A Scaling-Invariant Algorithm for Linear Programming Whose Running Time Depends Only on the Constraint Matrix*

Daniel Dadush
Centrum Wiskunde & Informatica
The Netherlands

Sophie Huiberts
Centrum Wiskunde & Informatica
The Netherlands

Bento Natura
London School of Economics and Political Science
United Kingdom

László A. Végh
London School of Economics and Political Science
United Kingdom

ABSTRACT

Following the breakthrough work of Tardos (Oper. Res. '86) in the bit-complexity model, Vavasis and Ye (Math. Prog. '96) gave the first exact algorithm for linear programming in the real model of computation with running time depending only on the constraint matrix. For solving a linear program (LP) max $c^\top x$, $Ax = b$, $x \geq 0$, $A \in \mathbb{R}^{m\times n}$, Vavasis and Ye developed a primal-dual interior point method using a *'layered least squares'* (LLS) step, and showed that $O(n^{3.5} \log(\bar{\chi}_A + n))$ iterations suffice to solve (LP) exactly, where $\bar{\chi}_A$ is a condition measure controlling the size of solutions to linear systems related to A.

Monteiro and Tsuchiya (SIAM J. Optim. '03), noting that the central path is invariant under rescalings of the columns of A and c, asked whether there exists an LP algorithm depending instead on the measure $\bar{\chi}^*_A$, defined as the minimum $\bar{\chi}_{AD}$ value achievable by a column rescaling AD of A, and gave strong evidence that this should be the case. We resolve this open question affirmatively.

Our first main contribution is an $O(m^2n^2 + n^3)$ time algorithm which works on the linear matroid of A to compute a nearly optimal diagonal rescaling D satisfying $\bar{\chi}_{AD} \leq n(\bar{\chi}^*)^3$. This algorithm also allows us to approximate the value of $\bar{\chi}_A$ up to a factor $n(\bar{\chi}^*)^2$. This result is in (surprising) contrast to that of Tunçel (Math. Prog. '99), who showed NP-hardness for approximating $\bar{\chi}_A$ to within $2^{\text{poly}(\text{rank}(A))}$. The key insight for our algorithm is to work with ratios g_i/g_j of circuits of A—i.e., minimal linear dependencies $Ag = 0$—which allow us to approximate the value of $\bar{\chi}^*_A$ by a maximum geometric mean cycle computation in what we call the *'circuit ratio digraph'* of A.

While this resolves Monteiro and Tsuchiya's question by appropriate preprocessing, it falls short of providing either a truly scaling invariant algorithm or an improvement upon the base LLS analysis. In this vein, as our second main contribution we develop a *scaling invariant* LLS algorithm, which uses and dynamically maintains improving estimates of the circuit ratio digraph, together with a refined potential function based analysis for LLS algorithms in general. With this analysis, we derive an improved $O(n^{2.5} \log n \log(\bar{\chi}^*_A + n))$ iteration bound for optimally solving (LP) using our algorithm. The same argument also yields a factor $n/\log n$ improvement on the iteration complexity bound of the original Vavasis-Ye algorithm.

CCS CONCEPTS

• **Theory of computation → Linear programming**.

KEYWORDS

Linear programming, interior point methods, condition number, chi bar, circuits, linear matroids

ACM Reference Format:
Daniel Dadush, Sophie Huiberts, Bento Natura, and László A. Végh. 2020. A Scaling-Invariant Algorithm for Linear Programming Whose Running Time Depends Only on the Constraint Matrix. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3357713.3384326

1 INTRODUCTION

The linear programming (LP) problem in primal-dual form is to solve

$$\begin{aligned} \min\ & c^\top x & \max\ & y^\top b \\ & Ax = b & & A^\top y + s = c \\ & x \geq 0, & & s \geq 0, \end{aligned} \tag{LP}$$

where $A \in \mathbb{R}^{m\times n}$, $\text{rank}(A) = m \leq n$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ are given in the input, and $x, s \in \mathbb{R}^n$, $y \in \mathbb{R}^m$ are the variables. We consider the program in x to be the primal problem and the program in y, s to be the dual problem.

Khachiyan [18] used the ellipsoid method to give the first polynomial time LP algorithm in the bit-complexity model, that is, polynomial in the bit description length of A, b, c. Following Khachiyan's work, the now forty year old open question is whether there exists a *strongly polynomial time* algorithm for LP. The task is to solve LP using $\text{poly}(n, m)$ basic arithmetic operations. Furthermore, the algorithm must be in PSPACE, that is, the numbers occurring in the computations must remain polynomially bounded in the input size. Known strongly polynomially solvable LP problems classes include: feasibility for two variable per inequality systems [26],

*Supported by the ERC Starting Grants ScaleOpt–757481 and QIP–805241.

STOC '20, June 22–26, 2020, Chicago, IL, USA

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384326

the minimum-cost circulation problem [41], the maximum generalized flow problem [33, 50], and discounted Markov decision problems [52, 53].

For more general LP classes, for which strongly polynomial algorithms are not known, the principal line of attack has been to reduce the *numerical complexity* of LP algorithms. More precisely, the goal has been to develop algorithms whose number of arithmetic operations depend on natural *condition measures* of the base LP; at a high level, these condition measures attempt to finely measure the "intrinsic complexity" of the LP. An important line of work in this area has been to parametrize LPs by the "niceness" of their solutions (e.g. the depth of the most interior point), where relevant examples include the Goffin measure [11] for conic systems and Renegar's distance to ill-posedness for general LPs [35, 36], and bounded ratios between the nonzero entries in basic feasible solutions [4, 19].

Parametrizing by the Constraint Matrix. A second line of research, and the main focus of this work, makes no assumptions on the "niceness" of solutions and instead focuses on the complexity of the constraint matrix A. The first breakthrough in this area was given by Tardos [42], who showed that if A has integer entries and all square submatrices of A have determinant at most Δ in absolute value, then (LP) can be solved in time $\text{poly}(n, m, \log \Delta)$. This is achieved by finding the exact solutions to n^2 rounded LPs derived from the original LP, with the right hand side vector and cost function being integers of absolute value bounded in terms of n and Δ. From n such rounded problem instances, one can infer, via proximity results, that a constraint $x_i = 0$ must be valid for every optimal solution. The process continues by induction until the optimal primal face is identified.

Path-Following Methods and the Vavasis-Ye Algorithm. In a seminal work, Vavasis and Ye [49] introduced a new type of interior-point method that optimally solves (LP) within $O(n^{3.5} \log(\bar{\chi}_A + n))$ iterations, where the condition number $\bar{\chi}_A$ controls the size of solutions to certain linear systems related to the kernel of A (see Section 2 for the formal definition).

Before detailing the Vavasis-Ye (henceforth VY) algorithm, we recall the basics of path following interior-point methods. If both the primal and dual problems in (LP) are strictly feasible, the *central path* for (LP) is the curve $((x(\mu), y(\mu), s(\mu)) : \mu > 0)$ defined by

$$\begin{aligned} x(\mu)_i s(\mu)_i &= \mu, \quad \forall i \in [n] \\ Ax(\mu) &= b,\ x(\mu) > 0, \\ A^\top y(\mu) + s(\mu) &= c,\ s(\mu) > 0, \end{aligned} \tag{CP}$$

which converges to complementary optimal primal and dual solutions (x^*, y^*, s^*) as $\mu \to 0$, recalling that the optimality gap at time μ is exactly $x(\mu)^\top s(\mu) = n\mu$. We thus refer to μ as the normalized dualized gap. Methods that "follow the path" generate iterates that stay in a certain neighborhood around it while trying to achieve rapid multiplicative progress w.r.t. to μ, where given (x, y, s) close to the path, we define the effective μ as $\mu(x, y, s) = \sum_{i=1}^n x_i s_i / n$. In general, the direction of movement at each iteration is computed by solving a carefully chosen linear system. Given a target parameter μ' and starting point close to the path at parameter μ, standard path following methods [12] can compute a point at parameter below μ' in at most $O(\sqrt{n} \log(\mu/\mu'))$ iterations, and hence the quantity $\log(\mu/\mu')$ can be usefully interpreted as the length of the corresponding segment of the central path.

Crossover Events and Layered Least Squares Steps. At a very high level, Vavasis and Ye show that the central path can be decomposed into at most $\binom{n}{2}$ short but curved segments, possibly joined by long (apriori unbounded) but very straight segments. At the end of each curved segment, they show that a new ordering relation $x_i(\mu) > x_j(\mu)$—called a *'crossover event'*—is implicitly learned, where this relation did not hold at the start of the segment, but will hold at every point from the end of the segment onwards. These $\binom{n}{2}$ relations give a combinatorial way to measure progress along the central path. In contrast to Tardos's algorithm, where the main progress is setting variables to zero explicitly, the variables participating in crossover events cannot be identified, only their existence is shown.

At a technical level, the VY algorithm is a variant of the Mizuno-Todd-Ye [29] predictor-corrector method (MTY P-C). In predictor-corrector methods, corrector steps bring an iterate closer to the path, i.e., improve centrality, and predictor steps "shoot down" the path, i.e., reduce μ without losing too much centrality. VY's main algorithmic innovation was the introduction of a new predictor step, called the *'layered least squares'* (LLS) step, which crucially allowed them to cross each aforementioned "straight" segment of the central path in *a single step*, recalling that these straight segments may be arbitrarily long. To traverse the short and curved segments of the path, the standard predictor step, known as *affine scaling* (AS), in fact suffices.

To compute the LLS direction, the variables are decomposed into 'layers' $J_1 \cup J_2 \cup \ldots \cup J_p = [n]$. The goal of such a decomposition is to eventually learn a refinement of the optimal partition of the variables $B^* \cup N^* = [n]$, where $B^* := \{i \in [n] : x_i^* > 0\}$ and $N^* := \{i \in [n] : s_i^* > 0\}$ for the limit optimal solution (x^*, y^*, s^*).

The primal affine scaling direction can be equivalently described by solving a weighted least squares problem in $\text{Ker}(A)$, with respect to a weighting defined according to the current iterate. The primal LLS direction is obtained by solving a series of weighted least squares problems, starting with focusing only on the final layer J_p. This solution is gradually extended to the higher layers (which refers to layers with lower indices). The dual directions have analogous interpretations, with the solutions on the layers obtained in the opposite direction, starting with J_1. If we use the two-level layering $J_1 = B^*$, $J_2 = N^*$, and are sufficiently close to the limit (x^*, y^*, s^*) of the central path, then the LLS step reaches an exact optimal solution in a single step. We note that standard AS steps generically never find an exact optimal solution, and thus some form of "LLS rounding" is always necessary to achieve finite termination.

Of course, guessing B^* and N^* correctly is just as hard as solving (LP). Still, if we work with a "good" layering, these will reveal new information about the "optimal order" of the variables, where B^* is placed on higher layers than N^*. The crossover events correspond to swapping two wrongly ordered variables into the correct ordering. Namely, a variable $i \in B^*$ and $j \in N^*$ are currently ordered on the same layer, or j is in a higher layer than i. After the crossover event, i will always be placed on a higher layer than j.

Computing Good Layerings and the $\bar{\chi}_A$ Condition Measure. Given the above discussion, the obvious question is how to come up with "good" layerings? The philosophy behind LLS can be stated as saying that if modifying a set of variables x_I barely affects the variables in $x_{[n]\setminus I}$ (recalling that movement is constrained to $\Delta x \in \mathrm{Ker}(A)$), then one should optimize over x_I without regard to the effect on $x_{[n]\setminus I}$; hence x_I should be placed on lower layers.

VY's strategy for computing such layerings was to directly use the size of the coordinates of the current iterate x (where (x, y, s) is a point near the central path). In particular, assuming $x_1 \geq x_2 \geq \ldots \geq x_n$, the layering $J_1 \cup J_2 \cup \ldots \cup J_p = [n]$ corresponds to consecutive intervals constructed in decreasing order of x_i values. The break between J_i and J_{i+1} occurs if the gap $x_r/x_{r+1} > g$, where r is the rightmost element of J_i and $g > 0$ is a threshold parameter. Thus, the expectation is that if $x_i > gx_j$, then a small multiplicative change to x_j, subject to moving in $\mathrm{Ker}(A)$, should induce a small multiplicative change to x_i. By proximity to the central path, the dual ordering is reversed as mentioned above.

The threshold g for which this was justified in VY was a function of the $\bar{\chi}_A$ condition measure. We now provide a convenient definition, which immediately yields this justification (see Proposition 2.3). Letting $W = \mathrm{Ker}(A)$ and $\pi_I(W) = \{x_I : x \in W\}$, we define $\bar{\chi}_A := \bar{\chi}_W$ as the minimum number $M \geq 1$ such that for any $\emptyset \neq I \subseteq [n]$ and $z \in \pi_I(W)$, there exists $y \in W$ with $y_I = z$ and $\|y\| \leq M\|z\|$. Thus, a change of ε in variables in I can be lifted to a change of at most $\bar{\chi}_A\varepsilon$ in variables in $[n] \setminus I$. Crucially, $\bar{\chi}$ is a "self-dual" quantity. That is, $\bar{\chi}_W = \bar{\chi}_{W^\perp}$, where $W^\perp = \mathrm{range}(A^\top)$ is the movement subspace for the dual problem, justifying the reversed layering for the dual (see Sections 2 for more details).

The Question of Scale Invariance and $\bar{\chi}_A^*$. While the VY layering procedure is powerful, its properties are somewhat mismatched with those of the central path. In particular, variable ordering information has *no intrinsic meaning* on the central path, as the path itself is *scaling invariant*. Namely the central path point $(x(\mu), y(\mu), s(\mu))$ w.r.t. the problem instance (A, b, c) is in bijective correspondence with the central path point $(D^{-1}x(\mu), Dy(\mu), Ds(\mu)))$ w.r.t. the problem instance (AD, Dc, b) for any positive diagonal matrix D. The standard path following algorithms are also scaling invariant in this sense.

