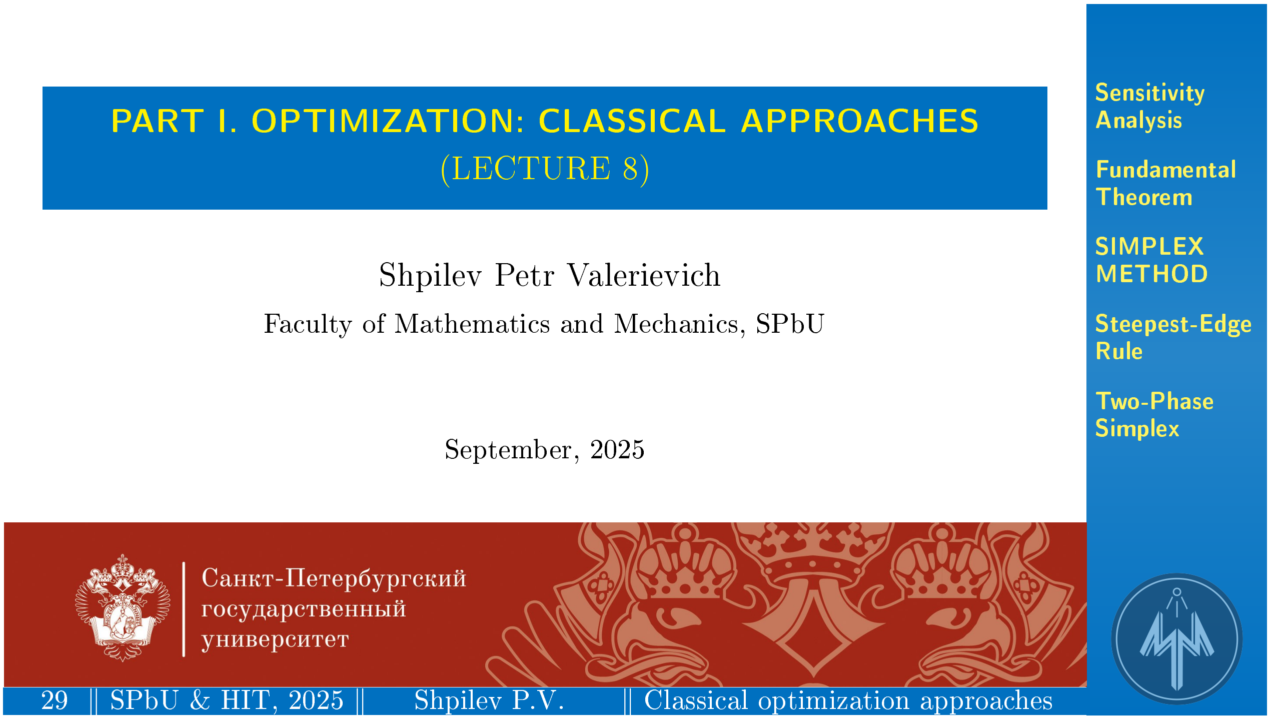


PART I. OPTIMIZATION: CLASSICAL APPROACHES (LECTURE 8)

Shpilev Petr Valerievich

Faculty of Mathematics and Mechanics, SPbU

September, 2025

Санкт-Петербургский
государственный
университет

Sensitivity Analysis

Fundamental Theorem

SIMPLEX METHOD

Steepest-Edge Rule

Two-Phase Simplex

Comments

In this lecture, we turn to linear programming and develop the simplex method as a central algorithm for solving such problems. We begin with sensitivity analysis, the role of full row rank, and the link between basic feasible points and polytope vertices, leading to the proof of the fundamental theorem of linear programming. The revised simplex method is then introduced, with emphasis on its KKT interpretation, pivoting mechanics, algebraic structure, and termination properties. Through examples and geometric illustrations, we trace the simplex iterates and examine practical implementation aspects, including LU updates, index selection, and the steepest-edge rule. Finally, we address initialization via the two-phase simplex method, detailing the process of finding a feasible starting point and carrying out Phase II optimization.

Sensitivity Analysis in Linear Programming

The multipliers λ, s for the primal problem (??), indicate the sensitivity of the optimal objective to perturbations in the constraints.

- For small perturbations Δb, Δx, $\Delta\lambda$, Δs (with Δx, Δs having zeros where x, s do, and $x^T s = 0$):
$$0 = x^T s = x^T \Delta s = (\Delta x)^T s = (\Delta x)^T \Delta s.$$
- By strong duality, for original and perturbed problems:
$$c^T x = b^T \lambda, \quad c^T(x + \Delta x) = (b + \Delta b)^T(\lambda + \Delta\lambda).$$
- Perturbed feasibility: $A(x + \Delta x) = b + \Delta b$, $A^T \Delta\lambda = -\Delta s$.
- Change in optimal objective:
$$\begin{aligned} c^T \Delta x &= (b + \Delta b)^T(\lambda + \Delta\lambda) - b^T\lambda \\ &= (b + \Delta b)^T \Delta\lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T A^T \Delta\lambda + (\Delta b)^T \lambda \\ &= (x + \Delta x)^T \Delta s + (\Delta b)^T\lambda = (\Delta b)^T \lambda. \end{aligned}$$
- For $\Delta b = \epsilon e_j$ (unit vector in $\mathbb{R}^n$), we have for all ϵ sufficiently small that:
$$c^T \Delta x = \epsilon \lambda_j,$$
i.e., the change in objective is λ_j times the size of the perturbation to b_j.

Comments

Sensitivity analysis in linear programming provides a framework for understanding how small perturbations in the input data affect the optimal solution. The central objects here are the Lagrange multipliers, usually denoted by λ, and the slack variables, denoted by s. These multipliers measure how sensitive the optimal objective value is to changes in the right-hand side of the constraints. To make this precise, one considers infinitesimal perturbations Δb in the constraints and studies the induced variations in the primal and dual variables, Δx, $\Delta\lambda$, and Δs. An important property of complementary slackness ensures that perturbations preserve orthogonality: the product of x and s remains zero, as do the mixed terms.

Strong duality tells us that the primal and dual objective values are always equal at optimality. When the system is perturbed, the balance between $c^T x$ and $b^T\lambda$ remains consistent. Careful expansion of this relation reveals that the change in the objective function depends solely on the perturbation to b multiplied by the multiplier λ. This shows that the dual variable λ acts as a shadow price: it measures the marginal impact on the objective of increasing a constraint by one unit. For example, if we perturb a single entry b_j by ϵ, then the change in the objective is exactly $\epsilon\lambda_j$, provided ϵ is small enough. This result justifies interpreting the multipliers as rates of change of the optimal value with respect to the constraints. In practice, this interpretation is fundamental in economics and operations research, where one is often interested not only in an optimal plan but also in its stability under fluctuations of resources or demand.

Full Row Rank and Basic Feasible Points

Assume the $m \times n$ matrix A in the primal problem $\min c^T x$, subject to $Ax = b$, $x \geq 0$, has full row rank.

- In practice, preprocessing removes redundancies and reformulates constraints (using slack, surplus, or artificial variables) to ensure this property.

Definition: Basic Feasible Point

A vector x is a *basic feasible point* for the primal problem if it is feasible ($Ax = b$, $x \geq 0$) and there exists a subset $\mathcal{B} \subseteq \{1, 2, \ldots, n\}$ such that:

- $\mathcal{B}$ contains exactly m indices.
- $i \notin \mathcal{B} \Rightarrow x_i = 0$ (that is, the bound $x_i \geq 0$ can be inactive only if $i \in \mathcal{B}$).
- The $m \times m$ matrix $B = [A_i]_{i \in \mathcal{B}}$, where A_i is the ith column of A, is nonsingular.

