Strongly polynomial linear programming

OpenAI

November 24, 2026

Abstract

We give a deterministic strongly polynomial algorithm for linear programming, answering Smale's ninth problem. For arbitrary real coefficients, the number of steps is bounded by a polynomial in the numbers of variables and constraints. For rational data, every intermediate number has bit length polynomial in the binary input length. The method also gives a strongly polynomial algorithm for convex quadratic programming. For finite turn-based mean-payoff games with perfect information, it gives strongly polynomial time for deterministic games and polynomial bit time for stochastic games with binary rational data.

1 Introduction

How much work does it take to optimize a linear function subject to linear inequalities? For rational coefficients, the ellipsoid method of Khachiyan [88, 89] gave an answer polynomial in the length of the input. Two linear programs with the same numbers of variables and constraints can have very different input lengths. Megiddo [104, 105] asked whether those two dimensions alone could bound the number of arithmetic operations. Tarjan raised the question for real data [135, pp. 1620–1621], and Smale made it his ninth problem [130]. We give an affirmative answer.

We maximize $c^\top x$ subject to $Ax \le b$, given $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ explicitly. A *strongly polynomial* algorithm takes polynomially many steps in m, n, counting exact scalar arithmetic, comparisons and control work. On rational input every intermediate number has bit length polynomial in the full binary input length.

Theorem 1 (Strongly polynomial linear programming)**.** *For arbitrary real A, b, c, a deterministic strongly polynomial algorithm returns one of the following certificates for* $\max\{c^\top x : Ax \le b\}$.

1. Finite optimum. *A strictly complementary primal–dual pair x, y with*
$$s = b - Ax \ge 0, \qquad y \ge 0, \qquad A^\top y = c, \qquad s^\top y = 0, \qquad s + y > 0.$$
2. Unbounded objective. *A feasible point x and an improving ray r with $Ar \le 0$ and $c^\top r > 0$.*
3. Infeasibility. *A vector $y \ge 0$ with $A^\top y = 0$, $b^\top y < 0$, and $|\operatorname{supp} y| \le 1 + \operatorname{rank} A$.*

We use a uniform sequential model [18]. The scalar operations are addition, subtraction, multiplication, division by a nonzero scalar, comparison and copying, with constants $0, 1$. Control operations on indices and explicit bit strings are charged at bit cost. The algorithm accesses scalar inputs only through the scalar operations. All vector inequalities are componentwise. On rational data we measure a number by its reduced fraction.

The arithmetic result extends to every ordered field. In particular, on rational input the certificates are exact rationals. If the input is supplied as integer numerator–denominator pairs, an implementation also keeps the integers in its unreduced pairs of polynomial bit length throughout the computation (Section 3.4).

How the algorithm finds a certificate. Consider first the feasibility of $Ax \le b$. A feasible point has nonnegative slacks $b - Ax$. A certificate of infeasibility instead gives nonnegative weights for a combination of the inequalities that yields a contradiction. Homogenizing the slack vectors puts these alternatives in two orthogonal subspaces. To decide feasibility it suffices to determine which subspace contains a nonnegative vector positive at one designated coordinate.

A positive entry of such a vector can be arbitrarily small compared with its largest entry. Changing the unit of a coordinate changes which entries can be seen at a fixed precision. The algorithm uses these changes of unit as queries. At each query it obtains reliable signs where the rescaled entries are largest. A search combining the signs from different queries identifies the subspace containing a witness, even though the algorithm does not know in advance which signs it can trust. It then sets coordinates to zero, one at a time whenever a witness survives, until linear algebra recovers the witness exactly.

The difficulty is making each query affordable. For a witness of minimal support, the ratios of its nonzero entries are ratios of subdeterminant magnitudes of a basis matrix for the two subspaces. Even in a fixed dimension, the logarithms of these magnitudes can be arbitrarily far apart. The algorithm gives short substitutes for the logarithms as they are needed, preserving the comparisons that determine which entries are large. This lets it compute each set of signs using ordinary polynomial-time LP on rational data whose bit lengths are bounded in the dimensions alone. Section 2 explains how the search, the changes of unit and the compression fit together.

1.1 Consequences

The same search applies to a convex quadratic objective under linear constraints. For arbitrary real coefficients we find an exact minimizer and its multipliers, or certify infeasibility or unboundedness, in deterministic strongly polynomial time (Corollary 6). The underlying result solves the monotone linear complementarity problem. Given M, q with $M + M^\top$ positive semidefinite, it finds $z \ge 0$ such that $Mz + q \ge 0$ and $z^\top(Mz + q) = 0$, or certifies that no such z exists (Theorem 5).

There is a second use for the signs returned by a query. If a point differs from the fixed point of a contraction in max norm, the map moves it toward that fixed point in every coordinate of greatest error. For a self-map of $[-1, 1]^n$ with Lipschitz constant at most $1 - 2^{-b}$, this gives a deterministic algorithm that locates the fixed point to within 2^{-k} in time polynomial in n, b, k, counting all bits exchanged with the evaluation oracle (Theorem 8).

In a finite turn-based mean-payoff game, two players with perfect information choose moves on a graph. One maximizes and the other minimizes the long-run average reward. The fixed-point algorithm and policy evaluation by LP give the exact values from every starting vertex and optimal strategies for both players. The running time is strongly polynomial when each move has a deterministic outcome and polynomial in the binary input length when outcomes are stochastic and the data rational. This includes polynomial time for simple stochastic reachability games. Theorem 11 gives the bounds for arbitrary real rewards and for discounted play.

1.2 Earlier approaches to strongly polynomial time

The first polynomial-time algorithms for general linear programming measure the size of a rational instance in bits. Khachiyan's ellipsoid algorithm [88, 89] draws on the cutting-plane methods of Shor and Yudin–Nemirovskii [128, 148]. In its feasibility argument, the volume of the enclosing ellipsoid falls by a controlled factor at each cut. Determinant bounds in terms of the input length tell the algorithm when that volume is too small to contain a feasible region of the required thickness [71]. Karmarkar's projective algorithm [87] gave a second route to polynomial bit complexity. Subsequent interior-point methods follow solutions of penalized problems as the penalty for approaching the boundary decreases [109]. Faster linear algebra reduces the work per solve [123, 139, 27, 140], while exact recovery still has to account for the input precision [64]. Even on the interval $[0, 1]$, the objective $x \mapsto 2^{-L}x$ gives the two vertices values only 2^{-L} apart. Intervals of radius 2^{-k} around these values are disjoint only for $k > L + 1$. A single exact sign comparison selects the optimal endpoint.

Removing dependence on the data. Tardos first obtained a strongly polynomial algorithm for minimum-cost circulation [133], then showed that the arithmetic operation count for a rational LP can depend on the encoding length of the constraint matrix alone [134]. This gives a strongly polynomial bound whenever the entries of the matrix have polynomial encoding length in the dimensions, including matrices with entries in $\{0, 1, -1\}$. Frank and Tardos [57] replace a vector by a short integer vector whose inner products with bounded integer vectors have the same signs. When the constraint matrix has short entries, this lets one compress the objective and the right-hand side while preserving the comparisons needed for optimization [71, §6.6].

A different source of strongly polynomial bounds is structure in the constraint matrix. Megiddo solved LP in linear time for fixed dimension [106], and solved feasibility for inequalities involving at most two variables each [105]. Generalized flow maximization supplies a related class with at most two nonzeros per column [143, 111]. Dadush, Koh, Natura, Olver and Végh obtained a strongly polynomial optimization algorithm for every matrix with at most two nonzeros in each row or at most two in each column [34]. Their later work treats classes for which the central path has polynomial straight-line complexity [35].

Geometry of the constraint matrix. For a real matrix A, Vavasis and Ye [141] gave a path-following algorithm whose iteration bound depends on the dimensions and the logarithm of a condition measure of A. Their layered least-squares steps can cross several numerical scales at once. A related condition measure in standard form $Ax = b$, $x \geq 0$ has a concrete description. A *circuit* of A is a nonzero kernel vector of minimal support. The *circuit imbalance* is the largest ratio of magnitudes of two nonzero coordinates in the same circuit [124, 50]. Dadush, Huiberts, Natura and Végh [33] obtained an iteration bound in terms of the best condition achievable by diagonal scaling of the columns. Circuit imbalance also controls the precision needed in an approximate-to-exact solver [37].

Even the best rescaling can leave large ratios in a fixed dimension. For $0 < u < 1$, the matrix

$$A = \begin{pmatrix} 1 & 0 & 1 & u \\ 0 & 1 & u & 1 \end{pmatrix}$$

has circuits $(-1, -u, 1, 0)$ and $(-u, -1, 0, 1)$. The ratios of the magnitudes of their first two coordinates are $1/u$ and u. Any positive diagonal column scaling multiplies these two ratios by

the same factor, so at least one ratio or its reciprocal is at least $1/u$. A separate obstruction comes from the central path itself. Allamigeon, Benchimol, Gaubert and Joswig constructed instances requiring exponentially many segments to follow the logarithmic central path in a prescribed neighborhood [3]. Allamigeon, Gaubert and Vandame extended this obstruction to path-following with self-concordant barriers [5].

Edge paths and rescalings. Early applications ranged from Kantorovich's production planning [86] to the computation of optimal strategies in zero-sum matrix games by LP duality [61]. Dantzig's simplex method exploits the geometry of these problems by moving between adjacent feasible bases [40]. Klee and Minty exhibited exponentially long runs for a classical choice of pivot [90]. Randomized pivot rules achieve subexponential bounds [84, 102], and smoothed analysis gives polynomial expected bounds in the dimensions and the inverse noise scale [132, 9]. Awoniyi has proposed a different pivot-based approach to strongly polynomial LP [8].

Even the shortest possible edge walk can be long. Kalai and Kleitman proved a quasipolynomial diameter bound for polyhedral graphs [85], and Santos disproved the original Hirsch bound of the number of facets minus the dimension [125]. The companion paper [116] constructs simple rational polytopes whose graph diameter grows faster than any polynomial in their number of facets. A circuit walk can leave the edge graph by taking a maximal feasible step in a circuit direction. Natura constructs polynomial-length monotone circuit walks between specified vertices [108].

The algorithm of Theorem 1 searches over coordinate rescalings. In earlier rescaling algorithms, finding that one coordinate must be small in every normalized feasible point can guide a change of scale that improves the geometry [13, 47, 26, 38, 118]. Here a rescaling selects the coordinates on which a primal or dual witness can be detected. Section 2 explains how the algorithm combines such observations, even when the initial system has arbitrarily large circuit imbalance.

Contents

1	Introduction	1
2	Finding exact certificates from partial signs	5
3	Linear programming by pricing	9
4	Calibrating the geometry to prices	14
5	Logarithmic compression on demand	17
6	Learning from extremal prices	23
7	Quadratic programming and complementarity	32
8	Contractions in max norm	38
9	Games on graphs	41
10	Tropical feasibility and further questions	50
	References	51

2 Finding exact certificates from partial signs

Consider first the feasibility problem $Ax \le b$. The algorithm searches for the support of a certificate. Once that support is known, the certificate itself can be recovered by linear algebra. The difficulty is that a coordinate essential to the certificate may be arbitrarily small compared with the others.

2.1 Which side contains a witness?

Collect all possible slack vectors, together with a homogenizing coordinate, into the subspace

$$C = \{(\alpha b - Ax, \alpha) : x \in \mathbb{R}^n,\ \alpha \in \mathbb{R}\}.$$

If C contains a nonnegative vector with positive last coordinate, division by that coordinate gives a feasible point. Its orthogonal complement consists of the vectors $(y, -b^\top y)$ with $A^\top y = 0$. A nonnegative vector there with positive last coordinate is precisely a Farkas certificate of infeasibility.

More generally, let $C \subseteq \mathbb{R}^N$ be any subspace and let $[N] = \{1, \dots, N\}$. Each coordinate i has a unique *side*. Its side is $+$ if some nonnegative vector in C has positive ith entry, and $-$ if such a vector lies in $C^\perp$. Farkas' lemma gives existence, and orthogonality rules out having both witnesses [53]. On the side of i one can choose a witness that is a *circuit*, a nonzero vector whose support is inclusion-minimal among the supports of nonzero vectors in its subspace [124].

Normalize a witness to have largest entry one. An entry of size 2^{-T} can be difficult to detect even when it is indispensable. Multiplying coordinate i of C by 2^{-p_i} can bring that entry into view. Multiplying the same coordinate of $C^\perp$ by 2^{p_i} preserves orthogonality. We call p_i its *price*. Figure 1 illustrates why the information at a single price vector may be incomplete.

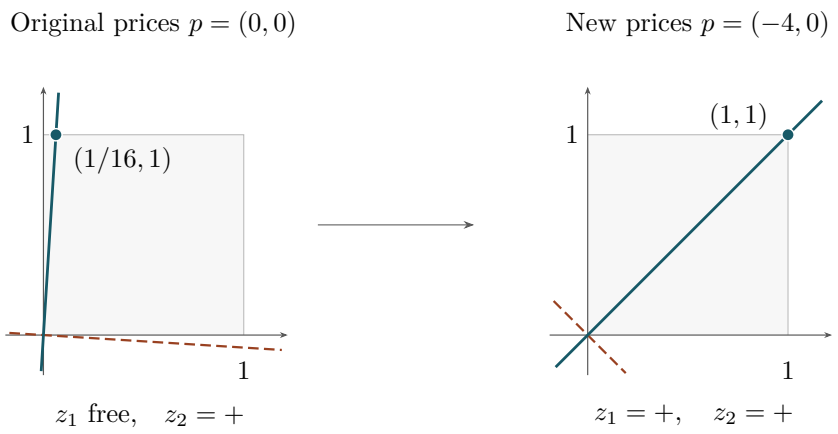

Figure 1. Making a small witness coordinate visible. The solid line is the scaled primal space for $C = \mathrm{span}\{(1/16, 1)\}$, and the dashed line is its orthogonal complement. The dot is the nonnegative primal vector of sup norm one. Multiplying its first coordinate by 16 and the corresponding dual coordinate by 1/16 preserves orthogonality. At the new prices both coordinates reach the visibility threshold 1/2, so both labels are forced. Coordinates 1 and 2 are horizontal and vertical, respectively.

At a given p, imagine a test that labels i with the correct side whenever some scaled, normalized nonnegative vector has ith entry at least 1/2. The other labels may be arbitrary. How could these answers decide *every* side? Fix a nonnegative circuit u on side $h \in \{-1, +1\}$. For each i in its support, the *slot* $s = (i, h)$ records the coordinate and its side. Assign it an integer *demand*

$$a_s = \lfloor \log_2 u_i \rfloor + b_u,$$

with a common shift b_u chosen so that all demands are positive. At any slot maximizing $a_s - hp_i$, the scaled entry is within a factor of two of the largest scaled entry of u. The label is therefore correct at every such maximizer, even though the test reveals neither the circuit nor the maximizing slots.

Theorem 2 (Extremal prices)**.** *Let $N, R \geq 1$ be integers and let $\mathcal{S} = [N] \times \{-1, +1\}$. There is a deterministic algorithm that queries price vectors $p \in \mathbb{Q}^N$, receives a label vector $z \in \{-1, +1\}^N$ for each query, and returns $x \in \mathbb{Q}^N$ with the following guarantee.*

A hypothesis consists of a nonempty set $S \subseteq \mathcal{S}$ and integer demands $a_s \in [-R, R]$ for $s \in S$. Its surplus at $s = (i, h)$ is $d_s(p) = a_s - hp_i$. Call the hypothesis consistent *if at every query and answer (p, z),*

$$s = (i, h) \in S, \qquad d_s(p) = \max_{u \in S} d_u(p) \geq 0 \quad \Longrightarrow \quad z_i = h.$$

For each consistent hypothesis the output meets all its demands, that is, $hx_i \geq a_s$ for every $s = (i, h) \in S$. The bit time, including all query and answer bits, is polynomial in N and $\log(R + 1)$.

For the ideal test, every nonnegative circuit defines a consistent hypothesis as soon as R bounds its demands. Since each coordinate has a witness circuit and its demand is positive, the sign of x_i gives its side. The prices returned by the theorem need only encode these signs. The separate queries have already supplied the evidence.

2.2 How to answer a price query

Two numerical obstacles remain. A circuit can have entry ratios of arbitrary magnitude, so the bound R from the ideal test can be arbitrarily large at fixed dimension. Even when a rational price p_i has few bits, the scaling factor 2^{p_i} may require prohibitively many bits to represent. The calibration procedure in Theorem 3 gives the search a short description of the geometry it needs. Figure 2 shows how a query is answered.

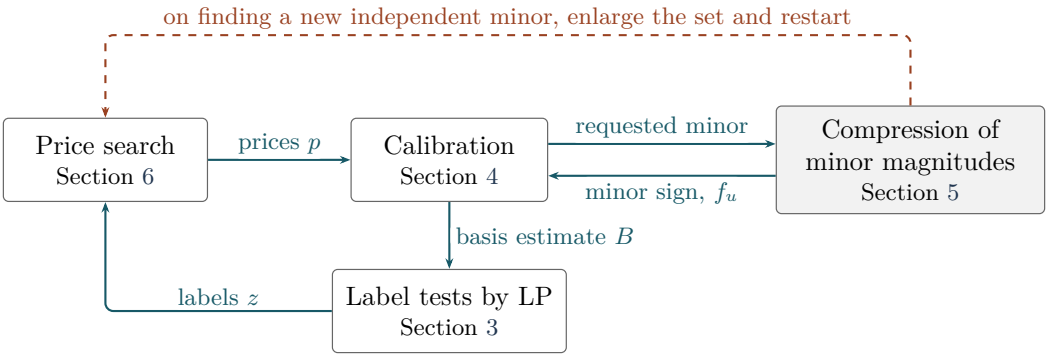

Figure 2. Answering queries during a side decision. The shaded step uses the original scalar entries. For a nonzero minor it supplies the sign and a short substitute f_u for the logarithm of its magnitude. The other steps use rational data of polynomial bit length. Each restart clears the price search and retains the enlarged set of minor indices. Once the search finishes, the signs of its returned prices give the sides.

Apply calibration to a row basis B_0 of $W = C \times C^{\perp}$, with one column for each slot. Between successive restarts, the queries share a single assignment of positive integer demands to every circuit of W. The bit length of these demands is bounded by a polynomial in the dimension and the chosen precision. At each queried p, calibration returns a rational matrix B. For the analysis there is a reciprocal diagonal scaling D with two properties. First, B closely approximates a row basis of WD in which each vector of sup norm one has bounded coefficients. Second, every slot of maximum surplus for a circuit has scaled entry of magnitude at least one third of its largest scaled entry. One D works for all circuits at this query. Its entries may be very large, but only B has to be computed.

To get labels from B, solve an ordinary bit-polynomial LP for each coordinate. Give i label $+$ if a bounded row combination of B is nearly nonnegative, stays in a bounded box and has $(i, +)$ entry at least $1/4$. Otherwise give it label $-$. The gap between $1/3$ and $1/4$ absorbs the approximation error. Opposing slots cannot both pass. In the true scaled space every vector v has $\sum_i v_{i+} v_{i-} = 0$ by orthogonality. If both tests passed, the midpoint of two passing combinations would have large entries in both slots and only tiny negative entries elsewhere, making this sum bounded away from zero. Section 3 makes the error bounds explicit. A circuit of C or $C^{\perp}$ embeds in W by zeros on the other side, so all witness circuits remain consistent through the final uninterrupted run of the price search.

Why minors suffice. Elimination and the description of circuits use the same data. Fix all but one column in a square minor of B_0, then vary the last column. The resulting determinants give a vector in the row space. Taking the fixed columns to be preceding pivots gives the next elimination row. Taking independent columns outside a circuit's support gives that circuit

up to scale. Consequently the magnitudes needed to choose pivots and assign demands are governed by the same numbers, the logarithms ℓ_u of the absolute nonzero minors.

Calibration uses substitutes f_u of short bit length. They preserve the signs, including zeros, of integer affine forms with bounded coefficients in a slight perturbation of the logarithms. To see why this suffices, choose a pivot using f_u and the queried prices. Record, on a fine grid, the scaled log differences from this pivot for the other entries. For tiny relative magnitudes only an upper bound is needed. Requiring prices to reproduce these intervals gives linear inequalities. Eliminating the prices leaves affine inequalities in the log data with bounded integer coefficients. They hold for f, hence also for the perturbed logs. A common price vector for the latter therefore exists. In the true geometry it realizes the same normalized cofactor entries to the accuracy needed for B and for the visibility bound (Section 4).

Why few scales suffice. There may be exponentially many nonzero minors. Section 5 extends Frank–Tardos rounding [57] to requests for one minor at a time. The reason this works is a counting argument. Suppose the subset sums of k minor logarithms all differ by more than two. The corresponding 2^k products of minors then have magnitudes that more than double in sorted order, so their 2^{2^k} subset sums are distinct. Yet each sum is the value of an integer polynomial in the matrix entries, of bounded degree and coefficient size. Counting these polynomials bounds k by a polynomial in the dimensions.

The procedure keeps a set of minor indices with this separation property for accurate proxies of the logs. Lattice reduction [95] provides a sequence of scales for the set. On a request, the procedure tries to express the new proxy in those scales using bounded integer digits and a small remainder. Success gives a short substitute, and all answers since the last restart extend to an assignment for every nonzero minor. Failure proves the index can be added to the separated set. The search then starts again with a larger set, which the counting bound permits only polynomially many times. Comparisons for extracting digits use ordinary arithmetic on the inputs. Small logarithms can be approximated directly. When a comparison involves an exceptionally large logarithm it can instead be decided by comparing products of powers of minors. In that case the input itself has enough bits, on rational data, to bound the bit length of the powers.

Once the search returns all side signs, actual witnesses can be recovered by testing which coordinates can be set to zero while retaining a witness at the desired coordinate. At minimal support the restricted space is a line. Solving on that line recovers the witness in the original scale. Applying feasibility to the recession and primal–dual optimality systems then gives the alternatives in Theorem 1.