This lead Monteiro and Tsuchiya [31] to ask whether a scaling invariant LLS algorithm exists. They noted that any such algorithm would then depend on the potentially much smaller parameter

$$\bar{\chi}_A^* := \inf_D \bar{\chi}_{AD}\,, \tag{1}$$

where the infimum is taken over the set of $n \times n$ diagonal matrices. Thus, Monteiro and Tsuchiya's question can be rephrased as to whether there exists an exact LP algorithm with running time $\mathrm{poly}(n, m, \log \bar{\chi}_A^*)$.

Substantial progress on this question was made in the followup works [21, 32]. The paper [32] showed that the number of iterations of the MTY predictor-corrector algorithm [29] can get from $\mu_0 > 0$ to $\eta > 0$ on the central path in $O(n^{3.5} \log \bar{\chi}^* + \min\{n^2 \log\log(\mu^0/\eta), \log(\mu^0/\eta)\})$ iterations. This is attained by showing that the standard AS steps are reasonably close to the LLS steps. This proximity can be used to show that the AS steps can traverse the curved parts of the central path in the same iteration complexity bound as the VY algorithm. Moreover, on the "straight" parts of the path, the rate of progress amplifies geometrically, thus attaining a log log convergence on these parts. Subsequently, [21] developed an affine invariant *trust region* step, which traverses the full path in $O(n^{3.5} \log(\bar{\chi}_A^* + n))$ iterations. However, each iteration is weakly polynomial in b and c. The question of developing an LP algorithm with complexity bound $\mathrm{poly}(n, m, \log \bar{\chi}_A^*)$ thus remained open.

A related open problem to the above is whether it is possible to compute a near-optimal rescaling D for program (1)? This would give an alternate pathway to the desired LP algorithm by simply preprocessing the matrix A. The related question of approximating $\bar{\chi}_A$ was already studied by Tunçel [45], who showed NP-hardness for approximating $\bar{\chi}_A$ to within a $2^{\mathrm{poly}(\mathrm{rank}(A))}$ factor. Taken at face value, this may seem to suggest that approximating the rescaling D should be hard.

A further open question is whether Vavasis and Ye's base crossover analysis can be improved. Ye in [**?**] showed that the iteration complexity can be reduced to $O(n^{2.5} \log(\bar{\chi}_A + n))$ for feasibility problems and further to $O(n^{1.5} \log(\bar{\chi}_A + n))$ for homogeneous systems, though the $O(n^{3.5} \log(\bar{\chi}_A + n))$ bound for optimization has remained unimproved since [49].

Our Contributions. In this work, we resolve all of the above questions in the affirmative. We detail our contributions below.

1. Finding an Approximately Optimal Rescaling. As our first contribution, we give an $O(m^2n^2 + n^3)$ time algorithm which works on the linear matroid of A to compute a diagonal rescaling matrix D which achieves $\bar{\chi}_{AD} \leq n(\bar{\chi}_A^*)^3$, given any $m \times n$ matrix A. Furthermore, this same algorithm allows us to approximate $\bar{\chi}_A$ to within a factor $n(\bar{\chi}_A^*)^2$. The algorithm bypasses Tunçel's hardness result by allowing the approximation factor to depend on A itself, namely on $\bar{\chi}_A^*$. This gives a simple first answer to Monteiro and Tsuchiya's question: by applying the Vavasis-Ye algorithm directly on the preprocessed A matrix, we may solve any LP with constraint matrix A using $O(n^{3.5}(\log \bar{\chi}_A^* + n))$ iterations. Note that the approximation factor $n(\bar{\chi}_A^*)^2$ increases the runtime only by a constant factor.

To achieve this result, we work directly with the circuits of A, where a circuit $C \subseteq [n]$ is $C = \mathrm{supp}(g)$ for a minimal linear dependency $Ag = 0$. With each circuit, we can associate a vector $g^C \in \mathrm{Ker}(A)$ with $\mathrm{supp}(g^C) = C$ that is unique up to scaling. By the *'circuit ratio'* of (i, j), we mean the largest ratio $|g_j^C/g_i^C|$ taken over every circuit C of A such that $i, j \in C$. As our first observation, we show that the maximum of all circuit ratios, which we call the *'circuit imbalance measure'*, in fact characterizes $\bar{\chi}_A$ up to a factor n. This measure was first studied by Vavasis [48], who showed that it lower bounds $\bar{\chi}_A$, though, as far as we are aware, our upper bound is new. The circuit ratios of each pair (i, j) induces a weighted directed graph we call the *circuit ratio digraph* of A. From here, our main result is that $\bar{\chi}_A^*$ is up to a factor n equal to the maximum geometric mean cycle in the circuit ratio digraph. Our approximation algorithm populates the circuit ratio digraph with ratios for each i, j using basic matroid techniques, and then computes a rescaling by solving the dual of the maximum geometric mean ratio cycle on the 'approximate circuit ratio digraph'.

2. Scaling Invariant LLS Algorithm. While the above yields an LP algorithm with poly$(n, m, \log \bar{\chi}_A^*)$ running time, it does not satisfactorily address Monteiro and Tsuchiya's question for a scaling invariant algorithm. As our second contribution, we use the circuit ratio digraph directly to give a natural scaling invariant LLS layering algorithm together with a scaling invariant crossover analysis.

At a conceptual level, we show that the circuit ratios give a scale invariant way to measure whether '$x_i > x_j$' and enable a natural layering algorithm. Let κ_{ij} be the circuit imbalance between i and j, i.e., the maximum value $|g_j/g_i|$ for a minimal kernel solution g containing i and j in the support. Given the circuit ratio graph induced by κ and a primal point x near the path, our layering algorithm can be described as follows. We first rescale the variables so that x becomes the all ones vector, which rescales κ_{ij} to $\kappa_{ij}x_i/x_j$. We then restrict the graph its edges of length $\geq 1/\text{poly}(n)$—the *long edges* of the (rescaled) circuit ratio graph—and let the layering $J_1 \cup J_2 \cup \ldots \cup J_p$ be a topological ordering of its strongly connected components (SCC) with edges going from left to right. Intuitively, variables that "affect each other" should be in the same layer, which motivates the SCC definition.

We note that our layering algorithm does not in fact have access to the true circuit ratios κ_{ij}, as these are NP-hard to compute. Getting a good enough initial estimate for our purposes however is easy: we let $\hat{\kappa}_{ij}$ be the ratio corresponding to an *arbitrary* circuit containing i and j. This already turns out to be within a factor $(\bar{\chi}_A^*)^2$ from the true value κ_{ij}, which we recall is the maximum over all such circuits. Our layering algorithm in fact learns better circuit ratio estimates if the "lifting costs" of our SCC layering, i.e., how much it costs to lift changes from lower layer variables to higher layers (as in the definition of $\bar{\chi}_A$), are larger than we expected them to be.

For our analysis, we define cross-overs in a scaling invariant way as follows. Before the crossover event, $\text{poly}(n)(\bar{\chi}_A^*)^n > \kappa_{ij}x_i/x_j$, and after the crossover event, $\text{poly}(n)(\bar{\chi}_A^*)^n < \kappa_{ij}x_i/x_j$ for all further central path points. Our analysis relies on $\bar{\chi}_A^*$ in only a minimalistic way, and does not require an estimate on the value of $\bar{\chi}_A^*$. Namely, it is only used to show that if $i, j \in J_q$, for a layer $q \in [p]$, then the rescaled circuit ratio $\kappa_{ij}x_i/x_j$ is in the range $(\text{poly}(n)\bar{\chi}_A^*)^{O(\pm|J_q|)}$. The argument to show this crucially utilizes the maximum geometric mean cycle characterization. Furthermore, unlike prior analyses [31, 49], our definition of a "good" layering (i.e., 'balanced' layerings, see Section 3.4), is completely independent of $\bar{\chi}_A^*$.

3. Improved Potential Analysis. As our third contribution, we improve the Vavasis-Ye crossover analysis using a new and simple potential function based approach. When applied to our new LLS algorithm, we derive an $O(n^{2.5} \log n \log(\bar{\chi}_A^* + n))$ iteration bound for path following, improving the polynomial term by an $\Omega(n/\log n)$ factor compared to the VY analysis.

Our potential function can be seen as a fine-grained version of the crossover events as described above. In case of such a crossover event, it is guaranteed that in every subsequent iteration, i is in a layer before j. Instead, we analyze less radical changes: an "event" parametrized by τ means that i and j are currently together on a layer of size $\leq \tau$, and after the event, i is on a layer before j, or if they are together on the same layer, then this layer must have size $\geq 2\tau$. For every LLS step, we can find a parameter τ such that an event of this type happens concurrently for at least $\tau - 1$ pairs within the next $O(\sqrt{n}\tau \log(\bar{\chi}_A^* + n))$ iterations,

Our improved analysis is also applicable to the original VY algorithm. Let us now comment on the relation between the VY algorithm and our new algorithm. The VY algorithm starts a new layer once $x_{\pi(i)} > gx_{\pi(i+1)}$ between two consecutive variables where the permutation π is a non-increasing order of the x_i variables. Here, $g = \text{poly}(n)\bar{\chi}$. Setting the initial 'estimates' $\hat{\kappa}_{ij} = g/\text{poly}(n)$ for a suitable polynomial, our algorithm runs the same way as the VY algorithm. Using these estimates, the layering procedure becomes much simpler: there is no need to verify 'balancedness' as in our general algorithm.

However, setting $g = \hat{\kappa}_{ij}$ has drawbacks. Most importantly, it does not give a lower bound on the true circuit ratio κ_{ij}—to the contrary, g will be an upper bound! In effect, this causes VY's layers to be "much larger" than ours, and for this reason, the connection to $\bar{\chi}^*$ is lost. Nevertheless, our potential function analysis can still be adapted to the VY algorithm to obtain the same $\Omega(n/\log n)$ improvement on the iteration complexity bound; see Section 4.1 for more details.

1.1 Related Work

Since the seminal works of Karmarkar [17] and Renegar [34], there has been a tremendous amount of work on speeding up and improving interior-point methods. In contrast to the present work, the focus of these works has mostly been to improve complexity of *approximately solving* LPs. Progress has taken many forms, such as the development of novel barrier methods, such Vaidya's volumetric barrier [46] and the recent entropic barrier of Bubeck and Eldan [3] and the weighted log-barrier of Lee and Sidford [22, 24], together with new path following techniques, such as the predictor-corrector framework [28, 29], as well as advances in fast linear system solving [23, 39]. For this last line, there has been substantial progress in improving IPM by amortizing the cost of the iterative updates, and working with approximate computations, see e.g. [5, 34, 46, 47]. Very recently, Cohen, Lee and Song [5] developed a new inverse maintenance scheme to get a randomized $\tilde{O}(n^{2.37}\log(1/\varepsilon))$-time algorithm for ε-approximate LP, which was derandomized by van den Brand [47]. For special classes of LP such as network flow problems, fast algorithms have been obtained by using fast Laplacian solvers, see e.g. [6, 25]. Given the progress above, we believe it to be an interesting problem to understand to what extent these new numerical techniques can be applied to speed up LLS computations, though we expect that such computations will require very high precision. We note that no attempt has been made in the present work to optimize the complexity of the linear algebra.

Ho and Tunçel [14] showed how to extend Tardos' framework to the real model of computation (i.e., to non-integral A), providing a blackbox alternative to the VY algorithm. The numerical complexity of the LPs arising in their reduction is controlled by the minimum and maximum subdeterminant of A restricted to non-singular submatrices and the minimum non-zero slack of any basic primal or dual solution over a certain grid of right hand sides and objectives.

With regard to LLS algorithms, the original VY algorithm required explicit knowledge of $\bar{\chi}_A$ to implement their layering algorithm. [27] showed that this could be avoided by computing all LLS steps associated with n candidate partitions and picking the best one. In particular, they showed that all such LLS steps can be computed in $O(m^2 n)$ time. [31] gave an alternate approach which computes a LLS partition directly from the coefficients of the AS step. We note that these methods crucially rely on the variable ordering, and hence are not scaling invariant. Kitahara and Tsuchiya [20], gave a 2-layer LLS step which achieves a running time depending only on $\bar{\chi}_A^*$ and right-hand side b, but with no dependence on the objective, assuming the primal feasible region is bounded.

A series of papers have studied the central path from a differential geometry perspective. Monteiro and Tsuchiya [30] showed that a curvature integral of the central path, first introduced by Sonnevend, Stoer, and Zhao [38], is in fact upper bounded by $O(n^{3.5}\log(\bar{\chi}_A^*+n))$. This has been extended to SDP and symmetric cone programming [16], and also studied in the context of information geometry [15].

Circuits have appeared in several papers on linear and integer optimization (see [8] and its references). The idea of using circuits within the context of LP algorithms also appears in [7]. They develop an augmentation framework for LP (as well ILP) and show that a simplex-like algorithm which takes steps according to the "best circuit" direction achieves linear convergence, though these steps are hard to compute.

Our algorithm makes progress towards strongly polynomial solvability of LP, by improving the dependence $\text{poly}(n, m, \log \bar{\chi})$ to $\text{poly}(n, m, \log \bar{\chi}^*)$. However, in a remarkable recent paper, Allamigeon et al. [2] have shown, using tools from tropical geometry, that path-following methods for the standard logarithmic barrier *cannot* be strongly polynomial. In particular, they give a parametrized family of instances, where, for sufficiently large parameter values, any sequence of iterations following the central path must be of exponential length—thus, $\bar{\chi}^*$ will be doubly exponential. We note that it is unclear whether their instance is robust to changing the barrier method itself; e.g., the weighted log-barrier [22].