- Such a set $\mathcal{B}$ is called a *basis* for the problem (??), and the corresponding matrix B is the *basis matrix*.

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

Comments

A central structural property of linear programs is the rank of the constraint matrix A. By assuming that A has full row rank, we eliminate redundancies among the equations and ensure that the system is well-posed. In practice, preprocessing techniques identify and remove such redundancies or reformulate constraints by introducing slack, surplus, or artificial variables so that the model achieves this rank condition. With this assumption, the concept of a basic feasible point becomes meaningful.

A feasible solution is any nonnegative vector x that satisfies $Ax = b$. Among these solutions, one calls x a basic feasible point if the support of x, that is, the set of indices of its positive components, has no more than m elements, where m is the number of constraints. Moreover, the corresponding m columns of A, collected into the basis matrix B, must be linearly independent, so that B is invertible. In this setting, the values of the basic variables are uniquely determined by solving the corresponding system ($Bx_{\mathcal{B}} = b$).

This construction is not arbitrary. Each basic feasible point corresponds to a vertex, or corner point, of the polyhedron defined by the constraints. The nonsingularity of B guarantees that exactly n hyperplanes intersect at this point, producing a vertex. Conversely, any vertex of the feasible region can be represented as a basic feasible point relative to some choice of basis. This equivalence, which we will formally prove a little bit later, establishes a bridge between algebraic descriptions of solutions and geometric intuition. It also provides the foundation for algorithms such as the simplex method, which explore feasible solutions by moving from one vertex to another. Understanding basic feasible points is therefore essential both for theory and computation in linear optimization.

Simplex Method and Fundamental Theorem

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

The simplex method examines only basic feasible points and converges to a solution of the primal problem $\min c^T x$, subject to $Ax = b$, $x \geq 0$, if:

(a) The problem has basic feasible points.

(b) At least one is a basic optimal point (a solution that is also a basic feasible point).

Theorem 32: Fundamental Theorem of Linear Programming

(i) If the primal problem has a nonempty feasible region, then there is at least one basic feasible point.

(ii) If the primal problem has solutions, then at least one solution is a basic optimal point.

(iii) If the nonempty feasible region is bounded, then the primal problem has an optimal solution.

Proof: Among all feasible vectors x ($Ax = b$, $x \geq 0$), choose one with the minimal number of nonzero components, say p, with nonzeros $x_1, x_2, \ldots, x_p$, satisfying:

$$\sum_{i=1}^{p} A_i x_i = b.$$

Comments

The simplex method is one of the most celebrated algorithms in optimization. Its core idea is that instead of searching through all feasible solutions, which may be infinitely many, one only needs to examine the basic feasible points. Because each such point corresponds to a vertex of the feasible region, the algorithm effectively navigates along the edges of the polyhedron from one vertex to another. It starts at some initial vertex and moves step by step to adjacent vertices, improving the objective function value at each move, until it reaches a point where no further improvement is possible. That point is then optimal.

The legitimacy of this approach rests on the Fundamental Theorem of Linear Programming. This theorem has three parts. First, whenever the feasible region is nonempty, there exists at least one basic feasible point. This assures that the simplex method always has a starting point. Second, if there exists an optimal solution, then one can be found among the basic feasible points. In other words, the search for an optimal solution can be restricted to vertices without loss of generality. Third, if the feasible region is both nonempty and bounded, then there necessarily exists an optimal solution.

These statements ensure that the simplex method is not only heuristic but theoretically justified. It is guaranteed to terminate with an optimal solution whenever one exists and the feasible set is bounded. The proof of the theorem relies on selecting feasible points with the minimal number of nonzero variables denoted by p, and then analyzing the linear independence of the corresponding columns of A.

Proof of Fundamental Theorem (i)

Suppose columns $A_1, \dots, A_p$ are linearly dependent, so:

$$A_p = \sum_{i=1}^{p-1} A_i z_i,$$

for scalars $z_1, \dots, z_{p-1}$. Define:

$$x(\epsilon) = x + \epsilon \left[z_1, \dots, z_{p-1}, -1, 0, \dots, 0\right]^T.$$

- $Ax(\epsilon) = b$ for any ϵ, and $x_i(\epsilon) > 0$ for $i = 1, \dots, p$ if $|\epsilon|$ is sufficiently small.
- There exists $\bar{\epsilon} \in (0, x_p]$ such that $x_i(\bar{\epsilon}) = 0$ for some $i = 1, \dots, p$, so $x(\bar{\epsilon})$ is feasible with at most $p - 1$ nonzeros, contradicting the minimality of p.

Hence, $A_1, \dots, A_p$ are linearly independent, so $p \leq m$.

- If $p = m$, then x is a basic feasible point with $\mathcal{B} = \{1, \dots, m\}$.
- If $p < m$, since A has full row rank, choose $m - p$ columns from $A_{p+1}, \dots, A_n$ to form m linearly independent vectors, and construct $\mathcal{B}$ by adding their indices to $\{1, \dots, p\}$. The proof of (i) is complete.

Comments

If the associated columns of A are linearly dependent, then one can form a nontrivial linear combination in which one column is expressed in terms of the others. Using this relation, it is possible to perturb the solution slightly, redistributing weight among the variables while preserving feasibility. The construction $x(\epsilon)$ illustrates this idea: the vector is adjusted by a small multiple of the dependence relation.

This perturbed solution remains feasible for sufficiently small ϵ, and its nonzero components remain positive. By choosing an appropriate value of ϵ, one can drive at least one variable to zero while keeping the others nonnegative. The resulting solution then has fewer than p nonzeros, contradicting the minimality assumption. The contradiction shows that the columns must in fact be linearly independent. Hence, the number of nonzero variables cannot exceed the number of constraints, so p is at most m.

If p equals m, then the solution is already a basic feasible point, since exactly m linearly independent columns correspond to the nonzero entries. If p is less than m, the full row rank assumption ensures that one can add extra columns from A to form a complete basis of m independent vectors. This allows extending the support of the solution to form a basis set, and the solution is then recognized as a basic feasible point. Thus, in every case, a feasible region that is nonempty contains at least one basic feasible point, completing the proof of the first statement.

Proof of Fundamental Theorem (ii)

Let x^* be a solution to $\min c^T x$, subject to $Ax = b$, $x \geq 0$, with minimal nonzero components p, say $x_1^*, \ldots, x_p^*$.

- If columns $A_1, \ldots, A_p$ are linearly dependent, define:
$$x^*(\epsilon) = x^* + \epsilon z,$$
where $z = \left[z_1, \ldots, z_{p-1}, -1, 0, \ldots, 0\right]^T$ as in the proof of (i).
- $x^*(\epsilon)$ is feasible for small $|\epsilon|$. Since x^* is optimal:
$$c^T(x^* + \epsilon z) \geq c^T x^* \Rightarrow \epsilon c^T z \geq 0,$$
for small ϵ (positive and negative), so $c^T z = 0$ and $c^T x^*(\epsilon) = c^T x^*$.
- As in (i), there exists $\bar{\epsilon} > 0$ such that $x^*(\bar{\epsilon})$ is feasible and optimal with at most $p - 1$ nonzeros, contradicting the minimality of p.
- Thus, $A_1, \ldots, A_p$ are linearly independent. Using the same reasoning as in (i), x^* is a basic feasible point, hence a basic optimal point.

The final statement (iii) is a consequence of finite termination of the simplex method. We comment on the latter property after consideration of this method.

□

Comments

The second part of the Fundamental Theorem concerns optimality. Consider an optimal solution x^* with the minimal number of nonzero variables, again denoted by p. If the corresponding columns of A are linearly dependent, then as before one can construct a perturbation by adding ϵ times a suitable vector z. The perturbed vector remains feasible for small values of ϵ, because the linear dependence guarantees that the equality constraints remain satisfied.

