quad \left\|\boldsymbol{u}^h\right\|_{\max} = \max\left\{\hat{R}, X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})\right\}.$$

If the 1-LEN *instance has a solution, then the* FHF *instance has a solution.*

Proof. According to the reduction described above, from any solution $\hat{\boldsymbol{x}}$ to the 1-LEN instance such that $\hat{\boldsymbol{A}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$, we can derive a solution $\boldsymbol{f}^h$ to the FHF instance. Concretely, we define a feasible flow $\boldsymbol{f}^h$ as follows:

- For incoming edges of $\{J_i^-, J_i^+\}$, we set
$$\boldsymbol{f}^h(\hat{x}_1^j) = \cdots = \boldsymbol{f}^h(\hat{x}_{\hat{m}}^j) = \hat{\boldsymbol{x}}(j) \le \hat{R}, \qquad \forall j \in [\hat{n}],$$
which satisfies the homologous constraint for the homologous edge sets $(\hat{x}_1^j, \cdots, \hat{x}_{\hat{m}}^j), j \in [\hat{n}]$, as well as the capacity constraint.
- For outgoing edges of $\{J_i^-, J_i^+\}$, we set
$$\boldsymbol{f}^h(\hat{b}_i) = \hat{\boldsymbol{b}}(i), \qquad \forall i \in \hat{m},$$
which satisfies the fixed flow constraint for edges $\hat{b}_1, \cdots, \hat{b}_{\hat{m}}$; and set
$$\boldsymbol{f}^h(e_i^+) = \boldsymbol{f}^h(e_i^-) = \sum_{j \in J_i^-} \hat{\boldsymbol{x}}(j) = \sum_{j \in J_i^+} \hat{\boldsymbol{x}}(j) - \hat{\boldsymbol{b}}(j),$$
which satisfies the homologous constraint for edge e_i^+, e_i^-, and the conservation of flows for vertices J_i^+, J_i^-.

Therefore, we conclude that $\boldsymbol{f}^h$ is a feasible flow to the FHF instance.

Now, we track the change of problem size after reduction. Based on the reduction method, given a 1-LENA instance with $\hat{n}$ variables, $\hat{m}$ linear equations, and $\mathrm{nnz}(\hat{\boldsymbol{A}})$ nonzero entries, it is straightforward to get the size of the reduced FHFA instance as follows.

1. $|V^h|$ vertices, where $|V^h| = 2\hat{m} + 2$.
2. $|E^h|$ edges, where $|E^h| = \text{nnz}(\hat{\boldsymbol{A}}) + 3\hat{m} \le 4\,\text{nnz}(\hat{\boldsymbol{A}})$, since $\hat{m} \le \text{nnz}(\hat{\boldsymbol{A}})$.
3. $|F^h|$ fixed flow edges, where $|F^h| = \hat{m}$.
4. h homologous edge sets, where $h = \hat{m} + \hat{n}$.
5. The maximum edge capacity is bounded by

$$\left\|\boldsymbol{u}^h\right\|_{\max} = \max\left\{\hat{R}, \left\|\hat{\boldsymbol{b}}\right\|_{\max}\right\} \le \max\left\{\hat{R}, X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})\right\}.$$

To estimate the reduction time, as there are $|V^h| = O(\hat{m})$ vertices and $|E^h| = O(\text{nnz}(\hat{\boldsymbol{A}}))$ edges in G^h, thus, performing such a reduction takes $O(\text{nnz}(\hat{\boldsymbol{A}}))$ time to construct G^h. □

4.4.2 1-LENA to FHFA

The above lemma shows the reduction between exactly solving a 1-LEN instance and exactly solving a FHF instance. Next, we generalize the case with exact solutions to the case that allows approximate solutions. First of all, we give a definition of the approximate version of FHF.

Definition 4.11 (FHF Approximate Problem (FHFA))**.** An FHFA instance is given by an FHF instance $(G, F, \boldsymbol{u}, \mathcal{H}, s, t)$ as in Definition 2.9, and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(G, F, \mathcal{H}, \boldsymbol{u}, s, t, \epsilon)$. We say an algorithm solves the FHFA problem, if, given any FHFA instance, it returns a flow $\boldsymbol{f} \ge \boldsymbol{0}$ that satisfies

$$\boldsymbol{u}(e) - \epsilon \le \boldsymbol{f}(e) \le \boldsymbol{u}(e) + \epsilon,\ \forall e \in F \tag{23}$$
$$0 \le \boldsymbol{f}(e) \le \boldsymbol{u}(e) + \epsilon,\ \forall e \in E \backslash F \tag{24}$$
$$\left|\sum_{u:(u,v)\in E} \boldsymbol{f}(u, v) - \sum_{w:(v,w)\in E} \boldsymbol{f}(v, w)\right| \le \epsilon,\ \forall v \in V \backslash \{s, t\} \tag{25}$$
$$\left|\boldsymbol{f}(v, w) - \boldsymbol{f}(v', w')\right| \le \epsilon,\ \forall (v, w), (v', w') \in H_i, H_i \in \mathcal{H} \tag{26}$$

or it correctly declares that the associated FHF instance is infeasible. We refer to the error in (23) and (24) as error in congestion, error in (25) as error in demand, and error in (26) as error in homology.

We can use the same reduction method and solution mapping method in the exact case to reduce a 1-LENA instance to an FHFA instance. Note that, though we still obtain $\hat{\boldsymbol{x}}$ by setting for each $j \in [\hat{n}]$,

$$\hat{\boldsymbol{x}}(j) = \boldsymbol{f}^h(\hat{x}_j^i), \qquad \text{for an arbitrary } i \in [\hat{m}]$$

Lemma 4.12 (1-LENA to FHFA)**.** *Given a 1-LENA instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R}, \epsilon^{1le})$ where $\hat{\boldsymbol{A}} \in \mathbb{Z}^{\hat{m}\times\hat{n}}$, $\hat{\boldsymbol{b}} \in \mathbb{Z}^{\hat{m}}$, if we use Lemma 4.10 to construct an FHF instance $(G^h, F^h, \mathcal{H}^h = (H_1, \cdots, H_h), \boldsymbol{u}^h, s, t)$ from the 1-LEN instance $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}, \hat{R})$, and let*

$$\epsilon^h = \frac{\epsilon^{1le}}{5\hat{n}X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})},$$

then the FHFA instance $(G^h, F^h, \mathcal{H}^h, \boldsymbol{u}^h, s, t, \epsilon^h)$ satisfies.

Furthermore, if $\boldsymbol{f}^h$ is a solution to the FHF(A) instance, then in time $O(\hat{n})$, we can compute a solution $\hat{\boldsymbol{x}}$ to the 1-LEN(A) instance, where the exact case holds when $\epsilon^h = \epsilon^{1le} = 0$.

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each entry of $\hat{\boldsymbol{x}}$ as $\boldsymbol{f}^h$ on certain edges. As $\hat{\boldsymbol{x}}$ has $\hat{n}$ entries, such a solution mapping takes $O(\hat{n})$ time.

In order to analyze errors, we firstly investigate the error in the ith linear equation in the 1-LENA instance $\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}(i)$. Let $\hat{\boldsymbol{x}}_i(j) = \boldsymbol{f}^h(\hat{x}_i^j)$ denote the amount of flow routed through in the ith section of G^h. For each edge $\hat{x}_i^j$ in the ith section of G^h, by error in homology defined in Eq. (26), we have

$$|\hat{\boldsymbol{x}}(j) - \hat{\boldsymbol{x}}_i(j)| \leq \epsilon^h, \qquad \forall j \in [\hat{n}]$$

Thus, we can bound $|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i|$ by

$$|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i| = |\hat{\boldsymbol{a}}_i^\top (\hat{\boldsymbol{x}} - \hat{\boldsymbol{x}}_i)| \leq \|\hat{\boldsymbol{a}}_i\|_1 \epsilon^h. \tag{27}$$

Next, we bound $\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i$. We use $\boldsymbol{f}^h(s, J_i^\pm)$ to denote the total incoming flow of vertex $J_i^\pm$. Then, we have

- $\left|\boldsymbol{f}^h(\hat{b}_i) - \hat{\boldsymbol{b}}(i)\right| \in \left[-\epsilon^h, \epsilon^h\right]$ because of error in congestion of fixed flow edge $\hat{b}_i$;
- $\left|\boldsymbol{f}^h(s, J_i^+) - (\boldsymbol{f}^h(e_i^+) + \boldsymbol{f}^h(\hat{b}_i))\right| \leq \epsilon^h$ because of error in demand of vertex J_i^+;
- $\left|\boldsymbol{f}^h(s, J_i^-) - \boldsymbol{f}^h(e_i^-)\right| \leq \epsilon^h$ because of error in demand of vertex J_i^-;
- $|\boldsymbol{f}^h(e_i^+) - \boldsymbol{f}^h(e_i^-)| \leq \epsilon^h$ because of error in homology between edge e_i^+, e_i^-.

Since $\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i = \boldsymbol{f}^h(s, J_i^+) - \boldsymbol{f}^h(s, J_i^-)$, the error of $\left|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i - \hat{\boldsymbol{b}}(i)\right|$ is an accumulation of the above four errors, which gives

$$\left|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i - \hat{\boldsymbol{b}}(i)\right| \leq 4\epsilon^h.$$

Combining with Eq. (27), we have

$$\begin{aligned}
|\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i)| &= |\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i + \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i - \hat{\boldsymbol{b}}(i)| \\
&\leq |\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i| + |\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}}_i - \hat{\boldsymbol{b}}(i)| \\
&\leq (\|\hat{\boldsymbol{a}}_i\|_1 + 4)\epsilon^h
\end{aligned}$$

To bound all the linear equations in 1-LENA uniformly, we have

$$\tau^{1le} \stackrel{\text{def}}{=} \max_{i \in [\hat{m}]} |\hat{\boldsymbol{a}}_i^\top \hat{\boldsymbol{x}} - \hat{\boldsymbol{b}}(i)| \leq \max_{i \in [\hat{m}]} \left\{(\|\hat{\boldsymbol{a}}_i\|_1 + 4)\epsilon^h\right\} \leq (\hat{n}X(\hat{\boldsymbol{A}}) + 4)\epsilon^h \leq 5\hat{n}X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})\epsilon^h.$$

As we set in the reduction that $\epsilon^h = \frac{\epsilon^{1le}}{5\hat{n}X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})}$, then we have

$$\tau^{1le} \leq 5\hat{n}X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})\epsilon^h = \epsilon^{1le},$$

indicating that $\hat{\boldsymbol{x}}$ is a solution to the 1-LENA instance.

□

4.5 FHF(A) to FPHF(A)

4.5.1 FHF to FPHF

We show the reduction from an FHF instance $(G^h, H^h, \mathcal{H}^h, \boldsymbol{u}^h, s, t)$ to an FPHF instance $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t)$. Suppose that $(v_1, w_1), \cdots, (v_k, w_k) \in E^h$ belong to a homologous edge set of size k in G^h. As shown in Figure 2[7], we replace $(v_i, w_i), i \in \{2, \cdots, k-1\}$ by two edges (v_i, z_i) and (z_i, w_i) such that z_i is a new vertex incident only to these two edges, and edge capacities of the two new edges are the same as that of the original edge (v_i, w_i). Then, we can construct $k-1$ pairs of homologous edges: (v_1, w_1) and (v_2, z_2); (z_2, w_2) and (v_3, z_3); $\cdots$; (z_i, w_i) and (v_{i+1}, z_{i+1}); $\cdots$; (z_{k-1}, w_{k-1}) and (v_k, w_k). In addition, no reduction is performed on non-homologous edges in G^h, and we trivially copy these edges to G^p.

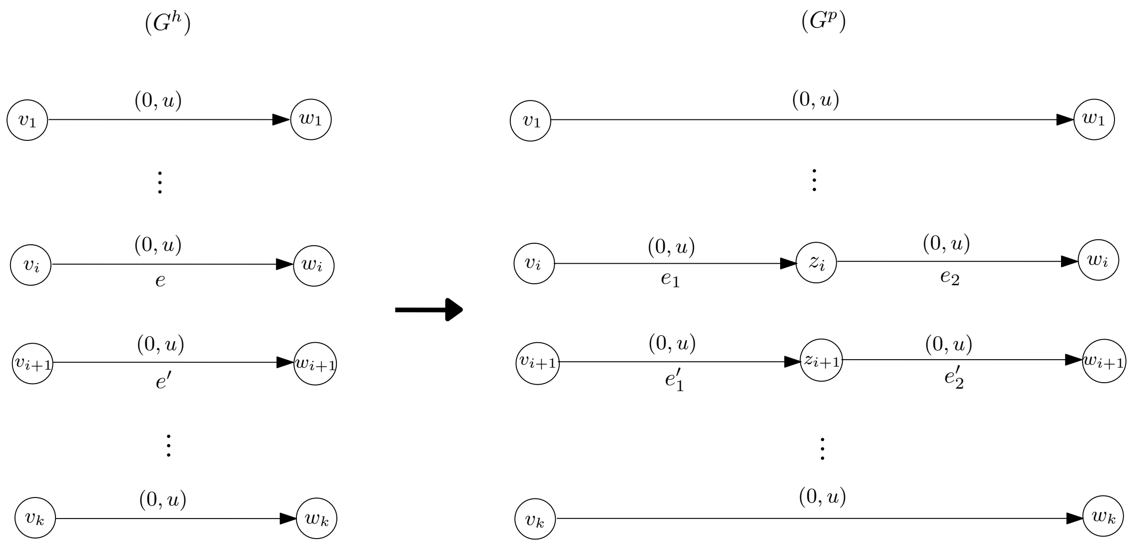

Figure 2: The reduction from FHF to FPHF.

If an FPHF solver returns $\boldsymbol{f}^p$ for the FPHF instance $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t)$, then we return $\boldsymbol{f}^h$ for the FHF instance $(G^h, H^h, \mathcal{H}^h, \boldsymbol{u}^h, s, t)$, by setting $\boldsymbol{f}^h(e) = \boldsymbol{f}^p(e_1)$ if e is a homologous edge and splits into two edges e_1, e_2 in the reduction such that e_1 precedes e_2, and setting $\boldsymbol{f}^h(e) = \boldsymbol{f}^p(e)$ otherwise. If the FPHF solver returns "infeasible" for the FPHF instance, then we return "infeasible" for the FHF instance.

Lemma 4.13 (FHF to FPHF)**.** *Given an* FHF *instance $(G^h, F^h, \mathcal{H}^h = (H_1, \cdots, H_h), \boldsymbol{u}^h, s, t)$, we can construct, in time $O(|E^h|)$, an* FPHF *instance $(G^p, F^p, \mathcal{H}^p = (H_1, \cdots, H_p), \boldsymbol{u}^p, s, t)$ such that*

$$|V^p| \leq |V^h| + |E^h|, \qquad |E^p| \leq 2|E^h|, \qquad |F^p| = |F^h|, \qquad p \leq |E^h|, \qquad \|\boldsymbol{u}^p\|_{\max} = \left\|\boldsymbol{u}^h\right\|_{\max}.$$

If the FHF *instance has a solution, then the* FPHF *instance has a solution.*

Proof. According to the reduction described above, from any solution $\boldsymbol{f}^h$ to the FHF instance, it is easy to derive a solution $\boldsymbol{f}^p$ to the FPHF instance. Concretely, for any homologous edge $e \in E^h$ that is split into two edges $e_1, e_2 \in E^p$, we set $\boldsymbol{f}^p(e_1) = \boldsymbol{f}^p(e_2) = \boldsymbol{f}^h(e)$. Since the vertex between e_1 and e_2 is only incident to these two edges, then the conservation of flows is satisfied on the inserted vertices. Moreover, since the edge capacity of e_1 and e_2 are the same as that of e, and they

[7]We use $(0, u)$ for an non-fixed flow edge of capacity u.

route the same amount of flows, thus the capacity constraint is also satisfied on the split edges. In addition, conservation of flows and capacity constraints are trivially satisfied for the rest vertices and edges. Therefore, $\boldsymbol{f}^p$ is a feasible flow to the FPHF instance.

Now, we track the change of problem size after reduction. Based on the reduction method, given an FHF instance with $|V^h|$ vertices, $|E^h|$ edges (including $|H^h|$ fixed flow edges, and h homologous edge sets $\mathcal{H}^h = \{H_1, \cdots, H_h\}$), we can compute the size of the reduced FPHFA instance as follows.

1. $|V^p|$ vertices. Since all vertices in V^h are maintained, and for each homologous edge set $H_i \in \mathcal{H}^h$, $|H_i| - 2$ new vertices are inserted, thus we have
$$\begin{aligned} |V^p| &= |V^h| + \sum_{i \in [h]} (|H_i| - 2) \\ &\leq |V^h| + |E^h|. \qquad\qquad \text{Because } \sum_{i \in [h]} |H_i| \leq |E^h| \end{aligned}$$

2. $|E^p|$ edges. Since all edges in E^h are maintained and a new edge is generated by inserting a new vertex, thus we have
$$|E^p| = |E^h| + \sum_{i \in [h]} (|H_i| - 2) \leq 2|E^h|.$$

3. $|F^p|$ fixed flow edges, where $|F^p| = |F^h|$. It is because all fixed flow edges in F^h are maintained, and no new fixed flow edges are generated by reduction of this step.

4. p pairs of homologous edges. Since each homologous edge set $H_i \in \mathcal{H}^h$ can be transformed into $|H_i| - 1$ pairs, then we have $p = \sum_{i \in [h]} (|H_i| - 1) \leq |E^h|$.

5. The maximum edge capacity is bounded by $\|\boldsymbol{u}^p\|_{\max} = \left\|\boldsymbol{u}^h\right\|_{\max}$, because the reduction of this step only separate edges by inserting new vertices without modifying edge capacities.