1.2 Organization

Section 2 begins with the necessary background on the condition measures $\bar{\chi}_A$ and $\bar{\chi}_A^*$. It culminates in the approximate $\bar{\chi}_A^*$ rescaling and $\bar{\chi}_A$ approximation algorithm. This algorithm relies upon the circuit imbalance measure in Section 2.1, the min-max characterization in Section 2.2, and a circuit finding algorithm in Section 2.3.

In Section 3, we develop our scaling invariant interior-point method. Interior-point preliminaries are given in Section 3.1, the layered least squares step is explained in Section 3.3, our scaling invariant layering algorithm is given in Section 3.4, and lastly, our overall algorithm is given in Section 3.5.

In Section 4, we describe the potential function proof for the improved iteration bound. Section 4.1 shows that our argument also leads to a factor $\Omega(n/\log n)$ improvement in the iteration complexity bound of the VY algorithm. Finally, in Section 5, we discuss the initialization of our interior-point method.

Proofs can be found in the full version of this paper, accessible at https://arxiv.org/abs/1912.06252.

2 FINDING AN APPROXIMATELY OPTIMAL RESCALING

Notation. Our notation will largely follow [31, 32]. We let $\mathbb{R}_{++}$ denote the set of positive reals, and $\mathbb{R}_+$ the set of nonnegative reals. For $n \in \mathbb{N}$, we let $[n] = \{1, 2, \ldots, n\}$. Let $e^i \in \mathbb{R}^n$ denote the ith unit vector, and $e \in \mathbb{R}^n$ the all 1s vector. For a vector $x \in \mathbb{R}^n$, we let $\text{Diag}(x) \in \mathbb{R}^{n\times n}$ denote the diagonal matrix with x on the diagonal. We let $\mathbf{D}$ denote the set of all positive $n \times n$ diagonal matrices. For $x, y \in \mathbb{R}^n$, we use the notation $xy \in \mathbb{R}^n$ to denote $xy = \text{Diag}(x)y = (x_i y_i)_{i\in[n]}$. The scalar product of the two vectors is denoted as $x^\top y$. For $p \in \mathbb{Q}$, we also use the notation x^p to denote the vector $(x_i^p)_{i\in[n]}$. Similarly, for $x, y \in \mathbb{R}^n$, we let x/y denote the vector $(x_i/y_i)_{i\in[n]}$. We denote the support of a vector $x \in \mathbb{R}^n$ by $\text{supp}(x) = \{i \in [n] : x_i \neq 0\}$.

For an index subset $I \subseteq [n]$, we use $\pi_I : \mathbb{R}^n \to \mathbb{R}^I$ for the coordinate projection. That is, $\pi_I(x) = x_I$, and for a subset $S \subseteq \mathbb{R}^n$, $\pi_I(S) = \{x_I : x \in S\}$. We let $\mathbb{R}^n_I = \{x \in \mathbb{R}^n : x_{[n]\setminus I} = 0\}$.

For a matrix $B \in \mathbb{R}^{n\times k}$, $I \subset [n]$ and $J \subset [k]$ we let $B_{I,J}$ denote the submatrix of B restricted to the set of rows in I and columns in J. We also use $B_{I,\bullet} = B_{I,[k]}$ and $B_J = B_{\bullet,J} = B_{[n],J}$. We let $B^\dagger \in \mathbb{R}^{k\times n}$ denote the pseudo-inverse of B.

We let $\text{Ker}(A)$ denote the kernel of the matrix $A \subseteq \mathbb{R}^{m\times n}$. Throughout, we assume that the matrix A in (LP) has full row rank, and that $n \geq 3$.

Subspace Formulation. Throughout the paper, we let $W = \text{Ker}(A) \subseteq \mathbb{R}^n$ denote the kernel of the matrix A. Using this notation, (LP) can be written in the form

$$\begin{array}{ll} \min c^\top x & \max d^\top (c - s) \\ x \in W + d & s \in W^\perp + c \\ x \geq 0, & s \geq 0, \end{array} \tag{2}$$

where $d \in \mathbb{R}^n$ satisfies $Ad = b$.

The condition number $\bar{\chi}$. The condition number $\bar{\chi}_A$ is defined as

$$\begin{aligned} \bar{\chi}_A &= \sup\left\{\left\|A^\top \left(ADA^\top\right)^{-1} AD\right\| : D \in \mathbf{D}\right\} \\ &= \sup\left\{\frac{\left\|A^\top y\right\|}{\|p\|} : y \text{ minimizes } \left\|D^{1/2}(A^\top y - p)\right\| \right. \\ &\qquad\qquad \left. \text{for some } 0 \neq p \in \mathbb{R}^n \text{ and } D \in \mathbf{D}\right\}. \end{aligned} \tag{3}$$

This condition number was first studied by Dikin [9], Stewart [40], and Todd [43], among others, and plays a key role in the analysis of the Vavasis-Ye interior point method [49]. There is an extensive literature on the properties and applications of $\bar{\chi}_A$, as well as its relations to other condition numbers. We refer the reader to the papers [14, 31, 49] for further results and references.

It is important to note that $\bar{\chi}_A$ only depends on the subspace $W = \text{Ker}(A)$. Hence, we can also write $\bar{\chi}_W$ for a subspace $W \subseteq \mathbb{R}^n$, defined to be equal to $\bar{\chi}_A$ for some matrix $A \in \mathbb{R}^{k\times n}$ with $W = \text{Ker}(A)$. We will use the notations $\bar{\chi}_A$ and $\bar{\chi}_W$ interchangeably.

The next lemma summarizes some important known properties of $\bar{\chi}_A$.

Proposition 2.1. *Let $A \in \mathbb{R}^{m\times n}$ with full row rank and $W = \text{Ker}(A)$.*

(i) If the entries of A are all integers, then $\bar{\chi}_A$ is bounded by $2^{O(L_A)}$, where L_A is the input bit length of A.
(ii) $\bar{\chi}_A = \max\{\|B^{-1}A\| : B$ non-singular $m \times m$-submatrix of $A\}$.
(iii) $\bar{\chi}_W = \bar{\chi}_{W^\perp}$.

Proof. Part (i) was proved in [49, Lemma 24]. For part (ii), see [44, Theorem 1] and [49, Lemma 3]. The duality statement (iii) was shown in [13]. □

In Proposition 3.8, we will also give another proof of (iii). We now define the lifting map, a key operation in this paper, and explain its connection to $\bar{\chi}_A$.

Definition 2.2. *Let us define the* lifting map $L_I^W : \pi_I(W) \to W$ *by*

$$L_I^W(p) = \arg\min\{\|z\| : z_I = p, z \in W\}.$$

Note that L_I^W is the unique linear map from $\pi_I(W)$ to W such that $L_I^W(p)_I = p$ and $L_I^W(p)$ is orthogonal to $W \cap \mathbb{R}^n_{[n]\setminus I}$.

We have following characterization. This will be the most suitable characterization of $\bar{\chi}_W$ for our purposes.

Proposition 2.3. *For a linear subspace $W \subseteq \mathbb{R}^n$,*

$$\bar{\chi}_W = \max\left\{\|L_I^W\| : I \subseteq [n], I \neq \emptyset\right\}.$$

The following notation will be convenient for our algorithm. For a subspace $W \subseteq \mathbb{R}^n$ and an index set $I \subseteq [n]$, if $\pi_I(W) \neq \{0\}$, we define the *lifting score*

$$\ell^W(I) := \sqrt{\|L_I^W\|^2 - 1}. \tag{4}$$

Otherwise, we define $\ell^W(I) = 0$. This means that for any $z \in \pi_I(W)$ and $x = L_I^W(z)$, $\|x_{[n]\setminus I}\| \leq \ell^W(I)\|z\|$.

The Condition Number $\bar{\chi}^*_A$. For every $D \in \mathbf{D}$, we can consider the condition number $\bar{\chi}_{WD} = \bar{\chi}_{AD^{-1}}$. We let

$$\bar{\chi}^*_W = \bar{\chi}^*_A = \inf\{\bar{\chi}_{WD} : D \in \mathbf{D}\}$$

denote the best possible value of $\bar{\chi}$ that can be attained by rescaling the coordinates of W. The main result of this section is the following theorem.

Theorem 2.4. *There is an $O(n^2m^2 + n^3)$ time algorithm that for any matrix $A \in \mathbb{R}^{m\times n}$ computes a t such that*

$$t \leq \bar{\chi}_W \leq tn(\bar{\chi}^*_W)^2$$

and a $D \in \mathbf{D}$ such that

$$\bar{\chi}^*_W \leq \bar{\chi}_{WD} \leq n(\bar{\chi}^*_W)^3.$$

2.1 The Circuit Imbalance Measure

We next introduce the *circuit imbalance measure*, a more combinatorial condition number, and show that it gives a good proxy to $\bar{\chi}_A$.

Definition 2.5. *For a linear subspace $W \subseteq \mathbb{R}^n$ and a matrix A such that $W = \ker(A)$, a circuit is an inclusion-wise minimal dependent set of columns of A. Equivalently, a circuit is a set $C \subseteq [n]$ such that $W \cap \mathbb{R}^n_C$ is one-dimensional and that no strict subset of C has this property. Any circuit is associated with a vector $g \in W$ with inclusion-wise minimal support. The set of circuits of W is denoted $\mathcal{C}_W$.*

Note that these are also known as the circuits in the linear matroid associated with A.

Definition 2.6. *For a circuit $C \in \mathcal{C}_W$, let $g^C \in W$ be such that $\mathrm{supp}(g^C) = C$. For $i, j \in C$, we let*

$$\kappa_{ij}^W(C) = \frac{\left|g_j^C\right|}{\left|g_i^C\right|}. \tag{5}$$

For any $i, j \in [n]$, we define the circuit ratio *as the maximum of $\kappa_{ij}^W(C)$ over all choices of the circuit C:*

$$\kappa_{ij}^W = \max\left\{\kappa_{ij}^W(C) : C \in \mathcal{C}_W, i, j \in C\right\}. \tag{6}$$

By convention we set $\kappa_{ij}^W = 0$ if there is no circuit supporting i and j. Further, we define the circuit imbalance measure *as*

$$\kappa_W = \max\left\{\kappa_{ij}^W : i, j \in [n]\right\}.$$

Minimizing over all coordinate rescalings, we define

$$\kappa^*_W = \min\{\kappa_{WD} : D \in \mathbf{D}\}.$$

We omit the index W whenever it is clear from context. In such cases, for $D = \mathrm{Diag}(d) \in \mathbf{D}$, we write $\kappa_{ij}^d = \kappa_{ij}^{WD}$ and $\kappa^d = \kappa_W^d = \kappa_{WD}$.

We want to remark that a priori it is not clear that κ^*_W is well-defined. Theorem 2.11 will show that the minimum of $\{\kappa_{WD} : D \in \mathbf{D}\}$ is indeed attained. Observe that $\kappa_{ij}^W(C)$ does not depend on the choice of g, since there is only a single choice up to scalar multiplication.

The circuit ratio, as well as the circuit imbalance measure, are self-dual.

Lemma 2.7. *For any subspace $W \subseteq \mathbb{R}^n$ and $i, j \in [n]$, $\kappa_{ij}^W = \kappa_{ji}^{W^\perp}$.*

The next theorem relates the circuit imbalance κ_W and the condition number $\bar{\chi}_W$. The lower bound was already proven in [48], and the upper bound is new, as far as we know.

Theorem 2.8. *For a linear subspace $W \subseteq \mathbb{R}^n$,*

$$\sqrt{1 + (\kappa_W)^2} \leq \bar{\chi}_W \leq \sqrt{1 + (n\kappa_W)^2}.$$

The next lemmas are key in the proof of Theorem 2.8, and will also be used later in the algorithm.

Lemma 2.9. *For $i \in I \subset [n]$ with $e^i \in \pi_I(W)$, let $z = L_I^W(e^i)$. Then for any $j \in \mathrm{supp}(z)$ we have $\kappa_{ij}^W \geq |z_j|$.*

For the next lemma, recall the definition of the lifting score $\ell^W(I)$ from (4).

Lemma 2.10. *There exists an algorithm Verify-Lift that, for a linear subspace $W \subseteq \mathbb{R}^n$ and an index set $I \subseteq [n]$, can efficiently identify $i \in I, j \in [n] \setminus I$ and $t \leq \kappa_{ij}^W$ such that $\ell^W(I) \leq nt$.*

Our LLS algorithm in Section 3 will use the subroutine described in Lemma 2.10. For a subspace $W \subseteq \mathbb{R}^n$, an index set $I \subseteq [n]$, and a threshold $\theta > 0$, the algorithm Verify-Lift(W, I, θ) outputs either of the answers *'pass'* or *'fail'*. If the answer is *'pass'*, then it is guaranteed that $\ell^W(I) \leq \theta$. If the answer is *'fail'*, then a pair of indices $i \in I, j \in [n] \setminus I$, and a bound t are returned, such that $\theta/n \leq t \leq \kappa_{i,j}^W$.

To implement Verify-Lift, we first need to select a minimal $I' \subset I$ such that $\dim(\pi_{I'}(W)) = \dim(\pi_I(W))$. This can be found by computing a matrix $M \in \mathbb{R}^{(n-m)\times n}$ such that $\text{range}(M) = W$, and selecting a maximal number of linearly independent columns of M_I. Then, we compute the matrix $B \in \mathbb{R}^{([n]\setminus I)\times I'}$ that implements the transformation $[L^W_{I'}]_{[n]\setminus I} : \pi_{I'}(W) \to \pi_{[n]\setminus I}(W)$. The algorithm returns the pair (i, j) corresponding to the entry maximizing $|B_{ji}|$.

2.2 A Min-Max Theorem on κ^*_W

We next provide a combinatorial min-max characterization on κ^*_W. Consider the *circuit ratio digraph* $G = ([n], E)$ on the node set $[n]$ where $(i, j) \in E$ if $\kappa(i, j) > 0$, that is, there exists a circuit $C \in \mathcal{C}$ with $i, j \in C$. An edge $(i, j) \in E$ is said to have weight $\kappa_{ij} = \kappa^W_{ij}$. (Note that $(i, j) \in E$ if and only if $(j, i) \in E$, but the weight of these two edges can be different.)