Now, since x^* is optimal, any perturbation cannot strictly improve the objective. This leads to the inequality $c^T(x^* + \epsilon z) \geq c^T x^*$. Simplifying, one deduces that $\epsilon c^T z$ must be nonnegative for both positive and negative values of ϵ. The only possibility is that $c^T z = 0$. Therefore, the perturbation does not change the objective value.

Using the same construction as before, one can choose ϵ so that at least one variable becomes zero, reducing the number of nonzero entries while preserving feasibility and optimality. This contradicts the assumption that p was minimal. Hence the columns must be linearly independent. Consequently, the optimal solution is a basic feasible point. Because it is also optimal, it qualifies as a basic optimal point.

This argument confirms that if an optimal solution exists, then one can always find an optimal solution among the basic feasible points. Together with the first part, this establishes the validity of the simplex method: one needs to examine only the vertices of the feasible region, and one of them will be optimal. The final part of the theorem, concerning boundedness, follows from the fact that the simplex method terminates after finitely many steps. So, we'll return to the proof of this part after reviewing this method.

Vertices of a three-dimensional polytope

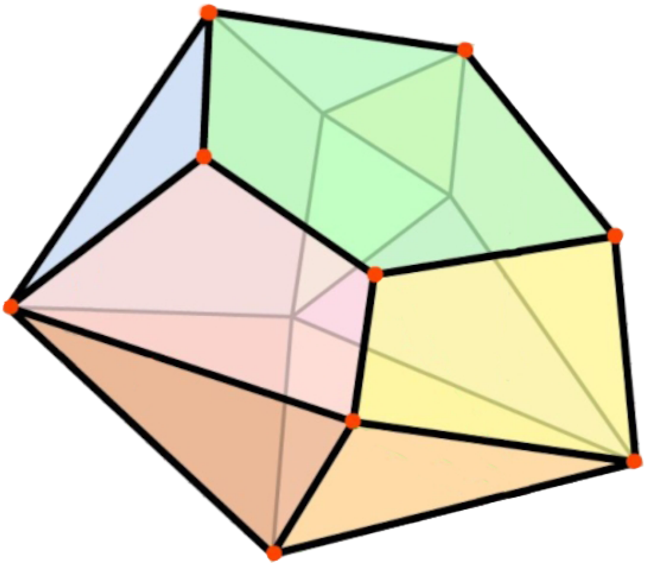


Figure: Vertices of a three-dimensional polytope

Comments

The feasible region determined by a finite system of linear equations and inequalities is always a convex polyhedron. In geometric terms, such an object is called a polytope whenever it is bounded. Each polytope is composed of faces of various dimensions: facets, edges, and vertices. The most important elements for linear programming are the vertices, since they encode the structure of possible solutions. A vertex of the feasible polytope is a point that cannot be expressed as a convex combination of two distinct feasible points; in other words, it does not lie strictly between any other feasible solutions. These special points are geometrically easy to recognize, and they play a central role in the simplex method.

From an algebraic perspective, the vertices of the feasible region coincide precisely with the basic feasible points introduced earlier. Each such point corresponds to a choice of a basis, that is, a set of linearly independent constraint columns, which uniquely determine the nonzero coordinates of the solution. This means that solving a linear program essentially reduces to moving from one vertex of the polytope to another, in search of the one that yields the smallest objective value. The entire power of the simplex algorithm relies on this fact: by focusing only on vertices, it can ignore the infinitely many points lying inside the polytope, while still guaranteeing that an optimal solution will be found among the finite set of basic feasible points. Thus, the geometry and algebra align perfectly: the corner points of the polytope are the basic feasible solutions, and they are the only candidates that need to be examined to locate an optimum.

Vertices and Basic Feasible Points

The feasible set $\{x \mid Ax = b, x \geq 0\}$ is a polytope, with vertices being points not on a straight line between two other points in the set.

- Geometrically, vertices are easily recognizable.
- Algebraically, vertices are exactly the basic feasible points.
- This links algebraic and geometric views, aiding understanding of the simplex method.

Theorem 33: Vertices and Basic Feasible Points

All basic feasible points for $\min c^T x$, subject to $Ax = b$, $x \geq 0$, are vertices of the feasible polytope $\{x \mid Ax = b, x \geq 0\}$, and vice versa.

Proof: Let x be a basic feasible point with $\mathcal{B} = \{1, 2, \dots, m\}$. Then, $B = [A_i]_{i=1,\dots,m}$ is nonsingular, and:

$$x_{m+1} = x_{m+2} = \cdots = x_n = 0.$$

If $x = \alpha y + (1-\alpha) z$ for feasible y, z and $\alpha \in (0,1)$, then $y_i = z_i = 0$ for $i = m+1, \dots, n$.

Comments

At this stage, we can formalize the deep connection between geometry and algebra in linear programming. Recall that the feasible region defined by a system of linear equations and nonnegativity constraints is a convex polyhedron, and its corner points are what we call vertices. Geometrically, a vertex is a point that cannot be represented as a convex combination of two other feasible points. In algebraic terms, however, we have already introduced the notion of a basic feasible point: a solution where at most m variables are nonzero, with m being the number of constraints, and the associated columns of matrix A forming a linearly independent set.

Theorem thirty-three tells us that these two concepts are not just related but in fact equivalent. Every basic feasible point corresponds exactly to a vertex of the feasible polyhedron, and conversely, every vertex corresponds to some basic feasible point. This equivalence closes the gap between intuition and formal proof: the geometry of polytopes aligns perfectly with the algebra of bases and feasible solutions.

This result is crucial because it guarantees that the simplex method, which moves from one basis to another, is in fact moving across the vertices of the feasible region, and thus no relevant candidates for optimality are ever missed.

The proof begins by assuming we have a basic feasible point with basis set $\mathcal{B}$. The corresponding basis matrix, B, is nonsingular, and all nonbasic variables are zero. Suppose we could write this point as a convex combination of two different feasible points. Then, because the nonbasic variables are all zero, the other points must also have zeros in these positions. Consequently, all three feasible solutions would agree on the same coordinates.

Proof: Vertices and Basic Feasible Points (Continued)

Define $x_b = (x_1, \dots, x_m)^T$, similarly y_b, z_b. Since $Ax = Ay = Az = b$:

$$Bx_b = By_b = Bz_b = b.$$

As B is nonsingular, $x_b = y_b = z_b$, so $x = y = z$, contradicting $y, z \neq x$. Thus, x is a vertex.

- Conversely, let x be a vertex with nonzeros $x_1, \dots, x_p$. If $A_1, \dots, A_p$ are linearly dependent, construct $x(\epsilon) = x + \epsilon z$ as in prior proofs.
  For small $|\epsilon|$, $x(\hat{\epsilon})$ and $x(-\hat{\epsilon})$ are feasible for some $\hat{\epsilon} > 0$. Since $x = x(0)$ lies between them, x is not a vertex, a contradiction.
- Thus, $A_1, \dots, A_p$ are linearly independent, $p \leq m$. If $p < m$, add $m - p$ indices to $\{1, \dots, p\}$ to form a basis $\mathcal{B}$, making x a basic feasible point.

□

Definition: Degeneracy

A basis $\mathcal{B}$ is *degenerate* if $x_i = 0$ for some $i \in \mathcal{B}$, where x is the basic feasible solution for $\mathcal{B}$. The linear program $\min c^T x$, subject to $Ax = b$, $x \geq 0$, is degenerate if it has at least one degenerate basis and nondegenerate, otherwise.

Comments

On the other hand since the basis matrix is invertible, this implies that the basic components also coincide, making all points identical. Therefore, the supposed convex combination reduces to trivial equality, showing that our basic feasible point is indeed a vertex.