To estimate the reduction time, inserting a new vertex takes a constant time and there are $O(|E^h|)$ new vertices to be inserted. Also, it takes constant time to copy each of the rest $O(|E^h|)$ non-homologous edges. Hence, the reduction of this step takes $O(|E^h|)$ time. □

4.5.2 FHFA to FPHFA

Definition 4.14 (FPHF Approximate Problem (FPHFA))**.** An FPHFA instance is given by an FPHF instance $(G, F, \mathcal{H} = \{H_1, \cdots, H_p\}, \boldsymbol{u}, s, t)$ as in Definition 2.10, and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(G, F, \mathcal{H}, \boldsymbol{u}, s, t, \epsilon)$. We say an algorithm solves the FPHFA problem, if, given any FPHFA instance, it returns a flow $\boldsymbol{f} \geq \boldsymbol{0}$ that satisfies

$$\boldsymbol{u}(e) - \epsilon \leq \boldsymbol{f}(e) \leq \boldsymbol{u}(e) + \epsilon, \ \forall e \in F \tag{28}$$

$$0 \leq \boldsymbol{f}(e) \leq \boldsymbol{u}(e) + \epsilon, \ \forall e \in E \backslash F \tag{29}$$

$$\left| \sum_{u:(u,v) \in E} \boldsymbol{f}(u, v) - \sum_{w:(v,w) \in E} \boldsymbol{f}(v, w) \right| \leq \epsilon, \ \forall v \in V \backslash \{s, t\} \tag{30}$$

$$|\boldsymbol{f}(v, w) - \boldsymbol{f}(y, z)| \leq \epsilon, \ \forall H_i \ni (v, w), (y, z), i \in [p] \tag{31}$$

or it correctly declares that the associated FPHF instance is infeasible. We refer to the error in (28) and (29) as error in congestion, error in (30) as error in demand, and error in (31) as error in pair homology.

We can use the same reduction and solution mapping method in the exact case to the approximate case.

Lemma 4.15 (FHFA to FPHFA)**.** *Given an* FHFA *instance* $(G^h, F^h, \mathcal{H}^h = (H_1, \cdots, H_h), \boldsymbol{u}^h, s, t, \epsilon^h)$, *if we use Lemma 4.13 to construct an* FPHF *instance* $(G^p, F^p, \mathcal{H}^p = (H_1, \cdots, H_p), \boldsymbol{u}^p, s, t)$ *from the* FHF *instance* $(G^h, H^h, \mathcal{H}^h, \boldsymbol{u}^h, s, t)$*, and let*

$$\epsilon^p = \frac{\epsilon^h}{|E^h|},$$

then the FPHFA *instance* $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t, \epsilon^p)$ *satisfies.*

Furthermore, if $\boldsymbol{f}^p$ *is a solution to the* FPHF(A) *instance, then in time* $O(\left|E^h\right|)$*, we can compute a solution* $\boldsymbol{f}^h$ *to the* FHF(A) *instance, where the exact case holds when* $\epsilon^p = \epsilon^h = 0$*.*

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each homologous or non-homologous entry of $\boldsymbol{f}^h$, and $\boldsymbol{f}^h$ has $|E^h|$ entries, such a solution mapping takes $O(|E^h|)$ time.

Now, we conduct an error analysis. By the error notions of FHFA (Definition 4.11), there are three types of error after solution mapping: (1) error in congestion; (2) error in demand; (3) error in homology.

1. Error in congestion.
   We first track the error of the upper bound of capacity. For any homologous edge e, the maximum flow to be routed in e is bounded by

   $$\boldsymbol{u}^h(e) + \tau_u^h = \boldsymbol{f}^p(e_1) \leq \boldsymbol{u}^p(e_1) + \epsilon^p = \boldsymbol{u}^h(e) + \epsilon^p,$$

   thus, $\tau_u^h \leq \epsilon^p$. This τ_u^h also applies to the other edges since no reduction is made on them.

   Next, we track the error of the lower bound of capacity. Since only fixed flow edges have lower bound of capacity, and no reduction is made on fixed flow edges, then we obtain trivially that $\tau_l^h \leq \epsilon^p$.

2. Error in demand.
   For simplicity, we denote $H^h = \bigcup_{i\in[h]} H_i$ as the set of all homologous edges. By Eq. (25) in Definition 4.11, the error in demand that we can achieve for vertices other than s, t in G^h is

computed as

$$
\begin{aligned}
\tau_d^h &= \max_{w\in V^h\backslash\{s,t\}} \left| \sum_{(v,w)\in E^h} \boldsymbol{f}^h(v,w) - \sum_{(w,u)\in E^h} \boldsymbol{f}^h(w,u) \right| \\
&\overset{(1)}{=} \max_{w\in V^h\backslash\{s,t\}} \left| \left(\sum_{(v,w)\in H^h} \boldsymbol{f}^h(v,w) + \sum_{(v,w)\in E^h\backslash H^h} \boldsymbol{f}^h(v,w) \right) - \left(\sum_{(w,u)\in H^h} \boldsymbol{f}^h(w,u) + \sum_{(w,u)\in E^h\backslash H^h} \boldsymbol{f}^h(w,u) \right) \right| \\
&\overset{(2)}{=} \max_{w\in V^h\backslash\{s,t\}} \left| \left(\sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_1) + \sum_{(v,w)\in E^h\backslash H^h} \boldsymbol{f}^p(v,w) \right) - \left(\sum_{e=(w,u)\in H^h} \boldsymbol{f}^p(e_1) + \sum_{(w,u)\in E^h\backslash H^h} \boldsymbol{f}^p(w,u) \right) \right| \\
&\overset{(3)}{\leq} \epsilon^p + \max_{w\in V^h\backslash\{s,t\}} \left| \left(\sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_1) + \sum_{(v,w)\in E^h\backslash H^h} \boldsymbol{f}^p(v,w) \right) - \left(\sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_2) + \sum_{(v,w)\in E^h\backslash H^h} \boldsymbol{f}^p(v,w) \right) \right| \\
&= \epsilon^p + \max_{w\in V^h\backslash\{s,t\}} \left| \sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_1) - \sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_2) \right| \\
&= \epsilon^p + \max_{w\in V^h\backslash\{s,t\}} \left| \sum_{e=(v,w)\in H^h} (\boldsymbol{f}^p(e_1) - \boldsymbol{f}^p(e_2)) \right| \\
&\leq \epsilon^p + \max_{w\in V^h\backslash\{s,t\}} \sum_{e=(v,w)\in H^h} |\boldsymbol{f}^p(e_1) - \boldsymbol{f}^p(e_2)| \\
&\overset{(4)}{\leq} \epsilon^p + |H^h|\epsilon^p \\
&\overset{(5)}{\leq} |E^h|\epsilon^p.
\end{aligned}
\tag{32}
$$

For step (1), we separate homologous edges from other edges that are incident to vertex w. For step (2), we apply the rule of mapping $\boldsymbol{f}^p$ back to $\boldsymbol{f}^h$. For step (3), an example of a vertex $w \in V^h\backslash\{s,t\}$ is illustrated in Figure 3, where w has an incoming homologous edge (v,w) and an outgoing homologous edge (w,u). We apply Eq. (30) in Definition 4.14 with respect to vertex w, so that we can replace the sum of outgoing flows of w by the sum of its incoming flows with an error in demand of G^p being introduced, i.e.,

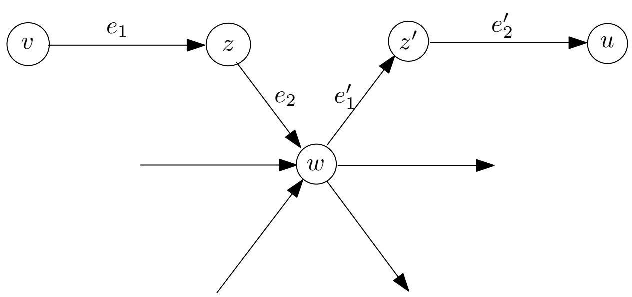

Figure 3: An example of the incoming and outgoing flows of vertex $w \in V^h\backslash\{s,t\}$ when get reduced to G^p. Suppose in G^p, $e = (v,w), e' = (w,u)$ are homologous edges incident to w, and the rest edges incident to w are non-homologous.

$$
\left| \left(\sum_{e=(w,u)\in H^h} \boldsymbol{f}^p(e_1) + \sum_{(w,u)\in E^h\backslash H^h} \boldsymbol{f}^p(w,u) \right) - \left(\sum_{e=(v,w)\in H^h} \boldsymbol{f}^p(e_2) + \sum_{(v,w)\in E^h\backslash H^h} \boldsymbol{f}^p(v,w) \right) \right| \leq \epsilon^p.
$$

For step (4), we apply Eq. (30) again with respect to the new inserted vertex. Since the new inserted vertex is only incident to e_1 and e_2, we have $|\boldsymbol{f}^p(e_1) - \boldsymbol{f}^p(e_2)| \leq \epsilon^p$. For step (5), we utilize $|E^h| - |H^h| \geq 1$, i.e., there are more than one non-homologous edges in G^h.

3. Error in homology.
The error in homology in G^h for a homologous edge set of size k get accumulated by $(k-1)$ times of error in pair homology in G^p. Thus,

$$\tau_h^h = \max_{\substack{e,e' \in H_i \in \mathcal{H}^h \\ |H_i| = k}} \left| \boldsymbol{f}^h(e) - \boldsymbol{f}^h(e') \right| = \max_{\substack{e,e' \in H_i \in \mathcal{H}^h \\ |H_i| = k}} \left| \boldsymbol{f}^p(e_1) - \boldsymbol{f}^p(e_1') \right| \leq (k-1)\epsilon^p \leq |E^h|\epsilon^p.$$

To summarize, as we set in the reduction that $\epsilon^p = \frac{\epsilon^h}{|E^h|}$, then we have

$$\tau_l^h, \tau_u^h, \tau_d^h, \tau_h^h \leq \epsilon^h,$$

indicating that $\boldsymbol{f}^h$ is a solution to the FHFA instance.

□

4.6 FPHF(A) to SFF(A)

4.6.1 FPHF to SFF

We show the reduction from an FPHF instance $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t)$ to an SFF instance $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$. Assume that $\{e, \hat{e}\} \in \mathcal{H}^p$ is an arbitrary homologous edge set in G^p. As shown in Figure 4, we map $\{e, \hat{e}\}$ in G^p to a gadget consisting edges $\{e_1, e_2, e_3, e_4, e_5 = \hat{e}_3, \hat{e}_1, \hat{e}_2, \hat{e}_4, \hat{e}_5\}$ in G^s. The key idea to remove the homologous requirement is to introduce a second commodity between a source-sink pair (s_2, t_2). Concretely, we impose the fixed flow constraints on $(e_4, \hat{e}_4)$, the selective constraint of commodity 1 on $(e_1, e_2, \hat{e}_1, \hat{e}_2)$, and the selective constraint of commodity 2 on $(e_3, e_5/\hat{e}_3, \hat{e}_5)$. Then, there is a flow of commodity 2 that routes through the directed path $e_3 \to e_4 \to e_5/\hat{e}_3 \to \hat{e}_4 \to \hat{e}_5$, and a flow of commodity 1 through paths $e_1 \to e_4 \to e_2$ and $\hat{e}_1 \to \hat{e}_4 \to \hat{e}_2$. The fixed flow constraint on $(e_4, \hat{e}_4)$ forces the flow of commodity 1 through the two paths to be equal, since the flows of commodity 2 on $(e_4, \hat{e}_4)$ are equal. Thus, the homologous requirement for edge e and $\hat{e}$ is simulated. In addition, as no reduction is performed on non-homologous edges in G^p, we trivially copy these edges to G^s, and restrict these edges to be selective for commodity 1.

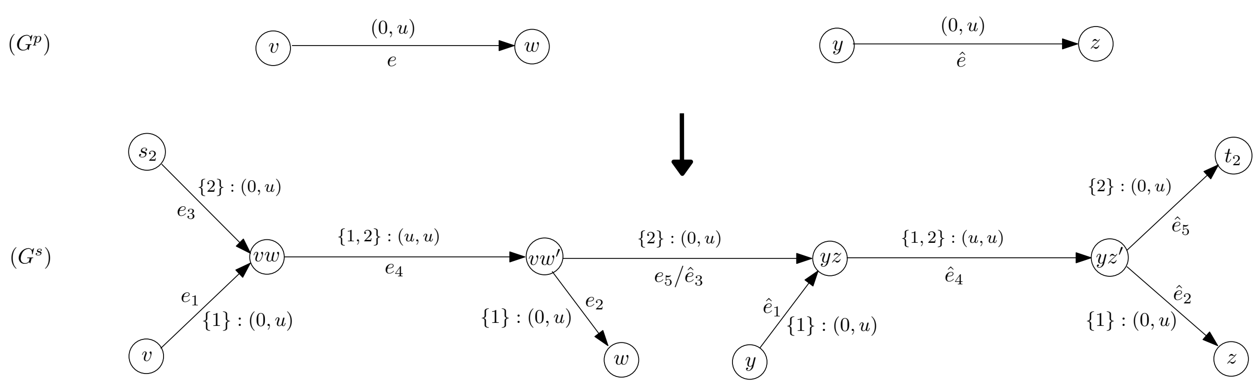

Figure 4: The reduction from FPHF to SFF.

If an SFF solver returns $\boldsymbol{f}^s$ for the SFF instance $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$, then we return $\boldsymbol{f}^p$ for the FPHF instance $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t)$ by the following method: for each pair of homologous edges e and $\hat{e}$, we set $\boldsymbol{f}^p(e) = \boldsymbol{f}^s_1(e_1), \boldsymbol{f}^p(\hat{e}) = \boldsymbol{f}^s_1(\hat{e}_1)$; for each non-homologous edge e', we set $\boldsymbol{f}^p(e') = \boldsymbol{f}^s_1(e')$. Note that $\boldsymbol{f}^s$ is a two-commodity flow[8] while $\boldsymbol{f}^p$ is a single-commodity flow. If the SFF solver returns "infeasible" for the SFF instance, then we return "infeasible" for the FPHF instance.

Lemma 4.16 (FPHF to SFF)**.** *Given an* FPHF *instance* $(G^p, F^p, \mathcal{H}^p = (H_1, \cdots, H_p), \boldsymbol{u}^p, s, t)$, *we can construct, in time* $O(|E^p|)$, *an* SFF *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$ *such that*

$$|V^s| = |V^p| + 4p + 2, \qquad |E^s| = |E^p| + 7p, \qquad |F^s| = |F^p| + 2p,$$

$$|S_1| = |E^p| + 2p, \qquad |S_2| = 3p, \qquad \|\boldsymbol{u}^s\|_{\max} = \|\boldsymbol{u}^p\|_{\max}.$$

If the FPHF *instance has a solution, then the* SFF *instance has a solution.*

Proof. According to the reduction described above, from any solution $\boldsymbol{f}^p$ to the FPHF instance, it is easy to derive a solution $\boldsymbol{f}^s$ to the SFF instance. Concretely, we define a feasible flow $\boldsymbol{f}^s$ as follows:

- For any pair of homologous edge $\{e, \hat{e}\} \in \mathcal{H}^p$, in its corresponding gadget in G^s, we set

  $$\boldsymbol{f}^s_1(e_1) = \boldsymbol{f}^s_1(e_4) = \boldsymbol{f}^s_1(e_2) = \boldsymbol{f}^p(e) = \boldsymbol{f}^p(\hat{e}) = \boldsymbol{f}^s_1(\hat{e}_1) = \boldsymbol{f}^s_1(\hat{e}_4) = \boldsymbol{f}^s_1(\hat{e}_2) \le u,$$

  where $u = \boldsymbol{u}^p(e) = \boldsymbol{u}^p(\hat{e})$, and set

  $$\boldsymbol{f}^s_2(e_1) = \boldsymbol{f}^s_2(e_2) = \boldsymbol{f}^s_2(\hat{e}_1) = \boldsymbol{f}^s_2(\hat{e}_2) = 0.$$

  It is obvious that $\boldsymbol{f}^s$ satisfies the selective constraint and the capacity constraint on edges $e_1, e_2, \hat{e}_1, \hat{e}_2$, and satisfies conservation of flows for commodity 1 on vertices vw, vw', yz, yz'.

  Moreover, we set

  $$\boldsymbol{f}^s_2(e_3) = \boldsymbol{f}^s_2(e_4) = \boldsymbol{f}^s_2(e_5) = \boldsymbol{f}^s_2(\hat{e}_4) = \boldsymbol{f}^s_2(\hat{e}_5) = u - \boldsymbol{f}^p(e) \le u,$$

  $$\boldsymbol{f}^s_1(e_3) = \boldsymbol{f}^s_1(e_4) = \boldsymbol{f}^s_1(e_5) = \boldsymbol{f}^s_1(\hat{e}_4) = \boldsymbol{f}^s_1(\hat{e}_5) = 0.$$

  Then it is obvious that $\boldsymbol{f}^s$ satisfies the selective constraint and the capacity constraint on edges $e_3, e_5, \hat{e}_5$, and satisfies the flow conservation constraint of commodity 2 on vertices vw, vw', yz, yz'.