Let H be a *cycle* in G, that is, a sequence of points $i_1, i_2, \ldots, i_k, i_{k+1} = i_1$. We use $|H| = k$ to denote the length of the cycle. (In our terminology, 'cycles' always refer to objects in G, whereas 'circuits' refer to the minimum supports in $\text{Ker}(A)$.)

We use the notation $\kappa(H) = \kappa_W(H) = \prod_{j=1}^k \kappa^W_{i_j i_{j+1}}$. For a vector $d \in \mathbb{R}^n_{++}$, we let $\kappa^d(H) = \kappa^d_W(H) = \kappa_{WD}(H)$ for $D = \text{Diag}(d)$. A simple but important observation is that such a rescaling does not change the value associated with the cycle, that is,

$$\kappa^d_W(H) = \kappa_W(H) \quad \forall d \in \mathbb{R}^n_{++} \quad \text{for any cycle } H \text{ in } G\,. \tag{7}$$

We are ready to formulate our theorem.

Theorem 2.11. *For a subspace $W \subset \mathbb{R}^n$, we have*

$$\kappa^*_W = \min_{d>0} \kappa^d_W = \max\left\{\kappa_W(H)^{1/|H|} : H \text{ is a cycle in } G\right\}\,.$$

The following example shows that $\kappa^* \le \bar{\chi}^*$ can be arbitrarily big.

Example 2.12. *Take $W = \text{span}((0, 1, 1, M), (1, 0, M, 1))$, where $M > 0$. Then $\{2, 3, 4\}$ and $\{1, 3, 4\}$ are circuits with $\kappa^W_{34}(\{2, 3, 4\}) = M$ and $\kappa^W_{43}(\{1, 3, 4\}) = M$. Hence, by Theorem 2.11, we see that $\kappa^* \ge M$.*

The following corollary of Theorem 2.11 particularly useful. It asserts that any arbitrary circuit containing i and j yields a $(\kappa^*)^2$ approximation to κ_{ij}.

Corollary 2.13. *We are given a linear subspace $W \subseteq \mathbb{R}^n$ and $i, j \in [n]$, $i \ne j$, and a circuit $C \in \mathcal{C}_W$ with $i, j \in C$. Let $g \in W$ be the corresponding vector with $\text{supp}(g) = C$. Then,*

$$\frac{\kappa^W_{ij}}{\left(\kappa^*_W\right)^2} \le \frac{|g_j|}{|g_i|} \le \kappa^W_{ij}\,.$$

2.3 Finding Circuits: A Detour in Matroid Theory

We now show how to efficiently obtain a family $\hat{C} \subseteq \mathcal{C}_W$ such that for any $i, j \in [n]$, $\hat{C}$ includes a circuit containing both i and j, provided there exists such a circuit.

We need some simple concepts and results from matroid theory. We refer the reader to [37, Chapter 39] or [10, Chapter 5] for definitions and background. Let $\mathcal{M} = ([n], \mathcal{I})$ be a matroid on ground set $[n]$ with independent sets $\mathcal{I} \subseteq 2^V$. The rank $\text{rk}(S)$ of a set $S \subseteq 2^{[n]}$ is the maximum size of an independent set contained in S. The maximal independent sets are called *bases*. All bases have the same cardinality $\text{rk}([n])$.

For the matrix $A \in \mathbb{R}^{m\times n}$, we will work with the linear matroid $\mathcal{M}(A) = ([n], \mathcal{I}(A))$, where a subset $I \subseteq [n]$ is independent if the columns $\{A_i : i \in I\}$ are linearly independent. Note that $\text{rk}([n]) = m$ under the assumption that A has full row rank.

The *circuits* of the matroid are the inclusion-wise minimal non-independent sets. Let $I \in \mathcal{I}$ be an independent set, and $i \in [n] \setminus I$ such that $I \cup \{i\} \notin \mathcal{I}$. Then, there exists a unique circuit $C(I, i) \subseteq I \cup \{i\}$ that is called the *fundamental circuit* of i with respect to I. Note that $i \in C(I, i)$.

The matroid $\mathcal{M}$ is *separable*, if the ground set $[n]$ can be partitioned to two nonempty subsets $[n] = S \cup T$ such that $I \in \mathcal{I}$ if and only if $I \cap S, I \cap T \in \mathcal{I}$. In this case, the matroid is the direct sum of its restrictions to S and T. In particular, every circuit is fully contained in S or in T.

For the linear matroid $\mathcal{M}(A)$, separability means that $\text{Ker}(A) = \text{Ker}(A_S) \oplus \text{Ker}(A_T)$. In this case, solving (LP) can be decomposed into two subproblems, restricted to the columns in A_S and in A_T, and $\bar{\chi}_A = \max\{\bar{\chi}_{A_S}, \bar{\chi}_{A_T}\}$.

Hence, we can focus on *non-separable* matroids. The following characterization is well-known, see e.g. [10, Theorems 5.2.5, 5.2.7–5.2.9]. For a hypergraph $H = ([n], \mathcal{E})$, we define the underlying graph $H_G = ([n], E)$ such that $(i, j) \in E$ if there is a hyperedge $S \in \mathcal{E}$ with $i, j \in S$. That is, we add a clique corresponding to each hyperedge. The hypergraph is called *connected* if the underlying graph $G = ([n], E)$ is connected.

Proposition 2.14. *For a matroid $\mathcal{M} = ([n], \mathcal{I})$, the following are equivalent:*

(i) $\mathcal{M}$ is non-separable.
(ii) The hypergraph of the circuits is connected.
(iii) For any base B of $\mathcal{M}$, the hypergraph formed by the fundamental circuits $\mathcal{C}^B = \{C(B, i) : i \in [n] \setminus B\}$ is connected.
(iv) For any $i, j \in [n]$, there exists a circuit containing i and j.

We are ready to describe the algorithm that will be used to obtain lower bounds on all κ_{ij} values. For a matrix $A \in \mathbb{R}^{m\times n}$, we let Find-Circuits($A$) denote the subroutine described in the lemma for the linear matroid $\mathcal{M}(A)$.

Theorem 2.15. *Given $A \in \mathbb{R}^{m\times n}$, there exists an $O(n^2m^2)$ time algorithm Find-Circuits(A) that obtains a decomposition of $\mathcal{M}(A)$ to a direct sum of non-separable linear matroids, and returns a family $\hat{\mathcal{C}}$ of circuits such that if i and j are in the same non-separable component, then there exists a circuit in $\hat{\mathcal{C}}$ containing both i and j. Further, for each $i \ne j$ in the same component, the algorithm returns a value $\hat{\kappa}_{ij}$ as the the maximum of $|g_j/g_i|$ such that $g \in W$, $\text{supp}(g) = C$ for some $C \in \hat{\mathcal{C}}$ containing i and j. For these values, $\hat{\kappa}_{ij} \le \kappa_{ij} \le (\kappa^*)^2\hat{\kappa}_{ij}$.*

The rescaling algorithm described in Theorem 2.4 functions by first running Find-Circuits(A) to approximate the circuit ratio graph. Taking $t = \sqrt{1 + \max_{(i,j)\in E} \hat{\kappa}^2_{ij}}$ approximates $\bar{\chi}_A$ per Theorem 2.8. A maximum-mean cycle computation allows us to compute the a suitable rescaling to approximately minimize κ^d_W in $O(n^3)$ time (see e.g. [1, Theorem 5.8]).

3 A SCALING-INVARIANT LAYERED LEAST SQUARES INTERIOR-POINT ALGORITHM

3.1 Preliminaries on Interior-Point Methods

In this section, we introduce the standard definitions, concepts and results from the interior-point literature that will be required for our algorithm. We consider an LP problem in the form (LP), or equivalently, in the subspace form (2) for $W = \mathrm{Ker}(A)$. We let

$$\mathcal{P}^{++} = \{x \in \mathbb{R}^n : Ax = b, x > 0\}$$
$$\mathcal{D}^{++} = \{(y,s) \in \mathbb{R}^{m+n} : A^\top y + s = c, s > 0\}\,.$$

Recall the *central path* defined in (CP), with $w(\mu) = (x(\mu), y(\mu), s(\mu))$ denoting the central path point corresponding to $\mu > 0$. We let $w^* = (x^*, y^*, s^*)$ denote the primal and dual optimal solutions to (LP) that correspond to the limit of the central path for $\mu \to 0$.

For a point $w = (x,y,s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$, the *normalized duality gap* is $\mu(w) = x^\top s/n$.

The *ℓ_2-neighborhood of the central path with opening $\beta > 0$* is the set

$$\mathcal{N}(\beta) = \left\{ w \in \mathcal{P}^{++} \times \mathcal{D}^{++} : \left\| \frac{xs}{\mu(w)} - e \right\| \le \beta \right\}$$

Throughout the paper, we will assume β is chosen from $(0, 1/4]$; in Algorithm 2 we use the value $\beta = 1/8$. The following proposition gives a bound on the distance between w and $w(\mu)$ if $w \in \mathcal{N}(\beta)$. See e.g. [12, Lemma 5.4].

Proposition 3.1. *Let $w = (x,y,s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$ and $\mu = \mu(w)$, and consider the central path point $w(\mu) = (x(\mu), y(\mu), s(\mu))$. For each $i \in [n]$,*

$$\frac{x_i}{1+2\beta} \le \frac{1-2\beta}{1-\beta} \cdot x_i \le x_i(\mu) \le \frac{x_i}{1-\beta}\,, \quad \text{and}$$
$$\frac{s_i}{1+2\beta} \le \frac{1-2\beta}{1-\beta} \cdot s_i \le s_i(\mu) \le \frac{s_i}{1-\beta}\,.$$

We will often use the following immediate corollary.

Corollary 3.2. *Let $w = (x,y,s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$, and $\mu = \mu(w)$. Then for each $i \in [n]$*

$$(1-\beta)\sqrt{\mu} \le \sqrt{s_i x_i} \le (1+2\beta)\sqrt{\mu}\,.$$

A key property of the central path is *"near monotonicity"*, formulated in the following lemma, see [49, Lemma 16].

Lemma 3.3. *Let $w = (x,y,s)$ be a central path point for μ and $w' = (x',y',s')$ be a central path point for $\mu' \le \mu$. Then $\|x'/x + s'/s\|_\infty \le n$. Further, for the optimal solution $w^* = (x^*, y^*, s^*)$ corresponding to the central path limit $\mu \to 0$, we have $\|x^*/x\|_1 + \|s^*/s\|_1 = n$.*

3.2 Predictor and Corrector Steps

Given $w = (x,y,s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$, the search directions commonly used in interior-point methods are obtained as the solution $(\Delta x, \Delta y, \Delta s)$ to the following linear system for some $\sigma \in [0,1]$.

$$A\Delta x = 0 \tag{8}$$
$$A^\top \Delta y + \Delta s = 0 \tag{9}$$
$$s\Delta x + x\Delta s = \sigma\mu e - xs \tag{10}$$

Predictor-corrector methods, such as the Mizuno-Todd-Ye Predictor-Corrector (MTY P-C) algorithm [29], alternate between two types of steps. In *predictor steps*, we use $\sigma = 0$. This direction is also called the *affine scaling direction*, and will be denoted as $\Delta w^{\mathrm{a}} = (\Delta x^{\mathrm{a}}, \Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$ throughout. In *corrector steps*, we use $\sigma = 1$. This gives the *centrality direction*, denoted as $\Delta w^{\mathrm{c}} = (\Delta x^{\mathrm{c}}, \Delta y^{\mathrm{c}}, \Delta s^{\mathrm{c}})$.

In the predictor steps, we make progress along the central path. Given the search direction on the current iterate $w = (x,y,s) \in \mathcal{N}(\beta)$, the step-length is chosen maximal such that we remain in $\mathcal{N}(2\beta)$, i.e.

$$\alpha^{\mathrm{a}} := \sup\{\alpha \in [0,1] : \forall \alpha' \in [0,\alpha] : w + \alpha' \Delta w^{\mathrm{a}} \in \mathcal{N}(2\beta)\}.$$

Thus, we obtain a point $w^+ = w + \alpha^{\mathrm{a}} \Delta w^{\mathrm{a}} \in \mathcal{N}(2\beta)$. The corrector step finds a next iterate $w^{\mathrm{c}} = w^{\mathrm{a}} + \Delta w^{\mathrm{c}}$, where Δw^{c} is the centrality direction computed at w^{a}. The next proposition summarizes well-known properties, see e.g. [51, Section 4.5.1].

Proposition 3.4. *Let $w = (x,y,s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$.*

(i) For the affine scaling step, we have $\mu(w^+) = (1-\alpha)\mu(w)$.

(ii) The affine scaling step-length is

$$\alpha^{\mathrm{a}} \ge \max\left\{ \frac{\beta}{\sqrt{n}}, 1 - \frac{\|\Delta x^{\mathrm{a}} \Delta s^{\mathrm{a}}\|}{\beta \mu(w)} \right\}.$$

(iii) For $w^+ \in \mathcal{N}(2\beta)$, and $w^{\mathrm{c}} = w^+ + \Delta w^{\mathrm{c}}$, we have $\mu(w^{\mathrm{c}}) = \mu(w^+)$ and $w^{\mathrm{c}} \in \mathcal{N}(\beta)$.

(iv) After a sequence of $O(\sqrt{n}t)$ predictor and corrector steps, we obtain an iterate $w' = (x',y',s') \in \mathcal{N}(\beta)$ such that $\mu(w') \le \mu(w)/2^t$.