To complete the proof of equivalence, we now consider the reverse direction: suppose we begin with a vertex of the feasible polyhedron. We ask whether this point can be described as a basic feasible solution. Assume the vertex has p nonzero coordinates, and let the corresponding columns of A be denoted A_1 through A_p. If these columns were linearly dependent, then we could construct a perturbation by adding ϵ times some vector z, as in earlier arguments. For small positive or negative values of ϵ, the perturbed solutions would remain feasible. But then our vertex would lie strictly between two distinct feasible points, which contradicts the definition of a vertex. Therefore, the associated columns must be linearly independent.

Since we have linear independence, the number p of nonzero components cannot exceed m, the number of constraints. If p equals m, then the vertex is already a basic feasible point. If p is strictly less than m, then we can extend the set of columns by adding additional independent ones, completing a full basis of size m. This construction proves that every vertex corresponds to some basic feasible point, establishing the equivalence in both directions.

This brings us naturally to the notion of degeneracy. A basis is called degenerate if the associated basic feasible solution has one or more basic variables equal to zero. In other words, although the column is part of the basis, the corresponding coordinate vanishes. A linear program is called degenerate if it admits at least one degenerate basis. Otherwise, it is nondegenerate. Degeneracy plays an important role in the analysis of the simplex method, because it can lead to situations where the algorithm cycles without making progress in the objective value. Later, we will examine strategies developed to overcome this issue, ensuring that the simplex method remains efficient and reliable.

Introduction to the Revised Simplex Method

There are a number of variants the simplex method; we start with the *revised simplex method*, a variant of the simplex method for solving $\min c^T x$, subject to $Ax = b$, $x \geq 0$ and further consider some other versions.

- All iterates are basic feasible points, i.e., vertices of the feasible polytope $\{x \mid Ax = b, x \geq 0\}$.
- Most steps move to an adjacent vertex, changing one index in the basis $\mathcal{B}$, typically reducing $c^T x$.
- Unbounded case: Move along an edge reducing $c^T x$, without reaching a vertex.
- Key task: Choose which index to remove from $\mathcal{B}$, replacing it with one from outside $\mathcal{B}$.

Define the nonbasic index set $\mathcal{N} = \{1, \ldots, n\} \backslash \mathcal{B}$, and matrices:

$$B = [A_i]_{i \in \mathcal{B}}, \quad N = [A_i]_{i \in \mathcal{N}}.$$

Partition vectors: $x_{\mathcal{B}} = [x_i]_{i \in \mathcal{B}}$, $x_{\mathcal{N}} = [x_i]_{i \in \mathcal{N}}$, similarly for $s_{\mathcal{B}}$, $s_{\mathcal{N}}$, $c_{\mathcal{B}}$, $c_{\mathcal{N}}$.
From $Ax = b$ (??):

$$Bx_{\mathcal{B}} + Nx_{\mathcal{N}} = b.$$

The primal iterate is:

$$x_{\mathcal{B}} = B^{-1}b, \quad x_{\mathcal{N}} = 0.$$

Comments

We now move to a computational perspective. The simplex method in its original form requires explicit manipulation of the full constraint matrix, which is often inefficient for large-scale problems. To address this, a refined approach known as the revised simplex method was developed. The revised version retains the essential structure of moving from one basic feasible point to another, but it does so in a more economical way by focusing on the basis matrix and related quantities, rather than handling all variables simultaneously.

At each iteration, the algorithm maintains a current basis B, and the set of basic and nonbasic variables is determined accordingly. The basic variables are obtained by solving the system $Bx_{\mathcal{B}} = b$, while all nonbasic variables are set to zero. In this way, each iterate is guaranteed to be a basic feasible solution, that is, a vertex of the feasible region. The algorithm then decides whether it can reduce the objective function by exchanging one index of the basis with one outside it. Such an exchange corresponds geometrically to moving along an edge of the polyhedron to an adjacent vertex.

In most cases, the algorithm makes progress toward improving the objective function. However, if the problem is unbounded, the algorithm detects a direction along which the objective can be decreased indefinitely, without ever reaching another vertex. This situation indicates that no finite optimal solution exists.

The central computational task of the revised simplex method is determining which variable should leave the basis and which should enter. These decisions are crucial for efficiency, and we will later explore the rules used for pivot selection.

Revised Simplex Method: KKT Conditions

For a basic feasible point in $\min c^T x$, subject to $Ax = b$, $x \geq 0$, with basis $\mathcal{B}$, we derive primal and dual variables satisfying KKT conditions.

- Since B is nonsingular and $x_{\mathcal{B}} = B^{-1}b \geq 0$, x satisfies $Ax = b$ and $x \geq 0$.
- Set $s_{\mathcal{B}} = 0$ to satisfy complementarity ($x_i s_i = 0$).

From $A^T\lambda + s = c$, partition into:

$$B^T\lambda = c_{\mathcal{B}}, \quad N^T\lambda + s_{\mathcal{N}} = c_{\mathcal{N}}.$$

Solve for λ:

$$\lambda = B^{-T}c_{\mathcal{B}}.$$

Compute $s_{\mathcal{N}}$ (reduced costs of nonbasic variables $x_{\mathcal{N}}$):

$$s_{\mathcal{N}} = c_{\mathcal{N}} - N^T\lambda = c_{\mathcal{N}} - (B^{-1}N)^T c_{\mathcal{B}}.$$

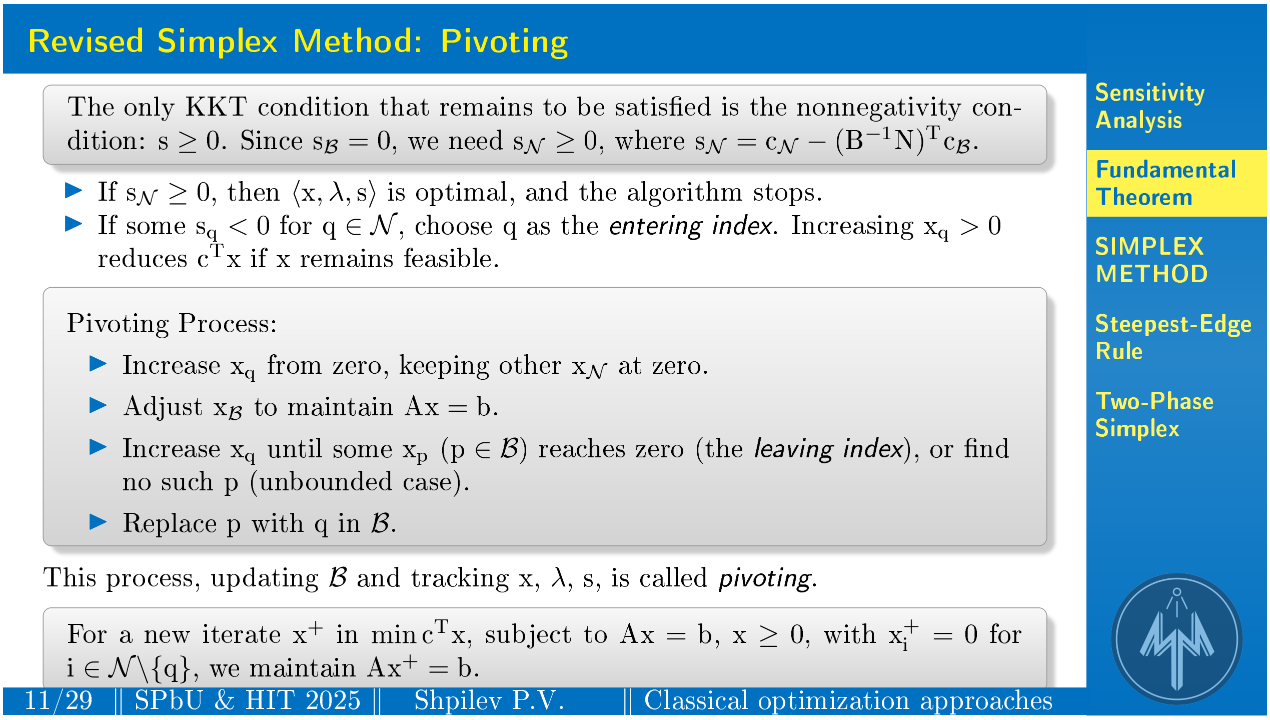
Computing $s_{\mathcal{N}}$ is called *pricing*. The components of $s_{\mathcal{N}}$ are often called the *reduced costs* of the nonbasic variables $x_{\mathcal{N}}$.