  It remains to verify if $\boldsymbol{f}^s$ satisfies the fixed flow constraint on edges $e_4, \hat{e}_4$. According to the above constructions, we have (we abuse the notation to also let $\boldsymbol{f}^s = \boldsymbol{f}^s_1 + \boldsymbol{f}^s_1$)

  $$\boldsymbol{f}^s(e_4) = \boldsymbol{f}^s_1(e_4) + \boldsymbol{f}^s_2(e_4) = \boldsymbol{f}^p(e) + (u - \boldsymbol{f}^p(e)) = u,$$

  $$\boldsymbol{f}^s(\hat{e}_4) = \boldsymbol{f}^s_1(\hat{e}_4) + \boldsymbol{f}^s_2(\hat{e}_4) = \boldsymbol{f}^p(\hat{e}) + (u - \boldsymbol{f}^p(\hat{e})) = u.$$

- For any non-homologous edge $e' \in E^p$, we also have $e' \in E^s$ since no reduction is made on this edge, and we copy it trivially to G^s. We set

  $$\boldsymbol{f}^s_1(e') = \boldsymbol{f}^p(e'), \qquad \boldsymbol{f}^s_2(e') = 0.$$

  Since $\boldsymbol{f}^p$ is a feasible flow in G^p, it is easy to check that $\boldsymbol{f}^s$ also satisfies the selective constraint for commodity 1 and the capacity constraint on non-homologous edges, as well as conservation of flows on vertices incident to non-homologous edges.

[8]We denote $\boldsymbol{f} = \boldsymbol{f}_1 + \boldsymbol{f}_2$ for two-commodity flows.

To conclude, $\boldsymbol{f}^s$ is a feasible flow to the SFF instance.

Now, we track the change of problem size after reduction. Based on the reduction method, given an FPHF instance with $|V^p|$ vertices, $|E^p|$ edges (including $|F^p|$ fixed flow edges, and p pairwise homologous edge sets $\mathcal{H}^p = \{H_1, \cdots, H_p\}$), we can compute the size of the reduced SFF instance as follows.

1. $|V^s|$ vertices. To bound it, first, all vertices in V^p are maintained. Then, for each pair of homologous edges (v, w) and (y, z), 4 auxiliary vertices vw, vw', yz, yz' are added. And finally, a source-sink pair of commodity 2 (s_2, t_2) is added. Therefore, we have $|V^s| = |V^p| + 4p + 2$.

2. $|E^s|$ edges. Since non-homologous edges are maintained, and each pair of homologous edges is replaced by a gadget with 9 edges, thus we have $|E^s| = (|E^p| - 2p) + 9p = |E^p| + 7p$.

3. $|F^s|$ fixed flow edges. First, all fixed flow edges in F^p are maintained since by Def. 2.10, fixed flow edges and homologous edges are disjoint, thus no reduction is made. Then, for each pair of homologous edges as shown in Figure 4, the reduction generate 2 fixed flow edges (i.e., $e_4, \hat{e}_4$). Hence, we have $|F^s| = |F^p| + 2p$.

4. $|S_i|$ edges that select the i-th commodity. First, all non-homologous edges in E^p are selective for the commodity 1. Then, for each pair of homologous edges as shown in Figure 4, the reduction generates 4 edges selecting commodity 1 (i.e., $e_1, e_2, \hat{e}_1, \hat{e}_2$) and 3 edges selecting commodity 2 (i.e., $e_3, e_5/\hat{e}_3, \hat{e}_5$). Thus, we have $|S_1| = (|E^p| - 2p) + 4p = |E^p| + 2p$ and $|S_2| = 3p$.

5. The maximum edge capacity is bounded by $\|\boldsymbol{u}^s\|_{\max} = \|\boldsymbol{u}^p\|_{\max}$. First, capacity of non-homologous edges is unchanged in G^s since no reduction is made. Then, for each pair of homologous edges $\{e, \hat{e}\}$ with capacity $\boldsymbol{u}^p(e)(= \boldsymbol{u}^p(\hat{e}))$, the capacity of the corresponding 9 edges are either selective edges with capacity $\boldsymbol{u}^p(e)$ or fixed flow edges with fixed flow $\boldsymbol{u}^p(e)$.

To estimate the reduction time, first, it takes constant time to reduce each pair of homologous edges since only a constant number of vertices and edges are added. Also, it takes constant time to copy each of the rest non-homologous edges, then the reduction of this step takes $O(|E^p|)$ time in total. □

4.6.2 FPHFA to SFFA

Definition 4.17 (SFF Approximate Problem (SFFA))**.** An SFFA instance is given by an SFF instance $(G, F, S_1, S_2, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ as in Definition 2.8, and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(G, F, S_1, S_2, \boldsymbol{u}, s_1, t_1, s_2, t_2, \epsilon)$. We say an algorithm solves the SFFA problem, if, given any SFFA instance, it returns a pair of flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \mathbf{0}$ that satisfies

$$\boldsymbol{u}(e) - \epsilon \leq \boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \quad \forall e \in F \tag{33}$$

$$0 \leq \boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \quad \forall e \in E \backslash F \tag{34}$$

$$\left| \sum_{u:(u,v) \in E} \boldsymbol{f}_i(u, v) - \sum_{w:(v,w) \in E} \boldsymbol{f}_i(v, w) \right| \leq \epsilon, \quad \forall v \in V \backslash \{s_i, t_i\}, \ i \in \{1, 2\} \tag{35}$$

$$\boldsymbol{f}_{\bar{i}}(e) \leq \epsilon, \quad \forall e \in S_i, \ \bar{i} = \{1, 2\} \backslash i \tag{36}$$

or it correctly declares that the associated SFF instance is infeasible. We refer to the error in (33) and (34) as error in congestion, error in (35) as error in demand, and error in (36) as error in type.

We can use the same reduction and solution mapping method in the exact case to the approximate case.

Lemma 4.18 (FPHFA to SFFA)**.** *Given an* FPHFA *instance* $(G^p, F^p, \mathcal{H}^p = (H_1, \cdots, H_p), \boldsymbol{u}^p, s, t, \epsilon^p)$*, if we use Lemma 4.16 to construct an* SFF *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$ *from the* FPHF *instance* $(G^p, F^p, \mathcal{H}^p, \boldsymbol{u}^p, s, t)$*, and let*

$$\epsilon^s = \frac{\epsilon^p}{11|E^p|},$$

then the SFFA *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2, \epsilon^s)$ *satisfies.*

Furthermore, if $\boldsymbol{f}^s$ *is a solution to the* SFF(A) *instance, then in time* $O(|E^p|)$*, we can compute a solution* $\boldsymbol{f}^p$ *to the* FPHF(A) *instance, where the exact case holds when* $\epsilon^s = \epsilon^p = 0$*.*

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each homologous or non-homologous entry of $\boldsymbol{f}^p$. Since $\boldsymbol{f}^p$ has $|E^p|$ entries, such a solution mapping takes $O(|E^p|)$ time.

Now, we conduct an error analysis. By the error notions of FPHPA (Definition 4.14), there are three types of error after solution mapping: (1) error in congestion; (2) error in demand; (3) error in pair homology.

1. Error in congestion.
   We first track the error of the upper bound of capacity. For any pair of homologous edges e and $\hat{e}$ with capacity $\boldsymbol{u}^p(e) = \boldsymbol{u}^p(\hat{e})$, the maximum flow to be routed in e and $\hat{e}$ are

   $$\boldsymbol{u}^p(e) + \tau_u^p = \boldsymbol{f}_1^s(e_1) \leq \boldsymbol{f}^s(e_1) \leq \boldsymbol{u}^s(e_1) + \epsilon^s = \boldsymbol{u}^p(e) + \epsilon^s,$$

   $$\boldsymbol{u}^p(\hat{e}) + \tau_u^p = \boldsymbol{f}_1^s(\hat{e}_1) \leq \boldsymbol{f}^s(\hat{e}_1) \leq \boldsymbol{u}^s(\hat{e}_1) + \epsilon^s = \boldsymbol{u}^p(\hat{e}) + \epsilon^s,$$

   thus, we have $\tau_u^p \leq \epsilon^s$. This bound also applies for non-homologous edges since no reduction is conducted.

   Next, we track the error of the lower bound of capacity. And we only need to take fixed flow edges (thus non-homologous edges) into consideration. For any fixed flow edge $e' \in F^p$, as it is copied to an edge selecting commodity 1 in G^s, then the minimum flow to be routed in e' can be lower bounded by

   $$\begin{aligned}
   \boldsymbol{u}^p(e') - \tau_l^p &= \boldsymbol{f}_1^s(e') \\
   &\geq \boldsymbol{f}^s(e') - \epsilon^s && \text{Because of error in type in } G^s \\
   &\geq \boldsymbol{u}^s(e') - 2\epsilon^s && \text{Because of error in congestion in } G^s \\
   &= \boldsymbol{u}^p(e') - 2\epsilon^s && \text{Because } \boldsymbol{u}^p(e') = \boldsymbol{u}^s(e')
   \end{aligned}$$

   Thus, we have $\tau_l^p \leq 2\epsilon^s$.

2. Error in demand.
   For simplicity, we denote $H^p = \bigcup_{i \in [p]} H_i$ as the set of all homologous edges. With the similar strategy of Eq. (32), the error in demand that we can achieve for vertices other than s, t in

G^p is computed as

$$\begin{aligned}
\tau_d^p &= \max_{w \in V^p \setminus \{s,t\}} \left| \sum_{(v,w)\in E^p} \boldsymbol{f}^p(v,w) - \sum_{(w,u)\in E^p} \boldsymbol{f}^p(w,u) \right| \\
&= \max_{w \in V^p \setminus \{s,t\}} \left| \left(\sum_{(v,w)\in H^p} \boldsymbol{f}^p(v,w) + \sum_{(v,w)\in E^p \setminus H^p} \boldsymbol{f}^p(v,w) \right) - \left(\sum_{(w,u)\in H^p} \boldsymbol{f}^p(w,u) + \sum_{(w,u)\in E^p \setminus H^p} \boldsymbol{f}^p(w,u) \right) \right| \\
&= \max_{w \in V^p \setminus \{s,t\}} \left| \left(\sum_{e=(v,w)\in H^p} \boldsymbol{f}_1^s(e_1) + \sum_{(v,w)\in E^p \setminus H^p} \boldsymbol{f}_1^s(v,w) \right) - \left(\sum_{e=(w,u)\in H^p} \boldsymbol{f}_1^s(e_1) + \sum_{(w,u)\in E^p \setminus H^p} \boldsymbol{f}_1^s(w,u) \right) \right| \\
&\le \epsilon^s + \max_{w \in V^p \setminus \{s,t\}} \left| \left(\sum_{e=(v,w)\in H^p} \boldsymbol{f}_1^s(e_1) + \sum_{(v,w)\in E^p \setminus H^p} \boldsymbol{f}_1^s(v,w) \right) - \left(\sum_{e=(v,w)\in H^p} \boldsymbol{f}_1^s(e_2) + \sum_{(v,w)\in E^p \setminus H^p} \boldsymbol{f}_1^s(v,w) \right) \right| \\
&= \epsilon^s + \max_{w \in V^p \setminus \{s,t\}} \left| \sum_{e=(v,w)\in H^p} \boldsymbol{f}_1^s(e_1) - \sum_{e=(v,w)\in H^p} \boldsymbol{f}_1^s(e_2) \right| \\
&= \epsilon^s + \max_{w \in V^p \setminus \{s,t\}} \left| \sum_{e=(v,w)\in H^p} (\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)) \right| \\
&\le \epsilon^s + \max_{w \in V^p \setminus \{s,t\}} \sum_{e=(v,w)\in H^p} \left| \boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2) \right| .
\end{aligned} \tag{37}$$

Now, we try to bound $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)|$. We apply Eq. (35) again with respect to vertex vw. Since vw is only incident to e_1, e_3, e_4, we have

$$|\boldsymbol{f}_1^s(e_1) + \boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(e_4)| \le \epsilon^s. \tag{38}$$

Similarly, by error in demand of vertex vw', we have

$$|\boldsymbol{f}_1^s(e_2) + \boldsymbol{f}_1^s(e_5) - \boldsymbol{f}_1^s(e_4)| \le \epsilon^s. \tag{39}$$

Combining Eq. (38) and Eq. (39), we have

$$|(\boldsymbol{f}_1^s(e_1) + \boldsymbol{f}_1^s(e_3)) - (\boldsymbol{f}_1^s(e_2) + \boldsymbol{f}_1^s(e_5))| \le 2\epsilon^s.$$

Moreover, we can also lower bound its left hand side by

$$\begin{aligned}
|(\boldsymbol{f}_1^s(e_1) + \boldsymbol{f}_1^s(e_3)) - (\boldsymbol{f}_1^s(e_2) + \boldsymbol{f}_1^s(e_5))| &= |(\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)) + (\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(e_5))| \\
&\ge |\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)| - |\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(e_5)| .
\end{aligned}$$

By rearranging, $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)|$ can be upper bounded by

$$\begin{aligned}
& |\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)| \\
\le & |\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(e_5)| + |(\boldsymbol{f}_1^s(e_1) + \boldsymbol{f}_1^s(e_3)) - (\boldsymbol{f}_1^s(e_2) + \boldsymbol{f}_1^s(e_5))| \\
\le & |\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(e_5)| + 2\epsilon^s \\
\le & \max\{\boldsymbol{f}_1^s(e_3), \boldsymbol{f}_1^s(e_5)\} + 2\epsilon^s && \text{Because } \boldsymbol{f}_1^s(e_3), \boldsymbol{f}_1^s(e_5) \ge 0 \\
\le & 3\epsilon^s,
\end{aligned} \tag{40}$$

where the last inequality comes from error in type for e_3 and e_5, since they are both selective edges for commodity 2.

Finally, applying the upper bound of $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)|$ Eq. (40) back to Eq. (37), we obtain $\tau_d^p \le \epsilon^s + 3|H^p|\epsilon^s \le 3|E^p|\epsilon^s$.

3. Error in pair homology.

   By Eq. (31) in Definition 4.14, the error in pair homology that we can achieve is computed as

$$
\begin{aligned}
\tau_h^p &= \max_{\mathcal{H}^p \ni H_i = \{(v,w),(y,z)\}} \left| \boldsymbol{f}^p(v,w) - \boldsymbol{f}^p(y,z) \right| \\
&= \max_{\mathcal{H}^p \ni H_i = \{e=(v,w), \hat{e}=(y,z)\}} \left| \boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1) \right|,
\end{aligned} \tag{41}
$$

   where the last equality comes from the rule of mapping $\boldsymbol{f}^s$ back to $\boldsymbol{f}^p$ for homologous edges.

   Now, we try to bound $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1)|$ for an arbitrary pair of homologous edges $e, \hat{e}$ with capacity $\boldsymbol{u}^p(e) (= \boldsymbol{u}^p(\hat{e}))$. By error in congestion on e_4 and $\hat{e}_4$, we have

$$
|\boldsymbol{f}^s(e_4) - \boldsymbol{f}^s(\hat{e}_4)| \le 2\epsilon^s.
$$

   By error in demand on vertices vw and yz for two commodities, we have

$$
\begin{aligned}
|\boldsymbol{f}^s(e_1) + \boldsymbol{f}^s(e_3) - \boldsymbol{f}^s(e_4)| &\le 2\epsilon^s, \\
|\boldsymbol{f}^s(\hat{e}_1) + \boldsymbol{f}^s(\hat{e}_3) - \boldsymbol{f}^s(\hat{e}_4)| &\le 2\epsilon^s.
\end{aligned}
$$

   Combining the above three inequalities, we have

$$
|(\boldsymbol{f}^s(e_1) + \boldsymbol{f}^s(e_3)) - (\boldsymbol{f}^s(\hat{e}_1) + \boldsymbol{f}^s(\hat{e}_3))| \le 6\epsilon^s.
$$

   Again, we can also lower bound its left hand side by

$$
\begin{aligned}
&|(\boldsymbol{f}^s(e_1) + \boldsymbol{f}^s(e_3)) - (\boldsymbol{f}^s(\hat{e}_1) + \boldsymbol{f}^s(\hat{e}_3))| \\
=& |(\boldsymbol{f}_1^s(e_1) + \boldsymbol{f}_2^s(e_1) + \boldsymbol{f}_1^s(e_3) + \boldsymbol{f}_2^s(e_3)) - (\boldsymbol{f}_1^s(\hat{e}_1) + \boldsymbol{f}_2^s(\hat{e}_1) + \boldsymbol{f}_1^s(\hat{e}_3) + \boldsymbol{f}_2^s(\hat{e}_3))| \\
\ge& |\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1)| - |(\boldsymbol{f}_2^s(e_1) - \boldsymbol{f}_2^s(\hat{e}_1)) + (\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(\hat{e}_3)) + (\boldsymbol{f}_2^s(e_3) - \boldsymbol{f}_2^s(\hat{e}_3))|.
\end{aligned}
$$

   By rearranging, we can upper bound $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1)|$ by

$$
\begin{aligned}
&|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1)| \\
\le& |(\boldsymbol{f}^s(e_1) + \boldsymbol{f}^s(e_3)) - (\boldsymbol{f}^s(\hat{e}_1) + \boldsymbol{f}^s(\hat{e}_3))| + |(\boldsymbol{f}_2^s(e_1) - \boldsymbol{f}_2^s(\hat{e}_1)) + (\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(\hat{e}_3)) + (\boldsymbol{f}_2^s(e_3) - \boldsymbol{f}_2^s(\hat{e}_3))| \\
\le& 6\epsilon^s + |\boldsymbol{f}_2^s(e_1) - \boldsymbol{f}_2^s(\hat{e}_1)| + |\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(\hat{e}_3)| + |\boldsymbol{f}_2^s(e_3) - \boldsymbol{f}_2^s(\hat{e}_3)|,
\end{aligned}
$$

   where we have

   - $|\boldsymbol{f}_2^s(e_1) - \boldsymbol{f}_2^s(\hat{e}_1)| \le \max\{\boldsymbol{f}_2^s(e_1), \boldsymbol{f}_2^s(\hat{e}_1)\} \le \epsilon^s$ because of error in type in G^s and $e_1, \hat{e}_1$ are selective edges for commodity 1;
   - $|\boldsymbol{f}_1^s(e_3) - \boldsymbol{f}_1^s(\hat{e}_3)| \le \max\{\boldsymbol{f}_1^s(e_3), \boldsymbol{f}_1^s(\hat{e}_3)\} \le \epsilon^s$ because of error in type in G^s and $e_3, \hat{e}_3$ are selective edges for commodity 2;
   - $|\boldsymbol{f}_2^s(e_3) - \boldsymbol{f}_2^s(\hat{e}_3)| \le 3\epsilon^s$, which can be obtained directly by symmetry from the bound of $|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(e_2)|$ in Eq. (40).