2.3 Learning all demands from extremal labels

It remains to explain how Theorem 2 turns labels into prices. First suppose we can afford one expert per hypothesis. Give each expert an initial unit of wealth and let it bet on a slot of greatest surplus, with the option of abstaining at utility zero. Choose prices to minimize the wealth-weighted sum of maximum utilities. At this minimum, bets on opposite signs of each coordinate balance if ties are split suitably. Multiplying wealth according to agreement with the labels can therefore keep total wealth from increasing, even with ties resolved against each expert. An expert for a consistent hypothesis never loses wealth. Whenever any of its demands is unmet, every maximizing slot has a true label and the expert's wealth grows by a fixed factor. This can happen only logarithmically many times in the number of experts. A majority

of queries can thus meet each consistent hypothesis's demands, and taking a coordinatewise median preserves each such bound. The choice of prices is related to the balanced points of Chen, Li and Yannakakis [23].

The full pool is too large to process. An expert who guesses the demands on just four slots, however, may miss the single largest surplus and receive no useful label. The recursive search in Section 6 amplifies the feedback, building on Soleymani and Farina [131] and on earlier algorithms for games [112, 113, 114]. From a proposal y, make a child call that returns prices x. Use the sign of $x_i - y_i$ as feedback at y, and query the label at x to break ties. Suppose the child has brought every surplus down to a threshold $t' \geq 0$. For any slot $s = (i, h)$ with $d_s(y) \geq t'$,

$$d_s(x) \leq t' \leq d_s(y) \qquad \Longrightarrow \qquad hx_i \geq hy_i.$$

Even equality gives correct feedback, since that slot then maximizes surplus at x. A single child can therefore give correct feedback on every slot above a common threshold.

The analysis assigns a weight distribution on slots and a threshold to each node for each consistent hypothesis. If few slots by weight violate the current threshold, the child retains it. When many violate it, the child takes a higher threshold at a surplus quantile. Success then gives correct feedback on an upper fraction of all slots, which four sampled slots have a constant chance to hit. Experts guess the threshold from the price history as well as the four demands. A median combines the children, and increasing the weight of persistently violated slots limits the necessary depth to $O(1 + \log N)$.

Polynomial running time also requires controlling the number of children. An expert invests a fraction of its wealth in a child while using the rest to bet locally. A failed child must return enough wealth to some experts to compensate for losses on its feedback. This accounting allows a constant number of children below the root and rules out failure at the root. Finally, experts guessing the same four slots and the same threshold pivot can be grouped by their at most five demand values. The regions on which those experts act alike have fixed dimension, so their collective bets can be computed in polynomial bit time.

Other uses of the price theorem. For monotone linear complementarity, the signs identify which member of each complementary pair must vanish for a solution on the selected support. A convex quadratic test supplies labels, after which linear feasibility gives the solution (Section 7). For a contraction in the max norm, a coordinate of greatest fixed-point error moves toward the fixed point under the map. This gives extremal labels from function values (Section 8). Discounted value equations for turn-based games furnish piecewise-affine examples. Section 9 uses these equations together with LP policy evaluation to recover exact values and strategies.

3 Linear programming by pricing

For a subspace $C \subseteq \mathbb{R}^N$, consider the two nonnegative cones in C and $C^\perp$. At each coordinate i, exactly one of these cones contains a vector positive at i. In the homogenized slack space of a linear program, the two possibilities at the homogenizing coordinate give a feasible point and a certificate of infeasibility, respectively. We first determine which cone contains the witness, then recover the witness itself with inclusion-minimal support.

Call C the positive side and $C^\perp$ the negative side. Orthogonality excludes simultaneous witnesses at i. If the system $v \in C,\ v \geq e_i$ is infeasible, Farkas' lemma gives a nonnegative

vector in $C^\perp$ positive at i, proving that one side always has a witness. Moreover, a witness u of inclusion-minimal support is a circuit. If there were a second independent vector on its support in the same subspace, a linear combination would give a nonzero direction supported there and vanishing at i. Move from u along that direction, choosing the sign that decreases some entry, until the first entry vanishes. This yields a witness with smaller support. For the relation with elementary vectors and the classical theorems of Minty and Tucker, see [124, Theorem 6 and its corollary] and [138].

3.1 Reciprocal scaling and visibility

To use Theorem 2, we need a label at each coordinate of each queried price vector. Scale coordinate i on the positive side by a positive factor and on the negative side by its reciprocal. This preserves orthogonality and can bring a small circuit entry into view. The following theorem chooses the effective scaling for each query and describes its effect using a rational matrix of short bit length.

Write $\mathcal{S} = [N] \times \{-1, +1\}$ and abbreviate the slots $(i, \pm 1)$ by $i\pm$. A *reciprocal scaling* is a positive diagonal matrix D on these slots with $D_{i+}D_{i-} = 1$ for all i. For row vectors, scaling acts by right multiplication. The scale for a query p will make every slot maximizing surplus visible, even when the factors $2^{-p_i}, 2^{p_i}$ would be too costly to represent.

Theorem 3 (Calibration)**.** *Given an ordered row basis B_0 of a real subspace W on the paired slots of $[N]$, $N \geq 1$, and an integer precision parameter $P \geq 1$, there is a deterministic procedure answering online price queries $p \in \mathbb{Q}^N$ by rational matrices B. On a query it may instead declare a restart.*

At any time, all answers since the most recent restart, or since initialization, admit the same assignment of demands. For each circuit u of W the assignment gives an integer $a_s \in [1, R]$ at each slot s in its support. For each such answer B at prices p, there is a reciprocal scaling D with the following properties.

- *B approximates a true row basis $B_* = LB_0D$ of WD to within 2^{-P} entrywise, with L lower triangular with positive diagonal. Entries of B_* have absolute value at most 2. Any vector in WD of sup norm one has row coefficients in this basis of absolute value at most 2^{100N^2}.*
- *For every circuit u, at each slot $s = (i, h)$ maximizing surplus $a_s - hp_i$ on the support of u,*

$$|(uD)_s| \;\geq\; \tfrac{1}{3}\|uD\|_\infty.$$

The integer bound $R = 2^{\mathrm{poly}(N,P)}$ is available at initialization, and the total number of restarts is polynomial in N, P. The procedure is strongly polynomial when the value of P, the number of queries, and the bit lengths of the prices count as additional discrete input size. It returns the answers in explicit binary form.

The algorithm receives the approximate basis B. The true basis B_*, the scaling D, and the demands certify the properties of that answer. The triangular factor L will also preserve the row order and orientation needed in Section 7.

3.2 From approximate labels to exact witnesses

Embed the two sides in $W = C \times C^{\perp}$, with C on the positive slots and $C^{\perp}$ on the negative slots. A witness circuit of either side is still a circuit in W after extension by zero on the other side. The quadratic form

$$Q(y) = \sum_{i=1}^{N} y_{i+} y_{i-}$$

vanishes on W and on every reciprocally scaled space WD. This identity lets us label a coordinate even though calibration returns only an approximate basis.

Testing one side. Apply Theorem 3 to a row basis of W, taking

$$H = 2^{100N^2}, \qquad \epsilon = H^{-10}, \qquad 2^{-P} = H^{-100}. \tag{3.1}$$

Run the algorithm of Theorem 2 with the resulting bound R. At each price query, request the calibrated basis. If calibration declares a restart, begin a fresh price search, retaining the state of the calibration procedure. Otherwise let B be the matrix returned. Form the convex set

$$\mathcal{B}(B) = \{y = \xi B : \|\xi\|_{\infty} \le H, \quad -\epsilon \le y_s \le 2 \quad (s \in \mathcal{S})\}. \tag{3.2}$$

Give label $+$ at i if some $y \in \mathcal{B}(B)$ has $y_{i+} \ge 1/4$, and give label $-$ otherwise. Each test is linear feasibility on explicit short rational data, so ordinary bit-polynomial LP suffices [88, 71].

To see why testing the positive slot suffices, write a candidate as $y = \xi B$ and compare it with $y_* = \xi B_*$ in the true scaled space. There are at most $2N$ rows, so $\|y - y_*\|_{\infty} \le 2NH2^{-P} < \epsilon^2$. Since $Q(y_*) = 0$ and the entries of y belong to $[-\epsilon, 2]$, we have $Q(y) \le 5N\epsilon^2 \le H^{-1}$. The following observation will also apply in quadratic programming.

Lemma 3.1. *Let K be a convex set of vectors on the paired slots, with $-\epsilon \le y_s \le 2$ and $Q(y) \le H^{-1}$ for every $y \in K$. For each i, at most one of its two slots attains $1/4$ or more anywhere in K.*

Proof. If a vector attaining the threshold exists at each slot, their midpoint has both entries at least $(1/4 - \epsilon)/2 \ge 1/10$. Every other pair contributes at least -2ϵ. Thus the midpoint belongs to K and has $Q \ge 10^{-2} - 2N\epsilon > H^{-1}$, a contradiction. □

Deciding all coordinates. In the final run between restarts, calibration supplies a common assignment of positive demands to every circuit. Fix a nonnegative circuit u and consider any query p. If the slot $s = (i, h)$ maximizes the surplus of this circuit, its normalized scaled vector

$$y_* = uD/\|uD\|_{\infty} = \xi B_*$$

has $(y_*)_s \ge 1/3$ and admits coefficients with $\|\xi\|_{\infty} \le H$. All entries of y_* lie in $[0, 1]$. The same approximation estimate shows that ξB belongs to $\mathcal{B}(B)$ and attains at least $1/4$ at s. For $h = +1$ the positive test therefore succeeds. For $h = -1$ Lemma 3.1 forces the positive test to fail. In either case the returned label at i equals h.

Thus the slots and demands of each nonnegative circuit form a consistent hypothesis for Theorem 2. The returned prices x satisfy $hx_i \ge a_s \ge 1$ at *every* slot of that circuit. Every coordinate has a witness circuit, so $\operatorname{sgn}(x_i)$ identifies the side of every coordinate. Calibration restarts only polynomially often. In each run Theorem 2 bounds the number of queries and their bit lengths by a polynomial in $N, \log(R + 1)$. The calibrated answers and the rational label tests then have strongly polynomial total cost by Theorem 3.

Recovering one circuit. Fix a target coordinate i and take the subspace on its indicated side. For each coordinate $j \neq i$ in turn, tentatively impose $v_j = 0$ on the current subspace. Keep the restriction exactly when the side decision for this smaller subspace says it still contains a nonnegative vector positive at i. A rejected restriction can never become feasible later, since each remaining step shrinks the subspace. Consequently every final witness is positive on every unrestricted coordinate. The direction argument at the start of this section now shows that the final subspace is a line. Linear algebra recovers its generator, oriented to be positive at i, and hence a witness circuit in the original subspace. At each trial we compute a basis from the original input and the retained zero coordinates. The bit bounds are given in Section 3.4.

3.3 Certificates for a linear program

Proof of Theorem 1. For $Ax \leq b$ use the homogenized slack space and its orthogonal complement,

$$C = \{(\alpha b - Ax, \alpha) : x \in \mathbb{R}^n,\ \alpha \in \mathbb{R}\},$$
$$C^\perp = \{(y, -b^\top y) : A^\top y = 0\}.$$

Recover a witness circuit at the last coordinate. On the positive side, divide the witness by its last coordinate and solve for x by linear algebra. This gives $Ax \leq b$. On the negative side, the witness gives $y \geq 0$, $A^\top y = 0$, and $b^\top y < 0$. The codimension of $C^\perp$ is $1 + \operatorname{rank} A$. A circuit in a space of codimension k has at most $k + 1$ entries in its support, since its support indexes a minimally dependent set of columns of a defining matrix. The last entry of this circuit is positive, leaving at most $1 + \operatorname{rank} A$ entries for y.

This also solves feasibility with both strict and weak inequalities. Homogenize with a variable α, require $\alpha \geq 1$, and require each slack belonging to a strict inequality to be at least one. A solution of the original mixed system can be scaled to meet all these lower bounds. Dividing by α gives the converse.

For a feasible program, test $Ar \leq 0,\ c^\top r > 0$ by this procedure. A solution, together with the feasible point already found, certifies unboundedness. Otherwise LP duality gives finite primal and dual optima with a common value v. Strict complementarity [69] ensures that some optimal pair satisfies $b - Ax + y > 0$. For completeness, take a coordinate i whose slack is zero at every primal optimum. Maximize that slack subject to $Ax \leq b$ and $c^\top x \geq v$. Duality for this auxiliary program, which has value zero, gives $u \geq 0$ and $\lambda \geq 0$ with

$$A^\top(u + e_i) = \lambda c, \qquad b^\top(u + e_i) = \lambda v.$$

If $\lambda > 0$, dividing $u + e_i$ by λ gives a dual optimum positive at i. If $\lambda = 0$, adding $u + e_i$ to any dual optimum has the same effect. Averaging the primal witnesses for the coordinates admitting positive slack, and the dual witnesses for those with necessarily zero slack, gives a strictly complementary pair. If one of these collections is empty, any optimum on that side suffices.

The algorithm recovers the pair directly, without knowing v. Solve the single mixed linear feasibility problem

$$Ax \leq b, \quad y \geq 0, \quad A^\top y = c, \quad b^\top y = c^\top x, \quad b - Ax + y > 0.$$

The equality makes the duality gap $(b - Ax)^\top y$ zero. This returns the optimal pair with the required strict complementarity. □

3.4 Arithmetic and bit lengths

The bounds used above apply to one uniform program for all input dimensions. Our control operations on explicit strings (indices or short rationals generated by the algorithm) are charged full bit cost, and can use any polynomial-time bit algorithm on strings of polynomial length in the dimension. We take standard explicit input with dimensions and a nonzero end marker (so size is at least the dimensions, also in the bit bounds). To simulate the algorithms on the real machine of [18], store each explicit bit in a cell. Even sequential memory scans that increment or reset indices and copy cells incur only polynomial overhead. All polynomial bounds are fixed and effective. When such a bound is used as an algorithmic parameter, one can fix it by composing the estimates in the proof. Logarithms are to base two unless indicated by ln.

We can implement all scalar arithmetic using pairs (u, v), meaning u/v ($v > 0$), and perform only ring operations and comparisons aside from possible final divisions. This representation admits two input conventions.

- Input a scalar x directly, starting with $(x, 1)$. Measure bit lengths on rational data by *reduced* fractions (of the entries u, v themselves, and similarly for intermediates).
- Give each rational scalar as numerator and denominator (integer scalars). Compute with integer pairs and return the resulting unreduced pair for each output scalar. Measure bit lengths by the full sizes of these integers. This uses only the most restrictive division convention in [71, §1.3]. See also [34] for a discussion of arithmetic models.

These bounds give a polynomial-time Turing simulation. A simulation may reduce fractions in the first model and store the integer pairs as given in the second. Neither algorithm itself uses fraction reduction in scalar arithmetic. Short rationals *given or constructed in bits* may be freely placed in scalar computations by setting up their numerators and denominators bit by bit.

To cover both versions, on rational data take as a pair's log height the log of the least bound on absolute numerators and denominator for writing u, v with one positive denominator. All results and intermediates must have log height polynomial in total input length (including bit sizes of any input pairs). The usual pair operations incur log height at most the sum for their arguments plus one (take the product of their two common denominators), including for cross-products when comparing. This gives the bit bound for arithmetic by polynomial-size formulas (size counting all leaf occurrences).

We use ordinary linear algebra on explicit real matrices. We give the determinant routine explicitly because repeated unreduced fraction arithmetic in elimination could increase the bit lengths (see also [71, §1.4]). Multiply the given $k \times k$ matrix by the product D of all pair-denominators v, forming each new entry by multiplying its numerator by all denominators except its own. Thus the resulting matrix is $M = DB$, where B denotes the input matrix. Compute $k! \det M$ by Newton's identities. If c_i is the coefficient of t^i in $\det(I + tM)$, the recurrence is

$$i!c_i = \sum_{j=1}^{i} (-1)^{j-1} \frac{(i-1)!}{(i-j)!} ((i-j)!c_{i-j}) \operatorname{tr}(M^j), \qquad 1 \le i \le k.$$

Use these division-free ring operations (factorial ratios are products) and return $(k!c_k, k!D^k)$. On rational data all entries of M and D have a common denominator U of polynomial bit length with polynomial-bit numerators. The degree i results here can use denominator U^i,

with polynomial-bit numerators (by determinant expansion for the c_i, and similarly for matrix powers and then partial results in the recurrence). The algorithm applies the recurrence to the scalar entries of M using ring arithmetic. The common denominator U serves only to bound reduced bit lengths in the first input model. In the integer-pair model all entries of M are integers, so $U = 1$.

Use bordered minor tests to pivot for rank (an invertible minor of order less than the matrix rank can be enlarged, by the Schur complement), and Cramer's rule for solving and obtaining bases. A fixed number of successive linear-algebra computations and polynomial-size formulas therefore preserves polynomial bit length. In witness recovery the number of coordinate restrictions can grow with the dimension. Each basis there is computed from the original input and the current list of zero coordinates, which keeps the number of successive linear-algebra computations on the scalars bounded.

Ordered fields. For any fixed input shape, unroll the field algorithm to its uniform step bound. Each computation branch and its output conditions can be expressed in the language of ordered rings, with division by a nonzero denominator expressed by multiplication. Tarski's transfer principle thus gives correctness with the same step bound over every real closed field. For an arbitrary ordered field, run the argument in a real closure. All operations on inputs from the original field stay in that field, so the same certificates are obtained there.

4 Calibrating the geometry to prices

Theorem 3 asks for two guarantees from one scaling. Elimination must produce a basis with bounded entries and bounded expansion coefficients for unit vectors, and every circuit must have a large coordinate wherever its assigned surplus is maximal. The same matrix minors govern both requirements. We first express them as comparisons of minor magnitudes, then show how to make these comparisons using only short rational data.

4.1 Cofactors for elimination and for circuits

Let B_0 be the given ordered basis, with r rows. Index its nonzero minors by their row and column sets u, and write ℓ_u for the base-two logarithm of the absolute determinant. For a list T of $j-1$ distinct columns, $1 \le j \le r$, define a row vector c by

$$c_s = \det B_0[[j], (T, s)]. \tag{4.1}$$

The entry vanishes when s repeats a column of T. Expansion along the last column expresses c as a combination of the first j rows of B_0. The coefficient on row j is $\det B_0[[j-1], T]$, with the empty determinant equal to one.

There are two uses for this formula. In elimination, T is the list of earlier pivots. Its determinant is nonzero, so c supplies the next elimination row, with zeros at all earlier pivots. For a circuit, take $j = r$. The vectors of W supported on that circuit form a line, so the columns outside its support have rank $r-1$. Choosing $r-1$ independent columns there gives a cofactor vector that vanishes outside the circuit. It is nonzero because B_0 has rank r, and hence represents the circuit up to a scalar.

Consider for the moment the reciprocal scaling $D(p)_{i,h} = 2^{-hp_i}$. Given a cofactor list T and a slot $s = (i, h)$ on its support, let $u(s)$ be the associated minor index. For a vector y of values on the minor indices, set

$$\rho_s(y, p) = y_{u(s)} - hp_i, \qquad \rho_s(\ell, p) = \log |(cD(p))_s|. \tag{4.2}$$

Consequently a maximizing slot is a largest entry in absolute value, and the difference from that maximum gives the logarithm of a normalized entry magnitude.

The algorithm makes the comparisons using short rational substitutes f_u for the logs. These substitutes need not have the same numerical size as the logs. What we require is that for each query p there exist prices $\widehat{p}$ at which the true normalized entry magnitudes agree, to the requested absolute accuracy, with those computed from (f, p). One vector $\widehat{p}$ must work for all the cofactor rows at that query, including circuits on which the algorithm never requests a minor. Our construction will prove existence of $\widehat{p}$ by projecting a system of linear inequalities.

4.2 Linear types and online compression

For two real vectors y, y' on the same finite index set, write $y \sim_M y'$ when every integer affine form of coefficient 1-norm at most M, including its constant, has the same sign in $\{-1, 0, 1\}$ on y and y'. This agreement of linear types lets us transfer feasibility from one parameter vector to the other. Compare the use of Frank–Tardos rounding in [71, Lemma (6.6.1)].

Lemma 4.1 (Projection)**.** *Consider a finite system of linear inequalities, strict or weak, in k real unknowns and additional parameters y. Let all its affine forms have integer coefficients and coefficient 1-norm (including constant) at most A, where $A \geq 2$. Then feasibility is the same at any two parameter vectors with $y \sim_M y'$, where $M = (k + 1)A^{k+1}$.*

If there are no parameters and the system is feasible, it has a solution with both a common positive denominator and the absolute numerators at most $(A + 1)^{k+1}$.

Proof. Fix the parameters. Give every strict inequality a common additional margin δ and write the resulting inequalities weakly. The original system is feasible precisely when this weak system is feasible for some $\delta > 0$. For fixed δ, Farkas' lemma reduces feasibility to testing nonnegative combinations of the inequalities that eliminate the k unknowns. It suffices to test the extreme rays of this cone of multipliers.

On the support of an extreme ray the multiplier kernel has dimension one. Its support thus has size at most $k + 1$, and subdeterminants give an integral generator whose entries are at most A^k by Hadamard's inequality. Each resulting affine form on the parameters, with δ set to zero, has coefficient norm at most $(k + 1)A^{k+1}$. Require the projected inequality to be strict if a strict input inequality has a positive multiplier, and weak otherwise. These conditions are necessary. They are also sufficient, since the cone has finitely many extreme rays and one sufficiently small $\delta > 0$ preserves every margin required by a strict combination. Agreement of linear types preserves all these tests.

For the second assertion, let δ be a new unknown and add $0 \leq \delta \leq 1$ to the system with margins. Intersect with a closed orthant containing a solution with $\delta > 0$. The new polyhedron is pointed and has a vertex maximizing δ. Its constraint rows, with constants, have coefficient 1-norm at most $A+1$. Cramer's rule and Hadamard's inequality now give the claimed common denominator and numerator bounds. □

The next theorem, proved in Section 5, furnishes substitutes that agree in linear type with a small perturbation of the true logs. A price query will use only the entries requested during elimination. The guarantee for all the remaining minors is what makes the conclusion about unseen circuits possible.