Minimum norm viewpoint and residuals. For any point $w = (x,y,s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$ we define

$$\delta = \delta(w) = s^{1/2} x^{-1/2} \in \mathbb{R}^n. \tag{11}$$

With this notation, we can write (10) in the form

$$\delta \Delta x + \delta^{-1} \Delta s = -s^{1/2} x^{1/2}\,. \tag{12}$$

From Proposition 3.1, we see that if $w \in \mathcal{N}(\beta)$, and $\mu = \mu(w)$, then for each $i \in [n]$,

$$\sqrt{1-2\beta} \cdot \delta_i(w) \le \delta_i(w(\mu)) \le \frac{1}{\sqrt{1-2\beta}} \cdot \delta_i(w)\,. \tag{13}$$

The matrix $\mathrm{Diag}(\delta(w))$ will be often used for rescaling in the algorithm. That is, for the current iterate $w = (x,y,s)$ in the interior-point method, we will perform projections in the space $W\,\mathrm{Diag}(\delta(w))$. To simplify notation, for $\delta = \delta(w)$, we use L_I^δ and κ_{ij}^δ as shorthands for $L_I^{W\,\mathrm{Diag}(\delta)}$ and $\kappa_{ij}^{W\,\mathrm{Diag}(\delta)}$. The subspace $W = \mathrm{Ker}(A)$ will be fixed throughout.

It is easy to see from the optimality conditions that the components of the affine scaling direction $(\Delta x^{\mathrm{a}}, \Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$ are the optimal solutions of the following minimum-norm problems.

$$\Delta x^{\mathrm{a}} = \operatorname*{arg\,min}_{\Delta x \in \mathbb{R}^n} \{\|\delta(x + \Delta x)\|^2 : A\Delta x = 0\}$$
$$(\Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}}) = \operatorname*{arg\,min}_{(\Delta y, \Delta s) \in \mathbb{R}^m \times \mathbb{R}^n} \{\|\delta^{-1}(s + \Delta s)\|^2 : A^\top \Delta y + \Delta s = 0\} \tag{14}$$

Following [32], for a search direction $\Delta w = (\Delta x, \Delta y, \Delta s)$, we define the *residuals* as

$$Rx := \frac{\delta(x + \Delta x)}{\sqrt{\mu}}, \qquad Rs := \frac{\delta^{-1}(s + \Delta s)}{\sqrt{\mu}}\,. \tag{15}$$

Hence, the primal affine scaling direction Δx^{a} is the one that minimizes the ℓ_2-norm of the primal residual Rx, and the dual affine scaling direction $(\Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$ minimizes the ℓ_2-norm of the dual residual Rs. The next lemma summarizes simple properties of the residuals, see [32].

Lemma 3.5. *For $\beta \in (0, 1/4]$ such that $w = (x, y, s) \in \mathcal{N}(\beta)$ and the affine scaling direction $\Delta w = (\Delta x^{\mathrm{a}}, \Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$, we have*

(i)

$$Rx^{\mathrm{a}} Rs^{\mathrm{a}} = \frac{\Delta x^{\mathrm{a}} \Delta s^{\mathrm{a}}}{\mu}, \quad Rx^{\mathrm{a}} + Rs^{\mathrm{a}} = \frac{x^{1/2} s^{1/2}}{\sqrt{\mu}}, \tag{16}$$

(ii)

$$\|Rx^{\mathrm{a}}\|^2 + \|Rs^{\mathrm{a}}\|^2 = n,$$

(iii) We have $\|Rx^{\mathrm{a}}\|, \|Rs^{\mathrm{a}}\| \le \sqrt{n}$, and for each $i \in [n]$, we have $\max\{|Rx^{\mathrm{a}}_i|, |Rs^{\mathrm{a}}_i|\} \ge \frac{1}{2}(1-\beta)$.

(iv)

$$Rx^{\mathrm{a}} = -\frac{1}{\sqrt{\mu}} \delta^{-1} \Delta s^{\mathrm{a}}, \quad Rs^{\mathrm{a}} = -\frac{1}{\sqrt{\mu}} \delta \Delta x^{\mathrm{a}}.$$

For a subset $I \subset [n]$, we define

$$\varepsilon^{\mathrm{a}}_I(w) := \max_{i \in I} \min\{|Rx^{\mathrm{a}}_i|, |Rs^{\mathrm{a}}_i|\}, \quad \text{and} \quad \varepsilon^{\mathrm{a}}(w) := \varepsilon^{\mathrm{a}}_{[n]}(w). \tag{17}$$

The next claim shows that for the affine scaling direction, a small $\varepsilon(w)$ yields a long step; see [32, Lemma 2.5].

Lemma 3.6. *Let $w = (x, y, s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$. Then for the affine scaling step, we have*

$$\frac{\mu(w + \alpha^{\mathrm{a}} \Delta w^{\mathrm{a}})}{\mu(w)} \le \min\left\{1 - \frac{\beta}{\sqrt{n}}, \frac{\sqrt{n}\varepsilon^{\mathrm{a}}(w)}{\beta}\right\}.$$

3.3 Layered Least Squares Direction

Let $\mathcal{J} = (J_1, J_2, \ldots, J_p)$ be an *ordered partition* of $[n]$.[1] For $k \in [p]$, we use the notations $J_{<k} := J_1 \cup \ldots \cup J_{k-1}$ and $J_{>k} := J_{k+1} \cup \ldots \cup J_p$, and similarly $J_{\le k}$ and $J_{\ge k}$. We will also refer to the sets J_k as *layers*, and $\mathcal{J}$ as a *layering*. Layers with lower indices will be referred to as 'higher' layers.

Given $w = (x, y, s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$, and the layering $\mathcal{J}$, the *layered-least-squares (LLS) direction* is defined as follows. For the primal direction, we proceed backwards, with $k = p, p-1, \ldots, 1$. Assume the components on the lower layers $\Delta x^{\mathrm{ll}}_{J_{>k}}$ have already been determined. We define the components in J_k as the coordinate projection $\Delta x^{\mathrm{ll}}_{J_k} = \pi_{J_k}(X_k)$, where the affine subspace X_k is defined as the set of minimizers

$$X_k := \arg\min_{\Delta x \in \mathbb{R}^n} \{\|\delta_{J_k}(x_{J_k} + \Delta x_{J_k})\|^2 : A\Delta x = 0, \Delta x_{J_{>k}} = \Delta x^{\mathrm{ll}}_{J_{>k}}\}. \tag{18}$$

The dual direction Δs^{ll} is determined in the forward order of the layers $k = 1, 2, \ldots, p$. Assume we already fixed the components $\Delta s^{\mathrm{ll}}_{J_{<k}}$ on the higher layers. Then, $\Delta s^{\mathrm{ll}}_{J_k} = \pi_{J_k}(S_k)$ for

$$S_k = \arg\min_{\Delta s \in \mathbb{R}^n} \{\|\delta^{-1}_{J_k}(s_{J_k} + \Delta s_{J_k})\|^2 : \exists y \in \mathbb{R}^m, A^\top \Delta y + \Delta s = 0, \Delta s_{J_{<k}} = \Delta s^{\mathrm{ll}}_{J_{<k}}\}. \tag{19}$$

[1] In contrast to how ordered partitions were defined in [32], we use the term *ordered* only to the p-tuple $(J_1, \ldots, J_p)$, which is to be viewed independently of δ.

The component Δy^{ll} is obtained as the optimal Δy for the final layer $k = p$. We use the notation Rx^{ll} and $\varepsilon^{\mathrm{ll}}(w)$ analogously to the affine scaling direction. This search direction was first introduced in [49].

The affine scaling direction is a special case for the single element partition. In this case, the definitions (18) and (19) coincide with those in (14).

3.3.1 A Linear System Viewpoint. We now present an equivalent definition of the LLS step, generalizing the linear system (9)-(10). We use the subspace notation. With this notation, (9)-(10) for the affine scaling direction can be written as

$$s\Delta x^{\mathrm{a}} + x\Delta s^{\mathrm{a}} = -xs, \quad \Delta x^{\mathrm{a}} \in W, \quad \text{and} \quad \Delta s^{\mathrm{a}} \in W^\perp. \tag{20}$$

Recall that (20) is equivalent to $\delta\Delta x^{\mathrm{a}} + \delta^{-1}\Delta s^{\mathrm{a}} = -x^{1/2}s^{1/2}$.

Given the layering $\mathcal{J}$ and $w = (x, y, s)$, for each $k \in [p]$ we define the subspaces

$$W_{\mathcal{J},k} := \{x_{J_k} : x \in W, x_{J_{>k}} = 0\}$$
$$W^\perp_{\mathcal{J},k} := \{x_{J_k} : x \in W^\perp, x_{J_{<k}} = 0\}.$$

It is easy to see that these two subspaces are orthogonal complements. Analogously to (20), the primal LLS step Δx^{ll} is obtained as the unique solution to the linear system

$$\begin{aligned} \delta\Delta x^{\mathrm{ll}} + \delta^{-1}\Delta s &= -x^{1/2}s^{1/2}, \\ \Delta x^{\mathrm{ll}} \in W, \quad \text{and} \quad \Delta s &\in W^\perp_{\mathcal{J},1} \oplus \cdots \oplus W^\perp_{\mathcal{J},p}, \end{aligned} \tag{21}$$

and the dual LLS step Δs^{ll} is the unique solution to

$$\begin{aligned} \delta\Delta x + \delta^{-1}\Delta s^{\mathrm{ll}} &= -x^{1/2}s^{1/2}, \\ \Delta x \in W_{\mathcal{J},1} \oplus \cdots \oplus W_{\mathcal{J},p}, \quad \text{and} \quad \Delta s^{\mathrm{ll}} &\in W^\perp. \end{aligned} \tag{22}$$

It is important to note that Δs in (21) may be different from Δs^{ll}, and Δx in (22) may be different from Δx^{ll}. In fact, $\Delta s^{\mathrm{ll}} = \Delta s$ and $\Delta x^{\mathrm{ll}} = \Delta x$ can only be the case for the affine scaling step.

The following lemma proves that the above linear systems are indeed uniquely solved by the LLS step.

Lemma 3.7. *For $t \in \mathbb{R}^n, W \subseteq \mathbb{R}^n, \delta \in \mathbb{R}^n_{++}$, and $\mathcal{J} = (J_1, J_2, \ldots, J_p)$, let $w = \mathrm{LLS}^{W,\delta}_{\mathcal{J}}(t)$ be defined by*

$$\delta w + \delta^{-1} v = \delta t, \qquad w \in W, \qquad v \in W^\perp_{\mathcal{J},1} \oplus \cdots \oplus W^\perp_{\mathcal{J},p}.$$

Then $\mathrm{LLS}^{W,\delta}_{\mathcal{J}}(t)$ is well-defined and

$$\left\|\delta_{J_k}(t_{J_k} - w_{J_k})\right\| = \min\left\{\left\|\delta_{J_k}(t_{J_k} - z_{J_k})\right\| : z \in W, z_{J_{>k}} = w_{J_{>k}}\right\}$$

for every $k \in [p]$.

In the notation of the above lemma we have, for ordered partitions $\mathcal{J} = (J_1, J_2, \ldots, J_p)$, $\bar{\mathcal{J}} = (J_p, J_{p-1}, \ldots, J_1)$, and $(x, y, s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$ with $\delta = s^{1/2}x^{-1/2}$, that $\Delta x^{\mathrm{ll}} = \mathrm{LLS}^{W,\delta}_{\mathcal{J}}(-x)$ and $\Delta s^{\mathrm{ll}} = \mathrm{LLS}^{W^\perp,\delta^{-1}}_{\bar{\mathcal{J}}}(-s)$.

With these tools, we can prove that the lifting costs are self-dual. This explains the reverse order in the dual vs primal LLS step and justifies our attention on the lifting cost in a self-dual algorithm. The next proposition generalizes the result of [13]. Note that it gives a proof of Proposition 2.1(iii).

Proposition 3.8. *For a linear subspace $W \subseteq \mathbb{R}^n$ and index set $I \subseteq [n]$ with $J = [n] \setminus I$,*

$$\|L_I^W\| \leq \max\{1, \|L_J^{W^\perp}\|\}.$$

In particular, $\ell^W(I) = \ell^{W^\perp}(J)$.

3.3.2 Partition Lifting Scores. A key insight is that if the layering $\mathcal{J}$ is "well-separated", then we indeed have $x\Delta s^{\mathrm{ll}} + s\Delta x^{\mathrm{ll}} \approx -xs$, that is, the LLS direction is close to the affine scaling direction. This will be shown in Lemma 3.10. The notion of "well-separatedness" can be formalized as follows. Recall the definition of the lifting score (4). The lifting score of the layering $\mathcal{J} = (J_1, J_2, \ldots, J_p)$ of $[n]$ with respect to W is defined as

$$\ell^W(\mathcal{J}) := \max_{2 \leq k \leq p} \ell^W(J_{\geq k}).$$

For $\delta \in \mathbb{R}^n_{++}$, we use $\ell^{W,\delta}(I) := \ell^{W\,\mathrm{Diag}(\delta)}(I)$ and $\ell^{W,\delta}(\mathcal{J}) := \ell^{W\,\mathrm{Diag}(\delta)}(\mathcal{J})$. When the context is clear, we omit W and write $\ell^\delta(I) := \ell^{W,\delta}(I)$ and $\ell^\delta(\mathcal{J}) := \ell^{W,\delta}(\mathcal{J})$.

The following important duality claim asserts that the lifting score of a layering equals the lifting score of the reverse layering in the orthogonal complement subspace. It is an immediate consequence of Proposition 3.8.

Lemma 3.9. *Let $W \subseteq \mathbb{R}^n$ be a linear subspace, $\delta \in \mathbb{R}^n_{++}$. For an ordered partition $\mathcal{J} = (J_1, J_2, \ldots, J_p)$, let $\bar{\mathcal{J}} = (J_p, J_{p-1}, \ldots, J_1)$ denote the reverse ordered partition. Then, we have*

$$\ell^{W,\delta}(\mathcal{J}) = \ell^{W^\perp,\delta^{-1}}(\bar{\mathcal{J}}).$$

The next lemma summarizes key properties of the LLS steps and their relation to affine scaling steps, assuming the partition has a small lifting score.