   Putting all together, we have

$$
|\boldsymbol{f}_1^s(e_1) - \boldsymbol{f}_1^s(\hat{e}_1)| \le 11\epsilon^s.
$$

   Thus, by Eq. (41), we have $\tau_h^p \le 11\epsilon^s$.

To summarize, as we set in the reduction that $\epsilon^s = \frac{\epsilon^p}{11|E^p|}$, then we have

$$
\tau_u^p, \tau_l^p, \tau_d^p, \tau_h^p \le \epsilon^p,
$$

indicating that $\boldsymbol{f}^p$ is a solution to the FPHFA instance. $\square$

4.7 SFF(A) to 2CFF(A)

4.7.1 SFF to 2CFF

We show the reduction from an SFF instance $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$ to a 2CFF instance $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$. Assume that $e \in S_i$ is an arbitrary selective edge for commodity i in G^p. As shown in Figure 5, we map e in G^s to a gadget consisting of edges $\{e_1, e_2, e_3, e_4, e_5\}$ in G^f. Note that a selective edge e can be either a fixed flow edge or a non-fixed flow edge. In Figure 5, $l = u$ if e is a fixed flow edge and $l = 0$ if e is a non-fixed flow edge. Moreover, no reduction is performed on non-selective edges in G^s, and we trivially copy these edges to G^f.

The key idea to remove the selective requirement is utilizing edge directions and the source-sink pair (s_i, t_i) to simulate a selective edge e for commodity i. More specifically, in the gadget, the flow of commodity i routes through three directed paths: (1) $e_1 \to e_4$, (2) $e_5 \to e_2 \to e_4$, (3) $e_5 \to e_3$. The selective requirement is realized because e_4 is the only outgoing edge of xy and only flow of commodity i is allowed in e_4 (since its tail is t_i), thus in e_1. Similarly, e_5 is the only incoming edge of xy' and only flow of commodity i is allowed in e_5, thus in e_3. In addition, to ensure that e_1 and e_3 route the same amount of flow, flows in e_4 and e_5 must be equal by the conservation of flows. Therefore, we impose the fixed flow constraint on e_4 and e_5 by setting the fixed flow to be u. We remove e_2 if e is a fixed flow edge (in which case e_2 has capacity 0), and set capacity of e_2 to be u otherwise (in which case $u - l = u - 0 = u$).

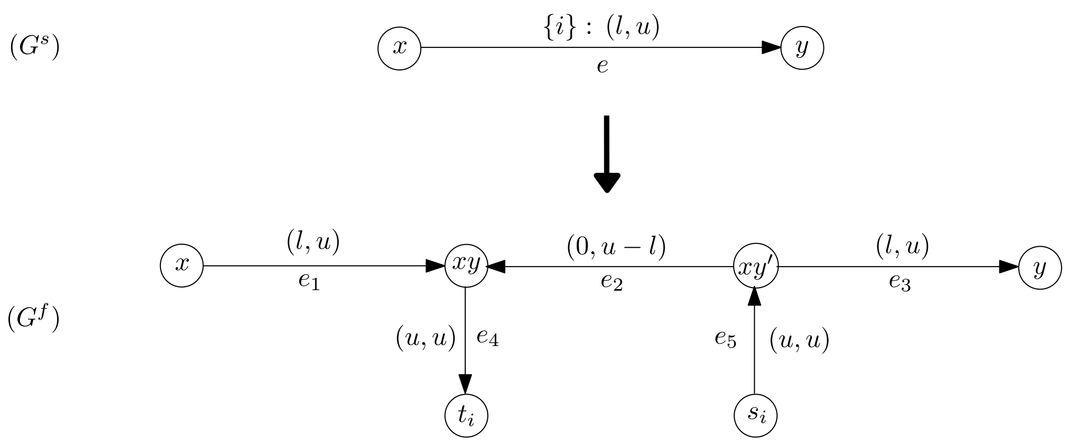

Figure 5: The reduction from SFF to 2CFF. $l = u$ if e is a fixed flow edge, and $l = 0$ if e is a non-fixed flow edge.

If a 2CFF solver returns $\boldsymbol{f}^f$ for the 2CFF instance $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$, then we return $\boldsymbol{f}^s$ for the SFF instance $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$ by the following method: for any selective edge $e \in S_i$, we set $\boldsymbol{f}_i^s(e) = \boldsymbol{f}_i^f(e_1), \boldsymbol{f}_{\bar{i}}^s(e) = \boldsymbol{f}_{\bar{i}}^f(e_1) = 0, \bar{i} \in \{1, 2\} \backslash i$; and for any non-selective edge $e \in E^s \backslash (S_1 \bigcup S_2)$, we map back trivially by setting $\boldsymbol{f}_i^s(e) = \boldsymbol{f}_i^f(e), i \in \{1, 2\}$. If the 2CFF solver returns “infeasible” for the 2CFF instance, then we return “infeasible” for the SFF instance.

Lemma 4.19 (SFF to 2CFF)**.** *Given an* SFF *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$*, we can construct, in time* $O(|E^s|)$*, a* 2CFF *instance* $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$ *such that*

$$|V^f| = |V^s| + 2(|S_1| + |S_2|), \;\; |E^f| = |E^s| + 4(|S_1| + |S_2|), \;\; |F^f| \leq 4(|F^s| + |S_1| + |S_2|), \;\; \left\|\boldsymbol{u}^f\right\|_{\max} = \|\boldsymbol{u}^s\|_{\max}.$$

If the SFF *instance has a solution, then the* 2CFF *instance has a solution.*

Proof. According to the reduction described above, from any solution $\boldsymbol{f}^s$ to the SFF instance, it is easy to derive a solution $\boldsymbol{f}^f$ to the 2CFF instance. Concretely, we define a feasible flow $\boldsymbol{f}^f$ as follows:

- For any selective edge $e \in S_i$, in its corresponding gadget in G^f, we set

$$l \leq \boldsymbol{f}_i^f(e_1) = \boldsymbol{f}_i^f(e_3) = \boldsymbol{f}_i^s(e) \leq u,$$

$$\boldsymbol{f}_i^f(e_4) = \boldsymbol{f}_i^f(e_5) = u,$$

$$0 \leq \boldsymbol{f}_i^f(e_2) = u - \boldsymbol{f}_i^s(e) \leq u - l,$$

  where $l = 0$ if e is a non-fixed flow edge, or $l = u = \boldsymbol{u}^s(e)$ if e is a fixed flow edge. And we set

$$\boldsymbol{f}_{\bar{i}}^f(e_1) = \boldsymbol{f}_{\bar{i}}^f(e_2) = \boldsymbol{f}_{\bar{i}}^f(e_3) = \boldsymbol{f}_{\bar{i}}^f(e_4) = \boldsymbol{f}_{\bar{i}}^f(e_5) = 0, \qquad \bar{i} = \{1,2\}\backslash i.$$

  It is obvious that $\boldsymbol{f}^f$ satisfies the capacity constraint on edges $(e_1, \ldots, e_5)$, and the flow conservation constraint on vertices xy, xy'.

- For any non-selective edge $e' \in E^s \backslash (S_1 \bigcup S_2)$, we also have $e' \in E^f$ since no reduction is made on this edge, and we copy it trivially to G^f. We set

$$\boldsymbol{f}_i^f(e') = \boldsymbol{f}_i^s(e'), \qquad i \in \{1,2\}.$$

  Since $\boldsymbol{f}^s$ is a feasible flow in G^s, it is easy to check that $\boldsymbol{f}^f$ also satisfies the capacity constraint on non-selective edges, as well as the flow conservation constraint on vertices incident to non-selective edges. Moreover, if e' is a fixed flow edge, the fixed flow edge constraint is also satisfied.

To conclude, $\boldsymbol{f}^f$ is a feasible flow to the 2CFF instance.

Now, we track the change of problem size after reduction. Based on the reduction method, given an SFF instance with $|V^s|$ vertices, $|E^s|$ edges (including $|F^s|$ fixed flow edges, and $|S_i|$ edges selecting commodity i), we can compute the size of the reduced 2CFF instance as follows.

1. $|V^f|$ vertices. As all vertices in V^s are maintained, and for each selective edge $(x,y) \in (S_1 \cup S_2)$, 2 auxiliary vertices xy, xy' are added, we have $|V^f| = |V^s| + 2(|S_1| + |S_2|)$.

2. $|E^f|$ edges. Since all non-selective edges in $E^s \backslash (S_1 \cup S_2)$ are maintained, and each selective edge $e \in (S_1 \cup S_2)$ is replaced by a gadget with 5 edges $(e_1, \cdots, e_5)$, we have

$$|E^f| = (|E^s| - |S_1| - |S_2|) + 5(|S_1| + |S_2|) = |E^s| + 4(|S_1| + |S_2|).$$

3. $|F^f|$ fixed flow edges. First, all non-selective fixed flow edges $e \in F^s \backslash (S_1 \cup S_2)$ are maintained since no reduction is made. Then, for all selective fixed flow edges $e \in F^s \bigcap (S_1 \cup S_2)$, the reduction generates 4 fixed flow edges (i.e., e_1, e_3, e_4, e_5). Finally, for all non-fixed flow selective edges $e \in (S_1 \cup S_2) \backslash F^s$, the reduction generates 2 fixed flow edges (i.e., e_4, e_5). Hence, we have

$$|F^f| = |F^s \backslash (S_1 \cup S_2)| + 4|F^s \cap (S_1 \cup S_2)| + 2|(S_1 \cup S_2) \backslash F^s| \leq 4|F^s \cup S_1 \cup S_2| \leq 4(|F^s| + |S_1| + |S_2|).$$

4. The maximum edge capacity is bounded by $\left\|\boldsymbol{u}^f\right\|_{\max} = \left\|\boldsymbol{u}^s\right\|_{\max}$. First, capacity of all non-selective edges in $E^s \backslash (S_1 \cup S_2)$ is unchanged in G^f since no reduction is made. Then, for all selective edges $e \in (S_1 \cup S_2)$ with capacity (or fixed flow) $\boldsymbol{u}^s(e)$, capacity of the 5 edges $(e_1, \ldots, e_5)$ in the gadget are with capacity at most $\boldsymbol{u}^s(e)$.

To estimate the reduction time, it is observed that it takes constant time to reduce each selective edge in $S_1 \cup S_2$ since only a constant number of vertices and edges are added. And it takes constant time to copy each of the other non-selective edges in $E^s \backslash (S_1 \cup S_2)$. Thus, the reduction of this step takes $O(|E^s|)$ time. □

4.7.2 SFFA to 2CFFA

Definition 4.20 (2CFF Approximate Problem (2CFFA))**.** A 2CFFA instance is given by a 2CFF instance $(G, F, \boldsymbol{u}, s_1, t_1, s_2, t_2)$ as in Definition 2.7, and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(G, F, \boldsymbol{u}, s_1, t_1, s_2, t_2, \epsilon)$. We say an algorithm solves the 2CFFA problem, if, given any 2CFFA instance, it returns a pair of flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \mathbf{0}$ that satisfies

$$\boldsymbol{u}(e) - \epsilon \leq \boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \quad \forall e \in F \tag{42}$$

$$0 \leq \boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \quad \forall e \in E \backslash F \tag{43}$$

$$\left| \sum_{u:(u,v) \in E} \boldsymbol{f}_i(u, v) - \sum_{w:(v,w) \in E} \boldsymbol{f}_i(v, w) \right| \leq \epsilon, \quad \forall v \in V \backslash \{s_i, t_i\}, \ i \in \{1, 2\} \tag{44}$$

or it correctly declares that the associated 2CFF instance is infeasible. We refer to the error in (42) and (43) as error in congestion, error in (44) as error in demand.

We can use the same reduction method and solution mapping method in the exact case to the approximate case. Note that, different from the exact case where $\boldsymbol{f}^s_{\bar{i}}(e) = \boldsymbol{f}^f_{\bar{i}}(e_1) = 0$ if $e \in S_i$, it can be nonzero in the approximate case, which gives rise to error in type in G^s.

Lemma 4.21 (SFFA to 2CFFA)**.** *Given an* SFFA *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2, \epsilon^s)$*, if we use Lemma 4.19 to construct a* 2CFF *instance* $\left(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2\right)$ *from the* SFF *instance* $(G^s, F^s, S_1, S_2, \boldsymbol{u}^s, s_1, t_1, s_2, t_2)$*, and let*

$$\epsilon^f = \frac{\epsilon^s}{6|E^s|},$$

then the 2CFFA *instance* $\left(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2, \epsilon^f\right)$ *satisfies.*

Furthermore, if $\boldsymbol{f}^f$ *is a solution to the* 2CFF(A) *instance, then in time* $O(|E^s|)$*, we can compute a solution* $\boldsymbol{f}^s$ *to the* SFF(A) *instance, where the exact case holds when* $\epsilon^f = \epsilon^s = 0$*.*

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each selective or non-selective entry of $\boldsymbol{f}^s$. As $\boldsymbol{f}^s$ has $|E^s|$ entries, such a solution mapping takes $O(|E^s|)$ time.

Now, we conduct an error analysis. By the error notions of SFFA (Definition 4.17), there are three types of error after solution mapping: (1) error in congestion; (2) error in demand; (3) error in type.

1. Error in congestion.
   Since we map solution back trivially for non-selective edges in $E^s \backslash (S_1 \cup S_2)$, error in congestion

of G^f (i.e., $\epsilon_l^f, \epsilon_u^f$) still holds for G^s. Therefore, we focus on how error in congestion changes when mapping solution back to selective edges in $S_1 \cup S_2$.

We first track the error of the upper bound of capacity. For any selective edge e, the maximum flow to be routed is

$$\boldsymbol{u}^s(e) + \tau_u^s = \sum_{i=\{1,2\}} \boldsymbol{f}_i^s(e) = \sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e_1) \leq \boldsymbol{u}^f(e) + \epsilon^f = \boldsymbol{u}^s(e) + \epsilon^f,$$

thus, $\tau_u^s \leq \epsilon^f$.

Next, we track the error of the lower bound of capacity. We only need to take selective fixed flow edges into consideration since there is no error of the lower bound of capacity for non-fixed flow edges. If e is a fixed flow edge, then e_1 is also a fixed flow edge with the same capacity as e. Hence, we can lower bound the flow routed through e by error in congestion of e_1 in G^f.

$$\boldsymbol{u}^s(e) - \tau_l^s = \boldsymbol{f}^s(e) = \boldsymbol{f}^f(e_1) \geq \boldsymbol{u}^s(e) - \epsilon^f,$$

and thus $\tau_l^s \leq \epsilon^f$.

2. Error in demand.
For simplicity, we denote $S = S_1 \bigcup S_2$. We consider commodity $i, i \in \{1,2\}$, then we analyze error in demand for vertices other than s_i, t_i.