Theorem 4 (Online compression)**.** *Given an $r \times d$ real matrix ($0 \le r \le d$, $d \ge 1$) and $K \ge 1$ an integer, index its nonzero minors by u (specifying row and column sets) and let ℓ_u be the base-two logarithm of their absolute values. There is a deterministic procedure that, on request of an individual index u, supplies a rational f_u or declares a restart.*

At any time the answers since the most recent restart (if no restart, all answers) admit a common assignment f of rationals to all *the indices with a common positive denominator and absolute numerators each bounded by $F = 2^{\mathrm{poly}(d,K)}$, such that*

$$f \sim_{2^K} g \quad \text{for some } g \text{ with } \|g - \ell\|_\infty \le 2^{-K}. \tag{4.3}$$

The total number of restarts is bounded by a polynomial in d. Initialization and each request (including responses with explicit bits or restarts) take strongly polynomial time, counting the value of K as additional input size. The integer bound F is available at initialization.

4.3 The common lift

For P in Theorem 3, set

$$E = P + 6, \qquad e = 2^{-E}, \qquad A = (E+5)2^E$$

and take K least with $2^K \ge (N+1)A^{N+1}$. Fix price data p and assignments f, g for B_0 as in (4.3). We encode the normalized cofactor magnitudes at (f, p) as follows. For each nonzero cofactor vector take the slot t maximizing $\rho_t(f, p)$, using a fixed tie rule. For each slot s on the same support write

$$w = \rho_s(f,p) - \rho_t(f,p) \le 0, \qquad \Delta = \rho_s(y,\widehat{p}) - \rho_t(y,\widehat{p}).$$

In Δ the vector y is the parameter and the N components of $\widehat{p}$ are the unknowns. Impose

$$\begin{cases} e\lfloor w/e \rfloor \le \Delta \le e\lceil w/e \rceil, & w \ge -E, \\ \Delta \le -E, & w < -E. \end{cases}$$

For a ratio smaller than 2^{-E} it suffices to preserve that upper bound. Ratios above the cutoff need only one grid cell to specify their log. All constraints hold at $(y, \widehat{p}) = (f, p)$. Multiplying by 2^E gives integer forms of coefficient norm at most A, independent of the bit lengths of the original prices and the proxies. Lemma 4.1 therefore gives a solution $\widehat{p}$ at $y = g$, since $f \sim_{2^K} g$. The system includes every cofactor list, so the solution is a single price vector shared by all of them. The algorithm never forms this system.

Replacing g by the true log vector ℓ changes each difference by at most $2 \cdot 2^{-K} \le 2e$. Thus at the lifted prices the true difference is within $3e$ of w for $w \ge -E$, and at most $-E + 2e$ for $w < -E$. Taking $D = D(\widehat{p})$ yields

$$\left| \frac{|(cD)_s|}{|(cD)_t|} - 2^w \right| \le 8e. \tag{4.4}$$

In the first case this follows from the mean value theorem applied to 2^x for $x \le 3e$. In the second, both magnitudes are at most 2^{-E+2e}.

4.4 The calibration procedure

Use Theorem 4 on B_0, with K as just specified. Take $R = 4F$. To answer a price query p, grow a sequence of pivot slots T, starting empty. At row $j = 1, \dots, r$, take the cofactor vector with T consisting of the previous pivots. To see there will be a pivot available, recall that this vector c is in the span of the first j rows. Its coefficient on original row j is $d_{j-1} = \det B_0[[j-1], T]$, nonzero by construction ($d_0 = 1$). Thus $c \neq 0$. Request f values for all its nonzero entries. If a restart occurs, abandon the query with that report. Otherwise compute the row as follows.

- As pivot at this row take slot t maximizing $\rho_s(f, p)$ on the support, breaking ties by the rule fixed above. This gives $d_j = c_t \neq 0$ for the next step.
- For row j of the answer B, keep zeros, and at each supported slot s put a rational within e of
$$\operatorname{sgn}(d_{j-1} c_s) \cdot 2^{\rho_s(f,p) - \rho_t(f,p)}.$$

One query visits at most r cofactor rows and requests at most $2Nr$ minor proxies. Signs and zeros use minors of the original matrix, computed as in Section 3.4. Theorem 4 bounds the cost of the requests and the total number of restarts. The rational approximations take polynomial bit time. For exponents below $-E$, use zero. Otherwise, split off the integer power, leaving 2^q with $0 \leq q < 1$. Use $\ln 2 = \sum_{j \geq 1} 2^{-j}/j$ and then the series for $\exp(q \ln 2)$, e.g., through $E + 10$ in each, by geometric tail bounds.

Completion of the proof of Theorem 3 given Theorem 4. Fix any uninterrupted sequence of answers from initialization or the last restart up to the present time. Theorem 4 extends all the returned proxies to one vector f. For each query in the sequence choose the scaling D from its common lift. For row j of B_*, use the corresponding cD with sign adjusted by $\operatorname{sgn}(d_{j-1})$ and scaled to absolute entry one at the pivot. Thus it now has a positive coefficient on row j of $B_0 D$, giving $B_* = L B_0 D$ with L as required. By (4.4), B is within $9e \leq 2^{-P}$ entrywise, and B_* has entries bounded by two in absolute value. Any vector of sup norm one in WD can be expanded with coefficient on row j bounded by $3^{j-1} \leq 2^{100N^2}$ in absolute value. Indeed, subtract row multiples in pivot order. The pivot entry has magnitude one, so a residual of norm M requires a coefficient of magnitude at most M. The row itself has norm at most two, leaving a residual of norm at most $3M$.

For each circuit choose one cofactor representative and assign demands

$$a_s = 1 + F + \lfloor f_{u(s)} \rfloor$$

on its support. Since $|f_{u(s)}| \leq F$, these demands belong to $[1, 1 + 2F] \subseteq [1, R]$. They use the same f at every query in the sequence. Fix a circuit, with t as in the common lift (so maximizing $\rho_s(f, p)$ for this representative). If $s = (i, h)$ maximizes surplus $a_s - h p_i$, then $w = \rho_s(f, p) - \rho_t(f, p) > -1$, by the bound of one on rounding loss. So in the common lift, $\rho_s(\ell, \widehat{p}) - \rho_t(\ell, \widehat{p})$ is greater than $-1 - 3e$. At any slot on this support, the true scaled log difference from t is at most $3e$. Thus after scaling the circuit by D, the entry at s has magnitude at least $2^{-1-6e} \geq 1/3$ times the sup norm. □

5 Logarithmic compression on demand

Calibration requests the sizes of particular minors as it chooses pivots. The choices must also be consistent with minors that never occur in a pivot calculation, since such minors can enter

a circuit. This is the extension requirement in Theorem 4. Write ℓ_u for the base-two logarithm of the magnitude of a nonzero minor. We seek short rational answers to individual requests that extend to a vector of the same linear type as a small perturbation of ℓ.

The procedure learns a set J of minor indices. Between restarts it uses the logarithms at these indices to set common scales for all its answers. A failure to represent a requested logarithm at those scales enlarges J. The bound on the number of restarts comes from the fact that products of minors are polynomials of bounded degree in the same matrix entries.

Throughout, an integer is *short* if its bit length is polynomial in d, K. Fix

$$D = (d+1)^6, \qquad \eta^{-1} = \text{the least power of two at least } 4D \cdot 2^K.$$

We first work with fixed proxies v_u satisfying $|v_u - \ell_u| \le \eta$ and access to the signs of integer affine forms in these proxies. The final subsection constructs proxies for which the required sign tests can be implemented in the arithmetic model of Section 3.4.

5.1 A bound on independent logarithms

Call a set I of minor indices *dissociated* if the subset sums of $(v_u)_{u \in I}$ are pairwise at distance greater than two. Equivalently, every nonzero coefficient vector in $\{-1, 0, 1\}^I$ gives a sum of absolute value greater than two. This is a version with tolerance of the usual additive notion [96]. If I is maximal, every proxy is within two of a signed subset sum on I. Indeed, for an index outside I, any relation preventing its addition must have coefficient $+1$ or -1 on the new index.

Lemma 5.1. *Every dissociated set has size less than D.*

Proof. Suppose there is a dissociated set of size D. The logarithms of the magnitudes of its 2^D subset products of signed minors differ by more than $2 - D\eta > 1$. In increasing order of magnitude each product therefore has magnitude greater than the sum of all preceding magnitudes. It follows that all 2^{2^D} subset sums of these products are distinct at the input matrix. Two unequal subsets differ at a largest product, whose contribution survives cancellation.

These sums are consequently distinct integer polynomials in the at most d^2 matrix entries. Each has degree at most dD and coefficients of absolute value at most $2^D(d!)^D$, by the determinant expansion. Counting possible coefficients at each monomial gives

$$2^{2^D} \le \left(1 + 2^{D+1}(d!)^D\right)^{(dD+1)^{d^2}}.$$

Two logarithms yield $D \le d^2 \log(dD+1) + \log(2 + D(1+d^2)) \le (7d^2+8)\log(d+1)$, contradicting $D = (d+1)^6$. □

Goemans's bound on geometrically decreasing sequences of small integer combinations uses a related counting argument [36, Lemma 3.5 in the preprint], attributed there via Radzik [122]. Here the integer combinations are products of minors expanded in monomials. Counting subset sums brings the number of monomials into the dissociation bound only logarithmically.

5.2 Extending partial answers

Let S contain the indices answered since the last restart. To certify answers on S, we shall enforce agreement of linear type at a larger coefficient bound M on S alone. Lemma 5.2 shows why this suffices. A maximal dissociated set expresses all the missing logarithms, to within a rounded offset, in fewer than D variables. Projecting out these variables determines M.

Lemma 5.2 (Extension)**.** *Set*

$$A = 2^K(D+4)/\eta, \qquad M = (D+1)A^{D+1}.$$

Suppose on a subset S of the indices we have

$$f_S \sim_M g_S, \qquad \|g_S - v_S\|_\infty \le \eta, \tag{5.1}$$

where f_S has a short common denominator and short numerators. Then f_S, g_S extend to vectors f, g satisfying (4.3) on all the minor indices. The vector f again has a short common denominator and short numerators.

Proof. Choose a maximal dissociated set I. For every $u \notin S$, write v_u as a signed subset sum on I plus an offset in $[-2, 2]$. Round the offset to the η-grid, and let $h_u(x)$ be the resulting affine form in $x \in \mathbb{R}^I$. Set $g_u = h_u(v_I)$. On S and its complement alike, $|g_u - \ell_u| \le 2\eta \le 2^{-K}$.

Now impose the sign conditions

$$(y_S,\, (h_u(x))_{u \notin S}) \sim_{2^K} g.$$

Here y_S is a parameter and x is an independent variable, even when I meets S. Write each sign as a strict inequality or as two weak inequalities in the zero case. After clearing the grid denominator, each row has integer coefficients of total absolute value at most A. For $y_S = g_S$ we have the solution $x = v_I$. Since $|I| < D$ by Lemma 5.1 and $f_S \sim_M g_S$, Lemma 4.1 gives a solution also at $y_S = f_S$.

If U bounds the common denominator and the absolute numerators of f_S, the rows after substitution and clearing that denominator have coefficient norm at most AU. The second assertion of Lemma 4.1 supplies x with a common denominator and absolute numerators at most $(AU+1)^{D+1}$. Set $f_u = h_u(x)$ for $u \notin S$. The product of the two common denominators and $1/\eta$ works for the whole vector. Both it and the absolute numerators are bounded by $(D+2)(AU+1)^{D+1}U/\eta$. □

5.3 Extracting common scales

The maximal set I in Lemma 5.2 serves only to prove existence. The procedure maintains its own dissociated set J, starting with the empty set. To decide whether it can answer a request, it needs a test that succeeds on every proxy within two of a signed subset sum on J. Enumerating those sums would cost exponential time. Instead we expand v_J at a sequence of common scales. At each scale the vector of digits is integral and the remaining error is a small fraction of the scale. This is the simultaneous approximation step of Frank–Tardos rounding [57].

Lemma 5.3. *Let $w \in \mathbb{R}^m$, and fix integers $\epsilon^{-1} \ge 2$ and $H \ge 0$. Put $C = 2^{m(m+1)}(2/\epsilon)^{m+1}$. Given the signs of integer-coefficient affine forms on w, one can find integer vectors $z_1, \dots, z_p$ and positive scales $\lambda_1, \dots, \lambda_p$, represented by rational linear forms on w, such that*

$$p \le m, \qquad \|z_j\|_\infty \le C, \qquad \lambda_j > H/C,$$
$$\left\| w - \sum_{i=1}^{j} \lambda_i z_i \right\|_\infty \le \epsilon\lambda_j \quad (1 \le j \le p), \qquad \left\| w - \sum_{i=1}^{p} \lambda_i z_i \right\|_\infty \le H.$$

Counting each sign query as one operation, the bit time and the lengths of all queries and outputs are polynomial in m and the bit lengths of ϵ^{-1}, H.

Proof. Start with remainder $w' = w$. If $X = \|w'\|_\infty > H$, use binary search to find a rational vector u within $\epsilon/(2C)$ of w'/X in each coordinate. Consider the lattice of vectors $(qu - z, q/C)$ for integral q, z, of determinant $1/C$. Lattice reduction [95], as in [71, Theorem (5.3.19)], produces in polynomial bit time a nonzero vector of Euclidean norm at most $2^{m/2}C^{-1/(m+1)} \leq \epsilon/2$. Such a vector must have $q \neq 0$, since a nonzero integer vector has length at least one. Choose its sign to give $1 \leq q \leq C$. Then

$$\|qw'/X - z\|_\infty \leq \epsilon, \qquad \|z\|_\infty \leq q.$$

For the second bound use integrality and $\epsilon < 1$.

Take $\lambda = X/q > H/C$ and replace w' by $w' - \lambda z$. The new remainder has norm at most $\epsilon\lambda$. At a zero coordinate the digit is zero. At a coordinate attaining X in absolute value the digit equals $+q$ or $-q$, so that coordinate now vanishes as well. Thus at most m steps are needed to reach residual norm at most H. For $m = 0$ the expansion is empty.

Each comparison for the binary search can be made after multiplying by $X > 0$. At every stage X is a signed coordinate of the remainder, chosen by comparisons, so both the remainders and the scales are rational linear forms in the original w. A step multiplies their common denominator by at most q and increases the maximum absolute numerator by at most a factor $2C$. Clearing these denominators in the comparisons, including those on the $\epsilon/(2C)$-grid, gives sign queries of the asserted length. □

For a homogeneous form, a nonzero integer digit contributes at least one whole scale. Choosing the residual small compared with that scale lets the first nonzero digit determine the sign. We record the usual Frank–Tardos consequence used to round offsets in Section 8 and, through that computation, to solve games in Section 9.

Lemma 5.4. *Given $w \in \mathbb{R}^m$ and an integer $M \geq 1$, one can find an integer vector preserving the sign on w of every homogeneous linear form with integer coefficients of 1-norm at most M. The procedure is strongly polynomial with the bit length of M counted as additional input size, and returns the entries in bits.*

Proof. Apply Lemma 5.3 with $\epsilon = 1/(2M)$ and $H = 0$ (cf. [71, Theorem (6.2.13)]). For a form whose first nonzero digit occurs on z_j, the remaining contribution has magnitude at most $M\epsilon\lambda_j < \lambda_j$. A form vanishing on every digit also vanishes on w, since the final remainder is zero. Consequently the integer vector $\sum_{j=1}^{p}(2CM+1)^{p-j}z_j$ has the required signs. Its first nonzero digit dominates the geometric sum of all later digits, whose evaluations have magnitude at most CM each. The short linear forms queried in Lemma 5.3 can be evaluated as in Section 3.4. □

For affine forms, append a coordinate equal to one before applying Lemma 5.4. Its rounded value is positive by sign agreement. Dividing the rounded vector by that value preserves the homogeneous signs and restores the last coordinate to one.

5.4 Answering a request

Start each sequence of requests by applying Lemma 5.3 to v_J. Use M from Lemma 5.2 and set

$$m = |J|, \qquad \epsilon^{-1} = 100(m+1)M, \qquad H = 100MC,$$

where C is determined by m, ϵ in Lemma 5.3. All extracted scales λ_j exceed $100M$. They remain fixed throughout the sequence.

For a request at u, look for integer digits $b_{u1}, \dots, b_{up}$ of magnitude at most mC satisfying

$$\begin{aligned} \left| v_u - \sum_{i=1}^{j} \lambda_i b_{ui} \right| &\le \frac{\lambda_j}{10M} \qquad (1 \le j \le p), \\ \left| v_u - \sum_{i=1}^{p} \lambda_i b_{ui} \right| &\le mH + 2. \end{aligned} \tag{5.2}$$

Once the previous digits are fixed, the permitted interval for b_{uj} has length $1/(5M) < 1$. Binary search therefore finds the unique digit or detects failure. Check the last residual bound even if there are no digits.

A failed request enlarges the dissociated set. If v_u lies within two of a signed subset sum on J, take the same signed sum of the digit vectors of v_J at each scale. This gives digits of magnitude at most mC with residuals at most $m\epsilon\lambda_j + 2 \le \lambda_j/(10M)$ at position j and at most $mH + 2$ at the end. Thus failure implies that u lies outside J and that $J \cup \{u\}$ is dissociated. In that case retain the enlarged set, discard the expansion and previous answers, and declare a restart. Each restart increases $|J|$ by one, so Lemma 5.1 bounds the total number by $D - 1$.

A successful request gives a short answer. Round the final residual down to the η-grid by binary search, obtaining e_u with $|e_u| \le mH+2$ and error at most η. For $B = 10M(mH+3)$, return f_u and define its comparison value g_u by

$$f_u = \sum_{j=1}^{p} B^{p+1-j} b_{uj} + e_u, \qquad g_u = \sum_{j=1}^{p} \lambda_j b_{uj} + e_u.$$

Replacing the true scales by powers of B makes f_u rational with common denominator $1/\eta$ and absolute numerator at most B^{p+2}/η. Retaining the true scales in g_u gives $|g_u - v_u| \le \eta$.

To verify $f_S \sim_M g_S$, write an integer affine form of norm at most M as $a_0 + \sum_{u \in S} a_u x_u$. At position j, its combined digit is the integer $\sum_{u \in S} a_u b_{uj}$. If all combined digits vanish, the two evaluations equal $a_0 + \sum_{u \in S} a_u e_u$. Otherwise let j be the first nonzero combined digit. For g_S, the magnitude contributed after position j, including the constant term, is bounded by

$$M \max\{\lambda_j/(10M) + \eta, 1\} < \lambda_j$$

by (5.2) and the rounding error. For f_S, each later combined digit and the final affine term in e have magnitude at most $B/10$. Their geometric sum is smaller than B^{p+1-j}. The digit at j therefore determines both signs. The answers made in this sequence satisfy (5.1), and Lemma 5.2 extends them to all minor indices.

Only J and its expansion are needed to answer the next request. The comparisons in (5.2) involve at most $m + 1$ proxies. The product of the denominators q in Lemma 5.3 clears coefficients in the scales, and a further factor $10M/\eta$ clears the residual and grid comparisons. All these integers are short since $p \le m < D$. Thus initialization and each request require polynomial bit time with sign access. A uniform integer bound F for Theorem 4 is available at initialization by using $U = B^{p+2}/\eta$ in Lemma 5.2 and taking the largest bound over $0 \le p \le m < D$.

5.5 Implementing the sign tests

A sign test on exact logarithms amounts to comparing a product of powers of minors with a power of two. An exponent can have short bit length and still produce very large rational intermediates. We use rounded logarithms whenever their magnitudes are small. A large logarithm, when it occurs, certifies that the rational input itself has sufficient bit length for the power comparison.

Before defining the proxies, fix a short power of two T such that every integer sign query used above has coefficient 1-norm at most $T\eta$. This bound is uniform over choices of the proxies. For example, the common denominators and absolute numerators for the scale coefficients are bounded by $(2C)^m$, and the other coefficients in the queries were bounded above in terms of d, K. These estimates allow T to be fixed for the worst case $m < D$.

Determining the proxy. For each minor needed in a comparison, compute its absolute value and invert it if necessary to obtain $z_0 = 2^{|\ell_u|} \geq 1$. Compute powers 2^k by squaring for $k = 1, 2, 4, \ldots, T$, stopping at the first k for which $z_0 < 2^k$. If no test succeeds then $|\ell_u| \geq T$ and we take $v_u = \ell_u$, represented by the minor itself. Otherwise binary search using powers of two supplies the integer part of $\log z_0$. The scalar powers computed in these tests have exponents at most $\max\{1, 2|\ell_u|\}$. On rational data this is polynomial in the input length by the determinant bound of Section 3.4. The operation count is polynomial in d, K, since T is short.

After removing that integer power of two, the remaining factor z lies in $[1, 2)$. Compute the first $\log(1/\eta) + 6$ terms of

$$\ln z = \sum_{j \geq 1} \frac{(1 - 1/z)^j}{j}$$

and divide by the corresponding partial sum at two. Both series tails admit a geometric bound, and the partial sum at two is at least $1/2$, so the quotient approximates $\log z$ to error at most $\eta/4$. Use binary search to round to the nearest point of the η-grid, breaking ties deterministically. Restoring the integer part and sign produces explicit bits for a proxy within η of ℓ_u. The fractional-part computation uses polynomial-size formulas in z and short constants, and needs only comparisons to perform the rounding.

Evaluating an affine form. If every participating minor has $|\ell_u| < T$, evaluate the query on the explicit proxy bits. Otherwise substitute the rounded proxies and multiply by $1/\eta$. The result is an integer constant plus an integer combination of true logarithms. The constant and each coefficient have magnitude at most $2T^2$, by the norm bound $T\eta$ on the original form. Exponentiation to base two reduces the sign test to comparing a product of integer powers of two and of absolute minor values with one.