Comments

To fully justify the revised simplex method, we recall that linear programming problems can be expressed as primal–dual pairs, with optimality characterized by the Karush–Kuhn–Tucker, or KKT, conditions. For a given basis B, we can explicitly construct both the primal and dual variables satisfying these conditions.

On the primal side, since the basis matrix B is invertible, we can solve for the basic variables as $x_{\mathcal{B}} = B^{-1}b$. By definition, the nonbasic variables are zero. This ensures feasibility: the equality constraints are satisfied, and the variables remain nonnegative. Furthermore, the complementarity condition requires that each basic variable is paired with a dual slack variable equal to zero. Thus, for all indices in the basis, the slack variables vanish.

On the dual side, the relation $A^T\lambda + s = c$ can be partitioned into two systems: $B^T\lambda = c_{\mathcal{B}}$, and $N^T\lambda + s_{\mathcal{N}} = c_{\mathcal{N}}$. Solving the first system gives the dual multipliers λ as $B^{-T}c_{\mathcal{B}}$. Substituting into the second expression yields the reduced costs: $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T\lambda$. This can also be written as $c_{\mathcal{N}} - (B^{-1}N)^T c_{\mathcal{B}}$.



Revised Simplex Method: Pivoting

The only KKT condition that remains to be satisfied is the nonnegativity condition: $s \geq 0$. Since $s_{\mathcal{B}} = 0$, we need $s_{\mathcal{N}} \geq 0$, where $s_{\mathcal{N}} = c_{\mathcal{N}} - (B^{-1}N)^T c_{\mathcal{B}}$.

- If $s_{\mathcal{N}} \geq 0$, then $\langle x, \lambda, s \rangle$ is optimal, and the algorithm stops.
- If some $s_q < 0$ for $q \in \mathcal{N}$, choose q as the *entering index*. Increasing $x_q > 0$ reduces $c^T x$ if x remains feasible.

Pivoting Process:

- Increase x_q from zero, keeping other $x_{\mathcal{N}}$ at zero.
- Adjust $x_{\mathcal{B}}$ to maintain $Ax = b$.
- Increase x_q until some x_p ($p \in \mathcal{B}$) reaches zero (the *leaving index*), or find no such p (unbounded case).
- Replace p with q in $\mathcal{B}$.

This process, updating $\mathcal{B}$ and tracking x, λ, s, is called *pivoting*.

For a new iterate x^+ in $\min c^T x$, subject to $Ax = b$, $x \geq 0$, with $x_i^+ = 0$ for $i \in \mathcal{N} \setminus \{q\}$, we maintain $Ax^+ = b$.

Comments

At this stage of the algorithm, all conditions except nonnegativity have already been satisfied. The central question becomes whether the reduced costs, denoted by the vector $s_{\mathcal{N}}$, are nonnegative. If they are, we have reached optimality. If, however, some component is negative, that variable offers a direction in which the objective can be decreased. The corresponding index is chosen as the entering variable, and we attempt to increase it from zero while preserving feasibility.

Maintaining feasibility requires adjustments of the basic variables. Specifically, when we raise the entering variable, the equality constraint $Ax = b$ must still hold. This forces a compensating decrease in some basic variables. If one of these basic variables is driven to zero, it can no longer remain in the basis and must leave. This is the essence of pivoting: exchanging one variable from the basis with one from the nonbasis.

The procedure has two possible outcomes. In the generic case, some basic variable hits zero, and we update the basis accordingly, moving to a new vertex of the feasible polytope. In the exceptional case, no basic variable is forced to zero, which indicates that the objective can be improved indefinitely. This is the unbounded situation, where the problem has no finite solution.

Pivoting is not only a computational mechanism but also a geometric one: it corresponds to moving along an edge of the feasible polyhedron. Each pivot replaces one defining hyperplane of the vertex with another, ensuring that we always remain on a vertex while searching for improvement. This edge-following nature explains the efficiency of the simplex method in practice.

Revised Simplex Method: Pivoting Algebra

- Since $Ax^+ = Bx_{\mathcal{B}}^+ + A_q x_q^+ = Bx_{\mathcal{B}} = Ax$, we derive:
$$x_{\mathcal{B}}^+ = x_{\mathcal{B}} - B^{-1}A_q x_q^+. \tag{i}$$
- Geometrically, this is a move along a polytope edge, reducing $c^T x$, until a new vertex where $x_p = 0$ for some $p \in \mathcal{B}$.
- Update $\mathcal{B}$: Remove p, add q.

Effect on Objective:
$$c^T x^+ = c_{\mathcal{B}}^T x_{\mathcal{B}}^+ + c_q x_q^+ = c_{\mathcal{B}}^T x_{\mathcal{B}} - c_{\mathcal{B}}^T B^{-1} A_q x_q^+ + c_q x_q^+.$$
Since $\lambda = B^{-T} c_{\mathcal{B}}$, we have $c_{\mathcal{B}}^T B^{-1} = \lambda^T$, and $A_q^T \lambda = c_q - s_q$, so:
$$c_{\mathcal{B}}^T B^{-1} A_q x_q^+ = (c_q - s_q) x_q^+.$$
Thus:
$$c^T x^+ = c_{\mathcal{B}}^T x_{\mathcal{B}} - (c_q - s_q) x_q^+ + c_q x_q^+ = c^T x + s_q x_q^+.$$

Since $s_q < 0$, it follows that the step (i) produces a decrease in the primal objective function $c^T x$ whenever $x_q^+ > 0$.

Comments

To formalize the pivot operation, we consider the structure of the constraints. When the entering variable is increased, the basic variables must adjust to keep the equality system valid. Algebraically, this relation is expressed as $x_{\mathcal{B}}^+ = x_{\mathcal{B}} - B^{-1}A_q x_q^+$.

This equation captures the trade-off: raising the entering variable reduces some of the basic ones proportionally. The vector $B^{-1}A_q$ specifies exactly how the adjustment is distributed among the basic components.

Geometrically, the formula describes motion along an edge of the feasible polytope. The path is linear until a boundary is hit, meaning one of the basic variables becomes zero. At that point, the vertex changes, and the new basis reflects the exchange between leaving and entering variables.

The impact on the objective can also be made precise. Substituting into the cost function gives $c^T x^+ = c^T x + s_q x_q^+$.

Here, s_q is the reduced cost associated with the entering variable. If this reduced cost is negative, then increasing the corresponding variable decreases the objective function. The magnitude of improvement is proportional to the step length. This shows clearly why negative reduced costs signal opportunities for descent.

Thus, the algebra provides both a computational rule and an intuitive explanation: pivoting moves us to another vertex while ensuring that each step strictly reduces the objective, provided the reduced cost is negative. This mechanism guarantees systematic progress toward optimality or unboundedness, leaving no ambiguity in the direction of search.

Revised Simplex Method: Termination

If $x_{\mathcal{B}}^{+} = x_{\mathcal{B}} - B^{-1}A_q x_q^{+} \geq 0$ for all $x_q^{+} > 0$, the linear program $\min c^T x$, subject to $Ax = b$, $x \geq 0$, is unbounded. The simplex method identifies a ray in the feasible polytope where $c^T x \to -\infty$.