- **Case 1:** For $s_{\bar{i}}, t_{\bar{i}}$
Since incident edges for $s_{\bar{i}}, t_{\bar{i}}$ decreases in G^s, error in demand on $s_{\bar{i}}, t_{\bar{i}}$ will not increase. Thus, $\tilde{\tau}_{di}^s \leq \epsilon^f$.
- **Case 2:** For vertices other than $s_{\bar{i}}, t_{\bar{i}}$
By Eq. (35) in Definition 4.17, error in demand of commodity i that we can achieve for vertices other than $\{s_1, t_1, s_2, t_2\}$ in G^s is computed as

$$\begin{aligned}
\bar{\tau}_{di}^s &= \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left| \sum_{(x,y)\in E^s} \boldsymbol{f}_i^s(x,y) - \sum_{(y,z)\in E^s} \boldsymbol{f}_i^s(y,z) \right| \\
&= \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left| \left(\sum_{(x,y)\in S} \boldsymbol{f}_i^s(x,y) + \sum_{(x,y)\in E^s\setminus S} \boldsymbol{f}_i^s(x,y) \right) - \left(\sum_{(y,z)\in S} \boldsymbol{f}_i^s(y,z) + \sum_{(y,z)\in E^s\setminus S} \boldsymbol{f}_i^s(y,z) \right) \right| \\
&= \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left| \left(\sum_{e=(x,y)\in S} \boldsymbol{f}_i^f(e_1) + \sum_{(x,y)\in E^s\setminus S} \boldsymbol{f}_i^f(x,y) \right) - \left(\sum_{e=(y,z)\in S} \boldsymbol{f}_i^f(e_1) + \sum_{(y,z)\in E^s\setminus S} \boldsymbol{f}_i^f(y,z) \right) \right| \\
&\leq \epsilon^f + \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left| \left(\sum_{e=(x,y)\in S} \boldsymbol{f}_i^f(e_1) + \sum_{(x,y)\in E^s\setminus S} \boldsymbol{f}_i^f(x,y) \right) - \left(\sum_{e=(x,y)\in S} \boldsymbol{f}_i^f(e_3) + \sum_{(x,y)\in E^s\setminus S} \boldsymbol{f}_i^f(x,y) \right) \right| \\
&= \epsilon^f + \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left| \sum_{e=(x,y)\in S} \boldsymbol{f}_i^f(e_1) - \sum_{e=(x,y)\in S} \boldsymbol{f}_i^f(e_3) \right| \\
&\leq \epsilon^f + \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \sum_{e=(x,y)\in S} \left| \boldsymbol{f}_i^f(e_1) - \boldsymbol{f}_i^f(e_3) \right| \\
&= \epsilon^f + \max_{y \in V^s \setminus \{s_1,t_1,s_2,t_2\}} \left\{ \sum_{e=(x,y)\in S_1} \left| \boldsymbol{f}_i^f(e_1) - \boldsymbol{f}_i^f(e_3) \right| + \sum_{\hat{e}=(x,y)\in S_2} \left| \boldsymbol{f}_i^f(\hat{e}_1) - \boldsymbol{f}_i^f(\hat{e}_3) \right| \right\}.
\end{aligned} \tag{45}$$

Now, we try to bound $\left|\boldsymbol{f}_i^f(e_1) - \boldsymbol{f}_i^f(e_3)\right|$. By error in demand on xy and xy', we have

$$\left|\boldsymbol{f}_i^f(e_1) + \boldsymbol{f}_i^f(e_2) - \boldsymbol{f}_i^f(e_4)\right| \leq \epsilon^f,$$

$$\left|\boldsymbol{f}_i^f(e_3) + \boldsymbol{f}_i^f(e_2) - \boldsymbol{f}_i^f(e_5)\right| \leq \epsilon^f.$$

Combining the above two inequalities gives

$$\left|\boldsymbol{f}_i^f(e_1) - \boldsymbol{f}_i^f(e_3)\right| \leq 2\epsilon^f + \left|\boldsymbol{f}_i^f(e_4) - \boldsymbol{f}_i^f(e_5)\right|. \tag{46}$$

Now, we assume $e \in S_1$. We can get the same bound for $\hat{e} \in S_2$ by symmetry. Eq. (46) can be further bounded by

$$\begin{aligned}
\left|\boldsymbol{f}_1^f(e_1) - \boldsymbol{f}_1^f(e_3)\right| &\leq 2\epsilon^f + \left|\boldsymbol{f}_1^f(e_4) - \boldsymbol{f}_1^f(e_5)\right| \\
&= 2\epsilon^f + \left|(\boldsymbol{f}^f(e_4) - \boldsymbol{f}_2^f(e_4)) - (\boldsymbol{f}^f(e_5) - \boldsymbol{f}_2^f(e_5))\right| \\
&\leq 2\epsilon^f + \left|\boldsymbol{f}^f(e_4) - \boldsymbol{f}^f(e_5)\right| + \left|\boldsymbol{f}_2^f(e_4) - \boldsymbol{f}_2^f(e_5)\right| \\
&\overset{(1)}{\leq} 4\epsilon^f + \left|\boldsymbol{f}^f(e_4) - \boldsymbol{f}^f(e_5)\right| \\
&\overset{(2)}{\leq} 6\epsilon^f.
\end{aligned} \tag{47}$$

and

$$\left|\boldsymbol{f}_2^f(e_1) - \boldsymbol{f}_2^f(e_3)\right| \leq 2\epsilon^f + \left|\boldsymbol{f}_2^f(e_4) - \boldsymbol{f}_2^f(e_5)\right| \overset{(3)}{\leq} 4\epsilon^f. \tag{48}$$

For step (1), we use $\boldsymbol{f}_2^f(e_4), \boldsymbol{f}_2^f(e_5) \leq \epsilon^f$ by error in demand on t_1, s_1 in G^f. For step (2) and (3), we use the error in congestion for e_4 and e_5 in G^f.
By symmetry, for $\hat{e} \in S_2$, we have

$$\left|\boldsymbol{f}_1^f(\hat{e}_1) - \boldsymbol{f}_1^f(\hat{e}_3)\right| \leq 4\epsilon^f, \tag{49}$$

$$\left|\boldsymbol{f}_2^f(\hat{e}_1) - \boldsymbol{f}_2^f(\hat{e}_3)\right| \leq 6\epsilon^f. \tag{50}$$

Finally, applying Eq. (47) and Eq. (49) to Eq. (45), we obtain

$$\begin{aligned}
\bar{\tau}_{d1}^s &\leq \epsilon^f + \max_{y \in V^s \setminus \{s_1, t_1\}} \left\{ \sum_{e=(x,y) \in S_1} \left|\boldsymbol{f}_1^f(e_1) - \boldsymbol{f}_1^f(e_3)\right| + \sum_{\hat{e}=(x,y) \in S_2} \left|\boldsymbol{f}_1^f(\hat{e}_1) - \boldsymbol{f}_1^f(\hat{e}_3)\right| \right\} \\
&\leq \epsilon^f + \max_{y \in V^s \setminus \{s_1, t_1\}} \left\{ \sum_{e=(x,y) \in S_1} 6\epsilon^f + \sum_{\hat{e}=(x,y) \in S_2} 4\epsilon^f \right\} \\
&\leq \epsilon^f + (6|S_1| + 4|S_2|)\epsilon^f \leq 6(|S_1| + |S_2|)\epsilon^f.
\end{aligned} \tag{51}$$

Similarly, applying Eq. (48) and Eq. (50) to Eq. (45), we obtain

$$\bar{\tau}_{d2}^s \leq \epsilon^f + (4|S_1| + 6|S_2|)\epsilon^f \leq 6(|S_1| + |S_2|)\epsilon^f. \tag{52}$$

Combining the above cases, we can bound the error in demand uniformly by

$$\tau_{di}^s := \max\{\tilde{\tau}_{di}^s, \bar{\tau}_{di}^s\} \leq 6|E^s|\epsilon^f, \qquad i \in \{1,2\}.$$

3. Error in type.
   By Eq. (36) in Definition 4.17, error in type for edges selecting commodity i that we can achieve is computed as

$$\begin{aligned}\tau_{ti}^s = \max_{e \in S_i} \boldsymbol{f}_{\bar{i}}^s(e) = \max_{e \in S_i} \boldsymbol{f}_{\bar{i}}^f(e_1) \\ \leq \max_{e \in S_i} (\boldsymbol{f}_{\bar{i}}^f(e_1) + \boldsymbol{f}_{\bar{i}}^f(e_2)) \quad \text{Becasue } \boldsymbol{f}_{\bar{i}}^f(e_2) \geq 0\end{aligned}$$

   Again, we first consider $e \in S_1$. For an arbitrary edge $e = (x, y) \in S_1$, it is known that e_4 and e_5 are incident to t_1 and s_1. By error in demand on vertex xy, we have

$$\left| \boldsymbol{f}_2^f(e_1) + \boldsymbol{f}_2^f(e_2) - \boldsymbol{f}_2^f(e_4) \right| \leq \epsilon^f.$$

   As $\boldsymbol{f}_2^f(e_4) \leq \epsilon^f$ because of error in demand on t_1 in G^f, then we can bound

$$\boldsymbol{f}_2^f(e_1) + \boldsymbol{f}_2^f(e_2) \leq 2\epsilon^f.$$

   Since e is an arbitrary edge in S_1, thus $\tau_{t1}^s \leq 2\epsilon^f$. By symmetry, we also have $\tau_{t2}^s \leq 2\epsilon^f$.

To summarize, as we set in the reduction that $\epsilon^f = \frac{\epsilon^s}{6|E^s|}$, then we have

$$\tau_u^s, \tau_l^s, \tau_{d1}^s, \tau_{d2}^s, \tau_{t1}^s, \tau_{t2}^s \leq \epsilon^s,$$

indicating that $\boldsymbol{f}^s$ is a solution to the SFFA instance.

□

4.8 2CFF(A) to 2CFR(A)

4.8.1 2CFF to 2CFR

We show the reduction from a 2CFF instance $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$ to a 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$. First of all, we add two new sources $\bar{s}_1, \bar{s}_2$ and two new sinks $\bar{t}_1, \bar{t}_2$. Then, for each edge $e \in E^f$, we map it to a gadget consisting edges $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ in G^r, as shown in the upper part of Figure 6. Additionally, there is another gadget with 5 edges in G^r that connects the original sink t_i and source s_i, as shown in the lower part of Figure 6. Capacity of these edges is the sum capacities of all edges in G^f, i.e., $M^f = \sum_{e \in E^f} \boldsymbol{u}^f(e)$. Additionally, we set $R_1 = R_2 = 2M^f$, indicating that at least $2M^f$ unit of flow should be routed from $\bar{s}_i$ to $\bar{t}_i$, $i \in \{1, 2\}$.

The key idea to remove the fixed flow constraint is utilizing edge directions and the requirements that $2M^f$ units of the flow of commodity i to be routed from the new source $\bar{s}_i$ to the new sink $\bar{t}_i$, $i \in \{1, 2\}$. It is noticed that all edges that are incident to the new sources and sinks should be saturated to fulfill the requirements. Therefore, for a fixed flow edge e in the first gadget, the incoming flow of vertex xy' and the outgoing flow of vertex xy must be $2u$. Since the capacity of e_2 is $2u - u = u$, then the flow of e_1, e_2, e_3 are forced to be u. As such, the fixed flow constraint can be simulated. Note that instead of simply copying non-fixed flow edges to G^r, we also need to map non-fixed flow edges to the designed gadget in this step. This guarantees that the requirement on flow values can be satisfied if the 2CFF instance is feasible.

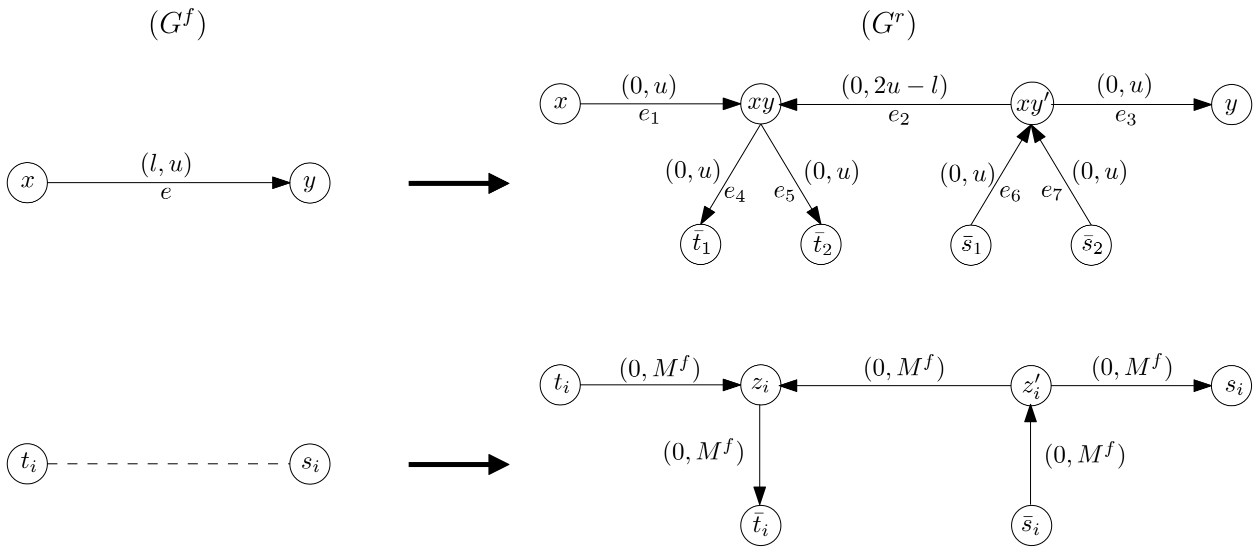

Figure 6: The reduction from 2CFF to 2CFR. $l = u$ if e is a fixed flow edge, and $l = 0$ if e is a non-fixed flow edge.

If a 2CFR solver returns $\boldsymbol{f}^r$ for the 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$, then we return $\boldsymbol{f}^f$ for the 2CFF instance $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$ by setting $\boldsymbol{f}_i^f(e) = \boldsymbol{f}_i^r(e_1), \forall e \in E^f, i \in \{1,2\}$. If the 2CFR solver returns "infeasible" for the 2CFR instance, then we return "infeasible" for the 2CFF instance.

Lemma 4.22 (2CFF to 2CFR)**.** *Given a* 2CFF *instance* $(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2)$*, we can construct, in time* $O(|E^f|)$*, a* 2CFR *instance* $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$ *such that*

$$|V^r| = |V^f| + 2|E^f| + 8, \quad |E^r| = 7|E^f| + 10, \quad \|\boldsymbol{u}^r\|_{\max} = \max\left\{2\left\|\boldsymbol{u}^f\right\|_{\max}, M^f\right\}, \quad R_1 = R_2 = 2M^f,$$

where $M^f = \sum_{e \in E^f} \boldsymbol{u}^f(e)$*. If the* 2CFF *instance has a solution, then the* 2CFR *instance has a solution.*

Proof. According to the reduction described above, from any solution $\boldsymbol{f}^f$ to the 2CFF instance, it is easy to derive a solution $\boldsymbol{f}^r$ to the 2CFR instance. Concretely, we define a feasible flow $\boldsymbol{f}^r$ as follows. For any edge $e \in E^f$, we set:

$$\boldsymbol{f}_i^r(e_1) = \boldsymbol{f}_i^r(e_3) = \boldsymbol{f}_i^f(e), \qquad i \in \{1,2\},$$
$$\boldsymbol{f}_i^r(e_2) = u - \boldsymbol{f}_i^f(e), \qquad i \in \{1,2\},$$
$$\boldsymbol{f}_1^r(e_4) = \boldsymbol{f}_1^r(e_6) = u, \qquad \boldsymbol{f}_2^r(e_5) = \boldsymbol{f}_2^r(e_7) = u,$$
$$\boldsymbol{f}_2^r(e_4) = \boldsymbol{f}_2^r(e_6) = \boldsymbol{f}_1^r(e_5) = \boldsymbol{f}_1^r(e_7) = 0,$$

where $u = \boldsymbol{u}^f(e)$. And we set

$$\boldsymbol{f}_i^r(t_i, z_i) = \boldsymbol{f}_i^r(z_i', s_i) = \sum_{w:(s_i,w) \in E^f} \boldsymbol{f}_i^f(s_i, w) \leq M^f, \qquad i \in \{1,2\},$$

$$\boldsymbol{f}_i^r(z_i', z_i) = M^f - \sum_{w:(s_i,w) \in E^f} \boldsymbol{f}_i^f(s_i, w), \qquad i \in \{1,2\},$$

$$\boldsymbol{f}_i^r(\bar{s}_i, z_i') = \boldsymbol{f}_i^r(z_i, \bar{t}_i) = M^f, \qquad i \in \{1,2\},$$

$$\boldsymbol{f}_{\bar{i}}^r(t_i, z_i) = \boldsymbol{f}_{\bar{i}}^r(z_i', s_i) = \boldsymbol{f}_{\bar{i}}^r(z_i', z_i) = \boldsymbol{f}_{\bar{i}}^r(\bar{s}_i, z_i') = \boldsymbol{f}_{\bar{i}}^r(z_i, \bar{t}_i) = 0, \qquad \bar{i} \in \{1,2\} \backslash i.$$

Now, we prove that $\boldsymbol{f}^r$ is a solution to the 2CFR instance.

1. Capacity constraint.
   Only the capacity constraint on edge e_2 is nontrivial.

   - If e is a fixed flow edge, then we have $l = u$, thus the capacity of e_2 is u. By construction, we have
     $$\sum_{i=\{1,2\}} \boldsymbol{f}_i^r(e_2) = 2u - \sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e) = 2u - u = u,$$
     where we use $\sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e) = u$.
   - If e is a non-fixed flow edge, then we have $l = 0$, thus the capacity of e_2 is $2u$. By construction, we have
     $$\sum_{i=\{1,2\}} \boldsymbol{f}_i^r(e_2) = 2u - \sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e) \leq 2u,$$
     where we use $\sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e) \geq 0$.

   Therefore, we conclude that $\boldsymbol{f}^r$ fulfills the capacity constraint.

2. Flow conservation constraint.
   Only the flow conservation constraint on vertices xy, xy', z_i, z_i' are nontrivial.

   - For vertex xy, we have
     $$\boldsymbol{f}_1^r(e_1) + \boldsymbol{f}_1^r(e_2) = \boldsymbol{f}_1^f(e) + (u - \boldsymbol{f}_1^f(e)) = u = \boldsymbol{f}_1^r(e_4),$$
     $$\boldsymbol{f}_2^r(e_1) + \boldsymbol{f}_2^r(e_2) = \boldsymbol{f}_2^f(e) + (u - \boldsymbol{f}_2^f(e)) = u = \boldsymbol{f}_2^r(e_5).$$
     We can prove for vertex xy' similarly.
   - For vertex $z_i, i \in \{1,2\}$, we have
     $$\boldsymbol{f}_i^r(t_i, z_i) + \boldsymbol{f}_i^r(z_i', z_i) = \sum_{w:(s_i,w)\in E^f} \boldsymbol{f}_i^f(s_i, w) + \left(M^f - \sum_{w:(s_i,w)\in E^f} \boldsymbol{f}_i^f(s_i, w)\right) = M^f = \boldsymbol{f}_i^r(z_i, \bar{t}_i).$$
     We can prove for vertex z_i' similarly.

   Therefore, we conclude that $\boldsymbol{f}^r$ fulfills the flow conservation constraint.

3. Requirement $R_1 = R_2 = 2M^f$.
   For commodity 1, we have
   $$R_1 = \sum_{w:(\bar{s}_1,w)\in E^r} \boldsymbol{f}_1^r(\bar{s}_1, w) = \boldsymbol{f}^r(\bar{s}_1, z_1') + \sum_{e\in E^f} \boldsymbol{f}_1^r(e_6) = M^f + \sum_{e\in E^f} \boldsymbol{u}^f(e) = 2M^f,$$
   $$R_1 = \sum_{u:(u,\bar{t}_1)\in E^r} \boldsymbol{f}_1^r(u, \bar{t}_1) = \boldsymbol{f}^r(z_1, \bar{t}_1) + \sum_{e\in E^f} \boldsymbol{f}_1^r(e_4) = M^f + \sum_{e\in E^f} \boldsymbol{u}^f(e) = 2M^f.$$
   We can prove that the requirement $R_2 = 2M^f$ is satisfied similarly.