Lemma 3.10. *Let $w = (x, y, s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$, let $\mu = \mu(w)$ and $\delta = \delta(w)$. Let $\mathcal{J} = (J_1, \ldots, J_p)$ be a layering with $\ell^\delta(\mathcal{J}) \leq \beta/(32n^2)$, and let $\Delta w^{\mathrm{ll}} = (\Delta x^{\mathrm{ll}}, \Delta y^{\mathrm{ll}}, \Delta s^{\mathrm{ll}})$ denote the LLS direction for the layering $\mathcal{J}$. Then the following properties hold.*

(i) We have for every $k \in [p]$

$$\|\delta_{J_k}\Delta x^{\mathrm{ll}}_{J_k} + \delta^{-1}_{J_k}\Delta s^{\mathrm{ll}}_{J_k} + x^{1/2}_{J_k}s^{1/2}_{J_k}\| \leq 6n\ell^\delta(\mathcal{J})\sqrt{\mu}, \tag{23}$$

and

$$\|\delta\Delta x^{\mathrm{ll}} + \delta^{-1}\Delta s^{\mathrm{ll}} + x^{1/2}s^{1/2}\| \leq 6n^{3/2}\ell^\delta(\mathcal{J})\sqrt{\mu}. \tag{24}$$

(ii) For the affine scaling direction $\Delta w^{\mathrm{a}} = (\Delta x^{\mathrm{a}}, \Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$,

$$\|Rx^{\mathrm{ll}} - Rx^{\mathrm{a}}\|, \|Rs^{\mathrm{ll}} - Rs^{\mathrm{a}}\| \leq 6n^{3/2}\ell^\delta(\mathcal{J}).$$

(iii) For the residuals of the LLS steps we have $\|Rx^{\mathrm{ll}}\|, \|Rs^{\mathrm{ll}}\| \leq \sqrt{2n}$. For each $i \in [n]$, $\max\{|Rx^{\mathrm{ll}}_i|, |Rs^{\mathrm{ll}}_i|\} \geq \frac{1}{2} - \frac{3}{4}\beta$.

(iv) Let $\varepsilon^{\mathrm{ll}}(w) = \max_{i\in[n]} \min\{|Rx^{\mathrm{ll}}_i|, |Rs^{\mathrm{ll}}_i|\}$, and define the step length as

$$\alpha := \sup\{\alpha' \in [0,1] : \forall \bar{\alpha} \in [0, \alpha'] : w + \bar{\alpha}\Delta w^{\mathrm{ll}} \in \mathcal{N}(2\beta)\}.$$

We obtain the following bounds on the progress in the LLS step:

$$\mu(w + \alpha\Delta w^{\mathrm{ll}}) = (1-\alpha)\mu, \quad \text{and}$$

$$\alpha \geq 1 - \frac{3\sqrt{n}\varepsilon^{\mathrm{ll}}(w)}{\beta}.$$

(v) We have $\varepsilon^{\mathrm{ll}}(w) = 0$ if and only if $\alpha = 1$. These are further equivalent to $w + \Delta w^{\mathrm{ll}} = (x + \Delta x^{\mathrm{ll}}, y + \Delta y^{\mathrm{ll}}, s + \Delta s^{\mathrm{ll}})$ being an optimal solution to (LP).

3.4 The Layering Procedure

Our algorithm performs LLS steps on a layering with a low lifting score. A further requirement is that within each layer, the circuit imbalances κ^δ_{ij} defined in (6) are suitably bounded. The rescaling here is with respect to $\delta = \delta(w)$ for the current iterate $w = (x, y, s)$. To define the precise requirement on the layering, we first introduce an auxiliary graph. Throughout we use the parameter

$$\gamma := \frac{\beta}{2^{10}n^5}. \tag{25}$$

The Auxiliary Graph. For a vector $\delta \in \mathbb{R}^n_{++}$ and $\sigma > 0$, we define the directed graph $G_{\delta,\sigma} = ([n], E_{\delta,\sigma})$ such that $(i,j) \in E_{\delta,\sigma}$ if $\kappa^\delta_{ij} \geq \sigma$. This is a subgraph of the *circuit ratio digraph* studied in Section 2, including only the edges where the circuit ratio is at least the threshold σ. Note that we do not have direct access to this graph, as we cannot efficiently compute the values κ^δ_{ij}.

At the beginning of the entire algorithm, we run the subroutine Find-Circuits(A) as in Theorem 2.15, where $W = \mathrm{Ker}(A)$. We assume the matroid $\mathcal{M}(A)$ is non-separable. For a separable matroid, we can solve the subproblems of our LP on the components separately. Thus, for each $i \neq j$, $i, j \in [n]$, we obtain an estimate $\hat{\kappa}_{ij} \leq \kappa_{ij}$. These estimates will be gradually improved throughout the algorithm.

Note that $\kappa^\delta_{ij} = \kappa_{ij}\delta_j/\delta_i$ and $\hat{\kappa}^\delta_{ij} = \hat{\kappa}_{ij}\delta_j/\delta_i$. If $\hat{\kappa}^\delta_{ij} \geq \sigma$, then we are guaranteed $(i,j) \in E_{\delta,\sigma}$.

Definition 3.11. *Define $\hat{G}_{\delta,\sigma} = ([n], \hat{E}_{\delta,\sigma})$ to be the directed graph with edges (i,j) such that $\hat{\kappa}^\delta_{ij} \geq \sigma$; clearly, $\hat{G}_{\delta,\sigma}$ is a subgraph of $G_{\delta,\sigma}$.*

Lemma 3.12. *Let $\delta \in \mathbb{R}^n_{++}$. For every $i \neq j$, $i, j \in [n]$, $\hat{\kappa}^\delta_{ij} \cdot \hat{\kappa}^\delta_{ji} \geq 1$. Consequently, for any $0 < \sigma \leq 1$, at least one of $(i,j) \in \hat{E}_{\delta,\sigma}$ or $(j,i) \in \hat{E}_{\delta,\sigma}$.*

Balanced layerings. We are ready to define the requirements on the layering in the algorithm. In the algorithm, $\delta = \delta(w)$ will correspond to the scaling of the current iterate $w = (x, y, s)$.

Definition 3.13. *Let $\delta \in \mathbb{R}^n_{++}$. The layering $\mathcal{J} = (J_1, J_2, \ldots, J_p)$ of $[n]$ is* δ-balanced *if*

(i) $\ell^\delta(\mathcal{J}) \leq \gamma$, and
(ii) J_k is strongly connected in $G_{\delta,\gamma/n}$ for all $k \in [p]$.

The following lemma shows that within each layer, the κ^δ_{ij} values are within a bounded range. This will play an important role in our potential analysis.

Lemma 3.14. *Let $0 < \sigma < 1$ and $t > 0$, and $i, j \in [n]$, $i \neq j$. If the graph $G_{\delta,\sigma}$ contains a directed path of at most $t-1$ edges from j to i, then*

$$\kappa^\delta_{ij} < \left(\frac{\bar{\chi}^*}{\sigma}\right)^t \quad \text{and} \quad \kappa^\delta_{ji} > \left(\frac{\sigma}{\bar{\chi}^*}\right)^t.$$

Description of the Layering Subroutine. Consider an iterate $w = (x, y, s) \in \mathcal{N}(\beta)$ of the algorithm with $\delta = \delta(w)$. The subroutine LAYERING$(\delta, \hat{\kappa})$, described in Algorithm 1, constructs a δ-balanced layering. We recall that the approximated auxilliary graph $\hat{G}_{\delta,\gamma/n}$ with respect to $\hat{\kappa}$ is as in Definition 3.11

Algorithm 1: LAYERING$(\delta, \hat{\kappa})$

Input : $\delta \in \mathbb{R}^n_{++}$ and $\hat{\kappa} \in \mathbb{R}^E_{++}$.
Output: δ-balanced layering $\mathcal{J} = (J_1, \ldots, J_p)$ and updated values $\hat{\kappa} \in \mathbb{R}^E_{++}$.

Compute the strongly connected components $C_1, C_2, \ldots, C_\ell$ of $\hat{G}_{\delta,\gamma/n}$, listed in the ordering imposed by $\hat{G}_{\delta,\gamma/n}$;
$\bar{E} \leftarrow \hat{E}_{\delta,\gamma/n}$;
for $k = 2, \ldots, \ell$ **do**
  Call VERIFY-LIFT$(W \operatorname{Diag}(\delta), C_{\geq k}, \gamma)$ that answers *'pass'* or *'fail'*;
  if *the answer is* 'fail' **then**
    Let $i \in C_{\geq k}$, $j \in C_{<k}$, and t be the output of VERIFY-LIFT such that $\gamma/n \leq t \leq \kappa^\delta_{ij}$;
    $\hat{\kappa}_{ij} \leftarrow t\delta_i/\delta_j$;
    $\bar{E} \leftarrow \bar{E} \cup \{(i, j)\}$;
Compute strongly connected components $J_1, J_2, \ldots, J_p$ of $([n], \bar{E})$, listed in the ordering imposed by $\hat{G}_{\delta,\gamma/n}$;
return $\mathcal{J} = (J_1, J_2, \ldots, J_p), \hat{\kappa}$.

We now give an overview of the subroutine LAYERING$(\delta, \hat{\kappa})$. We start by computing the strongly connected components (SCCs) of the directed graph $\hat{G}_{\delta,\gamma/n}$. The edges of this graph are obtained using the current estimates $\hat{\kappa}^\delta_{ij}$. According to Lemma 3.12, we have $(i, j) \in \hat{E}_{\delta,\gamma/n}$ or $(j, i) \in \hat{E}_{\delta,\gamma/n}$ for every $i, j \in [n]$, $i \neq j$. Hence, there is a linear ordering of the components $C_1, C_2, \ldots, C_\ell$ such that $(u, v) \in \hat{E}_{\delta,\gamma/n}$ whenever $u \in C_i$, $v \in C_j$, and $i < j$. We call this the ordering imposed by $\hat{G}_{\delta,\gamma/n}$.

Next, for each $k = 2, \ldots, \ell$, we use the subroutine VERIFY-LIFT$(W \operatorname{Diag}(\delta), C_{\geq k}, \gamma)$ described after Lemma 2.10. If the subroutine returns 'pass', then we conclude $\ell^\delta(C_{\geq k}) \leq \gamma$, and proceed to the next layer. If the answer is 'fail', then the subroutine returns as certificates $i \in C_{\geq k}$, $j \in C_{<k}$, and t such that $\gamma/n \leq t \leq \kappa^\delta_{ij}$. In this case, we update $\hat{\kappa}^\delta_{ij}$ to the higher value t. We add (i, j) to an edge set $\bar{E}$; this edge set was initialized to contain $\hat{E}_{\delta,\gamma/n}$. After adding (i, j), all components C_ℓ between those containing i and j will be merged into a single strongly connected component. To see this, recall that if $i' \in C_\ell$ and $j' \in C_{\ell'}$ for $\ell < \ell'$, then $(i', j') \in \hat{E}_{\delta,\gamma/n}$ according to Lemma 3.12.

Finally, we compute the strongly connected components of $([n], \bar{E})$. We let $J_1, J_2, \ldots, J_p$ denote their unique acyclic order, and return these layers.

Lemma 3.15. *The subroutine* LAYERING$(\delta, \hat{\kappa})$ *returns a δ-balanced layering in $O(nm^2 + n^2)$ time.*

The difficult part of the proof of the above lemma is showing the running time bound. We note that the weaker bound $O(n^2m^2)$ can be obtained by a simpler argument.

3.5 The Overall Algorithm

Algorithm 2: LP-SOLVE(A, b, c, w^0)

Input : $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and an initial feasible solution $w^0 = (x^0, y^0, s^0) \in \mathcal{N}(1/8)$ to (LP).
Output: Optimal solution $w^* = (x^*, y^*, s^*)$ to (LP).

Call FIND-CIRCUITS(A) to obtain the lower bounds $\hat{\kappa}_{ij}$ for each $i, j \in [n]$, $i \neq j$;
$k \leftarrow 0, \alpha \leftarrow 0$;
repeat
  /* Predictor step */
  Compute affine scaling direction $\Delta w^{\mathrm{a}} = (\Delta x^{\mathrm{a}}, \Delta y^{\mathrm{a}}, \Delta s^{\mathrm{a}})$ for w;
  if $\varepsilon^{\mathrm{a}}(w) < 10n^{3/2}\gamma$ **then** // Recall $\varepsilon^{\mathrm{a}}(w)$ defined in (17)
    $\delta \leftarrow (s^k)^{1/2}(x^k)^{-1/2}$;
    $(\mathcal{J}, \hat{\kappa}) \leftarrow$LAYERING$(\delta, \hat{\kappa})$;
    Compute Layered Least Squares direction $\Delta w^{\mathrm{ll}} = (\Delta x^{\mathrm{ll}}, \Delta y^{\mathrm{ll}}, \Delta s^{\mathrm{ll}})$ for the layering $\mathcal{J}$ and w;
    $\Delta w \leftarrow \Delta w^{\mathrm{ll}}$;
  else
    $\Delta w \leftarrow \Delta w^{\mathrm{a}}$;
  $\alpha \leftarrow \sup\{\alpha' \in [0, 1] : \forall \bar{\alpha} \in [0, \alpha'] : w + \bar{\alpha}\Delta w \in \mathcal{N}(1/4)\}$;
  $w' \leftarrow w^k + \alpha\Delta w$;
  /* Corrector step */
  Compute centrality direction $\Delta w^{\mathrm{c}} = (\Delta x^{\mathrm{c}}, \Delta y^{\mathrm{c}}, \Delta s^{\mathrm{c}})$ for w';
  $w^{k+1} \leftarrow w' + \Delta w^{\mathrm{c}}$;
  $k \leftarrow k + 1$;
until $\mu(w^k) = 0$;
return $w^k = (x^k, y^k, s^k)$.