- If the basis $\mathcal{B}$ is nondegenerate (i.e., $x_i > 0$ for all $i \in \mathcal{B}$), then $x_q^{+} > 0$, ensuring a strict decrease in $c^T x$.
- For a nondegenerate problem, every step reduces $c^T x$.

Theorem 34: Termination of the simplex method

If the linear program $\min c^T x$, subject to $Ax = b$, $x \geq 0$, is nondegenerate and bounded, the simplex method terminates at a basic optimal point.

Proof:

- Each step strictly decreases $c^T x$, so the same basic feasible point is never revisited.
- There are finitely many bases $\mathcal{B}$ (subsets of m indices from $\{1, \dots, n\}$), hence finitely many basic feasible points.
- Since the problem is bounded, the method terminates at a basic optimal point. □

Comments

The simplex method continues this process of exchanging variables until one of two situations arises. The first possibility is that no further negative reduced costs exist, which certifies optimality. The second is that the problem is unbounded. This occurs when, after selecting an entering variable, the corresponding direction allows all basic variables to remain nonnegative, even as the step length grows indefinitely. In such a case, the objective decreases without bound, and no finite solution exists.

For bounded problems, progress is guaranteed under the assumption of nondegeneracy. Nondegeneracy means that every basic variable is strictly positive, so each pivot produces an actual change in the objective value. Because the cost decreases strictly at every step, the method can never revisit the same basic feasible point.

The finite termination argument rests on combinatorial grounds. There are only finitely many possible bases, since a basis is determined by selecting m indices out of n. Each corresponds to a potential basic feasible solution. With every pivot leading to a new and distinct solution, the method cannot continue indefinitely without reaching either an optimal vertex or detecting unboundedness.

Thus, under the assumptions of boundedness and nondegeneracy, the simplex method is guaranteed to terminate with an optimal solution. The result is significant: despite the fact that the algorithm moves locally from vertex to vertex, it cannot cycle or diverge. The combination of finite basis count and strictly improving steps ensures convergence in a finite number of iterations.

This result gives us a proof of part (iii) of Theorem 32 in the case in which the linear program is nondegenerate. The proof of finite termination is considerably more complex when nondegeneracy of the primal problem is not assumed, as we discuss a little bit further.

Simplex iterates for a two-dimensional problem

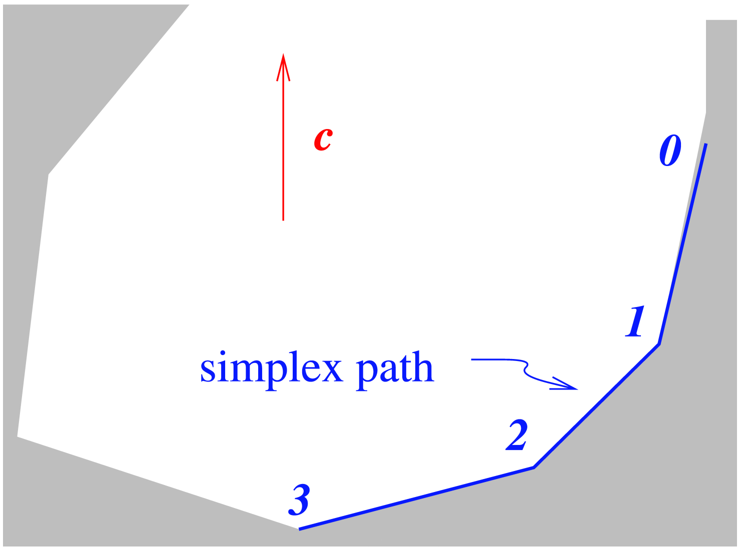


Figure: Simplex iterates for a two-dimensional problem.

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

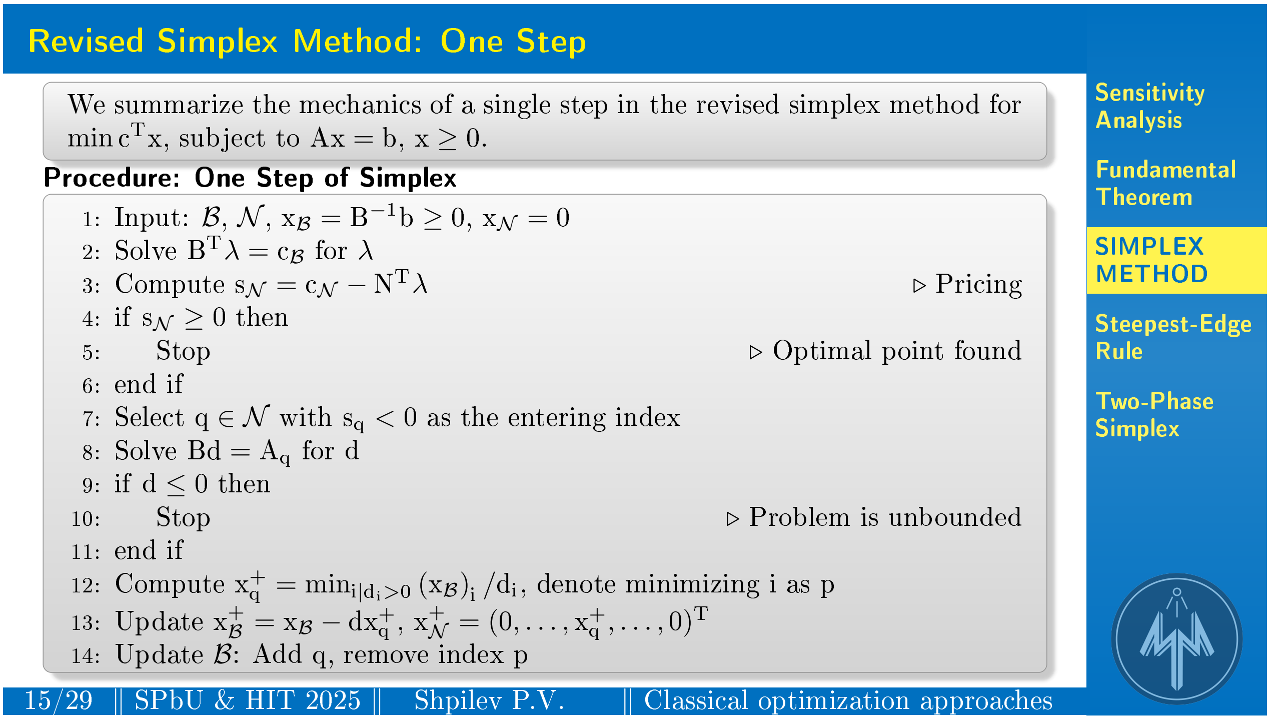
Comments

To gain intuition, it is helpful to visualize the method in two dimensions. In this case, the feasible region is a polygon defined by the intersection of linear inequalities. Each vertex corresponds to a basic feasible solution. The simplex method traces a path along the edges of this polygon, moving systematically from one vertex to another in search of improvement.

The entering and leaving variable rules translate geometrically into edge-following rules. When a negative reduced cost is detected, the method chooses an edge of the polygon that leads downhill in terms of the objective function. Movement continues along this edge until reaching the next vertex, which represents the exchange of one constraint for another. The new vertex then serves as the starting point for the next iteration.

This visualization explains why the simplex method often performs so efficiently in practice. Even though the number of potential vertices can be very large in higher dimensions, the method typically explores only a small fraction. By always following an improving edge, it avoids wandering through the interior and instead exploits the polyhedral structure directly.

In two dimensions, the process is particularly transparent: we can literally see the objective function decreasing as the algorithm marches along polygon edges. Each pivot corresponds to sliding down to a lower-cost vertex, step by step, until the minimum is reached or unboundedness is revealed. This geometric picture builds intuition that extends to higher dimensions, even though direct visualization is no longer possible.