To conclude, $\boldsymbol{f}^r$ is a feasible flow to the 2CFR instance.

Now, we track the change of problem size after reduction. Based on the reduction method, given a 2CFF instance with $|V^f|$ vertices and $|E^f|$ edges (including $|F^f|$ fixed flow edges), we can compute the size of the reduced 2CFR instance as follows.

1. $|V^r|$ vertices. First, all vertices in V^f are maintained. Then, for each edge $e = (x, y)$, two new auxiliary vertices xy, xy' are added. And finally, in the second gadget, $\{\bar{s}_i, \bar{t}_i, z_i, z_i'\}, i \in \{1, 2\}$ are added. Hence, we have $|V^r| = |V^f| + 2|E^f| + 8$.

2. $|E^r|$ edges. First, each edge $e \in E^f$ is replaced by the first gadget in Figure 6 with 7 edges $(e_1, \ldots, e_7)$. Then, 5 edges are added between t_i and s_i according to the second gadget in Figure 6, $i \in \{1, 2\}$. Thus, we have $|E^r| = 7|E^f| + 10$.

3. The maximum edge capacity is bounded by $\|\boldsymbol{u}^r\|_{\max} = \max\left\{2\left\|\boldsymbol{u}^f\right\|_{\max}, M^f\right\}$. For all fixed flow edges $e \in F^f$ with fixed flow $\boldsymbol{u}^f(e)$, the 7 edges $(e_1, \ldots, e_7)$ in the first gadget are all with capacity $\boldsymbol{u}^f(e)$. For all non-fixed flow edges in $e \in E^f \backslash F^f$ with capacity $\boldsymbol{u}^f(e)$, the capacity of e_2 becomes $2\boldsymbol{u}^f(e)$ and the capacity of the rest 6 edges in the first gadget remains $\boldsymbol{u}^f(e)$. Moreover, the 5 edges in the second gadget are all with capacity M^f, which is the sum of the capacity of all edges.

To estimate the reduction time, it is observed that it takes constant time to reduce each edge in E^f since only a constant number of vertices and edges are added. Thus, the reduction of this step takes $O(|E^f|)$ time. $\square$

4.8.2 2CFFA to 2CFRA

Definition 4.23 (2CFR Approximate Problem (2CFRA))**.** A 2CFRA instance is given by a 2CFR instance $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R_1, R_2)$ as in Definition 2.5, and an error parameter $\epsilon \in [0, 1]$, which we collect in a tuple $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R_1, R_2, \epsilon)$. We say an algorithm solves the 2CFRA problem, if, given any 2CFRA instance, it returns a pair of flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq \mathbf{0}$ that satisfies

$$\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \ \forall e \in E \tag{53}$$

$$\left| \sum_{u:(u,v)\in E} \boldsymbol{f}_i(u, v) - \sum_{w:(v,w)\in E} \boldsymbol{f}_i(v, w) \right| \leq \epsilon, \ \forall v \in V \backslash \{s_i, t_i\}, i \in \{1, 2\} \tag{54}$$

$$\left| \sum_{w:(s_i,w)\in E} \boldsymbol{f}_i(s_i, w) - R_i \right| \leq \epsilon, \qquad \left| \sum_{u:(u,t_i)\in E} \boldsymbol{f}_i(u, t_i) - R_i \right| \leq \epsilon, \ i \in \{1, 2\} \tag{55}$$

or it correctly declares that the associated 2CFR instance is infeasible. We refer to the error in (53) as error in congestion, error in (54), (55) as error in demand.

We can use the same reduction method and solution mapping method in the exact case to the approximate case.

Lemma 4.24 (2CFFA to 2CFRA)**.** *Given a* 2CFFA *instance* $\left(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2, \epsilon^f\right)$*, if we use Lemma 4.22 to construct a* 2CFR *instance* $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$ *from the* SFF *instance* $\left(G^f, F^f, \boldsymbol{u}^f, s_1, t_1, s_2, t_2\right)$*, and let*

$$\epsilon^r = \frac{\epsilon^f}{12|E^f|},$$

then the 2CFRA *instance* $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2, \epsilon^r)$ *satisfies.*

Furthermore, if $\boldsymbol{f}^r$ *is a solution to the* 2CFR(A) *instance, then in time* $O(|E^f|)$*, we can compute a solution* $\boldsymbol{f}^f$ *to the* 2CFF(A) *instance, where the exact case holds when* $\epsilon^r = \epsilon^f = 0$*.*

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each entry of $\boldsymbol{f}^f$, and $\boldsymbol{f}^f$ has $|E^f|$ entries, such a solution mapping takes $O(|E^f|)$ time.

Now, we conduct an error analysis. By the error notions of 2CFFA (Definition 4.20), there are two types of error after solution mapping: (1) error in congestion; (2) error in demand.

1. Error in congestion.
   We first track the error of the upper bound of capacity. For any edge $e \in E^f$, the maximum flow to be routed in e is
   $$\boldsymbol{u}^f(e) + \tau_u^f = \sum_{i=\{1,2\}} \boldsymbol{f}_i^f(e) = \sum_{i=\{1,2\}} \boldsymbol{f}_i^r(e_1) \le \boldsymbol{u}^r(e_1) + \epsilon^r = \boldsymbol{u}^f(e) + \epsilon^r,$$
   thus, $\tau_u^f \le \epsilon^r$.

   Next, we track the error of the lower bound of capacity for fixed flow edges. By error in demand on vertex $\bar{t}_1, \bar{t}_2$, as shown in Eq. (55), we have
   $$\left| \sum_{e\in E^f} \boldsymbol{f}_1^r(e_4) + \boldsymbol{f}^r(z_1, \bar{t}_1) - 2M^f \right| \le \epsilon^r,$$
   $$\left| \sum_{e\in E^f} \boldsymbol{f}_2^r(e_5) + \boldsymbol{f}^r(z_2, \bar{t}_2) - 2M^f \right| \le \epsilon^r.$$
   We now try to compute the minimum flow that can be routed through any fixed flow edge in F^f. For an arbitrary $\hat{e} \in F^f$, we can rearrange the above equations and have
   $$\begin{aligned}\boldsymbol{f}_1^r(\hat{e}_4) &\ge 2M^f - \epsilon^r - \sum_{e\in E^f\setminus\hat{e}} \boldsymbol{f}_1^r(e_4) - \boldsymbol{f}^r(z_1, \bar{t}_1)\\ &\ge \max\left\{0, 2M^f - \epsilon^r - \left(M^f - \boldsymbol{u}^f(\hat{e}) + (|E^f|-1)\epsilon^r\right) - (M^f + \epsilon^r)\right\}\\ &= \max\left\{0, \boldsymbol{u}^f(\hat{e}) - (|E^f|+1)\epsilon^r\right\}\end{aligned}$$
   Similarly, we have $\boldsymbol{f}_2^r(\hat{e}_5) \ge \max\left\{0, \boldsymbol{u}^f(\hat{e}) - (|E^f|+1)\epsilon^r\right\}$. Moreover, by error in demand on vertex xy, we have
   $$|\boldsymbol{f}_i^r(\hat{e}_1) + \boldsymbol{f}_i^r(\hat{e}_2) - \boldsymbol{f}_i^r(\hat{e}_4) - \boldsymbol{f}_i^r(\hat{e}_5)| \le \epsilon^r,\ i \in \{1,2\}.$$
   Therefore, for any fixed flow edge $\hat{e} = (x, y) \in F^f$, the minimum flow to be routed can be lower bounded by
   $$\begin{aligned}\boldsymbol{u}^f(\hat{e}) - \tau_l^f = \boldsymbol{f}^f(\hat{e}) &= \boldsymbol{f}^r(\hat{e}_1) = \boldsymbol{f}_1^r(\hat{e}_1) + \boldsymbol{f}_2^r(\hat{e}_1)\\ &\ge \boldsymbol{f}_1^r(\hat{e}_4) + \boldsymbol{f}_1^r(\hat{e}_5) - \boldsymbol{f}_1^r(\hat{e}_2) - \epsilon^r + \boldsymbol{f}_2^r(\hat{e}_4) + \boldsymbol{f}_2^r(\hat{e}_5) - \boldsymbol{f}_2^r(\hat{e}_2) - \epsilon^r\\ &\ge \boldsymbol{f}_1^r(\hat{e}_4) + \boldsymbol{f}_2^r(\hat{e}_5) - \boldsymbol{f}^r(\hat{e}_2) - 2\epsilon^r\\ &\overset{(1)}{\ge} \max\{0, 2(\boldsymbol{u}^f(\hat{e}) - (|E^f|+1)\epsilon^r) - (\boldsymbol{u}^f(\hat{e}) + \epsilon^r) - 2\epsilon^r\}\\ &= \max\{0, \boldsymbol{u}^f(\hat{e}) - (2|E^f|+5)\epsilon^r\}.\end{aligned} \tag{56}$$

For step (1), we plug in the lower bound of $\boldsymbol{f}_1^r(\hat{e}_4), \boldsymbol{f}_2^r(\hat{e}_5)$ that are proved previously, and the upper bound of $\boldsymbol{f}^r(\hat{e}_2)$. Thus, we have

$$\tau_l^f \leq (2|E^f| + 5)\epsilon^r \leq 7|E^f|\epsilon^r.$$

2. Error in demand.
   We focus on the error in demand for commodity 1 first, and then the error in demand for commodity 2 can be achieved directly by symmetry. Then we analyze error in demand for vertices other than s_1, t_1.

   - **Case 1:** For vertex s_2
     We have $\left|\boldsymbol{f}_1^r(z_2', s_2) - \sum_{e=(s_2,y)\in E^f} \boldsymbol{f}_1^r(e_1)\right| \leq \epsilon^r$ and $\boldsymbol{f}_1^r(z_2', s_2) \leq \boldsymbol{f}_1^r(\bar{s}_2, z_2') \leq \epsilon^r$, thus,

     $$\tilde{\tau}_{d1}^f = \sum_{e=(s_2,y)\in E^f} \boldsymbol{f}_1^f(e) = \sum_{e=(s_2,y)\in E^f} \boldsymbol{f}_1^r(e_1) \leq 2\epsilon^r.$$

   - **Case 2:** For vertex t_2
     We have $\left|\boldsymbol{f}_1^r(t_2, z_2) - \sum_{e=(x,t_2)\in E^f} \boldsymbol{f}_1^r(e_3)\right| \leq \epsilon^r$ and $\boldsymbol{f}_1^r(t_2, z_2) \leq \boldsymbol{f}_1^r(z_2, \bar{t}_2) \leq \epsilon^r$, thus,

     $$\sum_{e=(x,t_2)\in E^f} \boldsymbol{f}_1^r(e_3) \leq 2\epsilon^r,$$

     and

     $$\begin{aligned}\bar{\tau}_{d1}^f = &\sum_{e=(x,t_2)\in E^f} \boldsymbol{f}_1^f(e) = \sum_{e=(x,t_2)\in E^f} \boldsymbol{f}_1^r(e_1) \leq \\ &\sum_{e=(x,t_2)\in E^f} \boldsymbol{f}_1^r(e_3) + \sum_{e=(x,t_2)\in E^f} |\boldsymbol{f}_1^r(e_1) - \boldsymbol{f}_1^r(e_3)| \leq 2\epsilon^r + 10|E^f|\epsilon^r \leq 12|E^f|\epsilon^r,\end{aligned}$$

     where the last inequality comes from Eq. (57), which will be proved soon.

   - **Case 3:** For vertices other than s_2, t_2
     By Eq. (44) in Definition 4.20, error in demand of commodity 1 that we can achieve for vertices other than s_1, t_1 in G^f is computed as

     $$\begin{aligned}\hat{\tau}_{d1}^f = &\max_{y\in V^f\setminus\{s_1,t_1,s_2,t_2\}} \left|\sum_{(x,y)\in E^f} \boldsymbol{f}_1^f(x,y) - \sum_{(y,z)\in E^f} \boldsymbol{f}_1^f(y,z)\right| \\ = &\max_{y\in V^f\setminus\{s_1,t_1,s_2,t_2\}} \left|\sum_{e=(x,y)\in E^f} \boldsymbol{f}_1^r(e_1) - \sum_{e=(y,z)\in E^f} \boldsymbol{f}_1^r(e_1)\right| \\ \leq &\epsilon^r + \max_{y\in V^f\setminus\{s_1,t_1,s_2,t_2\}} \left|\sum_{e=(x,y)\in E^f} \boldsymbol{f}_1^r(e_1) - \sum_{e=(x,y)\in E^f} \boldsymbol{f}_1^r(e_3)\right| \\ \leq &\epsilon^r + \max_{y\in V^f\setminus\{s_1,t_1,s_2,t_2\}} \sum_{e=(x,y)\in E^f} |\boldsymbol{f}_1^r(e_1) - \boldsymbol{f}_1^r(e_3)|.\end{aligned} \tag{57}$$

     Now, we try to bound $|\boldsymbol{f}_1^r(e_1) - \boldsymbol{f}_1^r(e_3)|$. By error in demand on vertex xy and xy' with respect to commodity 1, we have

     $$|\boldsymbol{f}_1^r(e_1) + \boldsymbol{f}_1^r(e_2) - \boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_5)| \leq \epsilon^r,$$

$$|\boldsymbol{f}_1^r(e_2) + \boldsymbol{f}_1^r(e_3) - \boldsymbol{f}_1^r(e_6) - \boldsymbol{f}_1^r(e_7)| \leq \epsilon^r.$$

We also have $\boldsymbol{f}_1^r(e_5), \boldsymbol{f}_1^r(e_7) \leq \epsilon^r$. Combining these inequalities gives

$$|\boldsymbol{f}_1^r(e_1) - \boldsymbol{f}_1^r(e_3)| \leq |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| + 4\epsilon^r,$$

which can be plugged back into Eq. (57) to give

$$\hat{\tau}_{d1}^f \leq 5|E^f|\epsilon^r + \max_{y \in V^f \setminus \{s_1,t_1,s_2,t_2\}} \sum_{e=(x,y) \in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)|.$$

The following is a claim that shows the sum of $|\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)|$ for all edges $e \in E^f$ can be upper bounded.

Claim 4.25.

$$\sum_{e \in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| \leq 6|E^f|\epsilon^r.$$

By Claim 4.25, we have

$$\begin{aligned} \hat{\tau}_{d1}^f &\leq 5|E^f|\epsilon^r + \max_{y \in V^f \setminus \{s_1,t_1,s_2,t_2\}} \sum_{e=(x,y) \in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| \\ &\leq 5|E^f|\epsilon^r + \sum_{e \in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| \leq 11|E^f|\epsilon^r. \end{aligned}$$

Combining all three cases, we can bound the error in demand for commodity 1 uniformly by

$$\tau_{d1}^f := \max\{\tilde{\tau}_{d1}^f, \bar{\tau}_{d1}^f, \hat{\tau}_{d1}^f\} \leq 12|E^f|\epsilon^r.$$

And by symmetry of the two commodities, we can also bound the error in demand for commodity 2 by

$$\tau_{d2}^f \leq 12|E^f|\epsilon^r.$$

To summarize, as we set in the reduction that $\epsilon^r = \frac{\epsilon^f}{12|E^f|}$, then we have

$$\tau_u^f, \tau_l^f, \tau_{d1}^f, \tau_{d2}^f \leq \epsilon^f,$$

indicating that $\boldsymbol{f}^f$ is a solution to the 2CFFA instance. □

Now, we provide a proof to Claim 4.25.

Proof of Claim 4.25. We divide all edges in E^f into two groups:

$$E_I = \{e \in E^f \ s.t. \ \boldsymbol{f}_1^r(e_4) \leq \boldsymbol{f}_1^r(e_6)\},$$

$$E_{II} = \{e \in E^f \ s.t. \ \boldsymbol{f}_1^r(e_4) > \boldsymbol{f}_1^r(e_6)\}.$$

We denote

$$T_I = \sum_{e \in E_I} \boldsymbol{f}_1^r(e_4), \qquad S_I = \sum_{e \in E_I} \boldsymbol{f}_1^r(e_6);$$

$$T_{II} = \sum_{e \in E_{II}} \boldsymbol{f}_1^r(e_4), \qquad S_{II} = \sum_{e \in E_{II}} \boldsymbol{f}_1^r(e_6).$$

Then,

$$\sum_{e\in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| = (S_I - T_I) + (T_{II} - S_{II})$$

On one hand, by error in congestion, we have

$$S_I + T_{II} \le \sum_{e\in E^f} \boldsymbol{u}^f(e) + |E^f|\epsilon^r = M^f + |E^f|\epsilon^r. \tag{58}$$

On the other hand, applying error in demand on vertex t_i and s_i, we have

$$T_I + T_{II} \ge (2M^f - \epsilon^r) - \boldsymbol{f}_1^r(z_1, \bar{t}_1) \ge 2M^f - \epsilon^r - M^f - \epsilon^r = M^f - 2\epsilon^r,$$

$$S_I + S_{II} \ge (2M^f - \epsilon^r) - \boldsymbol{f}_1^r(\bar{s}_1, z_1') \ge 2M^f - \epsilon^r - M^f - \epsilon^r = M^f - 2\epsilon^r.$$

Thus,

$$S_I + T_{II} = S_I + (T_I + T_{II}) - T_I \ge M^f - 2\epsilon^r + (S_I - T_I),$$
$$S_I + T_{II} = (S_I + S_{II}) + T_{II} - S_{II} \ge M^f - 2\epsilon^r + (T_{II} - S_{II}).$$

Together with Eq. (58), we obtain

$$S_I - T_I \le S_I + T_{II} - M^f + 2\epsilon^r \le M^f + |E^f|\epsilon^r - M^f + 2\epsilon^r = (|E^f| + 2)\epsilon^r,$$

$$T_{II} - S_{II} \le S_I + T_{II} - M^f + 2\epsilon^r \le M^f + |E^f|\epsilon^r - M^f + 2\epsilon^r = (|E^f| + 2)\epsilon^r.$$

Hence, we have

$$\sum_{e\in E^f} |\boldsymbol{f}_1^r(e_4) - \boldsymbol{f}_1^r(e_6)| = (S_I - T_I) + (T_{II} - S_{II}) \le 2(|E^f| + 2)\epsilon^r \le 6|E^f|\epsilon^r,$$

which finishes the proof. □

4.9 2CFR(A) to 2CF(A)

4.9.1 2CFR to 2CF

We show the reduction from a 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$ to a 2CF instance $(G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf})$. We copy the entire graph structure of G^r to G^{2cf}. In addition, we add to G^{2cf} with two new sources $\bar{\bar{s}}_1, \bar{\bar{s}}_2$ and two new edges $(\bar{\bar{s}}_1, \bar{s}_1), (\bar{\bar{s}}_2, \bar{s}_2)$ with capacity R_1, R_2, respectively. We set $R^{2cf} = R_1 + R_2$.