Binary powering gives a polynomial operation count in d, K. For rational data in this branch, some $|\ell_u| \geq T$, so the determinant bound puts T itself within a polynomial in total input length. Each powering here could therefore be expanded into a formula with polynomially many leaves in that length, proving the required bound on intermediate log heights in both arithmetic conventions of Section 3.4. Each query uses the input matrix and short discrete data afresh. This proves Theorem 4 and finishes the proof of Theorem 3.

Remark 5.5. The dissociation bound uses only the number of input variables, the degree, and the coefficient 1-norm of the minor polynomials. The same compression procedure applies to

any family of nonzero values of integer polynomials in at most d^2 scalar inputs whose degrees are at most d and coefficient 1-norms at most $d!$. It suffices that the indices are short and that each evaluation has a deterministic strongly polynomial implementation in the chosen arithmetic convention.

6 Learning from extremal prices

In the LP algorithm, a query chooses scales for the coordinates and returns signs at those scales. The coordinates where a nonnegative circuit has its largest scaled entries receive the sign of its side. Theorem 2 turns this limited feedback into one price vector satisfying every consistent circuit hypothesis. We prove it in the abstract setting, with no access to the subspaces that generated the signs.

For integers $N, R \geq 1$, a hypothesis specifies integer demands $a_s \in [-R, R]$ on a nonempty set S of *slots*, the signed coordinates in $\mathcal{S} = [N] \times \{-1, +1\}$. For a vector y write $y_s = h y_i$ at slot $s = (i, h)$. The surplus $d_s(y) = a_s - y_s$ is the amount by which the price falls short of the demand. A query at y returns labels $z \in \{\pm 1\}^N$. Consistency requires $z_s = 1$ at *every* slot attaining $\max_{u \in S} d_u(y)$ whenever this maximum is nonnegative. The desired output has $d_s(y) \leq 0$ on all slots of every hypothesis consistent with the entire sequence of answers.

Consider a hypothesis with two demands on positive slots. If the prices fall short by five and two units, consistency requires a positive label on the first slot and permits either sign on the second. An expert guessing only the second demand could act on the wrong sign. Suppose that violations carry enough of a chosen probability distribution on the slots. A successful recursive call will then give correct feedback on a set of fixed positive mass. An expert sampling four slots from that distribution has a good chance to see one of them.

6.1 A market for feedback

In one round, consider a finite pool of experts e, each with wealth $w_e > 0$. An expert comes with a finite list of options. An option has utility $b - y_s$ at prices y, for some bid b and slot s; each expert also has an outside option with utility zero. Denote the maximum utility over expert e's options by $U_e(y)$.

Choose prices $y \in \mathbb{R}^N$ minimizing

$$\Phi(y) = \sum_e w_e U_e(y).$$

Introducing an upper bound on each expert's option utilities expresses this as a feasible linear program. The outside options bound its objective below by zero, so it attains a minimum. With rational data we may take a rational minimizer. When the feedback is $v \in \{\pm 1\}^N$, pay each maximizing slot option v_s, and pay the outside option zero. Define g_e to be the *least* payoff among expert e's maximizing options. The one-sided directional derivative of its utility satisfies $U_e'(y; v) = -g_e$, since the derivative of a maximum is the largest active slope in the given direction. Minimality of Φ gives, simultaneously for every feedback vector,

$$\sum_e w_e g_e = -\Phi'(y; v) \leq 0. \tag{6.1}$$

Consequently the update $w_e \leftarrow w_e(1 + \eta g_e)$, with rational $0 < \eta < 1$, preserves positivity and never increases total wealth.

There is a market interpretation of (6.1). At a minimizer, zero is a subgradient of Φ. Each expert's wealth can therefore be divided among its maximizing options so that the two signs of each coordinate receive equal total weight. This division would give total gain zero for every v. Taking the least payoff at each tie gives the inequality directly. Compare the market-clearing condition in matching markets [48, Chapter 10] and the halfspace condition in Blackwell approachability [16]. For multiplicative weights and boosting, see [98, 59, 7].

One expert per hypothesis. Give each hypothesis initial wealth one and bids a_s on its slots. In each round query labels at the minimizing prices and use them as feedback in the update. A hypothesis consistent with all answers has gain one whenever any of its demands is violated. Indeed, the maximum surplus is then positive and every maximizing option is a correctly labeled slot. In every other round its gain is nonnegative: a maximizing slot tied with the outside option has surplus zero and is also correctly labeled. Thus k rounds with an unmet demand give this expert wealth at least $(1+\eta)^k$. The initial total wealth M, the number of hypotheses, bounds that quantity.

Run an odd number of rounds greater than $2\log_{1+\eta} M$. Each consistent hypothesis has a majority of rounds meeting all of its demands, although the majority may differ among hypotheses. Taking coordinatewise medians meets them all because a signed coordinate of the median is the median of that signed coordinate. There are at most $(2R+2)^{2N}$ hypotheses, so for fixed η the query count is $O(N\log(R+1))$.

For demands $a_s = p_s$ on both signs of every coordinate, the expert's utility is $\|p-y\|_\infty$. The minimizing prices are then weighted geometric medians in this norm. Chen, Li and Yannakakis [23] use related balanced points to bound query complexity for contractions (see also [77, 55]). At $y = p$ both signs tie at zero, so these exact point demands cannot be consistent with the answer there. Section 8 uses approximate demands in applying the theorem.

Explicitly maintaining an expert for every hypothesis costs exponential time. With four guessed slots per expert, the implementation in Section 6.6 will need only a constant-dimensional space of guesses. The remaining question is how these experts can profit if they miss the one slot at which a label is guaranteed.

6.2 Amplifying the feedback

An internal node makes an odd number of *probes*. Each probe compares two price vectors, its own proposal and a price vector found by a recursive call.

1. Choose proposal prices y (ultimately this will be by a market). Call a fresh child with y added to the price history, the list of proposals along the path from the root.
2. Let x be the prices returned by that child. Query labels z at x, and form feedback at y:

$$v_i = \begin{cases} \operatorname{sgn}(x_i - y_i) & x_i \neq y_i, \\ z_i & x_i = y_i. \end{cases} \tag{6.2}$$

After all probes, return the coordinatewise median of the prices returned by the children. We will see how to end the recursion after logarithmically many levels.

Assigning thresholds. Fix the complete run and a hypothesis (S, a) consistent with it. Assign each node a threshold $t \geq 0$. A node *succeeds* for this hypothesis if its returned prices make every surplus at most t. The root's threshold is zero. The algorithm uses only the price history and the queried labels. We may choose these thresholds separately in the proof for each consistent hypothesis.

Suppose a child returns x and succeeds at its own threshold t'. Then the feedback at its parent's proposal y obeys

$$d_s(y) \geq t' \implies v_s = 1 \qquad (s \in S). \tag{6.3}$$

Indeed, $d_s(x) \leq t' \leq d_s(y)$ gives $x_s \geq y_s$. Strict inequality already gives $v_s = 1$; in case of equality the slot must maximize surplus at x with value t', so that $z_s = 1$ by consistency.

Thus if many surpluses at y exceed t, we can raise the child's threshold yet still aim to get feedback agreeing with many slots. Here is how we do the assignment, propagating also a probability distribution m on S, uniform at the root. For a node with m, t and a probe there, represent m by subdividing $[0, 1)$, with a piece of length m_s for each slot in order of nonincreasing surplus $d_s(y)$ (break ties deterministically). Call the probe

- *light* if $d_s(y) > t$ on mass at most $\delta = 1/5$. Then give the child the threshold $t' = t$.
- *heavy* otherwise. Give the child as threshold t' a high surplus quantile strictly above t: as *pivot* take any slot meeting the band

  $$[\gamma, \delta), \qquad \gamma = 3/16$$

  in positive length, and let t' be its surplus. (Later we will make a specific choice in this band.) Thus

  $$m\{s : d_s(y) \geq t'\} \geq \gamma, \qquad m\{s : d_s(y) > t'\} \leq \delta. \tag{6.4}$$

  See Figure 3.

A threshold is recorded either as zero or by the pivot slot and the level of the proposal at which its surplus gave the threshold. A light probe passes this description to the child. A heavy probe replaces it by the pivot and the current level.

To form the child's distribution m', multiply the first δ of the sorted mass by four and divide the resulting mass by $1 + 3\delta$. A slot crossing the cutoff receives a partial boost. Every slot with $d_s(y) > t'$ receives the full boost, by the light-probe rule or by (6.4). Once m and y are fixed, the same m' works for every allowed pivot. As with the thresholds, the distributions are used only in the proof.

Leaves. Take the root at level zero, all leaves at level D, the least positive odd number with

$$(5/4)^D > 2N.$$

At a leaf, return the coordinatewise median of the D price vectors in the history.

Lemma 6.1. *Each leaf succeeds: its returned prices achieve maximum surplus at most its threshold, for any assignment as above.*

Proof. Otherwise surplus at some slot s exceeds the leaf's threshold at the median. This also happened at more than half of the D proposals on the path. Since thresholds never decrease, at each such proposal y we had $d_s(y) > t'$, with t' the threshold assigned to the child. Thus

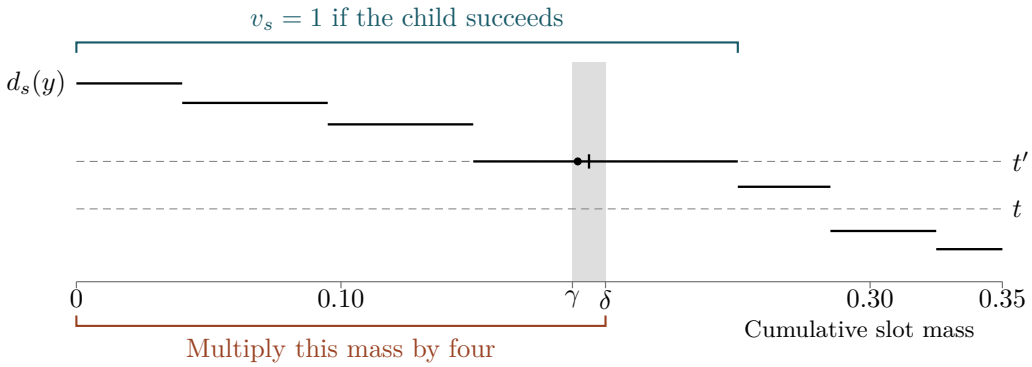

Figure 3. A heavy probe, with the first 0.35 of the slot mass shown. Each horizontal segment represents a slot, ordered by surplus. The dot selects a pivot in the shaded band $[\gamma, \delta) = [3/16, 1/5)$. Two adjacent slots have the pivot surplus t'. Success at t' guarantees correct feedback on mass 0.25, including both tied slots. Reweighting multiplies the first 0.20 of the mass by four before normalization.

the slot's weight was boosted by a factor of at least $4^{D/2}$ over the path before accounting for normalizations. Starting from mass $1/|S|$, the slot has ended with at least

$$\frac{1}{2N} \frac{4^{D/2}}{(1+3\delta)^D} > 1,$$

a contradiction. □

Recursive amplification with reweighting also appears in the contraction algorithm of Soleymani and Farina [131, §6] and the mean-payoff game algorithms of [112, 113, 114]. In [112, §§3–5], near-extreme deviations of hidden vectors imply shifted labeling requirements deeper in the recursion. That work also discusses the relation to Parys's recursion [117].

6.3 Betting on a few slots

If a majority of probes are light and their children succeed, the parent succeeds because those children met the parent's own threshold. A successful child in a heavy probe gives a different guarantee: feedback is correct on the slots at or above the pivot surplus, a set of mass at least γ. Four independent draws from m hit that set with probability at least $1 - (1-\gamma)^4 > 1/2$. This probability is the reason four slots suffice.

The algorithm uses the same pool E in every probe. An expert specifies the following data.

- four slots in $\mathcal{S}$ (an ordered tuple), guessing an integer demand in $[-R, R]$ on each;
- either threshold zero, or a level $j \in \{0, \dots, D-1\}$ and pivot slot in $\mathcal{S}$, again guessing such a demand on that slot.

Include all such experts (repetitions of slots allowed, even with different demands). Zero-threshold experts always bid; those who specify j only bid at levels deeper than j (otherwise they have only the outside option, giving zero gain).

In bidding on its four slots, an expert shifts its guessed demands by *subtracting* its threshold estimate τ. For an expert specifying j, τ is the surplus on its pivot slot at the proposal at

level j in the price history, using its guess for the pivot demand; τ may even be negative. For zero-threshold experts $\tau = 0$. Thus bidding gives the four option utilities

$$b - \tau - y_s$$

(one for each tuple position, with slot s and guess b). As before, experts keep the outside option of utility zero, and gains use the feedback v with worst tie breaking.

We analyze again with a consistent hypothesis and thresholds assigned as above. At each node consider a distribution ν of experts: sample the four slots independently from m, match this node's threshold description (including pivot and level if used), and use true demands throughout. Thus for these experts $\tau = t$ and sampled slot utilities are $d_s(y) - t$.

Lemma 6.2. *Fix a consistent hypothesis and any threshold assignment as in Section 6.2, and set $\eta = 1/16$. In any probe where the child succeeds, for market gains at the proposal y we have*

$$\sum_e \nu_e \ln(1 + \eta g_e) \geq \begin{cases} 0 & \text{if light,} \\ \epsilon := 1/256 & \text{if heavy.} \end{cases}$$

Here ν is from the parent node.

Proof. In the light case with $t' = t$, for these experts every maximizing option gets nonnegative feedback by (6.3) (including zero on the outside option).

In the heavy case with $t' > t$, if the tuple hits $\{s : d_s(y) \geq t'\}$, the gain is one: every maximizing option has surplus at least t' and hence correct sign, with no tie from the outside option. This happens for mass at least $1 - (1 - \gamma)^4$ of ν. Since gains are always at least -1, mean gain is at least $1 - 2(1 - \gamma)^4 > 1/8$. The bound follows from $\ln(1 + \eta g) \geq \eta g - \eta^2$ for these gains. □

Consider running these markets independently at every internal node, with unit initial wealth and an odd number of probes greater than $(2/\epsilon) \ln |E|$. Suppose all the children succeed. If half or more of the probes are heavy, then Lemma 6.2 makes the final log wealth, averaged under the fixed parent distribution ν, greater than $\ln |E|$. Yet no expert can exceed the initial total wealth $|E|$. A majority of the probes must be light, so the median rule gives success. Lemma 6.1 starts this induction at the leaves.

The resulting tree has logarithmic depth and $O(\log |E|)$ children at each internal node. To obtain a polynomial bound on its size we reduce the branching to a constant below the root. A child may then miss its threshold. The following wealth update lets its parent still make progress on that probe.

6.4 Investing recursively

Give the child a small share of each expert's wealth and retain the rest for the parent's market. If the child misses its threshold, we will show that some experts earn sufficiently large returns there to cover what they might lose on the feedback it supplies. Use the tree of Section 6.2, the expert pool E, the rate $\eta = 1/16$, and absolute constants

$$c = 2^{-14}, \qquad r = 2^{19} + 1. \tag{6.5}$$

Give each internal node r probes except the root, which gets

$$r_0 = r + 3072\lceil \log(|E| + 1) \rceil.$$

Each node receives as input the price history and a vector giving each expert positive wealth. At the root, start with wealth one each. Each node returns prices as well as a new wealth vector. A leaf simply returns median prices from the history, and returns the input wealth unchanged.

At an internal node, use the probes successively, starting with the input wealth and keeping any changes across probes. In each probe, split the current wealth w as shown in Figure 4:

1. Use $(1-c)w$ in the market: take proposal prices y minimizing Φ with that wealth.
2. Call the child with y added to the history and with wealth cw; let x and w' be the prices and wealth returned by the child.
3. Query labels at x and take feedback (6.2). Replace w_e for each expert e by

$$(1-c)w_e(1+\eta g_e) + w'_e, \tag{6.6}$$

using the gains in this probe's market.

Return median prices from the children, and the wealth vector after the last replacement.

Output the prices returned by the root. We leave implementation of the markets (including deterministic choices of minimizers) for later in this section.

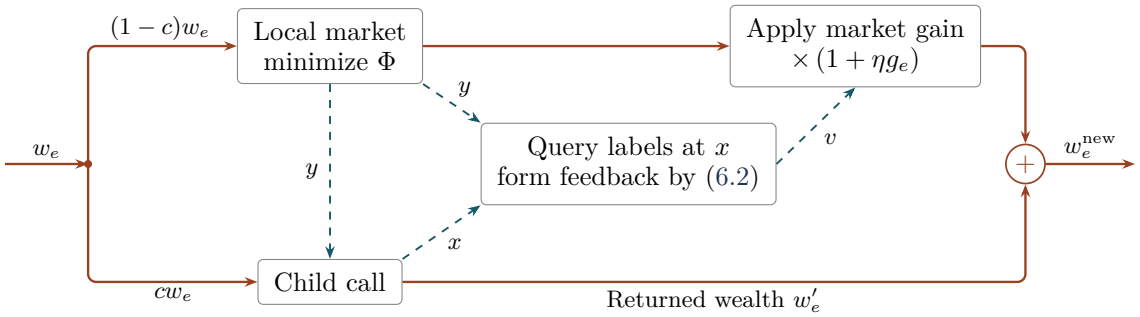

Figure 4. Wealth and prices in one probe. Solid arrows track the two shares of expert e's wealth. Dashed arrows show how the proposal y and the child's price vector x determine the feedback v for the local market. The sum $w_e^{\text{new}} = (1-c)w_e(1+\eta g_e) + w'_e$ funds the next probe or, after the last probe, returns to the parent.

Every node returns total wealth at most its input total. At a leaf the two totals agree. At an internal node (6.1) bounds the wealth paid by the market by its initial share, while the same claim for the child bounds its returned share. Recombining the two bounds and iterating over probes proves the claim.

6.5 Profits cover failures

Fix a consistent hypothesis. We will prove that a failed node gives a large return on its input wealth, averaged over experts whose demands agree with that hypothesis. This assertion is local: those experts may have had little wealth when they entered the node. At the root, however, every expert starts with wealth one and the total is at most $|E|$. There a sufficiently large return is impossible.

At each node let

$$\mu_e = \frac{\text{returned wealth for } e}{\text{input wealth for } e}, \qquad L_e = \ln(1 + (c/2)\mu_e)$$

for each expert. Use L_e to count the bonus from returns here, as a contribution to log wealth one level up. Indeed, if in a probe the child has such values μ'_e, L'_e, the wealth multiplier from (6.6) is

$$(1-c)(1+\eta g_e) + c\mu'_e \;\geq\; (1-c)(1+\eta g_e)\cdot(1+(c/2)\mu'_e)$$

as $(1-c)(1+\eta g_e) < 2$. Consequently for any internal node,

$$L_e \;\geq\; -\ln(2/c) + \sum_{\text{probes}} \big(\ln(1-c) + \ln(1+\eta g_e) + L'_e\big), \tag{6.7}$$

by taking logs and summing to bound $\ln \mu_e$. For each term, the primed value is from that probe's child. Each L'_e is nonnegative, even if the child's investment lost wealth. Thus $-\ln(1-c)$ bounds the loss per probe from sending wealth to the child, while a large child return gives an additional positive term.

To use a child's return in (6.7), we must average its bonuses with the parent's expert weights. The distribution ν from Lemma 6.2 samples four slots and specifies the *current* threshold. A heavy probe gives the child a new pivot description, so its ν' can be disjoint from the parent's ν. We reserve weight at the parent for experts with each possible future pivot level, including the one at which this probe takes place.

Comparing expert weights across one probe. For a fixed assignment of thresholds, give each node at level l the following nonnegative weights π on the experts.

- Include ν: the distribution with tuple slots i.i.d. drawn from m, matching this threshold description and using true demands.
- For each possible *future* pivot level $j \in \{l, \dots, D-1\}$, include mass 2^{l-j}, distributed by picking four slots and the pivot slot at that level all independently from m, with true demands. These experts have zero gains in the markets at this node, because they bid only deeper in the tree.

Thus π is nonnegative of total mass at most three.

Lemma 6.3. *For a complete run and a consistent hypothesis, there is an assignment (choosing all thresholds and pivots as allowed in Section 6.2) such that in every probe*

$$\sum_e \pi'_e L'_e \;\leq\; C \sum_e \pi_e L'_e, \qquad C = 80(5/2)^4 = 3125$$

where π, π' refer to the parent and child respectively, and L' is the vector of bonuses at the child.

Proof. Proceed down the tree. With m, t, π already determined for a node, consider choosing the pivot in any heavy probe here at random: take a uniform point in the band $[\gamma, \delta)$ to determine the pivot slot. This spreads out the choice: for any slot, the probability is at most $1/(\delta-\gamma) = 80$ times that under m.

We claim $\mathbb{E}\pi' \leq C\pi$ componentwise, including a light probe with its prescribed threshold. The child distribution m' is already fixed before choosing the pivot. Since $m' \leq 4m/(1+3\delta) = (5/2)m$, sampling the four tuple slots costs a factor of at most $(5/2)^4$.

- For the child's mass with its current threshold description: if light, compare with the parent's such mass, since the description does not change. If heavy, compare instead with the parent's future mass for $j = l$, using the extra factor of 80 for the chosen pivot.

- For the child's future mass at any pivot level $j \geq l+1$, compare with the parent's future mass for that level. Beyond the factor for the four slots there is a factor two for total mass for that level, and at most $5/2$ for the pivot slot probabilities.

The parent's current description and its reserve for each future level are distinct expert classes. The componentwise comparisons can therefore be combined with the single factor C.

The entire run was fixed before choosing pivots, so the child bonuses L'_e are fixed nonnegative numbers. Multiplying the componentwise expectation bound by L'_e and summing shows that at least one allowed pivot satisfies the claimed inequality. Fix such a choice in every heavy probe at this level and continue to the next level.

□

Lemma 6.4. *For a consistent hypothesis, assign thresholds as in Section 6.2, with the pivots chosen as in Lemma 6.3. If an internal node fails and has T probes, then*

$$\sum_e \pi_e L_e \;\geq\; \sigma T, \qquad \sigma = \epsilon/4 = 1/1024. \tag{6.8}$$

Proof. Proceed from the bottom of the tree. Fix such a node. We bound the contribution $\sum_e \pi_e(\ln(1+\eta g_e) + L'_e)$ in (6.7) per probe.