Algorithm 2 presents the overall algorithm LP-SOLVE(A, b, c, w^0). We assume that an initial feasible solution $w^0 = (x^0, y^0, s^0) \in \mathcal{N}(\beta)$ is given. We address this in Section 5, by adapting the extended system used in [49]. We note that this subroutine requires an upper bound on $\bar{\chi}^*$. Since computing $\bar{\chi}^*$ is hard, we can implement it by a doubling search on $\log \bar{\chi}^*$, as explained in Section 5. Other than for initialization, the algorithm does not require an estimate on $\bar{\chi}^*$.

The algorithm starts with the subroutine FIND-CIRCUITS(A) as in Theorem 2.15. The iterations are similar to the MTY Predictor-Corrector algorithm [29]. The main difference is that certain affine scaling steps are replaced by LLS steps. In every predictor step, we compute the affine scaling direction, and consider the quantity $\varepsilon^{\mathrm{a}}(w) = \max_{i\in[n]} \min\{|Rx^{\mathrm{a}}_i|, |Rs^{\mathrm{a}}_i|\}$. If this is above the threshold $10n^{3/2}\gamma$, then we perform the affine scaling step. However, in case $\varepsilon^{\mathrm{a}}(w) < 10n^{3/2}\gamma$, we use the LLS direction instead. In each such iteration, we call the subroutine LAYERING$(\delta, \hat{\kappa})$ (Algorithm 1) to compute the layers, and we compute the LLS step for this layering.

Another important difference is that the algorithm does not require a final rounding step. It terminates with the exact optimal

solution w^* once a predictor step is able to perform a full step with $\alpha = 1$.

Theorem 3.16. *For given $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and an initial feasible solution $w^0 = (x^0, y^0, s^0) \in \mathcal{N}(1/8)$, Algorithm 2 finds an optimal solution to (LP) in $O(n^{2.5} \log n \log(\bar{\chi}_A^* + n))$ iterations.*

4 THE POTENTIAL FUNCTION AND THE OVERALL ANALYSIS

Let $\mu > 0$ and $\delta(\mu) = s(\mu)^{1/2}x(\mu)^{-1/2}$ correspond to the point on the central path. For $i, j \in [n]$, $i \neq j$, we define

$$\varrho^\mu(i,j) := \frac{\log \kappa_{ij}^{\delta(\mu)}}{\log\left(3n\bar{\chi}_A^*/\gamma\right)},$$

and the main potentials in the algorithm as

$$\Psi^\mu(i,j) := \max\left\{1, \min\left\{2n, \inf_{0<\mu'<\mu} \varrho^{\mu'}(i,j)\right\}\right\}$$
$$\Psi(\mu) := \sum_{i,j\in[n],\, i\neq j} \log_2 \Psi^\mu(i,j).$$

The quantity $\Psi^\mu(i,j)$ is motivated by the bounds in Lemma 3.14. The next statement is an immediate consequence of this lemma and (13).

Lemma 4.1. *Let $w = (x, y, s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/4]$, let $\mu = \mu(w)$, and $\delta = \delta(w)$. Let $i, j \in [n]$, $i \neq j$. If the graph $G_{\delta,\gamma/(3n)}$ contains a path from j to i of at most $t-1$ edges, then $\varrho^\mu(i,j) < t$. If there is a path of at most $t-1$ edges from i to j, then $-t < \varrho^\mu(i,j)$.*

If $\Psi^\mu(i,j) \geq t$, then i and j cannot be together on a layer of size $\leq t$, and j cannot be on a layer preceding the layer containing i in any $\delta(w')$-balanced layering, where $w' = (x', y', s') \in \mathcal{N}(\beta)$ with $\mu(w') < \mu$.

Our potentials $\Psi^\mu(i,j)$ can be seen as fine-grained analogues of the crossover events analyzed in [31, 32, 49]. Roughly speaking, a crossover event corresponds to $\Psi^\mu(i,j)$ increasing above n, meaning that i and j cannot be contained in the same layer after the normalized duality gap decreases below μ.

In what follows, we formulate the key ideas for proving Theorem 3.16. Proofs can be found in the full version.

For a solution $w \in \mathcal{N}(\beta)$, Δw^{ll} refers to the LLS direction found in the algorithm, and Rx^{ll} and Rs^{ll} denote the residuals as in (15). For a subset $I \subset [n]$ recall the definition

$$\varepsilon_I^{\text{ll}}(w) := \max_{i\in I} \min\{|Rx_i^{\text{ll}}|, |Rs_i^{\text{ll}}|\}.$$

Another important quantity in the analysis is

$$\xi_I^{\text{ll}}(w) := \min\{\|Rx_I^{\text{ll}}\|, \|Rs_I^{\text{ll}}\|\}$$

for a subset $I \subset [n]$. For a layering $\mathcal{J} = (J_1, J_2, \ldots, J_p)$, we let

$$\xi_{\mathcal{J}}^{\text{ll}}(w) = \max_{k\in[p]} \xi_{J_k}^{\text{ll}}(w).$$

The key idea of the analysis is to extract information about the optimal solution $w^* = (x^*, y^*, s^*)$ from the LLS direction. The first main lemma shows that if $\|Rx_{J_q}^{\text{ll}}\|$ is large on some layer J_q, then for at least one index $i \in J_q$, $x_i^*/x_i \geq 1/\text{poly}(n)$; the analogous statement holds for $\|Rs_{J_q}^{\text{ll}}\|$.

Lemma 4.2. *Let $w = (x, y, s) \in \mathcal{N}(\beta)$ for $\beta \in (0, 1/8]$, and let $\mathcal{J} = (J_1, \ldots, J_p)$ be a $\delta(w)$-balanced layering, and let $\Delta w^{\text{ll}} = (\Delta x^{\text{ll}}, \Delta y^{\text{ll}}, \Delta s^{\text{ll}})$ be the corresponding LLS direction. Then the following statement holds for every $q \in [p]$:*

(i) There exists $i \in J_q$ such that

$$x_i^* \geq \frac{2x_i}{3\sqrt{n}} \cdot (\|Rx_{J_q}^{\text{ll}}\| - 2\gamma n). \tag{26}$$

(ii) There exists $j \in J_q$ such that

$$s_j^* \geq \frac{2s_j}{3\sqrt{n}} \cdot (\|Rs_{J_q}^{\text{ll}}\| - 2\gamma n). \tag{27}$$

We emphasize that the lemma only shows the existence of such indices i and j, but does not provide an efficient algorithm for identifying them. It is also useful to note that for any $i \in [n]$, $\max\{|Rx_i^{\text{ll}}|, |Rs_i^{\text{ll}}|\} \geq \frac{1}{2} - \frac{5}{4}\beta$ according to Lemma 3.10(iii). Thus, for each $q \in [p]$, we obtain a positive lower bound either in case (i) or in case (ii).

The next lemma shows how we can argue for increase in the potential function value for multiple pairs of variables, if we have lower bounds on both x_i^* and s_j^* for some $i, j \in [n]$, along with a lower bound on $\varrho^\mu(i,j)$.

Lemma 4.3. *Let $w = (x, y, s) \in \mathcal{N}(2\beta)$ for $\beta \in (0, 1/8]$, let $\mu = \mu(w)$ and $\delta = \delta(w)$. Let $i, j \in [n]$ and $2 \leq \tau \leq n$ such that for the optimal solution $w^* = (x^*, y^*, s^*)$, we have $x_i^* \geq \beta x_i/(2^{10}n^{5.5})$ and $s_j^* \geq \beta s_j/(2^{10}n^{5.5})$, and assume $\varrho^\mu(i,j) \geq -\tau$. Let μ' be the normalized duality gap after $\Omega(\sqrt{n}\tau \log(\bar{\chi}^* + n))$ iterations subsequent to the iterate w. Then $\Psi^{\mu'}(i,j) \geq 2\tau$, and for every $\ell \in [n] \setminus \{i, j\}$, either $\Psi^{\mu'}(i,\ell) \geq 2\tau$, or $\Psi^{\mu'}(\ell,j) \geq 2\tau$.*

We note that i and j as in the lemma are necessarily different, since $i = j$ would imply $0 = x_i^* s_i^* \geq \beta^2\mu/(2^{20}n^{11})$.

The overall potential argument in the proof of Theorem 3.16 uses Lemma 4.3 in three cases: $\xi_{\mathcal{J}}^{\text{ll}}(w) \geq 4\gamma n$ (Lemma 4.2 applies); $\xi_{\mathcal{J}}^{\text{ll}}(w) < 4\gamma n$ and $\ell^{\delta^+}(\mathcal{J}) \leq 3\gamma n$; and $\xi_{\mathcal{J}}^{\text{ll}}(w) < 4\gamma n$ and $\ell^{\delta^+}(\mathcal{J}) > 3\gamma n$. Here, δ^+ refers to the value of δ after the LLS step. Note that $\delta^+ > 0$ is well-defined, unless the algorithm terminated with an optimal solution. In the full version we show how Lemma 4.3 can be applied in each of the three cases.

4.1 The Iteration Complexity Bound for the Vavasis-Ye Algorithm

We now show that the potential analysis described above also gives an improved bound $O(n^{2.5} \log n \log(\bar{\chi}_A + n))$ for the original VY algorithm [49].

We recall the VY layering step. Order the variables via π such that $\delta_{\pi(1)} \leq \delta_{\pi(2)} \leq \ldots \leq \delta_{\pi(n)}$. The layers will be consecutive sets in the ordering; a new layer starts with $\pi(i+1)$ each time $\delta_{\pi(i+1)} > g\delta_{\pi(i)}$, for a parameter $g = \text{poly}(n)\bar{\chi}$.

As outlined in the Introduction, the VY algorithm can be seen as a special implementation of our algorithm by setting $\hat{\kappa}_{ij} = g\gamma/n$. With these edge weights, we have that $\hat{\kappa}_{ij}^\delta \geq \gamma/n$ precisely if $g\delta_j \geq \delta_i$.[2]

[2]For simplicity, in the Introduction we used $gx_i \geq x_j$ instead, which is almost the same in the proximity in the central path.

With these edge weights, it is easy to see that our Layering($\delta, \hat{\kappa}$) subroutine finds the exact same components as VY. Moreover, the layers will be the initial strongly connected components C_i of $G_{\delta,\gamma/n}$: due to the choice of g, this partition is automatically δ-balanced. There is no need to call Verify-Lift.

The essential difference compared to our algorithm is that the values $\hat{\kappa}_{ij} = g\gamma/n$ are not lower bounds on κ_{ij} as we require, but upper bounds instead. This is convenient to simplify the construction of the layering. On the negative side, the strongly connected components of $\hat{G}_{\delta,\gamma/n}$ may not anymore be strongly connected in $G_{\delta,\gamma/n}$. Hence, we cannot use Lemma 4.1, and consequently, Lemma 4.3 does not hold.

Still, the $\hat{\kappa}_{ij}$ bounds are overestimating κ_{ij} by at most a factor poly$(n)\bar{\chi}$. Therefore, the strongly connected components of $\hat{G}_{\delta,n/\gamma}$ are strongly connected in $G_{\delta,\sigma}$ for some $\sigma = 1/(\text{poly}(n)\bar{\chi})$.

Hence, the entire argument described in this section is applicable to the VY algorithm, with a different potential function defined with $\bar{\chi}$ instead of $\bar{\chi}^*$. This is the reason why the iteration bound in Lemma 4.3, and therefore in Theorem 3.16, also changes to $\bar{\chi}$ dependency.

It is worth noting that due to the overestimation of the κ_{ij} values, the VY algorithm uses a coarser layering than our algorithm. Our algorithm splits up the VY layers into smaller parts so that $\ell^{\delta}(\mathcal{J})$ remains small, but within each part, the gaps between the variables are bounded as a function of $\bar{\chi}_A^*$ instead of $\bar{\chi}_A$.

5 INITIALIZATION

Our main algorithm (Algorithm 2 in Section 3.5), requires an initial solution $w^0 = (x^0, y^0, s^0) \in \mathcal{N}(\beta)$. In this section, we remove this assumption by adapting the initialization method of [49] to our setting.

We use the "big-M method", a standard initialization approach for path-following interior point methods that introduces an auxiliary system whose optimal solutions map back to the optimal solutions of the original system. The primal-dual system we consider is

$$\begin{array}{ll|ll} \min & c^\top x + Me^\top \underline{x} & \max & y^\top b + 2Me^\top z \\ & Ax - A\underline{x} = b & & A^\top y + z + s = c \\ & x + \bar{x} = 2Me & & z + \bar{s} = 0 \\ & x, \bar{x}, \underline{x} \geq 0 & & -A^\top y + \underline{s} = Me \\ & & & s, \bar{s}, \underline{s} \geq 0. \end{array} \qquad \text{(Init-LP)}$$

The constraint matrix used in this system is

$$\hat{A} = \begin{pmatrix} A & 0 & -A \\ I & I & 0 \end{pmatrix}$$

The next lemma, asserts that the $\bar{\chi}$ condition number of $\hat{A}$ is not much bigger than that of A of the original system (LP).

Lemma 5.1 ([49, Lemma 23]). $\bar{\chi}_{\hat{A}} \leq 3\sqrt{2}(\bar{\chi}_A + 1)$.

We extend this bound for $\bar{\chi}^*$.

Lemma 5.2. $\bar{\chi}^*_{\hat{A}} \leq 3\sqrt{2}(\bar{\chi}^*_A + 1)$.