If a 2CF solver returns $\boldsymbol{f}^{2cf}$ for the 2CF instance $(G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf})$, then we return $\boldsymbol{f}^r$ for the 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$ by setting $\boldsymbol{f}_i^r(e) = \boldsymbol{f}_i^{2cf}(e), \forall e \in E^r, i \in \{1, 2\}$. If the 2CF solver returns "infeasible" for the 2CF instance, then we return "infeasible" for the 2CFR instance.

Lemma 4.26 (2CFR to 2CF)**.** *Given a 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$, we can construct, in time $O(|E^r|)$, a 2CF instance $(G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf})$ such that*

$$|V^{2cf}| = |V^r| + 2, \qquad |E^{2cf}| = |E^r| + 2, \qquad \left\|\boldsymbol{u}^{2cf}\right\|_{\max} = \max\{R_1, R_2\}, \qquad R^{2cf} = R_1 + R_2.$$

If the 2CFR instance has a solution, then the 2CF instance has a solution.

Proof. According to the reduction described above, if there exists a solution $\boldsymbol{f}^r$ to the 2CFR instance such that $F_1^r \geq R_1, F_2^r \geq R_2$, then there also exists a solution $\boldsymbol{f}^{2cf}$ to the 2CF instance because $F_1^{2cf} = R_1, F_2^{2cf} = R_2$, and thus $F_1^{2cf} + F_2^{2cf} = R^{2cf}$.

For the problem size after reduction, it is obvious that

$$|V^{2cf}| = |V^r| + 2, \qquad |E^{2cf}| = |E^r| + 2, \qquad \left\|\boldsymbol{u}^{2cf}\right\|_{\max} = \max\{R_1, R_2\}.$$

For the reduction time, it takes constant time to add two new vertices and edges, and it take $O(|E^r|)$ time to copy the rest edges. Thus, the reduction of this step can be performed in $O(|E^r|)$ time. □

4.9.2 2CFRA to 2CFA

Definition 4.27 (2CF Approximate Problem (2CFA))**.** A 2CFA instance is given by a 2CF instance $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R)$ and an error parameter $\epsilon \in [0,1]$, which we collect in a tuple $(G, \boldsymbol{u}, s_1, t_1, s_2, t_2, R, \epsilon)$. We say an algorithm solves the 2CFA problem, if, given any 2CFA instance, it returns a pair of flows $\boldsymbol{f}_1, \boldsymbol{f}_2 \geq 0$ that satisfies

$$\boldsymbol{f}_1(e) + \boldsymbol{f}_2(e) \leq \boldsymbol{u}(e) + \epsilon, \ \forall e \in E \tag{59}$$

$$\left| \sum_{u:(u,v)\in E} \boldsymbol{f}_i(u,v) - \sum_{w:(v,w)\in E} \boldsymbol{f}_i(v,w) \right| \leq \epsilon, \ \forall v \in V \backslash \{s_i, t_i\}, i \in \{1,2\} \tag{60}$$

$$\left| \sum_{w:(s_i,w)\in E} \boldsymbol{f}_i(s_i,w) - F_i \right| \leq \epsilon, \qquad \left| \sum_{u:(u,t_i)\in E} \boldsymbol{f}_i(u,t_i) - F_i \right| \leq \epsilon, \ i \in \{1,2\} \tag{61}$$

where $F_1 + F_2 = R$[9]; or it correctly declares that the associated 2CF instance is infeasible. We refer to the error in (59) as error in congestion, error in (60), (61) as error in demand.

We use the same reduction method and solution mapping method in the exact case to the approximate case.

Lemma 4.28 (2CFRA to 2CFA)**.** *Given a 2CFRA instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2, \epsilon^r)$, if we use Lemma 4.26 to construct a 2CF instance $(G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf})$ from the 2CFR instance $(G^r, \boldsymbol{u}^r, \bar{s}_1, \bar{t}_1, \bar{s}_2, \bar{t}_2, R_1, R_2)$, and let*

$$\epsilon^{2cf} = \frac{\epsilon^r}{4},$$

then the 2CFA instance $(G^{2cf}, \boldsymbol{u}^{2cf}, \bar{\bar{s}}_1, \bar{t}_1, \bar{\bar{s}}_2, \bar{t}_2, R^{2cf}, \epsilon^{2cf})$ satisfies.

Furthermore, if $\boldsymbol{f}^{2cf}$ is a solution to the 2CF(A) instance, then in time $O(|E^r|)$, we can compute a solution $\boldsymbol{f}^r$ to the 2CFR(A) instance, where the exact case holds when $\epsilon^{2cf} = \epsilon^r = 0$.

Proof. Based on the solution mapping method described above, it takes constant time to set the value of each entry of $\boldsymbol{f}^r$, and $\boldsymbol{f}^r$ has $|E^r|$ entries. Thus, such a solution mapping takes $O(|E^r|)$ time.

Now, we conduct an error analysis. By the error notions of 2CFRA (Definition 4.23), there are two types of error after solution mapping: (1) error in congestion; (2) error in demand.

[9] If we encode 2CF as an LP instance, and approximately solve the LP with at most ϵ additive error. Then, the approximately solution also agrees with the error notions of 2CF, except that we get $F_1 + F_2 \geq R - \epsilon$ instead of $F_1 + F_2 \geq R$. This inconsistency can be eliminated by setting $\epsilon' = 2\epsilon$, and slightly adjusting F_1, F_2 to F_1', F_2' such that $F_1' + F_2' \geq R$. This way, we obtain an approximate solution to 2CF with at most ϵ' additive error.

1. Error in congestion.
   Error in congestion does not increase since flows and edge capacities are unchanged for those edges other than $(\bar{\bar{s}}_1, \bar{s}_1), (\bar{\bar{s}}_2, \bar{s}_2)$. Therefore, we have $\tau_u^r \le \epsilon^{2cf}$.

2. Error in demand.
   It is noticed that only the incoming flow of vertex $\bar{s}_1, \bar{s}_2$ changes by solution mapping. Therefore, error in demand does not increase for vertices other than $\bar{s}_1, \bar{s}_2$, thus we only need to bound the error in demand of $\bar{s}_1, \bar{s}_2$. We consider commodity $i, i \in \{1, 2\}$.

   - **Case 1:** For vertex $\bar{s}_{\bar{i}}$
     We have $\tilde{\tau}_{di}^r = \sum_{w:(\bar{s}_{\bar{i}}, w)\in E} \boldsymbol{f}_i^r(\bar{s}_{\bar{i}}, w) \le 2\epsilon^{2cf}$ because error is accumulated twice over two vertices $\bar{\bar{s}}_{\bar{i}}, \bar{s}_{\bar{i}}$.
   - **Case 2:** For vertex $\bar{s}_i$
     We need to bound $\left|\sum_{w:(\bar{s}_i, w)\in E} \boldsymbol{f}_i^{2cf}(\bar{s}_i, w) - R_i\right|$. By construction, we have
     $$F_1^{2cf} + F_2^{2cf} = R^{2cf} = R_1 + R_2. \tag{62}$$
     By error in congestion on edges $(\bar{\bar{s}}_i, \bar{s}_i), i \in \{1, 2\}$, we have
     $$\boldsymbol{f}_i^{2cf}(\bar{\bar{s}}_i, \bar{s}_i) - R_i \le \boldsymbol{f}^{2cf}(\bar{\bar{s}}_i, \bar{s}_i) - R_i \le \epsilon^{2cf}. \tag{63}$$
     By error in demand on vertices $\bar{\bar{s}}_i, i \in \{1, 2\}$, we have
     $$\left|\boldsymbol{f}_i^{2cf}(\bar{\bar{s}}_i, \bar{s}_i) - F_i^{2cf}\right| \le \epsilon^{2cf}. \tag{64}$$
     By error in demand on vertices $\bar{s}_i, i \in \{1, 2\}$, we have
     $$\left|\boldsymbol{f}_i^{2cf}(\bar{\bar{s}}_i, \bar{s}_i) - \sum_{w:(\bar{s}_i, w)\in E} \boldsymbol{f}_i^{2cf}(\bar{s}_i, w)\right| \le \epsilon^{2cf}. \tag{65}$$
     Combining Eqs. (62), (63), (64) gives
     $$\left|F_i^{2cf} - R_i\right| \le 2\epsilon^{2cf}, \ i \in \{1, 2\}; \tag{66}$$
     and combining Eqs. (64), (65) gives
     $$\left|F_i^{2cf} - \sum_{w:(\bar{s}_i, w)\in E} \boldsymbol{f}_i^{2cf}(\bar{s}_i, w)\right| \le 2\epsilon^{2cf}, \ i \in \{1, 2\}. \tag{67}$$
     Eqs. (67), (66) gives
     $$\bar{\tau}_{di}^r = \left|\sum_{w:(\bar{s}_i, w)\in E} \boldsymbol{f}_i^{2cf}(\bar{s}_i, w) - R_i\right| \le 4\epsilon^{2cf}, \ i \in \{1, 2\}.$$

   Combining all cases, we can bound the error in demand for commodity i uniformly by
   $$\tau_{di}^r := \max\{\tilde{\tau}_{di}^r, \bar{\tau}_{di}^r \le 4\epsilon^{2cf}.\}$$

To summarize, as we set in the reduction that $\epsilon^{2cf} = \frac{\epsilon^r}{4}$, then we have
$$\tau_u^r, \tau_{d1}^r, \tau_{d2}^r \le \epsilon^r,$$
indicating that $\boldsymbol{f}^r$ is a solution to the 2CFRA instance. □

5 Main Theorem

Now, we are ready to prove the main theorem.

Theorem 5.1 (Restatement of Theorem 3.1.)**.** *Given an* LPA *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon^{lp})$ *where* $\boldsymbol{A} \in \mathbb{Z}^{m\times n}, \boldsymbol{b} \in Z^m, \boldsymbol{c} \in \mathbb{Z}^n, K \in \mathbb{Z}$*, we can construct, in time* $O(\mathrm{nnz}(\boldsymbol{A}) \log X)$ *where* $X = X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K)$*, a* 2CFA *instance* $(G^{2cf}, \boldsymbol{u}^{2cf}, s_1, t_1, s_2, t_2, R^{2cf}, \epsilon^{2cf})$ *such that*

$$|V^{2cf}|, |E^{2cf}| \le 10^6 \,\mathrm{nnz}(\boldsymbol{A})(3 + \log X),$$

$$\left\|\boldsymbol{u}^{2cf}\right\|_{\max}, R^{2cf} \le 10^8 \,\mathrm{nnz}^3(\boldsymbol{A}) R X^2 (2 + \log X)^2,$$

$$\epsilon^{2cf} \ge \frac{\epsilon^{lp}}{10^{24} \,\mathrm{nnz}^7(\boldsymbol{A}) R X^3 (3 + \log X)^6}.$$

If the LP *instance* $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R)$ *has a solution, then the* 2CF *instance* $(G^{2cf}, \boldsymbol{u}^{2cf}, s_1, t_1, s_2, t_2, R^{2cf})$ *has a solution. Furthermore, if* $\boldsymbol{f}^{2cf}$ *is a solution to the* 2CF(A) *instance, then in time* $O(\mathrm{nnz}(\boldsymbol{A}) \log X)$*, we can compute a solution* $\boldsymbol{x}$ *to the* LP(A) *instance, where the exact case holds when* $\epsilon^{2cf} = \epsilon^{lp} = 0$.

Proof. The theorem can be proved by putting all lemmas related to approximate problems in Section 4 together. Given an LPA instance $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K, R, \epsilon^{lp})$, for each problem along the reduction chain, we bound its problem size, problem error, reduction time, and solution mapping time, in terms of the input parameters of the LPA instance.

1. LPA
   - problem size: $n, m, \mathrm{nnz}(\boldsymbol{A}), R, X = X(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, K)$
   - problem error: ϵ^{lp}
   - reduction time: \
   - solution mapping time (Lemma 4.4) [10]: $O(n)$
2. LENA
   Note: for simplification, in the following calculations, we replace n, m with $\mathrm{nnz}(\boldsymbol{A})$ since wlog we can assume $1 \le n, m \le \mathrm{nnz}(\boldsymbol{A})$.
   - problem size (Lemma 4.1):

$$\begin{aligned}
\tilde{n} &= n + m + 1 \le 3 \,\mathrm{nnz}(\boldsymbol{A}) \\
\tilde{m} &= m + 1 \le 2 \,\mathrm{nnz}(\boldsymbol{A}) \\
\mathrm{nnz}(\tilde{\boldsymbol{A}}) &\le 4 \,\mathrm{nnz}(\boldsymbol{A}) \\
\tilde{R} &= 5mRX \le 5 \,\mathrm{nnz}(\boldsymbol{A}) RX \\
X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}) &= X
\end{aligned}$$

   - problem error (Lemma 4.4): $\epsilon^{le} = \epsilon^{lp}$
   - reduction time (Lemma 4.1): $O(\mathrm{nnz}(\boldsymbol{A}))$
   - solution mapping time (Lemma 4.7): $O(\tilde{n}) \le O(\mathrm{nnz}(\boldsymbol{A}))$

[10] The time needed to turn a solution to the next problem in the reduction chain to a solution to this problem.

3. 2-LENA (by Lemma 4.7 and Lemma 4.9)

   - problem size (Lemma 4.5):

$$\begin{aligned}
&\bar{n} \leq \tilde{n} + 4\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) \leq 8\,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
&\bar{m} \leq 3\tilde{m}\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) \leq 6\,\mathrm{nnz}(\boldsymbol{A})(1 + \log X) \\
&\mathrm{nnz}(\bar{\boldsymbol{A}}) \leq 17\,\mathrm{nnz}(\tilde{\boldsymbol{A}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) \leq 68\,\mathrm{nnz}(\boldsymbol{A})(1 + \log X) \\
&\bar{R} = 8\tilde{m}\tilde{R}X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\left(1 + \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) \leq 80\,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1 + \log X) \\
&X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) = 2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\tilde{R} = 10\,\mathrm{nnz}(\boldsymbol{A})RX^2
\end{aligned}$$

   - problem error (Lemma 4.7):

$$\epsilon^{2le} = \frac{\epsilon^{le}}{2X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})} = \frac{\epsilon^{lp}}{2X}$$

   - reduction time Lemma 4.5: $O\left(\mathrm{nnz}(\tilde{\boldsymbol{A}}) \log X(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}})\right) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$
   - solution mapping time (Lemma 4.9): $O(\bar{n}) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$

4. 1-LENA (by Lemma 4.9 and Lemma 4.12)

   - problem size (Lemma 4.8):

$$\begin{aligned}
&\hat{n} \leq 2\bar{n} \leq 16\,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
&\hat{m} \leq \bar{m} + \bar{n} \leq 14\,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
&\mathrm{nnz}(\hat{\boldsymbol{A}}) \leq 4\,\mathrm{nnz}(\bar{\boldsymbol{A}}) \leq 272\,\mathrm{nnz}(\boldsymbol{A})(1 + \log X) \\
&\hat{R} = 2\bar{R} \leq 160\,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1 + \log X) \\
&X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}}) = X(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}) = 10\,\mathrm{nnz}(\boldsymbol{A})RX^2
\end{aligned}$$

   - problem error (Lemma 4.9):

$$\epsilon^{1le} = \frac{\epsilon^{2le}}{\bar{n} + 1} \geq \frac{\epsilon^{lp}}{16\,\mathrm{nnz}(\boldsymbol{A})X(3 + \log X)}$$

   - reduction time (Lemma 4.8): $O(\mathrm{nnz}(\bar{\boldsymbol{A}})) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$
   - solution mapping time (Lemma 4.12): $O(\hat{n}) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$

5. FHFA

   - problem size (Lemma 4.10):

$$\begin{aligned}
&|V^h| = 2\hat{m} + 2 \leq 28\,\mathrm{nnz}(\boldsymbol{A})(3 + \log X) \\
&|E^h| \leq 4\,\mathrm{nnz}(\hat{\boldsymbol{A}}) \leq 1088\,\mathrm{nnz}(\boldsymbol{A})(1 + \log X) \\
&|F^h| = \hat{m} \leq 14\,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
&h = \hat{n} + \hat{m} \leq 30\,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
&\left\|\boldsymbol{u}^h\right\|_{\max} = \max\left\{\hat{R}, X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})\right\} \leq 160\,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1 + \log X)
\end{aligned}$$

- problem error (Lemma 4.12):

$$\epsilon^h = \frac{\epsilon^{1le}}{5\hat{n}X(\hat{\boldsymbol{A}}, \hat{\boldsymbol{b}})} \geq \frac{\epsilon^{lp}}{12800\,\mathrm{nnz}^3(\boldsymbol{A})RX^3(3+\log X)^2}$$