- The child succeeds: the contribution from log market gains $\ln(1+\eta g_e)$ is nonnegative, and at least ϵ if heavy, by Lemma 6.2 (the experts added for future pivots have gain zero). Child bonuses only help.
- The child fails: then the child must be internal by Lemma 6.1. Therefore

$$\sigma r \leq \sum_e \pi'_e L'_e \leq C \sum_e \pi_e L'_e$$

by induction and the choice of pivots. Thus this probe contributes at least

$$\ln(1-\eta) + \frac{\sigma r}{C} \geq -2\eta + \frac{\sigma r}{C} \geq \epsilon,$$

paying for possibly bad market outcomes (again future-pivot experts have zero gain).

By the median rule, failure implies that for a majority of probes either the probe was heavy or the child failed. So altogether

$$\sum_e \pi_e L_e \;\geq\; T\epsilon/2 - 3T(2c) - 3\ln(2/c) \;\geq\; \sigma T$$

by (6.7), $\ln(1-c) \geq -2c$, and the choices (6.5) ($6c < \epsilon/8,\ 3\ln(2/c) < r\epsilon/8,\ T \geq r$). □

At the root, $\mu_e \leq |E|$ since the total wealth cannot increase, so $L_e \leq \log(|E|+1)$ (recall log denotes base two). As $\sigma r_0 > 3\log(|E|+1)$, there cannot be a failure: apply Lemma 6.4 with assignments as in Lemma 6.3. As this holds for any hypothesis consistent with all answers, the returned prices meet all demands as required.

6.6 Running the markets

The number of probes is $O(r_0 r^{D-1})$. The expert choices give $|E|$ polynomial in N, R, so $r_0 = O(\log(NR+1))$. Since D is the least positive odd integer with $(5/4)^D > 2N$ and $\log_{5/4} r < 60$, the tree uses $O(N^{60}\log(NR+1))$ probes. The wealths also have polynomial bit length. Each probe adds only constantly many binary places to a common dyadic denominator, and all wealths are bounded by the root's initial total $|E|$.

The remaining cost is choosing the proposal at each probe. We first bound its bit length and then describe how to optimize over the implicit expert pool in time polynomial in $N, \log(R+1)$.

Prices at vertices. At each probe use an optimum vertex of the epigraph of Φ. Such vertices exist. Indeed, Φ even tends to infinity as $\|y\|_\infty$ grows: there are zero-threshold experts whose tuples repeat a single slot with guess zero, one for each slot, with strictly positive wealth. Thus the optimal face is nonempty and bounded.

We claim this keeps all prices half-integral, with polynomial bit length, through the tree. To see this, a vertex price vector is determined by ties in each expert's option utilities, i.e., by equations equating utilities of options of the same expert (including ties with the outside option). Indeed, otherwise there is a line through the price vector preserving all those ties; each U_e, hence also Φ, is affine near the point on this line, contradicting extremality. Equations from equating two slot utilities have form

$$y_i = \pm y_j + B, \qquad B \in \mathbb{Z}, \quad |B| \le 2R,$$

as the τ is common. Within any connected component of these edges on the N coordinates, express all its prices in terms of one of them via a spanning tree. If these equations in the component already determine the prices there, some remaining edge (possibly a loop) determines twice that one price as an integer of magnitude $O(NR)$. If not, a tie with the outside option must pin some price in the component, to an integer or to a signed history price plus an integer (in either case with the integer bounded by $2R$ in absolute value). By induction down the tree we keep half-integrality and prices of magnitude $O(DNR)$.

Cells of experts. Group experts by all their data except their demand guesses. Thus the number of groups is polynomial, and each group corresponds to the integer points of a box (one dimension per guess, at most five). Observe that for given prices (proposal and history), all comparisons among an expert's option utilities are comparisons of single guesses or differences of two guesses against constants. Thus we can subdivide the groups by these comparisons, creating cells of experts with a common set of maximizing options (by position on their lists).

More precisely, within each group partition the range $[-R, R]$ for each guess into intervals of integers, and similarly partition $[-2R, 2R]$ for each pairwise difference. Choosing one interval from each partition gives a cell of experts, requiring all of the bounds simultaneously (the cell may be empty). Start with each range as one interval. Every time we choose proposal prices, refine according to all of the option comparisons with these prices, keeping ties separate. Each partition now gets only a constant number of new cuts per probe, and each group has only a constant number of partitions. Thus the number of cells remains polynomial.

The outcomes (including ties) of all option comparisons at proposals so far are now common within cells. So gains in response to feedback are also common: we can identify all maximizing options at the proposal. Wealths too are common, since splits, updates and recombination

preserve this. Keep these per-expert data for each cell, copying them upon refinement to the subcells. This includes wealth data for any ongoing probes, which we need to recombine wealths. Compare Feodorov and Haslebacher's use of arrangements and volumes for fixed points [55]; here each expert has only a constant-dimensional guess, regardless of N.

To use these cells in choosing prices, we compute two kinds of sums: the number of experts in a given cell, and the sum there of any given guess coordinate. In fact we can sum polynomials by eliminating variables one by one. The range for a variable to eliminate runs from a maximum of lower bounds to a minimum of upper bounds, each bound an integer constant or a remaining variable plus an integer constant. So, on the remaining variables, we can subdivide according to which bounds achieve these two extrema (choose first in case of ties), and require lower at most upper. All these conditions have the same form (interval constraints on variables and differences). On each piece, summing over this range again gives a polynomial by the formulas for sums of powers (discrete antiderivatives, valid also with negative endpoints). Starting with constant and linear polynomials for our computations, the degree goes up by at most one per step, and bounds and coefficients undergo only constantly many arithmetic operations: both branching and depth are bounded by constants. This gives the sums in polynomial bit complexity.

Computing the market minimizer. The label oracle is queried at prices returned by the children. To choose the parent proposal, we construct a separate oracle from the finite market data. It separates points from the epigraph of Φ, which is defined by one inequality for every selection of one option per expert: the height must dominate their weighted sum of utilities. These inequalities have polynomial bit length because the history prices are short, the weights share a short denominator, and $\log |E|$ is polynomial in the input size.

At rational test prices, compute an affine sum as above attaining Φ. To do so, temporarily refine on the option comparisons at these prices. Choose a common maximizing option for the experts in each cell and compute the coefficients from their weighted utilities using counts and sums of guesses. This gives separation: compare the proposed height to Φ and use the computed inequality if violated (the normal can be scaled to sup norm one). The bit time is polynomial, counting also the bits of the test point as input.

Apply the classical bit-polynomial ellipsoid algorithm for well-described polyhedra [71, Theorem (6.4.9)] to this separator. All data passed to it have the bit bounds just proved. This is the LP subroutine used to implement (6.1).

Add the optimal height constraint (still polynomial bits), and take the lexicographically first optimal point, a vertex (cf. [71, Lemma (6.5.1)]), minimizing each coordinate successively. This works in polynomial time as well: the optimal face is bounded, and each additional constraint value fixing a price coordinate is a coordinate of the final vertex, hence still short.

This gives the required deterministic implementation; medians and labels too use polynomially many bits, and the bounds hold regardless of labels. This proves Theorem 2 and so completes the proof of Theorem 1.

7 Quadratic programming and complementarity

At a minimum of a convex quadratic over a polyhedron, each inequality has either zero slack or zero multiplier. If we knew which entry to set to zero in every pair, we could recover the point and its multipliers by solving linear equations and inequalities. We find those zeros by

applying the pricing method of Section 3 to the more general monotone linear complementarity problem [30, 31].

Given $M \in \mathbb{R}^{N\times N}$ and $q \in \mathbb{R}^N$, the linear complementarity problem asks for $z, w \in \mathbb{R}^N$ satisfying

$$z \geq 0, \qquad w = Mz + q \geq 0, \qquad z^\top w = 0. \tag{7.1}$$

Here M is *monotone*, meaning $M + M^\top \succeq 0$. Nonnegativity makes the last equality equivalent to $z_i w_i = 0$ for every i.

Theorem 5 (Monotone linear complementarity)**.** *There is a deterministic strongly polynomial algorithm that, for any monotone linear complementarity problem* (7.1)*, returns a solution or an infeasibility certificate $u \geq 0$ satisfying $M^\top u \leq 0$ and $q^\top u < 0$.*

For convex quadratic programming the earlier bit-polynomial algorithm is due to Kozlov, Tarasov and Khachiyan [92]. Interior-point methods for monotone complementarity include those of Kojima, Mizuno and Yoshise [91]. Granot and Skorin-Kapov obtained Tardos-type bounds with matrix dependence for strictly convex quadratics [70], and Cui, Li and Miao obtained such bounds for separable quadratics [32]. Végh's strongly polynomial algorithms for network flows with convex costs include separable quadratic costs [142].

Monotonicity enters at two distinct steps. It ensures that feasibility of the linear constraints in (7.1) suffices for existence of a complementary solution. To *find* one, we also use it to construct a convex set that approximates every scaled, normalized solution and has small complementarity gap throughout. The midpoint argument of Lemma 3.1 then makes the coordinate labels unambiguous.

7.1 Linear feasibility and existence of a solution

By Cottle's theorem [29], a monotone LCP has a solution whenever its linear constraints are feasible. For the proof, we first record the attainment argument for convex quadratics, which will also give the unboundedness certificate in Corollary 6.

Consider $\frac{1}{2}x^\top Gx + c^\top x$ on a nonempty polyhedron $\{x : Ax \leq b\}$, where G is symmetric positive semidefinite. The Frank–Wolfe attainment theorem [58] in this setting follows from the following alternative. There may be a direction

$$d: \quad Ad \leq 0, \qquad Gd = 0, \qquad c^\top d < 0, \tag{7.2}$$

giving rays of descent to $-\infty$. Otherwise $c = Gv - A^\top \mu$ for some v, μ with $\mu \geq 0$, by Farkas. Writing $G = T^\top T$, the objective is then

$$\tfrac{1}{2}\|T(x+v)\|_2^2 - \mu^\top Ax + \text{const}.$$

The pairs $(T(x+v), \mu^\top Ax)$ attainable with $Ax \leq b$ form a polyhedron, with second component bounded above. So in this image space the objective has compact sublevel sets on the polyhedron. There is an optimum.

Now minimize $z^\top w$, $w = Mz + q$, subject to $z, w \geq 0$. If feasible there is an optimum by attainment, since the objective is a convex quadratic and bounded below by zero. Any such optimum solves the LCP. Indeed, in the variable z it also minimizes the linearization there, so by LP duality there are multipliers $u, v \geq 0$ with

$$w + M^\top z = v + M^\top u, \qquad w^\top u = z^\top v = 0.$$

Dotting the stationarity equation (with all terms on the left) with $z - u$ gives

$$z^\top w + u^\top v + (z-u)^\top M(z-u) = 0.$$

All three terms are nonnegative, proving the claim.

7.2 Labeling for complementarity

First use Theorem 1 to test the linear constraints $-z \le 0,\ -Mz \le q$. In the infeasible case, the Farkas multipliers on $-Mz \le q$ form the desired certificate u. In the feasible case with $q = 0$, take $z = w = 0$. For $q \neq 0$ collect the scaled affine spaces in the single linear subspace

$$W = \{(z, Mz + \alpha q) : z \in \mathbb{R}^N,\ \alpha \in \mathbb{R}\}.$$

View these vectors $y = (z, w)$ as rows when used with bases and scalings. Use positive slots for z, negative for w, so $Q(y) = z^\top w$. Use as row basis B_0 a basis of the directions with $\alpha = 0$, say from unit vectors in the z-space, followed by $(0, q)$.

Choose a solution y, with $\alpha = 1$, of minimal support. It is a circuit of W. Indeed, if the subspace supported on $\operatorname{supp} y$ had dimension at least two, the linear functional α would vanish on a nonzero vector there. Move from y in one of the two directions along this vector until a positive entry first reaches zero. The resulting vector still has $\alpha = 1$, all its entries are nonnegative, and it retains at least one zero in each pair. It would therefore be a solution of smaller support.

As before, apply calibration (Theorem 3) and the extremal price theorem (Theorem 2), restarting price search on a calibration restart. Use the same parameters (3.1). At a queried price vector let $B_* = LB_0D$ be the true basis corresponding to the answer B as in Theorem 3. Our task is to give label h at i if slot (i, h) maximizes surplus for such a solution circuit (with its calibrated demands). After scaling the circuit by D and normalizing, the entry at such a slot is at least $1/3$. As in LP we will test for large entries using only B. To avoid conflicting labels, we again want a convex set of candidates on which Q is uniformly small, to use the midpoint argument (Lemma 3.1). We develop the constraints first with the true basis.

A convex constraint for a scaled solution. With the reciprocal scaling D, the complementarity gap $Q(y) = z^\top w$ is unchanged. On the subspace with $\alpha = 0$, the gap is $z^\top Mz \ge 0$. The first N rows of $B_* = LB_0D$ span this subspace after scaling because L is lower triangular. The last diagonal entry of L is positive, so a vector with $\alpha > 0$ has a positive coefficient on the last row.

Write the row coefficients as $\xi = (v^\top, \lambda)$ with $v \in \mathbb{R}^N$. Gathering the terms that contain λ gives

$$Q(\xi B_*) = v^\top A_* v - \lambda g_*(\xi),$$

where A_* is symmetric positive semidefinite and g_* is linear. In the scaled, normalized representation of a solution, $\lambda > 0$ and $Q(\xi B_*) = 0$. Hence $g_*(\xi) = v^\top A_* v/\lambda \ge 0$. Every such solution belongs to the convex set

$$\lambda \ge 0, \qquad g_*(\xi) \ge 0, \qquad v^\top A_* v \le \lambda g_*(\xi).$$

Indeed, $\sqrt{v^\top A_* v}$ is a seminorm and the geometric mean of the two nonnegative quantities on the right is concave. This is the rotated second-order cone constraint in the coordinates of B_*.

A label test on the rational basis. With B, use again the decomposition $Q(\xi B) = v^\top A v - \lambda g(\xi)$, taking A symmetric as before. Recall $H = 2^{100N^2}$, $\epsilon = H^{-10}$, and set $\widetilde{A} = A + H^{-50} I$. Use candidates $y = \xi B$ as in (3.2) (again with $\|\xi\|_\infty \le H$, $-\epsilon \le y_s \le 2$ for each entry), with the *additional* constraints

$$\lambda \ge 0, \qquad g(\xi) + \epsilon \ge 0, \qquad v^\top \widetilde{A} v \le (\lambda + \epsilon)(g(\xi) + \epsilon). \tag{7.3}$$

These height shifts let us work with only absolute accuracy on the basis entries, even when a solution has λ tiny or $g_*(\xi) = 0$.

Indeed, B is within H^{-100} of B_*, which is bounded by 2 entrywise. Hence A, g are within $20NH^{-100}$ of A_*, g_* coefficientwise, by taking dot products. Thus $\widetilde{A}$ is positive semidefinite, so the constraints are convex together as before. Also for $\|\xi\|_\infty \le H$,

$$|g(\xi)| \le H^2, \qquad |g(\xi) - g_*(\xi)| \le H^{-90}, \qquad |v^\top (\widetilde{A} - A_*) v| \le H^{-45}.$$

All candidates satisfy

$$Q(y) \le \epsilon(\lambda + g(\xi) + \epsilon) \le H^{-1}.$$

And for any solution y_0, write $y_0 D / \|y_0 D\|_\infty = \xi B_*$, with $\|\xi\|_\infty \le H$. To see that $y = \xi B$ is a candidate, observe that it approximates ξB_* within ϵ^2 entrywise, and (7.3) holds since in particular

$$(\lambda + \epsilon)(g(\xi) + \epsilon) \;\ge\; \lambda g_*(\xi) + \epsilon^2/2 \;\ge\; v^\top \widetilde{A} v.$$

Assign $+$ at i if the candidate set contains a point with $y_{i+} \ge 1/4$, and assign $-$ otherwise. Only short rational data enter this test. To carry it out using the bit-polynomial convex QP algorithm [92], divide by the positive quantity $\lambda + \epsilon$. Introduce $s = 1/(\lambda + \epsilon)$, $\bar{\xi} = s\xi$ and $\bar{y} = sy$. Multiply every linear constraint, including the label threshold, by s, and impose

$$s \ge \frac{1}{H + \epsilon}, \qquad \bar{\lambda} + \epsilon s = 1, \qquad \bar{v}^\top \widetilde{A} \bar{v} \le g(\bar{\xi}) + \epsilon s, \qquad \bar{\xi} = (\bar{v}^\top, \bar{\lambda}).$$

The last inequality is (7.3) multiplied by s^2. Conversely, dividing the barred variables by $s > 0$ recovers the original constraints. This perspective transformation [20, §2.3.3] leaves one convex quadratic inequality on a bounded polyhedron. Indeed, $\bar{\lambda} \ge 0$ gives $s \le 1/\epsilon$, and the scaled box constraints bound $\bar{\xi}, \bar{y}$. Test linear feasibility first. On a nonempty polyhedron, minimize the left side minus the right side of the quadratic inequality and accept exactly when its minimum is nonpositive.

Conclusion of the proof of Theorem 5. As in LP, consider the final uninterrupted price search, with demands given by calibration. A solution circuit, scaled and normalized at a query, has entry at least $1/3$ at each slot of maximal surplus on its support. A candidate therefore reaches $1/4$ at this slot. No candidate can reach the same threshold at the opposite slot by convexity and Lemma 3.1. As a result the labels are consistent with the solution circuit's demands.

Write x for the prices returned by the extremal price algorithm. In each pair allow only one slot to be positive, namely at the sign of x_i (arbitrary choice when zero). Return to the original system (7.1) and impose the indicated zeros. Each demand on the solution circuit's support is positive and met, so these zeros retain the solution circuit. Theorem 1 recovers a solution of the resulting linear system.

The number and sizes of the label tests obey the same bounds as in Section 3. Their convex QP data have short binary encodings. The two LP calls on the original data check initial feasibility and recover the solution. All steps are deterministic. □

A single solution (z_*, w_*) also gives a linear description of the entire solution set [101]. On the region $z, w \geq 0$, $w = Mz + q$, the identity

$$z^\top w = z^\top w_* + z_*^\top w + (z - z_*)^\top M(z - z_*)$$

splits the gap into three nonnegative terms. Their sum is zero precisely when

$$z^\top w_* + z_*^\top w = 0, \qquad (M + M^\top)(z - z_*) = 0.$$

The second equivalence uses positive semidefiniteness of $M + M^\top$. Consequently the solution set is a polyhedron. To find a solution of maximum support, test which entries can be positive by linear feasibility. Averaging the witnesses gives a solution positive on all those entries at once.

7.3 Quadratic optimization and conic feasibility

The first-order optimality conditions for a convex quadratic are an affine variational inequality [94, 93]. Their classical complementarity formulation [30, 31] yields the following algorithm.

Corollary 6 (Convex quadratic programming and affine variational inequalities)**.** *Both problems below are solvable in deterministic strongly polynomial time for arbitrary real data.*

1. Affine variational inequalities. *Given a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and monotone $G \in \mathbb{R}^{n \times n}$, $c \in \mathbb{R}^n$, find $x \in P$ such that $(Gx + c)^\top (x' - x) \geq 0$ for all $x' \in P$, or report that none exists.*
2. Convex quadratic programming. *For such P and symmetric $G \succeq 0$, minimize $\frac{1}{2} x^\top Gx + c^\top x$ over P, returning an optimum and multipliers, or certifying infeasibility or unboundedness.*

Proof. By LP duality the variational inequality at $x \in P$ holds if and only if there are multipliers satisfying

$$Gx + c + A^\top \mu = 0, \qquad \mu \geq 0, \qquad \mu^\top (b - Ax) = 0.$$

Write $x = Ru$ for $R = [I \; -I]$, with $u \geq 0$. The requirements on u, μ are exactly (7.1) for

$$z = \begin{pmatrix} u \\ \mu \end{pmatrix}, \qquad M = \begin{pmatrix} R^\top GR & R^\top A^\top \\ -AR & 0 \end{pmatrix}, \qquad q = \begin{pmatrix} R^\top c \\ b \end{pmatrix}.$$

Indeed, applying $R^\top$ includes each entry of $Gx + c + A^\top \mu$ and its opposite in the first block of w, forcing zero. Monotonicity is immediate, so apply Theorem 5.

For the quadratic objective, first test feasibility of P by Theorem 1. If P is empty this gives a Farkas certificate. Otherwise retain a feasible point x_0 and test the linear conditions (7.2) on d. Any such d gives the ray $x_0 + td$, $t \geq 0$, along which the objective decreases to $-\infty$. If no such direction exists, the attainment argument gives a minimum. Convexity makes the variational inequality necessary and sufficient for optimality, so the complementarity solution yields an optimizer and its multipliers. These reductions use a constant number of layers of scalar computations and calls as in Section 3.4. □

The LCP and convex QP algorithms and their certificates transfer to ordered fields by the argument in Section 3. Diagonalizing the symmetric form by congruence shows that its positive semidefiniteness persists in a real closure.

Projection. For any v and any nonempty polyhedron P, minimizing $\|x - v\|_2^2$ over P gives its Euclidean projection. If a polytope is given by vertices, barycentric variables express membership by linear constraints. This includes the minimum-norm point of a vertex-given simplex. See Hong and Verma [78] for earlier connections of strong polynomiality questions for convex QPs and projection, and De Loera, Haddock and Rademacher [44] for an LP reduction *to* the minimum-norm problem in an integer-input model, as well as exponential behavior of Wolfe's combinatorial algorithm [145]. Fujishige, Kitahara and Végh [60] give bounds using circuit imbalance for update-and-stabilize methods for box-constrained least squares.

A conic consequence. Another short application, using perspective again, is to a special case of second-order cone programming (see [99, 20] for background and applications).

Corollary 7 (Lorentz constraints with proportional quadratic parts)**.** *One can deterministically decide, in strongly polynomial time, real feasibility of linear weak inequalities together with a single Lorentz constraint*

$$\|F(x)\|_2 \le h(x) \tag{7.4}$$

for real affine maps F, h with vector and scalar output, respectively, and return a point in case of feasibility.