Revised Simplex Method: One Step

We summarize the mechanics of a single step in the revised simplex method for $\min c^T x$, subject to $Ax = b$, $x \geq 0$.

Procedure: One Step of Simplex

```
Input: $\mathcal{B}$, $\mathcal{N}$, $x_{\mathcal{B}} = B^{-1}b \geq 0$, $x_{\mathcal{N}} = 0$
Solve $B^T\lambda = c_{\mathcal{B}}$ for $\lambda$
Compute $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T\lambda$                 ▷ Pricing
if $s_{\mathcal{N}} \geq 0$ then
    Stop                                                  ▷ Optimal point found
end if
Select $q \in \mathcal{N}$ with $s_q < 0$ as the entering index
Solve $Bd = A_q$ for d
if $d \leq 0$ then
    Stop                                                  ▷ Problem is unbounded
end if
Compute $x_q^+ = \min_{i|d_i>0} (x_{\mathcal{B}})_i / d_i$, denote minimizing i as p
Update $x_{\mathcal{B}}^+ = x_{\mathcal{B}} - dx_q^+$, $x_{\mathcal{N}}^+ = (0, \ldots, x_q^+, \ldots, 0)^T$
Update $\mathcal{B}$: Add q, remove index p
```

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

Comments

The revised simplex method executes each iteration through a precise sequence of linear algebraic operations. We begin with a chosen basis, denoted by $\mathcal{B}$, and its complement, the nonbasis $\mathcal{N}$. The basic variables are obtained as $B^{-1}b$, ensuring feasibility, while all nonbasic variables remain equal to zero. From this point, dual multipliers are recovered by solving the system $B^T\lambda = c_{\mathcal{B}}$, where $c_{\mathcal{B}}$ denotes the cost coefficients corresponding to the basis. Once λ is available, we compute the reduced costs for the nonbasic variables as $c_{\mathcal{N}} - N^T\lambda$.

If every reduced cost is nonnegative, then no improvement is possible, and we have reached an optimal solution. Otherwise, some component is strictly negative, signaling a direction of descent. The corresponding index, denoted q, is chosen as the entering variable. To maintain feasibility when x_q increases, we must determine how the basic variables adjust. This requires solving $Bd = A_q$, where A_q is the q-th column of the matrix A. If the direction vector d has only nonpositive entries, then no basic variable is ever forced to zero, and the problem is unbounded. Otherwise, the ratio test identifies the maximum permissible step length: we take the minimum of the quotients between the current basic values and the positive entries of d. The minimizing index indicates the leaving variable, denoted p.

With these updates, we modify both the basis and the solution vector, ensuring feasibility and reducing the objective. This compact procedure guarantees that each step either moves us closer to optimality or certifies unboundedness, fully capturing the mechanics of a simplex pivot in algebraic form.

Revised Simplex Method: Example

Example

Consider: $\min -4x_1 - 2x_2$ s.t. $x_1 + x_2 + x_3 = 5$, $2x_1 + \frac{1}{2}x_2 + x_4 = 8$, $x \geq 0$, or in matrix form: $Ax = b$, with $A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & \frac{1}{2} & 0 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$.

- Basis $\mathcal{B} = \{3, 4\}$, so $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ (columns of A for x_3, x_4). Set $x_{\mathcal{B}} = B^{-1}b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$, $x_{\mathcal{N}} = (x_1, x_2) = (0, 0)$, $c^T x = 0$.
- Solve $B^T \lambda = c_{\mathcal{B}} = (0, 0)^T$: $\lambda = (0, 0)^T$. Compute $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$, where $c_{\mathcal{N}} = (-4, -2)^T$, $N = \begin{bmatrix} 1 & 1 \\ 2 & \frac{1}{2} \end{bmatrix}$: $s_{\mathcal{N}} = (-4, -2)^T - \begin{bmatrix} 1 & 2 \\ 1 & \frac{1}{2} \end{bmatrix} (0, 0)^T = (-4, -2)^T$.
- Since $s_1, s_2 < 0$, choose $q = 1$. Solve $Bd = A_1 = (1, 2)^T$: $d = (1, 2)^T$. As $d \not\leq 0$, not unbounded.
- Ratio test: $\min \left\{ \frac{(x_{\mathcal{B}})_i}{d_i} \mid d_i > 0 \right\} = \min \left\{ \frac{5}{1}, \frac{8}{2} \right\} = 4$, so $p = 2$ (corresponding to the index 4), $x_1^+ = 4$. Update $\mathcal{B} = \{3, 1\}$, $\mathcal{N} = \{4, 2\}$.

Comments

To deepen our understanding, let us analyze a concrete numerical example. Consider the objective function to minimize $-4x_1 - 2x_2$, subject to the constraints: $x_1 + x_2 + x_3 = 5$, and $2x_1 + \frac{1}{2}x_2 + x_4 = 8$, with all variables nonnegative. In matrix notation, the constraint matrix A is two by four, with rows $\begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 2 & \frac{1}{2} & 0 & 1 \end{bmatrix}$, while the right-hand side vector b is $\begin{bmatrix} 5 \\ 8 \end{bmatrix}$.

We initially select the basis indices three and four, corresponding to the slack variables. The basis matrix in this case is the identity matrix, so the basic variables equal five and eight, respectively, while the nonbasic decision variables x_1 and x_2 are zero. The initial objective value is zero. Next, we solve $B^T \lambda = 0$, which gives λ equal to the zero vector. Then we compute the reduced costs: $c_{\mathcal{N}}$ is the vector $(-4, -2)$. Subtracting $N^T \lambda$, which remains zero, we obtain reduced costs of -4 and -2. Both are negative, so improvement is possible.

Following the rule, we select x_1 to enter the basis. We then solve $Bd =$ column one of A, which is the vector $(1, 2)$. The result is $d = (1, 2)$. Because some components of d are positive, the problem is not unbounded. The ratio test is then applied: we take the minimum between $5/1$ and $8/2$, which are five and four. The minimum is four, corresponding to variable x_4, which therefore leaves the basis. The updated basis now contains indices three and one.

Revised Simplex Method: Example (Continued)

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

Continued

Continuing: with $A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & \frac{1}{2} & 0 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$, $c = \begin{bmatrix} -4 & -2 & 0 & 0 \end{bmatrix}^T$.

- Iteration 2: $\mathcal{B} = \{3, 1\}$, $B = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$, $x_{\mathcal{B}} = \begin{bmatrix} x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & -1/2 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$, $c^T x = -16$.
  Solve $\lambda = (B^T)^{-1} c_{\mathcal{B}} = \begin{bmatrix} 1 & 0 \\ -1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \end{bmatrix} = (0, -2)^T$.
  $\mathcal{N} = \{4, 2\}$, $c_{\mathcal{N}} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}$, $N = \begin{bmatrix} 0 & 1 \\ 1 & \frac{1}{2} \end{bmatrix}$. Compute: $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$.
  Choose $q = 2$ ($s_2 < 0$), solve: $Bd = A_2 \Rightarrow d = B^{-1} A_2 = \left(\frac{3}{4}, \frac{1}{4}\right)^T$.
  Ratio test: $\min\left\{\frac{4}{3}, 16\right\} = \frac{4}{3}$, $p = 1$ (corresponding to the index 3), $x_2^+ = \frac{4}{3}$.
  Update: $\mathcal{B} = \{2, 1\}$, $\mathcal{N} = \{4, 3\}$.
- Iteration 3: $\mathcal{B} = \{2, 1\}$, $B = \begin{bmatrix} 1 & 1 \\ \frac{1}{2} & 2 \end{bmatrix}$, $x_{\mathcal{B}} = \begin{bmatrix} \frac{4}{3} \\ \frac{11}{3} \end{bmatrix}$, $c^T x = -\frac{52}{3}$, $s_{\mathcal{N}} = \left(\frac{4}{3}, \frac{4}{3}\right)^T$.
  Since $s_{\mathcal{N}} \geq 0$, stop: optimal.