Also, for sufficiently large M, the optimal solutions of the original system are preserved. We let d be the min-norm solution to $Ax = b$, i.e., $d = A^\top(AA^\top)^{-1}b$.

Proposition 5.3. *Assume both primal and dual of (LP) are feasible, and $M > \max\{(\bar{\chi}_A + 1)\|c\|, \bar{\chi}_A\|d\|\}$. Every optimal solution (x, y, s) to (LP), can be extended to an optimal solution $(x, \underline{x}, \bar{x}, y, s, \underline{s}, \bar{s})$ to (Init-LP); and conversely, from every optimal solution $(x, \underline{x}, \bar{x}, y, z, s, \underline{s}, \bar{s})$ to (Init-LP), we obtain an optimal solution (x, y, s) by deleting the auxiliary variables.*

The next lemma is from [31, Lemma 4.4]. Recall that $w = (x, y, s) \in \mathcal{N}(\beta)$ if $\|xs/\mu(w) - e\| \leq \beta$.

Lemma 5.4. *Let $w = (x, y, s) \in \mathcal{P}^{++} \times \mathcal{D}^{++}$, and let $\nu > 0$. Assume that $\|xs/\nu - e\| \leq \tau$. Then $(1 - \tau/\sqrt{n})\nu \leq \mu(w) \leq (1 + \tau/\sqrt{n})\nu$ and $w \in \mathcal{N}(\tau/(1-\tau))$.*

The new system has the advantage that we can easily initialize the system with a feasible solution in close proximity to central path:

Proposition 5.5. *We can initialize system (Init-LP) close to the central path with initial solution $w^0 = (x^0, y^0, s^0) \in \mathcal{N}(1/8)$ and parameter $\mu(w^0) \approx M^2$ if $M > 15\max\{(\bar{\chi}_A + 1)\|c\|, \bar{\chi}_A\|d\|\}$.*

Detecting Infeasibility. For using the extended system (Init-LP), we still need to assume that both the primal and dual programs in (LP) are feasible. For arbitrary instances, we first need to check if this is the case, or conclude that the primal or the dual (or both) are infeasible.

This can be done by employing a two-phase method. The first phase decides feasibility by running (Init-LP) with data $(A, b, 0)$ and $M > \bar{\chi}_A\|d\|$. The objective value of the optimal primal-dual pair is 0 if and only if (LP) has a feasible solution. If the optimal primal/dual solution $(x^*, \underline{x}^*, \bar{x}^*, y^*, s^*, \underline{s}^*, \bar{s}^*)$ has positive objective value, we can extract an infeasibility certificate.

Feasibility of the dual of (LP) can be decided by running (Init-LP) on data $(A, 0, c)$ and $M > (\bar{\chi}_A + 1)\|c\|$ with the same argumentation: Either the objective of the dual is 0 and therefore the dual optimal solution $(y^*, \underline{s}^*, s^*, \bar{s}^*)$ corresponds to a feasible dual solution of (LP) or the objective value is negative and we extract a dual infeasibility certificate.

Finding the Right Value of M. Whereas Algorithm 2 does not require any estimate on $\bar{\chi}^*$ or $\bar{\chi}$, for the initialization we need to set $M \geq \max\{(\bar{\chi}_A + 1)\|c\|, \bar{\chi}_A\|d\|\}$ as in Proposition 5.3.

A straightforward guessing approach (attributed to J. Renegar in [49]) starts with a constant guess, say $\bar{\chi}_A = 100$, constructs the extended system, and runs the algorithm. In case the optimal solution to the extended system does not map to an optimal solution of (LP), we restart with $\bar{\chi}_A = 100^2$ and try again; we continue squaring the guess until an optimal solution is found.

This would still require a series of $\log\log \bar{\chi}_A$ guesses, and thus, result in a dependence on $\bar{\chi}_A$ in the running time. However, if we initially rescale our system using the near-optimal rescaling Theorem 2.4, the we can turn the dependence from $\bar{\chi}_A$ to $\bar{\chi}^*_A$. The overall iteration complexity remains $O(n^{2.5}\log n \log(\bar{\chi}^*_A + n))$, since the running time for the final guess on $\bar{\chi}^*_A$ dominates the total running time of all previous computations due to the repeated squaring.

Note that this guessing technique can handle bad guesses gracefully. For the first phase, if neither a feasible solution to (LP) is returned nor a Farkas' certificate can be extracted, we have proof

that the guess was too low by the above paragraph. Similarly, in phase two, when feasibility was decided in the affirmative for primal and dual, an optimal solution to (Init-LP) that corresponds to an infeasible solution to (LP) serves as a certificate that another squaring of the guess is necessary.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc.

[2] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. 2018. Log-barrier interior point methods are not strongly polynomial. *SIAM Journal on Applied Algebra and Geometry* 2, 1 (2018), 140–178.

[3] Sébastien Bubeck and Ronen Eldan. 2014. The entropic barrier: a simple and optimal universal self-concordant barrier. arXiv preprint arXiv:1412.1587.

[4] Sergei Chubanov. 2014. A polynomial algorithm for linear optimization which is strongly polynomial under certain conditions on optimal solutions. (2014). http://www.optimization-online.org/DB_HTML/2014/12/4710.html.

[5] Michael B Cohen, Yin Tat Lee, and Zhao Song. 2019. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 938–942.

[6] Samuel I Daitch and Daniel A Spielman. 2008. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th annual ACM symposium on Theory of Computing*. 451–460.

[7] Jesús A De Loera, Raymond Hemmecke, and Jon Lee. 2015. On Augmentation Algorithms for Linear and Integer-Linear Programming: From Edmonds–Karp to Bland and Beyond. *SIAM Journal on Optimization* 25, 4 (2015), 2494–2511.

[8] Jesús A De Loera, Sean Kafer, and Laura Sanità. 2019. Pivot Rules for Circuit-Augmentation Algorithms in Linear Optimization. *arXiv preprint arXiv:1909.12863* (2019).

[9] II Dikin. 1967. Iterative solution of problems of linear and quadratic programming. *Doklady Akademii Nauk* 174, 4 (1967), 747–748.

[10] András Frank. 2011. *Connections in Combinatorial Optimization.* Number 38 in Oxford Lecture Series in Mathematics and its Applications. Oxford University Press.

[11] Jean-Louis Goffin. 1980. The relaxation method for solving systems of linear inequalities. *Mathematics of Operations Research* 5, 3 (1980), 388–414.

[12] Clovis C Gonzaga. 1992. Path-following methods for linear programming. *SIAM review* 34, 2 (1992), 167–224.

[13] Clovis C. Gonzaga and Hugo J. Lara. 1997. A note on properties of condition numbers. *Linear Algebra Appl.* 261, 1 (1997), 269 – 273.

[14] Jackie CK Ho and Levent Tunçel. 2002. Reconciliation of Various Complexity and Condition Measures for Linear Programming Problems and a Generalization of Tardos' Theorem. In *Foundations of Computational Mathematics*. World Scientific, 93–147.

[15] Satoshi Kakihara, Atsumi Ohara Ohara, and Takashi Tsuchiya. 2013. Information geometry and interior-point algorithms in semidefinite programs and symmetric cone programs. *Journal of Optimization Theory and Applications* 157 (2013), 749–780.

[16] Satoshi Kakihara, Atsumi Ohara Ohara, and Takashi Tsuchiya. 2014. Curvature integrals and iteration complexities in SDP and symmetric cone programs. *Computational Optimization and Applications* 57 (2014), 623–665.

[17] Narendra Karmarkar. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC)*. 302–311.

[18] Leonid G Khachiyan. 1979. A polynomial algorithm in linear programming. In *Doklady Academii Nauk SSSR*, Vol. 244. 1093–1096.

[19] Tomonari Kitahara and Shinji Mizuno. 2013. A bound for the number of different basic solutions generated by the simplex method. *Mathematical Programming* 137, 1-2 (2013), 579–586.

[20] Tomonari Kitahara and Takashi Tsuchiya. 2013. A simple variant of the Mizuno–Todd–Ye predictor-corrector algorithm and its objective-function-free complexity. *SIAM Journal on Optimization* 23, 3 (2013), 1890–1903.

[21] Guanghui Lan, Renato DC Monteiro, and Takashi Tsuchiya. 2009. A polynomial predictor-corrector trust-region algorithm for linear programming. *SIAM Journal on Optimization* 19, 4 (2009), 1918–1946.

[22] Yin Tat Lee and Aaron Sidford. 2014. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{\text{rank}})$ iterations and faster algorithms for maximum flow. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 424–433.

[23] Yin Tat Lee and Aaron Sidford. 2015. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. 230–249.

[24] Yin Tat Lee and Aaron Sidford. 2019. Solving Linear Programs with $\tilde{O}(\sqrt{\text{rank}})$ Linear System Solves. arXiv preprint 1910.08033.

[25] Aleksander Madry. 2013. Navigating central path with electrical flows: From flows to matchings, and back. In *Proceedings of the 54th IEEE Annual Symposium on Foundations of Computer Science*. IEEE, 253–262.

[26] Nimrod Megiddo. 1983. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.* 12, 2 (1983), 347–353.

[27] Nimrod Megiddo, Shinji Mizuno, and Takashi Tsuchiya. 1998. A modified layered-step interior-point algorithm for linear programming. *Mathematical Programming* 82, 3 (1998), 339–355.

[28] Sanjay Mehrotra. 1992. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization* 2, 4 (1992), 575–601.

[29] Shinji Mizuno, Michael Todd, and Yinyu Ye. 1993. On Adaptive-Step Primal-Dual Interior-Point Algorithms for Linear Programming. *Mathematics of Operations Research - MOR* 18 (11 1993), 964–981. https://doi.org/10.1287/moor.18.4.964

[30] Renato D.C. Monteiro and Takashi Tsuchiya. 2008. A strong bound on the integral of the central path curvature and its relationship with the iteration-complexity of primal-dual path-following LP algorithms. *Mathematical Programming* 115, 1 (2008), 105–149.

[31] Renato D. C. Monteiro and Takashi Tsuchiya. 2003. A Variant of the Vavasis-Ye Layered-Step Interior-Point Algorithm for Linear Programming. *SIAM Journal on Optimization* 13, 4 (2003), 1054–1079.

[32] Renato D. C. Monteiro and Takashi Tsuchiya. 2005. A New Iteration-Complexity Bound for the MTY Predictor-Corrector Algorithm. *SIAM Journal on Optimization* 15, 2 (2005), 319–347.

[33] Neil Olver and László A. Végh. 2017. A simpler and faster strongly polynomial algorithm for generalized flow maximization. In *Proceedings of the Forty-Ninth Annual ACM Symposium on Theory of Computing (STOC)*. 100–111.

[34] James Renegar. 1988. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming* 40, 1-3 (1988), 59–93.

[35] James Renegar. 1994. Is it possible to know a problem instance is ill-posed?: some foundations for a general theory of condition numbers. *Journal of Complexity* 10, 1 (1994), 1–56.

[36] James Renegar. 1995. Incorporating condition measures into the complexity theory of linear programming. *SIAM Journal on Optimization* 5, 3 (1995), 506–524.

[37] Alexander Schrijver. 2003. *Combinatorial Optimization – Polyhedra and Efficiency.* Springer.

[38] György Sonnevend, Josef Stoer, and Gongyun Zhao. 1991. On the complexity of following the central path of linear programs by linear extrapolation II. *Mathematical Programming* 52, 1-3 (1991), 527–553.

[39] Daniel A Spielman and Shang-Hua Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*.

[40] G.W. Stewart. 1989. On scaled projections and pseudoinverses. *Linear Algebra Appl.* 112 (1989), 189 – 193. https://doi.org/10.1016/0024-3795(89)90594-6

[41] Éva Tardos. 1985. A strongly polynomial minimum cost circulation algorithm. *Combinatorica* 5, 3 (01 Sep 1985), 247–255.

[42] Éva Tardos. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* (1986), 250–256.

[43] Michael J. Todd. 1990. A Dantzig–Wolfe-Like Variant of Karmarkar's Interior-Point Linear Programming Algorithm. *Operations Research* 38, 6 (1990), 1006–1018. https://doi.org/10.1287/opre.38.6.1006 arXiv:https://doi.org/10.1287/opre.38.6.1006

[44] Michael J. Todd, Levent Tunçel, and Yinyu Ye. 2001. Characterizations, bounds, and probabilistic analysis of two complexity measures for linear programming problems. *Mathematical Programming* 90, 1 (01 Mar 2001), 59–69.

[45] Levent Tunçel. 1999. Approximating the complexity measure of Vavasis-Ye algorithm is NP-hard. *Mathematical Programming* 86, 1 (01 Sep 1999), 219–223.

[46] Pravin M Vaidya. 1989. Speeding-up linear programming using fast matrix multiplication. In *Proceedings of the 30th IEEE Annual Symposium on Foundations of Computer Science*. 332–337.

[47] Jan van den Brand. 2020. A Deterministic Linear Program Solver in Current Matrix Multiplication Time. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*.

[48] Stephen A Vavasis. 1994. Stable numerical algorithms for equilibrium systems. *SIAM J. Matrix Anal. Appl.* 15, 4 (1994), 1108–1131.

[49] Stephen A. Vavasis and Yinyu Ye. 1996. A primal-dual interior point method whose running time depends only on the constraint matrix. *Mathematical Programming* 74, 1 (1996), 79–120.

[50] László A. Végh. 2017. A Strongly Polynomial Algorithm for Generalized Flow Maximization. *Mathematics of Operations Research* 42, 2 (2017), 179–211.

[51] Yinyu Ye. 1997. *Interior-Point Algorithms: Theory and Analysis.* John Wiley and Sons, New York.

[52] Yinyu Ye. 2005. A new complexity result on solving the Markov decision problem. *Mathematics of Operations Research* 30, 3 (2005), 733–749.

[53] Yinyu Ye. 2011. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research* 36, 4 (2011), 593–603.