More generally we can decide feasibility, within the same bound, for arbitrarily many Lorentz constraints if the Hessians $\nabla^2(\|F(x)\|_2^2 - h(x)^2)$ (one for each constraint) span at most one dimension (e.g., ball constraints of the form $\|x - c\|_2 \le r$ for given c, r). We can return an exact feasible point here using an output representation $a + b\sqrt{d}$, giving scalar $d \ge 0$ and vectors a, b in the field generated by the input.

Proof. In the single-constraint case, test first for feasibility with $h(x) = 0$, by LP (since we then just require $F(x) = 0$ for (7.4)). For $h(x) > 0$ use variables $s = 1/h(x), y = sx$, writing the conditions as $s > 0$ together with

$$Ay \le sb, \qquad H(y, s) = 1, \qquad f(y, s) := \|T(y, s)\|_2^2 \le 1,$$

with $Ax \le b$ the given linear constraints and H, T the linear forms or maps obtained by homogenizing h, F using s. Find a point satisfying all but possibly the norm constraint (or conclude infeasibility) by LP. If this point does not already give a solution, minimize f over the linear conditions with $s > 0$ relaxed to $s \ge 0$, by Corollary 6. A minimum greater than one proves infeasibility. If the minimum is less than one, interpolate between its value and the value at the point from LP, choosing a positive weight on the latter so the interpolated value is one. The corresponding convex combination has $s > 0$ and $f \le 1$ by convexity. If the minimum is one, we need a minimizer with $s > 0$. Strict convexity of the squared norm in the T-image makes that image the same at every minimizer. Fix it to the value already found and test for $s > 0$ by LP.

Now for several constraints of the indicated type, require the right-hand sides nonnegative and square the constraints. Besides linear constraints the result has the form

$$\ell_j(x) \le q(x), \qquad q(x) \le r_k(x) \qquad \text{(respective indices } j, k)$$

for affine ℓ_j, r_k and one quadratic form q, after rescaling inequalities (all quadratic parts are multiples of one). Impose also all $\ell_j(x) \le r_k(x)$.

On this linear region, test separately for feasibility with the lower bounds, and for feasibility with the upper bounds. For the lower bounds, try each j as a maximizing index. Impose $\ell_j \geq \ell_{j'}$ for all j' and enforce $\ell_j \leq q$ using its original Lorentz constraint. The upper bounds are treated in the same way by enumerating a minimizing index. If either list is empty, its separate feasibility test consists of linear constraints alone. If there are both kinds, with feasibility for each separately, consider the segment between the two witnesses. At each point, the lower bounds all hold or the upper bounds all hold (since $\ell_j \leq r_k$). These are closed conditions, so somewhere on the segment both hold. We can find such a point by testing endpoints and enumerating roots of nonconstant quadratic or linear constraint polynomials on the segment (an endpoint of the closed feasible portion suffices). Testing at such a root takes ordinary scalar operations and comparisons by isolating a radical and squaring (checking signs first). Only constantly many layers of subroutine calls or polynomial-size arithmetic formulas are needed, so we get strong polynomiality as claimed. □

A radical in the generalization can be necessary for a witness. For example, $\|(2x, 1)\|_2 \leq 3$ and $\|(1, 1)\|_2 \leq x$ together force $x = \sqrt{2}$.

8 Contractions in max norm

At a point x far from a fixed point v, a contraction in max norm must move each coordinate of greatest error toward v. The signs of these movements give exactly the kind of partial information used by Theorem 2. We use them to locate v with a running time polynomial in the number of bits of accuracy.

Let $C = [-1, 1]^n$, $n \geq 1$. A self-map $f : C \to C$ with

$$\|f(x) - f(y)\|_\infty \leq (1 - \gamma)\|x - y\|_\infty, \qquad 0 < \gamma \leq 1,$$

has a unique fixed point v [10]. Direct iteration may require order $1/\gamma$ evaluations to halve the error. We assume an oracle that accepts a dyadic point $x \in C$ and a precision m. It returns a dyadic point of C with m bits after the binary point and sup norm error at most 2^{-m} from $f(x)$. Dimensions and accuracy parameters are given in unary.

Theorem 8 (Approximation in max norm)**.** *For $\gamma = 2^{-b}$ we can, given this access to the contraction f and nonnegative integers b, k, return a dyadic $x \in C$ with $\|x - v\|_\infty \leq 2^{-k}$ in deterministic bit time polynomial in n, b, k, including all query and answer bits. For a nonexpansive map ($\gamma = 0$ allowed in the Lipschitz condition), we can instead return a dyadic $x \in C$ with residual $\|x - f(x)\|_\infty \leq 2^{-k}$, in time polynomial in n, k.*

For a contraction, a residual bound e also gives the distance bound e/γ. Shellman and Sikorski [127] obtained polylogarithmic dependence on inverse accuracy in fixed dimension, with an exponent growing with the dimension. Chen, Li and Yannakakis [23] balanced the remaining possible locations to obtain a residual of at most ϵ in $O(n \log(1/\epsilon))$ exact evaluations. Computing the query points incurs an additional cost. Feodorov and Haslebacher [55] combined faster balancing with dimension splits [24] to obtain subexponential arithmetic time in $n + \log(1/\epsilon)$ with exact evaluations. See also the subsequent work on decomposition [25].[1] Soleymani and Farina [131] obtained quasipolynomial arithmetic complexity for approximating

[1] For other norms see [129, 80, 77, 41], for monotone maps see [11], and for the complexity of total search see [42, 54].

the location, counting exact evaluations and using logarithms of inverse gap and inverse error as parameters. Theorem 2 combines balancing with recursive amplification of extremal labels.

Proof. First assume $\gamma = 2^{-b} > 0$ and set

$$\epsilon = 2^{-k}, \qquad \delta = \gamma\epsilon/16, \qquad R = 2/\delta.$$

Extend f to $\mathbb{R}^n$ by clipping each input coordinate to $[-1, 1]$. This preserves the Lipschitz bound and the fixed point. Run Theorem 2 with $N = n$ and demand bound R. At a queried price vector p, set $x = \delta p$. To obtain a label, clip x to C, round it to the nearest δ-grid point, and request $b + k + 4$ bits from the oracle. Subtract the original x from the answer and return the signs, breaking ties arbitrarily. Input rounding contributes at most $\delta/2$ to the evaluation error and the oracle at most δ. Thus the labels recover the sign of every component of $f(x) - x$ with magnitude greater than 2δ.

For the analysis, the fixed point defines one hypothesis with both slots (i, h), $h = \pm 1$, at every coordinate. Assign the demand

$$a_{i,h} = \left\lfloor \frac{hv_i - \epsilon/2}{\delta} \right\rfloor.$$

These integers lie in $[-R, R]$. The algorithm need not know them. Writing $d_{i,h}(p) = a_{i,h} - hp_i$, we have

$$h(v_i - x_i) - \delta < \delta d_{i,h}(p) + \epsilon/2 \le h(v_i - x_i).$$

Suppose that (i, h) maximizes the surplus and its surplus is nonnegative. With $u = h(v_i - x_i)$ the displayed bounds give $u \ge \epsilon/2$ and $\|v - x\|_\infty \le u + \delta$. Since $f(v) = v$, the movement of coordinate i satisfies

$$h(f_i(x) - x_i) \ge u - (1 - \gamma)(u + \delta) \ge \gamma\epsilon/2 - \delta = 7\delta.$$

Even after rounding, the reported sign is h. This proves consistency of the hypothesis.

At the returned prices all its surpluses are nonpositive, so the corresponding x satisfies $\|x - v\|_\infty \le \epsilon/2 + \delta$. Clipping and rounding to the same grid yield a dyadic point within ϵ of v. The running time is polynomial in n, b, k because $\log R = b + k + 5$, and every oracle input and answer is a string of polynomial length.

For a nonexpansive f, apply the contraction result to $g = (1 - \epsilon/2)f$ with target distance $\epsilon/4$ as in [23, 131]. At the fixed point v_g, the residual $\|v_g - f(v_g)\|_\infty$ is at most $\epsilon/2$ because $f(v_g) \in C$. Moving to the returned point changes this residual by at most $2(\epsilon/4)$, by nonexpansiveness. To simulate an m-bit query to g, request $m + 1$ bits from f, multiply by $1 - \epsilon/2$, and round to the nearest m-bit dyadic. The total error is at most 2^{-m}. □

8.1 Exact piecewise-affine computation

For an explicitly given piecewise-affine map, a small residual identifies a region on which an exact fixed point can be found by LP. Consider a circuit with n inputs and n outputs, built from scalar affine, minimum and maximum gates [52, 54]. A gate output may feed several later gates. A *selection* specifies one attaining operand at each minimum or maximum. Adding a variable for each gate output gives linear constraints for the inputs and outputs realizing a selection. Ties are allowed, so each such cell is closed.

Corollary 9 (Exact fixed points for rational circuits)**.** *Given such a circuit with binary rational coefficients, a deterministic algorithm uses polynomial bit time to return either an exact fixed point in C or a counterexample to the nonexpansive self-map property. A counterexample consists of one point mapped outside C or two points violating nonexpansiveness. All points returned are rational.*

Proof. Fix a selection and minimize e over the realizing cell in C, subject to $-e \le f_i(x) - x_i \le e$ for every i. Every such LP has encoding length bounded by a common polynomial in the circuit length. By the usual vertex bounds [71], its minimum, if positive, is at least the reciprocal of an integer of polynomial bit length. Choose k, polynomial in the input length, so that 2^{-k} is strictly smaller than this lower bound. Every cell realizing a residual at most 2^{-k} then contains an exact fixed point. This precision argument is due to Etessami and Yannakakis [52].

Run Theorem 8 for this k, in its nonexpansive version. Evaluate the circuit exactly before rounding each oracle reply. With the input and a realizing selection prescribed, the gate values are the unique solution of a rational linear system. Vertex bounds therefore give polynomial bit length for all reduced gate values on short-bit inputs. Include evaluation at the point output by that theorem. If an evaluation breaks range or violates nonexpansiveness upon comparison with past evaluations, stop and return this counterexample.

If all pass, these exact evaluations extend to a nonexpansive self-map of the cube by McShane extension [103] (coordinatewise, take the minimum of recorded values plus sup norm distance to the corresponding sample, then clip). So the residual guarantee holds, even for the original circuit, at the output point. This extension argument for total search (cf. [23]) applies also to each prefix of passing evaluations, so the time bound holds in any case. Thus we need only make a selection at the output point and solve linear feasibility for a fixed point in that cell by bit-polynomial LP. □

8.2 Offsets and asymptotics

A discounted game gives affine gates whose slopes contain transition probabilities and discount factors, while rewards enter as offsets. The offsets may be arbitrary reals. To study vanishing discount, we also need to recover a single selection valid throughout an interval $0 < t < t_*$. We do both by determining which linear constraints are tight. Their determinant tests depend linearly on the offsets, permitting Frank–Tardos rounding in the spirit of Tardos's algorithm for LP [134].

Proposition 10 (Real offsets and a vanishing parameter)**.** *Allow affine gates in Corollary 9 with operands y_j and output*

$$z = \sum_j a_j(t) y_j + b(t),$$

where the polynomials are given by coefficient lists. The slope polynomials a_j have binary rational coefficients, and the offset polynomial b has arbitrary real coefficients. There is a deterministic strongly polynomial algorithm returning either a fixed point in C or a counterexample to the nonexpansive self-map property as in Corollary 9. The coordinates are rational functions of t, represented by ratios of real polynomials, valid for all sufficiently small $t > 0$. With a fixed point we can also return a selection, valid in the same sense. Here degrees and slope encoding lengths count as part of discrete input size.

This includes nonparametric input (and then the output can be taken constant).

Proof. We solve on short-bit data and transfer the result back. For this we recognize possible outcomes by determinant signs.

Indeed, at fixed t, use evaluation constraints with selections prescribed. Then asking for a fixed point in C gives linear conditions. To ask if the outputs z, z' at points $x, x' \in C$ can violate nonexpansiveness, specify an output coordinate i and sign h, and consider

$$h(z_i - z'_i) \geq \tau + u, \qquad \|x - x'\|_\infty \leq \tau \leq 2, \qquad 0 \leq u \leq 1.$$

A solution with $u > 0$ witnesses a violation. For a range violation prescribe a single selection, coordinate i and sign h, and impose $hz_i \geq 1 + u$ with $0 \leq u \leq 1$. Again success means $u > 0$.

With only its weak inequalities, each system describes a polytope, since the inputs lie in the cube and the gate outputs are bounded there. A successful outcome thus has a vertex witness, obtained by maximizing u when a violation is sought. A square invertible system of tight rows determines this vertex. Its determinant and the bordered determinants formed with each inequality row and the right-hand side column determine whether the vertex satisfies the inequalities. They also determine the sign of u.

All these determinants are univariate polynomials with coefficients linear in the vector w of all offset coefficients together with a constant 1 (use the 1 entry also for tests independent of offsets). After a common positive denominator is cleared, these linear forms have integer coefficients of ℓ_1-norm bounded by $2^{\mathrm{poly}(L)}$, uniformly over selections, with L our input size counting each offset coefficient as one entry. Indeed, clear all slope denominators by their product, then use the determinant expansion (degrees polynomial, at most polynomially many factors per term, at most $2^{\mathrm{poly}(L)}$ terms). Offsets appear only in the right-hand side column.

By Lemma 5.4 use short-bit offsets (normalizing the rounded vector so the constant is again one) preserving all these coefficient signs. At the new offsets, use a positive $t_0 = 2^{-\mathrm{poly}(L)}$ small enough that the lowest nonzero coefficient decides every determinant sign at t_0 whenever the determinant is nonzero as a polynomial. After clearing the new denominators, its coefficients are integers of polynomial bit length. A geometric series bounds the terms after the lowest nonzero one, so this sign agrees with that at all sufficiently small positive t. Identically zero determinants stay zero.

Solve the resulting short-bit problem (new offsets, $t = t_0$) by Corollary 9. Recover a successful vertex and defining tight rows as above, by ordinary bit-polynomial LP (use selections at the obtained points, maximizing u where applicable, then getting a vertex). The same determinant signs hold with original offsets at all sufficiently small positive t. Return the corresponding vertex by Cramer's rule, with the numerator and denominator polynomials formed by Lagrange interpolation. The degree bounds above allow interpolation from evaluations at polynomially many small integers, avoiding polynomial division. This is strongly polynomial by Section 3.4 (interpolation uses linear combinations with short-bit coefficients). □

9 Games on graphs

In a mean-payoff game, two players choose a walk in a directed graph. Max wants the walk to earn a high average reward and Min wants it to earn a low one. We compute the value from every starting state and optimal policies for both players. The algorithm is strongly

polynomial when each action determines its outcome. If an action draws a random outcome, the step bound also counts the bits specifying its probabilities. The proof combines two earlier results. A fixed-point computation proposes actions for the two players, and linear programming determines what each proposed policy guarantees against a best response.

Arena and payoffs. The arena is finite and explicit. Each of its $n \geq 1$ states belongs to Max or Min, whose turn it is to choose an action there. Every state has at least one action. An action a has a nonempty finite set E_a of outcomes. Outcome e leads to state $j(e)$ and pays a real reward r_e. Conditional on the entire history and the chosen action, e has probability p_e. Each action belongs to a single state and each outcome to a single action. Players observe all actions and outcomes and may randomize as a function of the entire history. A *policy* is a pure positional strategy, specifying one action at each state owned by that player.

For a play with outcomes $e_1, e_2, \ldots$, the ordinary mean payoff is

$$\liminf_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} r_{e_k}.$$

We also allow *priority mean payoff* [68]. Each outcome has a weight $w_e > 0$ and a positive integer priority q_e. Let $q = \liminf_k q_{e_k}$, the least priority visited infinitely often. The payoff is

$$\liminf_{N\to\infty} \frac{\sum_{k\leq N:\, q_{e_k}=q} w_{e_k} r_{e_k}}{\sum_{k\leq N:\, q_{e_k}=q} w_{e_k}}.$$

The denominator is eventually positive. Equal priorities and unit weights give ordinary mean payoff. Priorities are discrete input and can be replaced by their ranks $1, \ldots, Q$, with Q at most the number of outcomes. In a *discounted* game, each outcome has a discount factor $0 \leq \beta_e < 1$. Its payoff is

$$\sum_{k\geq 1} r_{e_k} \prod_{l<k} \beta_{e_l}.$$

For each of these payoffs Max maximizes its expectation and Min minimizes it.

Theorem 11 (Values and optimal policies)**.** *For discounted and priority mean-payoff games on finite turn-based arenas of perfect information, there is a deterministic algorithm that computes the exact value vector v and a pair of policies. From every initial state i, the Max policy guarantees expected payoff at least v_i and the Min policy guarantees at most v_i against every strategy of the opposing player.*

- *In a two-player game with binary rational transition probabilities, the step bound is polynomial in the size of the arena, including its discrete data and the transition probability bit lengths. Rewards, weights, and discount factors may be arbitrary real scalars.*
- *In a Markov decision process, where only one player has choices, the step bound is polynomial in the size of the arena and its discrete data even for arbitrary real transition probabilities supplied as scalars.*

On rational input all intermediate numbers and outputs have bit lengths polynomial in the complete input length. In particular, both cases have polynomial bit complexity on binary rational input.

In a deterministic arena there is one outcome per action, so the two-player algorithm is strongly polynomial, including for arbitrary real rewards. Against pure strategies the policy guarantees hold for the resulting individual play. For both mean-payoff objectives the same values and policies apply when the liminf is replaced by a limsup.

Reachability and parity. Making a target absorbing and assigning reward one there and zero elsewhere turns the mean payoff of each play into the indicator of reaching the target. Theorem 11 therefore gives a polynomial-time algorithm for simple stochastic games, whose threshold problem Condon placed in NP ∩ coNP [28]. For reductions among simple stochastic games, discounted games and mean-payoff games, see Zwick–Paterson [150] and Andersson–Miltersen [6].

In a parity game Max wins when the least priority occurring infinitely often is even. Set $r_e = \mathbf{1}_{\{q_e \text{ even}\}}$ before relabeling the priorities. The priority mean payoff of each play then equals its win indicator [68], so the algorithm computes the optimal win probability even with binary rational transitions. The deterministic parity problem also reduces to ordinary deterministic mean payoff with exponentially large rewards [82]. Earlier quasipolynomial algorithms for deterministic parity games include the breakthrough of Calude, Jain, Khoussainov, Li and Stephan [22] and Parys's recursive approach [117]. The distinct *mean-payoff parity* objective imposes a parity condition on a mean-payoff game [115].

Earlier algorithms. Shapley [126] introduced discounted stochastic games and the contraction argument, allowing simultaneous moves. Gillette [65] studied long-run average reward. Positional optimality for perfect-information games has a long history, including Liggett–Lippman [97] and, for deterministic games, Ehrenfeucht–Mycielski [49].

Pseudopolynomial algorithms for integer rewards use finite horizons [150] or an energy objective with initial credit [21]. See also [72, 120] and algorithms parameterized by the number of chance positions [19, 4]. Randomized strategy iteration draws on subexponential simplex methods [84, 102, 76]. It has been applied to simple stochastic games [100] and mean-payoff games [14, 15], with an LP-type interpretation [73] and improved expected bounds [149]. Complementarity formulations are available for discounted and simple stochastic games [83, 74, 62], although their matrices can fail the monotonicity hypothesis of Section 7 [74, Theorem 23].

Ohlmann [110] gave a symmetric recursive algorithm for deterministic mean-payoff games. Recent deterministic and randomized algorithms give quasipolynomial bounds for binary-encoded integer rewards [112, 113], and a deterministic quasipolynomial bound is known for turn-based stochastic games [114]. For connections with contractions see [131]. Truffet's proposals use max-atoms and tropical optimization [136, 137]. See [113, Section 1 and Appendix A] for discussion.

Proof of Theorem 11. We first solve the discounted fixed-point equation when its slopes have short binary encodings. Taking discounts to one then yields mean-payoff values and policies by Blackwell optimality. To remove the dependence on discount bits, we use LP to evaluate a fixed policy with arbitrary real data. These one-player evaluations give lower and upper guarantees on the two-player value. A clipped fixed-point computation halves the gap between the guarantees, and a bound on the possible scales of a positive gap gives exact termination.

9.1 Discounted values and the mean-payoff limit

At a state i, write ext_i for the maximum over actions if Max owns i and for the minimum if Min owns it. The Bellman (Shapley) operator for a discounted game is

$$T_i(x) = \operatorname*{ext}_{a \text{ at } i} \sum_{e \in E_a} p_e(r_e + \beta_e x_{j(e)}). \tag{9.1}$$

It preserves the coordinate order and contracts the sup norm by a factor at most $\bar{\beta} = \max_e \beta_e < 1$. Suppose $x \le T(x)$. At each Max state choose an action attaining the maximum. At a Min state every action has score at least x_i. Under this Max policy, iteration of the one-step inequality gives expected discounted reward through N steps, including discounted terminal value x, at least x_i from start i. The terminal contribution has magnitude at most $\bar{\beta}^N \|x\|_\infty$, so the same guarantee holds for infinite play. Reversing the inequality gives Min's upper guarantee. At the fixed point these bounds coincide. An extremizing action in each state gives an optimal policy for its owner.

Suppose the discount factors and probabilities are given as binary rationals. Scale the rewards by $1/S$, where

$$S = \max\{1, \max_e |r_e|/(1-\beta_e)\}.$$

Every term $r_e/S + \beta_e x_{j(e)}$ then lies in $[-1, 1]$ on the unit cube. The circuit for (9.1) has binary rational slopes and arbitrary real offsets, as required by Proposition 10. Multiplying its fixed point by S recovers the value vector. The extremizing actions survive the scaling. The arithmetic uses only a constant number of layers of scalar formulas and subroutine calls.