- reduction time (Lemma 4.10): $O(\mathrm{nnz}(\hat{\boldsymbol{A}})) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$
- solution mapping time (Lemma 4.15): $O(\left|E^h\right|) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$

6. FPHFA

   - problem size (Lemma 4.13):

$$\begin{aligned}
&|V^p| \leq |V^h| + |E^h| \leq 1116\,\mathrm{nnz}(\boldsymbol{A})(3+\log X)\\
&|E^p| \leq 2|E^h| \leq 2176\,\mathrm{nnz}(\boldsymbol{A})(1+\log X)\\
&|F^p| = |F^h| \leq 14\,\mathrm{nnz}(\boldsymbol{A})(2+\log X)\\
&p \leq |E^h| \leq 1088\,\mathrm{nnz}(\boldsymbol{A})(1+\log X)\\
&\|\boldsymbol{u}^p\|_{\max} = \left\|\boldsymbol{u}^h\right\|_{\max} \leq 160\,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1+\log X)
\end{aligned}$$

   - problem error (Lemma 4.15):

$$\epsilon^p = \frac{\epsilon^h}{|E^h|} \geq \frac{\epsilon^{lp}}{2\cdot 10^7\,\mathrm{nnz}^4(\boldsymbol{A})RX^3(3+\log X)^3}$$

   - reduction time (Lemma 4.13): $O(|E^h|) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$
   - solution mapping time (Lemma 4.18): $O(|E^p|) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$

7. SFFA

   - problem size (Lemma 4.16):

$$\begin{aligned}
&|V^s| = |V^p| + 4p + 2 \leq 5468\,\mathrm{nnz}(\boldsymbol{A})(3+\log X)\\
&|E^s| = |E^p| + 7p \leq 9792\,\mathrm{nnz}(\boldsymbol{A})(1+\log X)\\
&|F^s| = |F^p| + 2p \leq 2190\,\mathrm{nnz}(\boldsymbol{A})(2+\log X)\\
&|S_1| = |E^p| + 2p \leq 4352\,\mathrm{nnz}(\boldsymbol{A})(1+\log X)\\
&|S_2| = 3p \leq 3264\,\mathrm{nnz}(\boldsymbol{A})(1+\log X)\\
&\|\boldsymbol{u}^s\|_{\max} = \|\boldsymbol{u}^p\|_{\max} \leq 160\,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1+\log X)
\end{aligned}$$

   - problem error (Lemma 4.18):

$$\epsilon^s = \frac{\epsilon^p}{11|E^p|} = \frac{\epsilon^{lp}}{5\cdot 10^{11}\,\mathrm{nnz}^5(\boldsymbol{A})RX^3(3+\log X)^4}$$

   - reduction time (Lemma 4.16): $O(|E^p|) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$
   - solution mapping time (Lemma 4.21): $O(|E^s|) = O(\mathrm{nnz}(\boldsymbol{A})\log X)$

8. 2CFFA

- problem size (Lemma 4.19):

$$\begin{aligned}
|V^f| &= |V^s| + 2(|S_1| + |S_2|) \leq 20700 \,\mathrm{nnz}(\boldsymbol{A})(3 + \log X) \\
|E^f| &= |E^s| + 4(|S_1| + |S_2|) \leq 40256 \,\mathrm{nnz}(\boldsymbol{A})(1 + \log X) \\
|F^f| &\leq 4(|F^s| + |S_1| + |S_2|) \leq 39224 \,\mathrm{nnz}(\boldsymbol{A})(2 + \log X) \\
\left\|\boldsymbol{u}^f\right\|_{\max} &= \|\boldsymbol{u}^s\|_{\max} \leq 160 \,\mathrm{nnz}^2(\boldsymbol{A})RX^2(1 + \log X)
\end{aligned}$$

- problem error (Lemma 4.21):

$$\epsilon^f = \frac{\epsilon^s}{6|E^s|} \geq \frac{\epsilon^{lp}}{3 \cdot 10^{16} \,\mathrm{nnz}^6(\boldsymbol{A})RX^3(3 + \log X)^5}$$

- reduction time (Lemma 4.19): $O(|E^s|) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$
- solution mapping time (Lemma 4.24): $O(|E^f|) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$

9. 2CFRA

- problem size (Lemma 4.22):

$$\begin{aligned}
&|V^r| = |V^f| + 2|E^f| + 8 \leq 2 \cdot 10^5 \,\mathrm{nnz}(\boldsymbol{A})(3 + \log X), \\
&|E^r| = 7|E^f| + 10 \leq 3 \cdot 10^5 \,\mathrm{nnz}(\boldsymbol{A})(3 + \log X), \\
&\|\boldsymbol{u}^r\|_{\max} = \max\left\{2\left\|\boldsymbol{u}^f\right\|_{\max}, M^f\right\} \leq |E^f| \left\|\boldsymbol{u}^f\right\|_{\max} \leq 7 \cdot 10^6 \,\mathrm{nnz}^3(\boldsymbol{A})RX^2(1 + \log X)^2, \\
&R_1 = R_2 = 2M^f \leq 1.4 \cdot 10^7 \,\mathrm{nnz}^3(\boldsymbol{A})RX^2(1 + \log X)^2
\end{aligned}$$

- problem error (Lemma 4.24):

$$\epsilon^r = \frac{\epsilon^f}{12|E^f|} \geq \frac{\epsilon^{lp}}{1.1 \cdot 10^{23} \,\mathrm{nnz}^7(\boldsymbol{A})RX^3(3 + \log X)^6}$$

- reduction time (Lemma 4.22): $O(|E^f|) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$
- solution mapping time (Lemma 4.28): $O(|E^r|) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$

10. 2CFA

- problem size (Lemma 4.26):

$$\begin{aligned}
|V^{2cf}|, |E^{2cf}| &\leq 10^6 \,\mathrm{nnz}(\boldsymbol{A})(3 + \log X), \\
\left\|\boldsymbol{u}^{2cf}\right\|_{max}, R^{2cf} &\leq 10^8 \,\mathrm{nnz}^3(\boldsymbol{A})RX^2(2 + \log X)^2
\end{aligned}$$

- problem error (Lemma 4.28):

$$\epsilon^{2cf} = \frac{\epsilon^r}{4} \geq \frac{\epsilon^{lp}}{5 \cdot 10^{23} \,\mathrm{nnz}^7(\boldsymbol{A})RX^3(3 + \log X)^6}$$

- reduction time (Lemma 4.26): $O(|E^r|) = O(\mathrm{nnz}(\boldsymbol{A}) \log X)$
- solution mapping time: \

$\square$

By applying Theorem 5.1, we can prove Corollary 3.2, which states that if there exists a fast high-accuracy 2CFA solver, then there also exists an almost equally fast high-accuracy LPA solver.

References

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. "Network Flows". In: (1993) (cit. on p. 1).

[AR98] Y. Aumann and Y. Rabani. "An O (Log k) Approximate Min-Cut Max-Flow Theorem and Approximation Algorithm". In: *SIAM Journal on Computing* 27.1 (1998), pp. 291–301 (cit. on p. 1).

[BKV09] C. Barnhart, N. Krishnan, and P. H. Vance. "Multicommodity Flow Problems". en. In: *Encyclopedia of Optimization*. Ed. by C. A. Floudas and P. M. Pardalos. Boston, MA: Springer US, 2009, pp. 2354–2362. ISBN: 978-0-387-74759-0 (cit. on p. 1).

[Che+20] L. Chen, G. Goranci, M. Henzinger, R. Peng, and T. Saranurak. "Fast Dynamic Cuts, Distances and Effective Resistances via Vertex Sparsifiers". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 1135–1146 (cit. on p. 2).

[Chr+11] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. "Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs". In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. 2011, pp. 273–282 (cit. on p. 2).

[CLS21] M. B. Cohen, Y. T. Lee, and Z. Song. "Solving Linear Programs in the Current Matrix Multiplication Time". In: *Journal of the ACM (JACM)* 68.1 (2021), pp. 1–39 (cit. on p. 2).

[CSW10] C. Chekuri, F. B. Shepherd, and C. Weibel. "Flow-Cut Gaps for Integer and Fractional Multiflows". In: *arXiv:1008.2136 [cs]* (2010). arXiv: `1008.2136 [cs]` (cit. on p. 1).

[Din70] E. A. Dinic. "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation". In: *Soviet Math. Doklady*. Vol. 11. 1970, pp. 1277–1280 (cit. on p. 2).

[ET75] S. Even and R. E. Tarjan. "Network Flow and Testing Graph Connectivity". In: *SIAM journal on computing* 4.4 (1975), pp. 507–518 (cit. on p. 2).

[Eva76] J. R. Evans. "A Combinatorial Equivalence between A Class of Multicommodity Flow Problems and the Capacitated Transportation Problem". en. In: *Mathematical Programming* 10.1 (1976), pp. 401–404. ISSN: 1436-4646 (cit. on p. 1).

[Eva78] J. R. Evans. "The Simplex Method for Integral Multicommodity Networks". en. In: *Naval Research Logistics Quarterly* 25.1 (1978), pp. 31–37. ISSN: 00281441, 19319193 (cit. on p. 1).

[FF56] L. R. Ford and D. R. Fulkerson. "Maximal Flow through a Network". In: *Canadian journal of Mathematics* 8 (1956), pp. 399–404 (cit. on p. 2).

[Fle00] L. K. Fleischer. "Approximating Fractional Multicommodity Flow Independent of the Number of Commodities". In: *SIAM Journal on Discrete Mathematics* 13.4 (2000), pp. 505–520. ISSN: 0895-4801 (cit. on p. 1).

[GK07] N. Garg and J. Könemann. "Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems". In: *SIAM Journal on Computing* 37.2 (2007), pp. 630–652. ISSN: 0097-5397 (cit. on p. 1).

[GLP21] Y. Gao, Y. P. Liu, and R. Peng. "Fully Dynamic Electrical Flows: Sparse Maxflow Faster Than Goldberg-Rao". In: *arXiv:2101.07233 [cs]* (2021). arXiv: `2101.07233 [cs]` (cit. on p. 2).

[GR98] A. V. Goldberg and S. Rao. "Beyond the Flow Decomposition Barrier". In: *Journal of the ACM* 45.5 (1998), pp. 783–797. ISSN: 0004-5411 (cit. on p. 2).

[Hu63] T. C. Hu. "Multi-Commodity Network Flows". In: *Operations Research* 11.3 (1963), pp. 344–360. ISSN: 0030-364X (cit. on p. 1).

[Ita78] A. Itai. "Two-Commodity Flow". In: *Journal of the ACM (JACM)* 25.4 (1978), pp. 596–611 (cit. on pp. 1, 3, 9).

[JS21] A. Jambulapati and A. Sidford. "Ultrasparse Ultrasparsifiers and Faster Laplacian System Solvers". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2021, pp. 540–559 (cit. on p. 2).

[Kar84] N. Karmarkar. "A New Polynomial-Time Algorithm for Linear Programming". In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. 1984, pp. 302–311 (cit. on pp. 1, 3).

[Kel+13] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. "A Simple, Combinatorial Algorithm for Solving SDD Systems in Nearly-Linear Time". In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. 2013, pp. 911–920 (cit. on p. 2).

[Kel+14] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. "An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and Its Multicommodity Generalizations". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2014, pp. 217–226 (cit. on pp. 1, 2).

[Ken78] J. L. Kennington. "A Survey of Linear Cost Multicommodity Network Flows". In: *Operations Research* 26.2 (1978), pp. 209–236. ISSN: 0030-364X (cit. on p. 1).

[Kha80] L. G. Khachiyan. "Polynomial Algorithms in Linear Programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72 (cit. on p. 1).

[KLS20] T. Kathuria, Y. P. Liu, and A. Sidford. "Unit Capacity Maxflow in Almost \$ O (M\$\^{}{\$4/3\$}\$) \$ Time". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2020, pp. 119–130 (cit. on p. 2).

[KMP10] I. Koutis, G. L. Miller, and R. Peng. "Approaching Optimality for Solving SDD Linear Systems". In: *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. FOCS '10. USA: IEEE Computer Society, 2010, pp. 235–244. ISBN: 978-0-7695-4244-7 (cit. on p. 2).

[KMP11] I. Koutis, G. L. Miller, and R. Peng. "A Nearly-m Log n Time Solver for Sdd Linear Systems". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE, 2011, pp. 590–598 (cit. on p. 2).

[KS16] R. Kyng and S. Sachdeva. "Approximate Gaussian Elimination for Laplacians-Fast, Sparse, and Simple". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 573–582 (cit. on p. 2).

[KWZ20] R. Kyng, D. Wang, and P. Zhang. "Packing LPs Are Hard to Solve Accurately, Assuming Linear Equations Are Hard". In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 279–296 (cit. on p. 3).

[Kyn+19] R. Kyng, R. Peng, S. Sachdeva, and D. Wang. "Flows in Almost Linear Time via Adaptive Preconditioning". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 902–913 (cit. on p. 2).

[KZ20] R. Kyng and P. Zhang. "Hardness Results for Structured Linear Systems". In: *SIAM Journal on Computing* 49.4 (2020), FOCS17–280 (cit. on p. 3).

[Lei+95] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas. "Fast Approximation Algorithms for Multicommodity Flow Problems". en. In: *Journal of Computer and System Sciences* 50.2 (1995), pp. 228–243. ISSN: 0022-0000 (cit. on p. 1).

[LLR95] N. Linial, E. London, and Y. Rabinovich. "The Geometry of Graphs and Some of Its Algorithmic Applications". In: *Combinatorica* 15.2 (1995), pp. 215–245 (cit. on p. 1).

[LR89] T. Leighton and S. Rao. *An Approximate Max-Flow Min-Cut Theorem for Uniform Multicommodity Flow Problems with Applications to Approximation Algorithms*. Tech. rep. MASSACHUSETTS INST OF TECH CAMBRIDGE MICROSYSTEMS RESEARCH CENTER, 1989 (cit. on p. 1).

[LR99] T. Leighton and S. Rao. "Multicommodity Max-Flow Min-Cut Theorems and Their Use in Designing Approximation Algorithms". In: *Journal of the ACM (JACM)* 46.6 (1999), pp. 787–832 (cit. on p. 1).

[LRS13] Y. T. Lee, S. Rao, and N. Srivastava. "A New Approach to Computing Maximum Flows Using Electrical Flows". In: *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*. 2013, pp. 755–764 (cit. on p. 2).

[LS20] Y. P. Liu and A. Sidford. "Faster Energy Maximization for Faster Maximum Flow". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020, pp. 803–814 (cit. on p. 2).

[Mad10] A. Madry. "Faster Approximation Schemes for Fractional Multicommodity Flow Problems via Dynamic Graph Algorithms". In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 121–130. ISBN: 978-1-4503-0050-6 (cit. on p. 1).

[Mad13] A. Madry. "Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 253–262 (cit. on p. 2).

[Mad16] A. Madry. "Computing Maximum Flow with Augmenting Electrical Flows". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016, pp. 593–602 (cit. on p. 2).

[Mus+19] C. Musco, P. Netrapalli, A. Sidford, S. Ubaru, and D. P. Woodruff. "Spectrum Approximation Beyond Fast Matrix Multiplication: Algorithms and Hardness". In: *arXiv:1704.04163 [cs, math]* (2019). arXiv: `1704.04163 [cs, math]` (cit. on p. 3).

[OMV00] A. Ouorou, P. Mahey, and J.-P. Vial. "A Survey of Algorithms for Convex Multicommodity Flow Problems". In: *Management Science* 46.1 (2000), pp. 126–147. ISSN: 0025-1909 (cit. on p. 1).

[Pen16] R. Peng. "Approximate Undirected Maximum Flows in o (m Polylog (n)) Time". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2016, pp. 1862–1867 (cit. on p. 1).

[PS14] R. Peng and D. A. Spielman. “An Efficient Parallel Solver for SDD Linear Systems”. In: *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*. 2014, pp. 333–342 (cit. on p. 2).

[Ren88] J. Renegar. “A Polynomial-Time Algorithm, Based on Newton’s Method, for Linear Programming”. In: *Mathematical programming* 40.1 (1988), pp. 59–93 (cit. on pp. 1, 2, 3).

[She13] J. Sherman. “Nearly Maximum Flows in Nearly Linear Time”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 2013, pp. 263–269 (cit. on pp. 1, 2).

[She17] J. Sherman. “Area-Convexity, L∞ Regularization, and Undirected Multicommodity Flow”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 452–460 (cit. on p. 1).

[ST04] D. A. Spielman and S.-H. Teng. “Nearly-Linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems”. In: *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*. STOC ’04. New York, NY, USA: Association for Computing Machinery, 2004, pp. 81–90. ISBN: 978-1-58113-852-8 (cit. on p. 2).

[Vai89] P. M. Vaidya. “Speeding-up Linear Programming Using Fast Matrix Multiplication”. In: *30th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1989, pp. 332–337 (cit. on p. 1).

[van+21] J. van den Brand, Y. T. Lee, Y. P. Liu, T. Saranurak, A. Sidford, Z. Song, and D. Wang. “Minimum Cost Flows, MDPs, and L1-Regression in Nearly Linear Time for Dense Instances”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 859–869. ISBN: 978-1-4503-8053-9 (cit. on p. 2).

[Wan18] I.-L. Wang. “Multicommodity Network Flows: A Survey, Part I: Applications and Formulations”. In: *International Journal of Operations Research* 15.4 (2018), pp. 145–153 (cit. on p. 1).

[WW18] V. V. Williams and R. R. Williams. “Subcubic Equivalences Between Path, Matrix, and Triangle Problems”. In: *Journal of the ACM* 65.5 (2018), 27:1–27:38. ISSN: 0004-5411 (cit. on p. 3).