The same contraction estimate gives an approximation result for discounted games *with simultaneous moves*. Here the players choose independently randomized actions at each state, and the outcome distribution depends on the pair of actions. All history is public. Shapley's operator replaces the extremum in (9.1) by the value of the one-stage matrix game [126]. Optimal mixed actions at its fixed point give the same finite-horizon guarantee. With the scaling above, the map preserves the cube and contracts it by $\bar{\beta}$, since an entrywise perturbation of size ϵ changes a matrix-game value by at most ϵ. For binary rational input of length L, $1 - \bar{\beta} \ge 2^{-L}$. Each evaluation can use bit-polynomial LP. Theorem 8 therefore approximates the value vector to absolute error 2^{-k} in bit time polynomial in L and k. Undoing the scaling costs only polynomially many extra bits of accuracy. See [51, 11, 55] for related fixed-point methods.

From discounts to mean payoff. In a turn-based priority game, introduce the parameter $t > 0$ and set

$$\beta_e = 1 - w_e t^{q_e}, \qquad \widetilde{r}_e = (1-\beta_e) r_e. \tag{9.2}$$

Use q_e after relabeling. For small t, the resulting discounted payoff is the expected reward in a process that stops after outcome e with probability $1 - \beta_e$ and pays r_e upon stopping [68]. Lower priorities have larger stopping probabilities as t tends to zero.

For binary rational probabilities and weights, the slopes in the discounted operator are polynomials with binary rational coefficients. The scaling $S = \max\{1, \max_e |r_e|\}$ works for every small $t > 0$. Proposition 10 supplies the discounted values as rational functions of t and a pair of *Blackwell optimal* policies, optimal for all sufficiently small $t > 0$. For background on Blackwell optimality see [17, 43, 66, 67]. Determinant signs have also been used to establish stabilization as discounts tend to one [6, 63]. See also [114, Section 2.1].

Gimbert and Zielonka's Theorem 6 [68] transfers these policies to priority mean payoff and identifies the values with the limits of the discounted values. Their theorem uses rewards, priorities and weights on states. To apply it here, duplicate each state to record its incoming outcome. Give each original state an additional starting copy with fixed dummy data. From a specified starting copy the observed histories in the two games determine one another, and an original policy lifts by ignoring the incoming outcome. The initial state datum adds a fixed first discounted payoff and multiplies the remaining payoff by a positive factor tending to one. The additive term tends to zero. Thus the lifted policies are Blackwell optimal, and the extra initial datum does not affect priority mean payoff. Applying the theorem with parameter $1 - t$ gives its limsup conclusion. Exchanging players and negating the rewards gives the liminf conclusion with the same policies.

The transfer holds for arbitrary real game data. The encoding assumptions above enter the computation of the Blackwell policies. Once the value functions are known, their limits are obtained from their lowest nonzero coefficients. The values are bounded, and an identically zero numerator gives value zero.

9.2 The one-player case

In a Markov decision process (MDP) only one player chooses among actions. After negating rewards if necessary, take this player to be Max. We allow arbitrary real probabilities. More generally, we allow rewards and discounts to be polynomials in t of explicitly given degree, provided that the discounts belong to $[0, 1)$ for all sufficiently small $t > 0$. We will compute a single optimal policy and rational value functions valid throughout such an interval.

Dynamic programming [12, 121, 79] and LP [45] are classical approaches to MDPs. Ye [146, 147] gave strongly polynomial algorithms for a fixed discount factor, including simplex and policy iteration. Hansen, Miltersen and Zwick [75] extended the strategy iteration bound to two players with a fixed discount. Post and Ye [119] gave a strongly polynomial simplex algorithm for deterministic MDPs with variable discounts. See also [149].

For a fixed discounted Max MDP, the value vector is the unique minimizer of the linear program

$$\text{minimize} \quad \sum_i x_i \qquad \text{subject to} \quad x \geq T(x).$$

There is one inequality for each action. The fixed point is feasible, and every feasible x bounds it above by iterating $x \geq T(x)$. Theorem 1 solves the LP for arbitrary scalar data.

For polynomial data, impose primal feasibility, dual feasibility and equality of the two objectives, and write the joint system in standard form $A(t)y = b(t)$, $y \geq 0$. Let m be its number of variables and D a bound on its coefficient degrees. At each sufficiently small $t > 0$ there is a basic feasible solution. Along some sequence tending to zero the same basic columns and independent defining rows occur. Cramer's rule gives all coordinates over a common denominator, with numerator and denominator degrees at most mD. The unused equations hold identically, because they hold on an infinite sequence. The inequalities hold throughout a sufficiently small positive interval, because every rational function has a stable sign there. For related asymptotic LP methods see [81, 56, 144].

It suffices, therefore, to seek polynomials Y, s of degrees at most mD satisfying

$$A(t)Y(t) = s(t)b(t), \qquad Y(t) \geq 0, \qquad s(t) > 0 \quad \text{for all sufficiently small } t > 0.$$

The equality is a system of linear equations in the coefficients. For each polynomial in a sign condition, the first nonzero coefficient in order of increasing degree determines the sign. Starting at degree zero, impose nonnegativity of the current coefficient in every active condition. Use Theorem 1 to find a feasible point at which every coefficient that can be positive is positive simultaneously. The coefficient system is homogeneous, so positivity of a coefficient can be tested by adding the bound that it be at least one. Average the witnesses from the successful tests. If a coefficient is forced to zero, prescribe that equality and advance that condition by one degree. A weak condition whose coefficients are all zero is satisfied. The strictly positive denominator condition cannot be exhausted, since each relaxation contains every true solution. When no coefficient is forced to zero, the current point satisfies every sign condition. The number of advances is at most the number of coefficients. Each LP uses the original coefficient data and the prescribed zeros.

Divide Y by s and recover the value coordinates from the standard-form variables. Choose an action tight identically in t at each state. Such an action exists since the action scores are rational functions and their order stabilizes for small $t > 0$. This yields a Blackwell policy for (9.2), including arbitrary positive real weights. All degrees are polynomial and the coefficient arithmetic uses a constant number of layers of sums and products, as in Section 3.4.

9.3 Two players with arbitrary discounts

For two players, the Bellman operator may contract by an arbitrarily small amount even with a fixed number of states. We will halve the gap between two policy guarantees by estimating the values on a scale set by that gap. Only the transition probabilities will enter the slopes of the fixed-point computation at this scale.

First fix the discounts and rewards. An action a at state i has expected reward $r_a = \sum_{e \in E_a} p_e r_e$, transition row P_a, and discounted transition row B_a, where

$$P_a x = \sum_{e \in E_a} p_e x_{j(e)}, \qquad B_a x = \sum_{e \in E_a} p_e \beta_e x_{j(e)}.$$

Thus the action score in (9.1) is $r_a + B_a x$. We have $0 \le B_a \le P_a$ coordinatewise. Write $\lambda_a = 1 - B_a \mathbf{1} > 0$ for the mass lost to discounting. The row $P_a - B_a$ has mass λ_a.

Policy guarantees. Start with any Max policy and any Min policy. Fix Max's policy and solve the resulting Min MDP to get its vector of guarantees L. Fix Min's policy and solve the Max MDP to get U. Then $L \le v \le U$. If $W = \|U - L\|_\infty = 0$, these policies are optimal and $v = L$. For $W > 0$ we construct two new policies whose width is at most $W/2$.

The clipped update. Given $y \in \mathbb{R}^n$ with $\|v - y\|_\infty \le 2H$, $H > 0$, we seek a correction u that halves the error bound. Keep the discounted score at the center y, propagate the correction using P_a, and clip each coordinate to $[-H, H]$. Thus we solve

$$u_i = \Big[\operatorname*{ext}_{a \text{ at } i} \Phi_a(u) \Big]_{-H}^{H}, \qquad \Phi_a(u) = r_a + B_a y - y_i + P_a u. \tag{9.3}$$

Here brackets denote projection onto the indicated interval. This map is nonexpansive, because P_a is a probability row and clipping preserves the Lipschitz bound. The discounts enter only

the offsets. After scaling coordinates by $1/H$, Proposition 10 finds a fixed point in the unit cube. For earlier uses of clipping in mean-payoff games see [113, 112].

Every solution u of (9.3) satisfies

$$\|v - (y + u)\|_\infty \le H. \tag{9.4}$$

To prove the lower bound, translate by the clipping radius and put $z = y + u - H\mathbf{1}$. For every action a at i,

$$r_a + B_a z - z_i = \Phi_a(u) - u_i + H\lambda_a - (P_a - B_a)u \;\ge\; \Phi_a(u) - u_i, \tag{9.5}$$

because $u \le H\mathbf{1}$ and $P_a - B_a$ has mass λ_a. If $u_i > -H$, taking the extremum in (9.5) gives $z_i \le T_i(z)$. If $u_i = -H$, the initial error bound gives $z_i = y_i - 2H \le v_i$. A positive maximum $d = \max_i(z_i - v_i)$ would therefore be attained at a state where $z_i \le T_i(z)$. But monotonicity and contraction would give

$$d \le T_i(z) - T_i(v) \le T_i(v + d\mathbf{1}) - T_i(v) \le \bar\beta d < d.$$

Thus $z \le v$. Negate values and exchange the owners to get $v \le y + u + H\mathbf{1}$, proving (9.4).

If $y < v$ componentwise, (9.4) also forces $u_i > -H$ at every state. The maximizing actions for $\Phi_a(u)$ in (9.3) then certify $z \le T(z)$ in (9.5). They give a Max policy guaranteeing $z \ge v - 2H\mathbf{1}$. Figure 5 shows the translation in a two-state game. With discount $1/2$, let Max at A and Min at B choose between staying and moving to the other state. Let their action scores be

$$T_A(x) = \max\{3/4 + x_A/2,\ x_B/2\}, \qquad T_B(x) = \min\{1 + x_B/2,\ -1/4 + x_A/2\}.$$

The fixed point is $v = (3/2, 1/2)$, with Max staying and Min moving. At the low estimate in the figure, staying at A gives the larger clipped-problem action score and thus certifies the lower guarantee at both states.

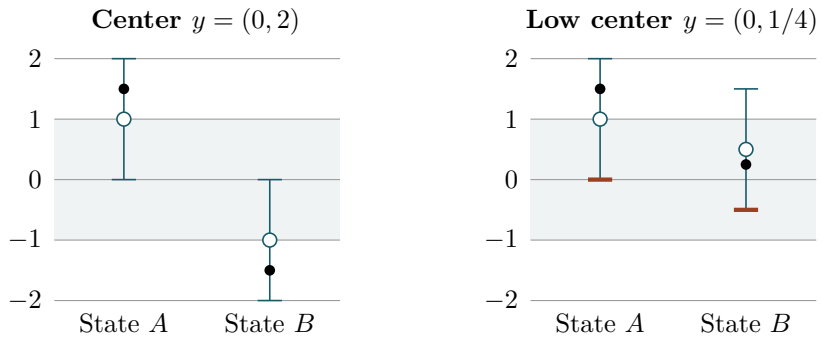

Figure 5. Clipped updates for the two-state game with $H = 1$. The shaded band is the allowed range of the correction u. Each vertical interval is $[u_i - 1, u_i + 1]$, with the correction at its open circle and the true error $v_i - y_i$ at its solid dot. In the right panel $y < v$ in both states, so the lower endpoints are certified policy guarantees relative to y. Adding y gives the absolute lower bound $z = (0, -1/4)$.

Halving the width. The midpoint $y = (L+U)/2$ has error at most $W/2$. Apply the clipped update three times with radii $W/4, W/8, W/16$, replacing y by $y + u$ each time. By (9.4) the

resulting y_* has error at most $W/16$. Set $H = W/8$. The lower estimate $y_* - H\mathbf{1}$ is strictly below v with error at most $2H$, so one more clipped solve gives a Max policy guaranteeing at least $v - (W/4)\mathbf{1}$. Likewise $y_* + H\mathbf{1}$ gives a Min policy guaranteeing at most $v + (W/4)\mathbf{1}$. The two new MDP guarantees thus have width at most $W/2$.

The possible scales of a policy error. Halving alone gives an approximation. To get exact termination in polynomially many iterations, we show that each positive width lies within a factor $2^{p(N)}$ of one of polynomially many positive numbers, for a fixed polynomial p. Here N is the length of the discrete input, including the probability bits. The scales may depend on the real input. The algorithm does not compute them.

Each of L, U is the payoff vector of a pair of policies, since a best response in an MDP is positional and works from every state. Moreover,

$$\max\{\|v - L\|_\infty, \|U - v\|_\infty\} \le W \le 2\max\{\|v - L\|_\infty, \|U - v\|_\infty\}.$$

It therefore suffices to control the error of each policy pair. Write Δ for the vector of its payoffs minus v and suppose that $G = \|\Delta\|_\infty > 0$. The *action gaps* $\delta_a = r_a + B_a v - v_i$, defined for every action a at i, are common to all pairs. For the action used by the pair at i,

$$\Delta_i = \delta_a + B_a \Delta.$$

Thus $g = \max_{a \text{ used}} |\delta_a| \le 2G$, and $g > 0$ by contraction. When G is much larger than g, the transition chain must retain a large error across a whole closed set of states.

Indeed, change the signs of Δ, δ if necessary so that some $\Delta_i = G$. Write $D_i = G - \Delta_i$. For the used action a at i,

$$\lambda_a G + B_a D = D_i + \delta_a \le D_i + g, \qquad P_a D \le 2(D_i + g).$$

For the second inequality use $D \le 2G\mathbf{1}$ and $(P_a - B_a)\mathbf{1} = \lambda_a$. Every positive transition has probability at least $2^{-\mathrm{poly}(N)}$. Along a transition $i \to j$, the last display bounds D_j by $2^{\mathrm{poly}(N)}(D_i + g)$. Each state reachable from a maximum can be reached by a path of length at most $n - 1$. Starting with $D_i = 0$ at the maximum, we obtain $D_j \le Kg$ throughout its reachable set for some uniform $K = 2^{\mathrm{poly}(N)}$.

If $G \le 2Kg$, the single gap of magnitude g supplies the scale. Otherwise $G/2 \le \Delta_j \le G$ on the reachable set. This set is closed under the undiscounted transitions of the pair. Choose a stationary distribution supported there, assigning mass d_a to the used action in each state. Such a distribution can be taken at a vertex of the stationarity equations with nonnegative coordinates summing to one. These equations use only transition probabilities, so every positive d_a has polynomial bit length. Stationarity and the recursion for Δ give

$$\sum_a d_a \delta_a = \sum_a d_a (P_a - B_a)\Delta.$$

The rows $P_a - B_a$ are nonnegative, stay in the same reachable set, and have masses λ_a. Hence

$$(G/2)\sum_a d_a \lambda_a \le \sum_a d_a \delta_a \le G \sum_a d_a \lambda_a.$$

The weighted discount loss $\sum_a d_a \lambda_a$ is within a factor $2^{\mathrm{poly}(N)}$ of the largest λ_a in the support. The numerator in the middle is a nonzero short rational combination of the action gaps.

It remains to bound the number of scales for such numerators. Fix the polynomial bit bound for their coefficient vectors. Take a maximal collection of nonzero magnitudes of combinations of the gaps that differ pairwise by factors greater than $2^{p(N)}$, where p is chosen below. By maximality each nonzero magnitude is within that factor of one in the collection. The selected coefficient vectors are linearly independent. Indeed, any dependence contains a dependence on at most $m+1$ vectors, where m is the number of actions. Clearing denominators and applying determinant bounds gives integer dependence coefficients of ℓ_1-norm less than $2^{p(N)}$ for a sufficiently large fixed polynomial p. Applying that dependence to the combination values gives a contradiction. The term of largest magnitude among the values with nonzero coefficient exceeds the sum of the magnitudes of all smaller terms even after weighting by their coefficients.

There are at most m numerator scales. For $G > 2Kg$, divide one of them by one of the m discount losses λ_a. For $G \le 2Kg$, use one of the m individual gap magnitudes. After increasing p to absorb the comparison factors, these at most $m^2 + m$ scales work for every pair, hence for every positive W. A sequence of successive halvings has at most $2p(N) + 1$ widths in the interval from $2^{-p(N)}$ to $2^{p(N)}$ times any one scale. The algorithm therefore reaches $W = 0$ after polynomially many iterations.

Arithmetic and parameter dependence. Only the two discrete policies pass between iterations. Within each iteration a constant number of MDP evaluations and clipped solves suffice. The step count is polynomial in N, and the bit bounds of Section 3.4 apply to each iteration.

Finally let the rewards and discounts be polynomials in t, valid for all sufficiently small positive t. Their degrees count in the input size, and the probabilities are fixed binary rationals. The MDP algorithm of Section 9.2 returns best responses and guarantees as rational functions. Compare two such functions using their signs at 0^+, so the policy selections hold throughout a sufficiently small positive interval.

In a clipped solve the height H and offsets $c_a = r_a + B_a y - y_i$ are now rational functions. Choose a common denominator d, positive near 0^+, so that dH and every dc_a are polynomials. Let $K > 0$ bound $|dH|$ on $0 < t < 1$, for example one plus the sum of the absolute values of its coefficients. In the scaled coordinates $\xi = du/K$, Equation (9.3) has offsets dc_a/K, rows P_a, and clipping endpoints $\pm dH/K$. These endpoints lie in $[-1, 1]$ near 0^+. The resulting nonexpansive map of the unit cube fits Proposition 10, which yields a rational ξ and the required extremizing actions. Recover $u = K\xi/d$.

Degrees remain polynomial because each iteration has a constant number of layers of calls, and passes only policies to the next iteration. Products and sums of polynomially many polynomials of these degrees can be computed by evaluation and interpolation using polynomial-size formulas in their coefficients, as required in Section 3.4. Every finite sequence of symbolic iterations agrees with a run at a common sufficiently small $t > 0$. The pointwise iteration bound is uniform in t, so the symbolic algorithm has the same bound. Applied to (9.2), it supplies Blackwell optimal policies even for arbitrary positive real weights. The mean-payoff limit of Section 9.1 completes the proof of Theorem 11.

10 Tropical feasibility and further questions

The prices in the LP algorithm enter a comparison by addition to a logarithmic magnitude. A system of tropical linear inequalities makes such comparisons directly. Akian, Gaubert and Guterman [1] related its feasibility to the values of a mean-payoff game (see also [107, 46]). Theorem 11 thus gives one further algorithmic consequence.

Corollary 12 (Tropical linear feasibility)**.** *Let a_{ri}, b_{ri} be explicitly given entries in $\mathbb{R} \cup \{-\infty\}$ of two $m \times n$ matrices. For the system*

$$\max_i(a_{ri} + x_i) \;\leq\; \max_j(b_{rj} + x_j) \qquad \text{for every row } r,$$

a deterministic strongly polynomial algorithm finds a solution in $(\mathbb{R} \cup \{-\infty\})^n$ finite at every coordinate that can be finite in any solution. An empty maximum equals $-\infty$.

Proof. Use the game of [1]. At coordinate i, Min may choose any row r with a_{ri} finite. At row r, Max may choose any coordinate j with b_{rj} finite. The two successive edges have rewards $-a_{ri}$ and b_{rj}. Give a vertex with no move a self-loop of reward zero if it belongs to Min, and of reward -1 if it belongs to Max.

A coordinate i finite in a solution has nonnegative game value. After Min chooses a row there, Max takes a term attaining the right-hand maximum. Its coordinate j is finite and $-a_{ri} + b_{rj} \geq x_i - x_j$. Summing over the play telescopes. The finite coordinates form a finite set, so the bounded endpoint contribution vanishes in the mean payoff.

Conversely, compute the values and fix an optimal policy for Max by Theorem 11. From a coordinate of nonnegative value, Min can reach no negative-weight cycle under this policy, since Min could then repeat the cycle. In particular, every reachable Max vertex has a move in the original game. At each coordinate reachable from a nonnegative-value start, set x_i to the least total weight of a walk starting at i with Max's policy fixed, allowing the empty walk. Erasing cycles shows that a simple walk attains the minimum. At all other coordinates put $x_i = -\infty$. For each permitted pair of moves i, r, j from a finite coordinate, concatenating the two moves with a minimizing walk from j gives $x_i \leq -a_{ri} + b_{rj} + x_j$. Hence the row inequalities hold. Dynamic programming over walks of length at most the number of vertices computes the minima in strongly polynomial time. □

A constant term in a maximum can be homogenized by adding one coordinate. The original system is feasible precisely when this coordinate can be finite. In that case subtract its value from every coordinate to recover a solution.

Further questions. Allamigeon, Benchimol, Gaubert and Joswig [2] showed how a strongly polynomial combinatorial simplex algorithm could yield strongly polynomial time for mean-payoff games. The algorithm of Theorem 1 uses quantitative comparisons of minors as well as their signs. Some comparisons in Section 5 have degree exponential in the dimensions. Can a strongly polynomial LP algorithm use predicates of polynomial degree throughout?

Theorem 1 takes the constraints as explicit input. For a polyhedron given by a separation oracle, known exact algorithms also involve geometric parameters [39]. A further question concerns transition probabilities in games. Theorem 11 permits arbitrary real probabilities for one player. For two players its step bound still depends on the bit lengths of the probabilities. Can that dependence be removed?

References

[1] M. Akian, S. Gaubert, and A. Guterman, *Tropical polyhedra are equivalent to mean payoff games*, Internat. J. Algebra Comput. **22** (2012), doi:10.1142/S0218196711006674.

[2] X. Allamigeon, P. Benchimol, S. Gaubert, and M. Joswig, *Combinatorial simplex algorithms can solve mean payoff games*, SIAM J. Optim. **24** (2014), 2096–2117.

[3] X. Allamigeon, P. Benchimol, S. Gaubert, and M. Joswig, *Log-barrier interior point methods are not strongly polynomial*, SIAM J. Appl. Algebra Geom. **2** (2018), 140–178.

[4] X. Allamigeon, S. Gaubert, R. D. Katz, and M. Skomra, *Universal complexity bounds based on value iteration for stochastic mean payoff games and entropy games*, Inform. and Comput. **302** (2025), 105236.

[5] X. Allamigeon, S. Gaubert, and N. Vandame, *No self-concordant barrier interior point method is strongly polynomial*, Proc. STOC 2022, 515–528.

[6] D. Andersson and P. B. Miltersen, *The complexity of solving stochastic games on graphs*, Proc. ISAAC 2009, 112–121.

[7] S. Arora, E. Hazan, and S. Kale, *The multiplicative weights update method: a meta-algorithm and applications*, Theory Comput. **8** (2012), 121–164.

[8] S. Awoniyi, *A strongly polynomial-time algorithm for the general linear programming problem*, arXiv:2503.12041 (2025; revised 2026).

[9] E. Bach and S. Huiberts, *Optimal smoothed analysis of the simplex method*, Proc. FOCS 2025, 1829–1856.

[10] S. Banach, *Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales*, Fund. Math. **3** (1922), 133–181.

[11] E. Batziou, J. Fearnley, S. Gordon, R. Mehta, and R. Savani, *Monotone contractions*, Proc. STOC 2025, 507–517.

[12] R. Bellman, *Dynamic programming*, Princeton Univ. Press, 1957.

[13] U. Betke, *Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem*, Discrete Comput. Geom. **32** (2004), 317–338.

[14] H. Björklund, S. Sandberg, and S. Vorobyov, *A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games*, Proc. MFCS 2004, 673–685.

[15] H. Björklund and S. Vorobyov, *A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games*, Discrete Appl. Math. **155** (2007), 210–229.

[16] D. Blackwell, *An analog of the minimax theorem for vector payoffs*, Pacific J. Math. **6** (1956), 1–8.

[17] D. Blackwell, *Discrete dynamic programming*, Ann. Math. Statist. **33** (1962), 719–726.

[18] L. Blum, M. Shub, and S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bull. Amer. Math. Soc. (N.S.) **21** (1989), 1–46.

[19] E. Boros, K. Elbassioni, V. Gurvich, and K. Makino, *A pseudo-polynomial algorithm for mean payoff stochastic games with perfect information and few random positions*, Inform. and Comput. **267** (2019), 74–95.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[21] Luboš Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin, *Faster algorithms for mean-payoff games*, Formal Methods System Design **38** (2011), 97–118.

[22] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan, *Deciding parity games in quasi-polynomial time*, SIAM J. Comput. **51** (2022), STOC17-152–STOC17-188.

[23] X. Chen, Y. Li, and M. Yannakakis, *Computing a fixed point of contraction maps in polynomial queries*, J. ACM **72** (2025), Art. 25.

[24] X. Chen, Y. Li, and M. Yannakakis, *Quadratic speedup for computing contraction fixed points*, arXiv:2602.10296 (2026).

[25] X. Chen, Y. Li, and M. Yannakakis, *A decomposition theorem and a simple algorithm for contraction fixed points*, arXiv:2611.01302 (2026).

[26] S. Chubanov, *A polynomial projection algorithm for linear feasibility problems*, Math. Program. **153** (2015), 687–713.

[27] M. B. Cohen, Y. T. Lee, and Z. Song, *Solving linear programs in the current matrix multiplication time*, Proc. STOC 2019, 938–942.

[28] A. Condon, *The complexity of stochastic games*, Inform. and Comput. **96** (1992), 203–224.

[29] R. W. Cottle, *Note on a fundamental theorem in quadratic programming*, J. Soc. Indust. Appl. Math. **12** (1964), 663–665.

[30] R. W. Cottle and G. B. Dantzig, *Complementary pivot theory of mathematical programming*, Linear Algebra Appl. **1** (1968), 103–125.

[31] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*, Academic Press, 1992.

[32] B. Cui, H. Li, and X. Miao, *Extending Tardos's theorem to separable convex quadratic programming*, arXiv:2610.07465v2 (2026).

[33] D. Dadush, S. Huiberts, B. Natura, and L. A. Végh, *A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix*, Math. Program. **204** (2024), 135–206.

[34] D. Dadush, Z. K. Koh, B. Natura, N. Olver, and L. A. Végh, *A strongly polynomial algorithm for linear programs with at most two nonzero entries per row or column*, Proc. STOC 2024, 1561–1572.

[35] D. Dadush, Z. K. Koh, B. Natura, N. Olver, and L. A. Végh, *Solving classes of linear programs with strongly polynomial straight line complexity in strongly polynomial time*, arXiv:2610.26073 (2026).

[36] D. Dadush, Z. K. Koh, B. Natura, and L. A. Végh, *An accelerated Newton–Dinkelbach method and its application to two variables per inequality systems*, Math. Oper. Res. **48** (2023), 1934–1958; arXiv:2004.08634.

[37] D. Dadush, B. Natura, and L. A. Végh, *Revisiting Tardos's framework for linear programming: Faster exact solutions using approximate solvers*, Proc. FOCS 2020, 931–942.

[38] D. Dadush, L. A. Végh, and G. Zambelli, *Rescaling algorithms for linear conic feasibility*, arXiv:1611.06427 (2016; revised 2019).

[39] D. Dadush, L. A. Végh, and G. Zambelli, *On finding exact solutions of linear programs in the oracle model*, arXiv:2606.11820 (2026; conference version, SODA 2022).

[40] G. B. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, Activity Analysis of Production and Allocation (T. C. Koopmans, ed.), Cowles Commission Monograph 13, Wiley, 1951, 339–347.

[41] C. Daskalakis, G. Farina, and B. H. Zhang, *Polynomial-time computation of ℓ_p-contraction fixed points for even p*, arXiv:2607.23863 (2026).

[42] C. Daskalakis and C. H. Papadimitriou, *Continuous local search*, Proc. SODA 2011, 790–804.

[43] L. de Alfaro, T. A. Henzinger, and R. Majumdar, *Discounting the future in systems theory*, Proc. ICALP 2003, 1022–1037.

[44] J. A. De Loera, J. Haddock, and L. Rademacher, *The minimum Euclidean-norm point in a convex polytope: Wolfe's combinatorial algorithm is exponential*, SIAM J. Comput. **49** (2020), 138–169.

[45] F. d'Epenoux, *A probabilistic production and inventory problem*, Management Sci. **10** (1963), 98–108.

[46] V. Dhingra and S. Gaubert, *How to solve large scale deterministic games with mean payoff by policy iteration*, Proc. VALUETOOLS 2006, Art. 12.

[47] J. Dunagan and S. Vempala, *A simple polynomial-time rescaling algorithm for solving linear programs*, Math. Program. **114** (2008), 101–114.

[48] D. Easley and J. Kleinberg, *Networks, crowds, and markets: reasoning about a highly connected world*, Cambridge Univ. Press, 2010.

[49] A. Ehrenfeucht and J. Mycielski, *Positional strategies for mean payoff games*, Internat. J. Game Theory **8** (1979), 109–113.

[50] F. Ekbatani, B. Natura, and L. A. Végh, *Circuit imbalance measures and linear programming*, Surveys in Combinatorics 2022, Cambridge Univ. Press, 64–114.

[51] K. Etessami, C. H. Papadimitriou, A. Rubinstein, and M. Yannakakis, *Tarski's theorem, supermodular games, and the complexity of equilibria*, Proc. ITCS 2020 (LIPIcs **151**), 18:1–18:19.

[52] K. Etessami and M. Yannakakis, *On the complexity of Nash equilibria and other fixed points*, SIAM J. Comput. **39** (2010), 2531–2597.

[53] J. Farkas, *Theorie der einfachen Ungleichungen*, J. Reine Angew. Math. **124** (1902), 1–27.

[54] J. Fearnley, S. Gordon, R. Mehta, and R. Savani, *Unique end of potential line*, J. Comput. System Sci. **114** (2020), 1–35.

[55] A. Feodorov and S. Haslebacher, *Faster approximate fixed points of ℓ_∞-contractions*, arXiv: 2604.01006 (2026).

[56] J. A. Filar, E. Altman, and K. E. Avrachenkov, *An asymptotic simplex method for singularly perturbed linear programs*, Oper. Res. Lett. **30** (2002), 295–307.

[57] A. Frank and É. Tardos, *An application of simultaneous Diophantine approximation in combinatorial optimization*, Combinatorica **7** (1987), 49–65.

[58] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Res. Logist. Quart. **3** (1956), 95–110.

[59] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, J. Comput. System Sci. **55** (1997), 119–139.

[60] S. Fujishige, T. Kitahara, and L. A. Végh, *An update-and-stabilize framework for the minimum-norm-point problem*, Math. Program. **210** (2025), 281–311.

[61] D. Gale, H. W. Kuhn, and A. W. Tucker, *Linear programming and the theory of games*, in Activity Analysis of Production and Allocation (T. C. Koopmans, ed.), Wiley, 1951, 317–335.

[62] B. Gärtner and L. Y. Rüst, *Simple stochastic games and P-matrix generalized linear complementarity problems*, Proc. FCT 2005, 209–220.

[63] S. Gaubert, J. Grand-Clément, and R. D. Katz, *Thresholds for sensitive optimality and Blackwell optimality in stochastic games*, Advances in Neural Information Processing Systems 38, 2025.

[64] M. Ghadiri, R. Peng, and S. S. Vempala, *The bit complexity of efficient continuous optimization*, Proc. FOCS 2023, 2059–2070.

[65] D. Gillette, *Stochastic games with zero stop probabilities*, Contributions to the Theory of Games, Vol. III (M. Dresher, A. W. Tucker, and P. Wolfe, eds.), Princeton Univ. Press, 1957, 179–187.

[66] H. Gimbert and W. Zielonka, *Deterministic priority mean-payoff games as limits of discounted games*, Proc. ICALP 2006 (II), 312–323.

[67] H. Gimbert and W. Zielonka, *Limits of multi-discounted Markov decision processes*, Proc. LICS 2007, 89–98.

[68] H. Gimbert and W. Zielonka, *Blackwell-optimal strategies in priority mean-payoff games*, Electron. Proc. Theor. Comput. Sci. **25** (2010), 7–21.

[69] A. J. Goldman and A. W. Tucker, *Theory of linear programming*, in Linear Inequalities and Related Systems (H. W. Kuhn and A. W. Tucker, eds.), Annals of Mathematics Studies 38, Princeton Univ. Press, 1956.

[70] F. Granot and J. Skorin-Kapov, *Towards a strongly polynomial algorithm for strictly convex quadratic programs: an extension of Tardos' algorithm*, Math. Programming **46** (1990), 225–236.

[71] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Algorithms and Combinatorics 2, Springer, 1988.

[72] V. A. Gurvich, A. V. Karzanov, and L. G. Khachiyan, *Cyclic games and an algorithm to find minimax cycle means in directed graphs*, USSR Comput. Math. and Math. Phys. **28** (1988), 85–91.

[73] N. Halman, *Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems*, Algorithmica **49** (2007), 37–50.

[74] T. D. Hansen and R. Ibsen-Jensen, *The complexity of interior point methods for solving discounted turn-based stochastic games*, arXiv:1304.1888v2 (2014).

[75] T. D. Hansen, P. B. Miltersen, and U. Zwick, *Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor*, J. ACM **60** (2013), Art. 1.

[76] T. D. Hansen and U. Zwick, *An improved version of the Random-Facet pivoting rule for the simplex algorithm*, Proc. STOC 2015, 209–218.

[77] S. Haslebacher, J. Lill, P. Schnider, and S. Weber, *Query-efficient fixpoints of ℓ_p-contractions*, Proc. FOCS 2025, 2037–2053.

[78] S.-P. Hong and S. Verma, *A note on the strong polynomiality of convex quadratic programming*, Math. Programming **68** (1995), 131–139.

[79] R. A. Howard, *Dynamic programming and Markov processes*, MIT Press, 1960.

[80] Z. Huang, L. Khachiyan, and K. Sikorski, *Approximating fixed points of weakly contracting mappings*, J. Complexity **15** (1999), 200–213.

[81] R. G. Jeroslow, *Asymptotic linear programming*, Oper. Res. **21** (1973), 1128–1141.

[82] M. Jurdziński, *Deciding the winner in parity games is in UP ∩ co-UP*, Inform. Process. Lett. **68** (1998), 119–124.

[83] M. Jurdziński and R. Savani, *A simple P-matrix linear complementarity problem for discounted games*, in Proc. CiE 2008, 283–293.

[84] G. Kalai, *A subexponential randomized simplex algorithm* (extended abstract), Proc. STOC 1992, 475–482.

[85] G. Kalai and D. J. Kleitman, *A quasi-polynomial bound for the diameter of graphs of polyhedra*, Bull. Amer. Math. Soc. (N.S.) **26** (1992), 315–316.

[86] L. V. Kantorovich, *Mathematical methods of organizing and planning production*, Management Sci. **6** (1960), 366–422 (translation; Russian original, 1939).

[87] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), 373–395.

[88] L. G. Khachiyan, *A polynomial algorithm in linear programming* (Russian), Dokl. Akad. Nauk SSSR **244** (1979), 1093–1096.

[89] L. G. Khachiyan, *Polynomial algorithms in linear programming* (Russian), Zh. Vychisl. Mat. Mat. Fiz. **20** (1980), 51–68.

[90] V. Klee and G. J. Minty, *How good is the simplex algorithm?* Inequalities III (O. Shisha, ed.), Academic Press, 1972, 159–175.

[91] M. Kojima, S. Mizuno, and A. Yoshise, *A polynomial-time algorithm for a class of linear complementarity problems*, Math. Programming **44** (1989), 1–26.

[92] M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan, *The polynomial solvability of convex quadratic programming*, Zh. Vychisl. Mat. Mat. Fiz. **20** (1980), 1319–1323 (in Russian).

[93] H. W. Kuhn, *Nonlinear programming. A historical view*, in Nonlinear Programming (R. W. Cottle and C. E. Lemke, eds.), SIAM–AMS Proc. 9, AMS, 1976, 1–26.

[94] H. W. Kuhn and A. W. Tucker, *Nonlinear programming*, in Proc. Second Berkeley Symp. Math. Statist. Probab., University of California Press, 1951, 481–492.

[95] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann. **261** (1982), 515–534.

[96] V. F. Lev and R. Yuster, *On the size of dissociated bases*, arXiv:1005.0155 (2010).

[97] T. M. Liggett and S. A. Lippman, *Stochastic games with perfect information and time average payoff*, SIAM Rev. **11** (1969), 604–607.

[98] N. Littlestone and M. K. Warmuth, *The weighted majority algorithm*, Inform. and Comput. **108** (1994), 212–261.

[99] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, *Applications of second-order cone programming*, Linear Algebra Appl. **284** (1998), 193–228.

[100] W. Ludwig, *A subexponential randomized algorithm for the simple stochastic game problem*, Inform. and Comput. **117** (1995), 151–155.

[101] O. L. Mangasarian, *A simple characterization of solution sets of convex programs*, Oper. Res. Lett. **7** (1988), 21–26.

[102] J. Matoušek, M. Sharir, and E. Welzl, *A subexponential bound for linear programming*, Algorithmica **16** (1996), 498–516.

[103] E. J. McShane, *Extension of range of functions*, Bull. Amer. Math. Soc. **40** (1934), 837–842.

[104] N. Megiddo, *Is binary encoding appropriate for the problem-language relationship?* Theoret. Comput. Sci. **19** (1982), 337–341.

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

[106] N. Megiddo, *Linear programming in linear time when the dimension is fixed*, J. ACM **31** (1984), 114–127.

[107] R. H. Möhring, M. Skutella, and F. Stork, *Scheduling with AND/OR precedence constraints*, SIAM J. Comput. **33** (2004), 393–415.

[108] B. Natura, *Circuit diameter of polyhedra is strongly polynomial*, arXiv:2602.06958 (2026).

[109] Y. Nesterov and A. Nemirovskii, *Interior-point polynomial algorithms in convex programming*, SIAM, 1994.

[110] P. Ohlmann, *A symmetric recursive algorithm for mean-payoff games*, arXiv:2603.07555 (2026).

[111] N. Olver and L. A. Végh, *A simpler and faster strongly polynomial algorithm for generalized flow maximization*, J. ACM **67** (2020), Art. 10.

[112] OpenAI, *Deterministic quasipolynomial-time mean-payoff games*, OpenAI Math Release preprint OAI:Deterministic-quasipolynomial-time-mean-payoff-games-September-25-2026, 2026.

[113] OpenAI, *Randomized quasipolynomial-time mean-payoff games*, OpenAI Math Release preprint OAI:Randomized-quasipolynomial-time-mean-payoff-games-September-25-2026, 2026.

[114] OpenAI, *Turn-Based Stochastic Mean-Payoff Games in Deterministic Quasipolynomial Time*, OpenAI Math Release preprint OAI:Turn-Based-Stochastic-Mean-Payoff-Games-in-Deterministic-Quasipolynomial-Time-October-5-2026, 2026.

[115] OpenAI, *Mean-payoff parity games in quasipolynomial time*, OpenAI Math Release preprint OAI:Mean-payoff-parity-games-in-quasipolynomial-time-October-5-2026, 2026.

[116] OpenAI, *Superpolynomial graph diameter of simple polytopes*, OpenAI Math Release preprint OAI:Superpolynomial-graph-diameter-of-simple-polytopes-November-24-2026, 2026.

[117] P. Parys, *Parity games: Zielonka's algorithm in quasi-polynomial time*, Proc. MFCS 2019, LIPIcs 138, 10:1–10:13.

[118] J. Peña and N. Soheili, *Projection and rescaling algorithm for finding maximum support solutions to polyhedral conic systems*, Math. Oper. Res. **47** (2022), 3304–3316.

[119] I. Post and Y. Ye, *The simplex method is strongly polynomial for deterministic Markov decision processes*, Math. Oper. Res. **40** (2015), 859–868.

[120] A. Puri, *Theory of hybrid systems and discrete event systems*, Ph.D. thesis, University of California at Berkeley, 1995.

[121] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, Wiley, 1994.

[122] T. Radzik, *Fractional combinatorial optimization*, Handbook of Combinatorial Optimization (D.-Z. Du and P. M. Pardalos, eds.), 1998, 429–478.

[123] J. Renegar, *A polynomial-time algorithm, based on Newton's method, for linear programming*, Math. Program. **40** (1988), 59–93.

[124] R. T. Rockafellar, *The elementary vectors of a subspace of* $\mathbb{R}^N$, Combinatorial Mathematics and Its Applications (R. C. Bose and T. A. Dowling, eds.), Univ. of North Carolina Press, 1969, 104–127.

[125] F. Santos, *A counterexample to the Hirsch conjecture*, Ann. of Math. (2) **176** (2012), 383–412.

[126] L. S. Shapley, *Stochastic games*, Proc. Natl. Acad. Sci. USA **39** (1953), 1095–1100.

[127] S. Shellman and K. Sikorski, *A recursive algorithm for the infinity-norm fixed point problem*, J. Complexity **19** (2003), 799–834.

[128] N. Z. Shor, *Cut-off method with space extension in convex programming problems* (Russian), Kibernetika (Kiev) (1977), no. 1, 94–95.

[129] K. Sikorski, C.-W. Tsay, and H. Woźniakowski, *An ellipsoid algorithm for the computation of fixed points*, J. Complexity **9** (1993), 181–200.

[130] S. Smale, *Mathematical problems for the next century*, Math. Intelligencer **20** (1998), no. 2, 7–15.

[131] A. Soleymani and G. Farina, *A quasipolynomial algorithm for* ℓ_∞ *contractions*, arXiv:2610.30122 (2026).

[132] D. A. Spielman and S.-H. Teng, *Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time*, J. ACM **51** (2004), 385–463.

[133] É. Tardos, *A strongly polynomial minimum cost circulation algorithm*, Combinatorica **5** (1985), 247–255.

[134] É. Tardos, *A strongly polynomial algorithm to solve combinatorial linear programs*, Oper. Res. **34** (1986), 250–256.

[135] R. E. Tarjan, *Efficient algorithms for network optimization*, in Proc. ICM 1983, vol. 2, PWN/North-Holland, 1984, 1619–1635.

[136] L. Truffet, *Looking for all solutions of a set of max-atoms solves the max atom problem in strongly polynomial time*, Modélisation des Systèmes Réactifs (MSR 2025), hal-05491586, 2025.

[137] L. Truffet, *Substitution for minimizing/maximizing a tropical linear (fractional) programming*, arXiv:2603.26423 (2026).

[138] A. W. Tucker, *Dual systems of homogeneous linear relations*, in Linear Inequalities and Related Systems (H. W. Kuhn and A. W. Tucker, eds.), Annals of Mathematics Studies 38, Princeton Univ. Press, 1956, 3–18.

[139] P. M. Vaidya, *Speeding-up linear programming using fast matrix multiplication* (extended abstract), Proc. FOCS 1989, 332–337.

[140] J. van den Brand, *A deterministic linear program solver in current matrix multiplication time*, Proc. SODA 2020, 259–278.

[141] S. A. Vavasis and Y. Ye, *A primal-dual interior point method whose running time depends only on the constraint matrix*, Math. Program. **74** (1996), 79–120.

[142] L. A. Végh, *A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives*, SIAM J. Comput. **45** (2016), 1729–1761.

[143] L. A. Végh, *A strongly polynomial algorithm for generalized flow maximization*, Math. Oper. Res. **42** (2017), 179–211.

[144] A. F. Veinott, Jr., *Discrete dynamic programming with sensitive discount optimality criteria*, Ann. Math. Statist. **40** (1969), 1635–1660.

[145] P. Wolfe, *Finding the nearest point in a polytope*, Math. Programming **11** (1976), 128–149.

[146] Y. Ye, *A new complexity result on solving the Markov decision problem*, Math. Oper. Res. **30** (2005), 733–749.

[147] Y. Ye, *The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate*, Math. Oper. Res. **36** (2011), 593–603.

[148] D. B. Yudin and A. S. Nemirovskii, *Informational complexity and efficient methods for the solution of convex extremal problems* (Russian), Ekonom. Mat. Metody **12** (1976), 357–369.

[149] U. Zwick, *Improved subexponential analysis of the Random-Action-Removal algorithm for 2-player turn-based games and non-binary AUSOs*, arXiv:2607.06334 (2026).

[150] U. Zwick and M. Paterson, *The complexity of mean payoff games on graphs*, Theoret. Comput. Sci. **158** (1996), 343–359.