Comments

Continuing with the example, we now examine subsequent iterations. With the updated basis containing indices three and one, the corresponding basis matrix is two by two with columns $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Its inverse, when multiplied by the right-hand side vector, yields new basic values: $x_3 = 1$, and $x_1 = 4$. The objective function evaluates to -16. Next, solving $B^T \lambda = c_{\mathcal{B}}$ which is $(0, -4)^T$ produces $\lambda = (0, -2)^T$.

The reduced costs are then computed. For the nonbasic variables, namely x_4 and x_2, we have $c_{\mathcal{N}} = (0, -2)$. Subtracting $N^T \lambda$, we obtain reduced costs of 2 and -1. Since x_2 has a negative reduced cost, it becomes the entering variable. To preserve feasibility, we solve $Bd =$ column two of A, which yields $d = (3/4, 1/4)$. The ratio test is applied: the minimum of $1/(3/4)$, and $4/(1/4)$, gives $4/3$. Therefore, x_3 is the leaving variable, and x_2 enters with value $4/3$.

The next basis contains indices two and one. Solving again, we obtain basic values $x_2 = 4/3$ and $x_1 = 11/3$. The objective function equals $-52/3$. At this stage, the reduced costs for the nonbasic variables are both positive $4/3$. Since no negative reduced costs remain, optimality is reached.

This sequence of calculations demonstrates the complete cycle of the revised simplex method: from an initial feasible vertex, through systematic pivots, to an optimal solution. Each iteration carefully balances feasibility and optimality conditions, guiding the process to termination.

Revised Simplex Method: Implementation Aspects

Enhancing the simplex method for $\min c^T x$, s.t. $Ax = b$, $x \geq 0$ requires addressing:

- Linear algebra: Maintain an LU factorization of basis matrix B to solve for λ and d.
- Entering index: Select q from negative components of $s_{\mathcal{N}}$.
- Degeneracy: Handle degenerate bases and steps where $x_q^+ \leq 0$ violates feasibility.

Linear Algebra: Solve $B^T\lambda = c_{\mathcal{B}}$, $Bd = A_q$ using LU factorization $LU = B$ (L unit lower triangular, U upper triangular).

- Avoid computing B^{-1} explicitly; use triangular substitutions.
- Update factorization per iteration (single column change in B).
- For sparse B, reorder rows/columns for stability and sparsity (e.g., Markowitz pivot strategy).
- Solve $Bd = A_q$ via: $Lz = A_q$, $Ud = z$.

Comments

Beyond the mechanics of individual pivots, efficient implementation of the revised simplex method requires attention to numerical linear algebra. Each iteration involves solving two types of systems: $B^T\lambda = c_{\mathcal{B}}$, and $Bd = A_q$. Although one might be tempted to compute B^{-1} explicitly, this is computationally wasteful and numerically unstable. Instead, we maintain a factorization of B, typically in the form of an LU decomposition, where L is unit lower triangular and U is upper triangular. This structure allows forward and backward substitutions, yielding solutions at far lower cost than recomputing inverses.

Because each pivot modifies only one column of the basis matrix, the factorization can be updated efficiently without starting from scratch. Specialized update formulas ensure that computational effort remains proportional to the matrix dimension, rather than its square or cube. Moreover, in large-scale problems, the basis matrix is often sparse. Preserving sparsity is crucial, since fill-in during factorization can destroy efficiency. Strategies such as the Markowitz pivot rule reorder rows and columns to minimize fill-in while maintaining numerical stability.

In practice, this careful treatment of linear algebra makes the revised simplex method highly scalable. The ability to exploit sparsity and incremental updates is what enables modern implementations to handle problems with tens or even hundreds of thousands of constraints and variables. Without these optimizations, the theoretical mechanics of the method would remain impractical at large scales. Thus, efficient factorization, stability control, and update strategies are as essential to the success of the simplex method as the underlying mathematics of pivoting.

Revised Simplex Method: Linear Algebra (Continued)

Solving $\mathrm{B}^{\mathrm{T}}\lambda = \mathrm{c}_{\mathcal{B}}$: Use LU factorization $\mathrm{LU} = \mathrm{B}$ via:

$$\mathrm{U}^{\mathrm{T}}\tilde{\lambda} = \mathrm{c}_{\mathcal{B}}, \quad \mathrm{L}^{\mathrm{T}}\lambda = \tilde{\lambda}.$$

- Updating LU Factors: After replacing index p with q in basis $\mathcal{B}$ after one step of the simplex method, update B to B^{+} (replace column B_{p} with A_{q}).
- $\mathrm{L}^{-1}\mathrm{B}^{+}$ is upper triangular except in column p.
- Apply cyclic permutation $\mathrm{P}_{\downarrow}$: Move column p to last position m, shift columns $\mathrm{p}+1,\ldots,\mathrm{m}$ left, and permute rows p to m. Result: $\mathrm{P}_{\downarrow}\mathrm{L}^{-1}\mathrm{B}^{+}\mathrm{P}_{\downarrow}^{\mathrm{T}}$ shifts non-triangular part to last row.
- Perform sparse Gaussian elimination on $\mathrm{P}_{\downarrow}\mathrm{L}^{-1}\mathrm{B}^{+}\mathrm{P}_{\downarrow}^{\mathrm{T}} = \mathrm{L}_{\downarrow}\mathrm{U}_{\downarrow}$:
  - $\mathrm{L}_{\downarrow}$: Identity except last row.
  - $\mathrm{U}_{\downarrow}$: Matches $\mathrm{P}_{\downarrow}\mathrm{L}^{-1}\mathrm{B}^{+}\mathrm{P}_{\downarrow}^{\mathrm{T}}$ except modified (m,m) element and zeroed off-diagonal last row elements.

Comments

When solving the dual system given by $\mathrm{B}^{\mathrm{T}}\lambda = \mathrm{c}_{\mathcal{B}}$, the LU factorization of the basis matrix provides a computationally efficient route. Recall that LU factorization decomposes the basis matrix B into the product of a lower triangular matrix L and an upper triangular matrix U. To solve for λ, we instead consider the system $\mathrm{U}^{\mathrm{T}}\tilde{\lambda} = \mathrm{c}_{\mathcal{B}}$, followed by $\mathrm{L}^{\mathrm{T}}\lambda = \tilde{\lambda}$. This two-step triangular system avoids directly computing the inverse of B, which would be numerically unstable and computationally expensive.

After a pivot step, the basis matrix changes: one column, say column p, is replaced with the entering column A_{q}. This produces the new basis matrix B^{+}. Rather than recomputing the full LU factorization from scratch, which would cost cubic time in the dimension m, we instead update the factorization incrementally. It can be noted that the matrix $\mathrm{L}^{-1}\mathrm{B}^{+}$ is nearly upper triangular: all columns except column p retain their triangular structure. Thus, the only disturbance is localized.

To systematically restore triangular form, we apply a cyclic permutation. This permutation, denoted $\mathrm{P}_{\downarrow}$, moves column p to the last column and permutes rows accordingly. After this rearrangement, the non-triangular structure is confined to the last row, making the system well suited for a sparse Gaussian elimination step. Performing elimination yields new triangular factors $\mathrm{L}_{\downarrow}$ and $\mathrm{U}_{\downarrow}$, which differ only minimally from the original ones. In this way, the revised simplex method maintains an efficient updating mechanism without the prohibitive cost of refactorization.

Cyclic row and column permutation

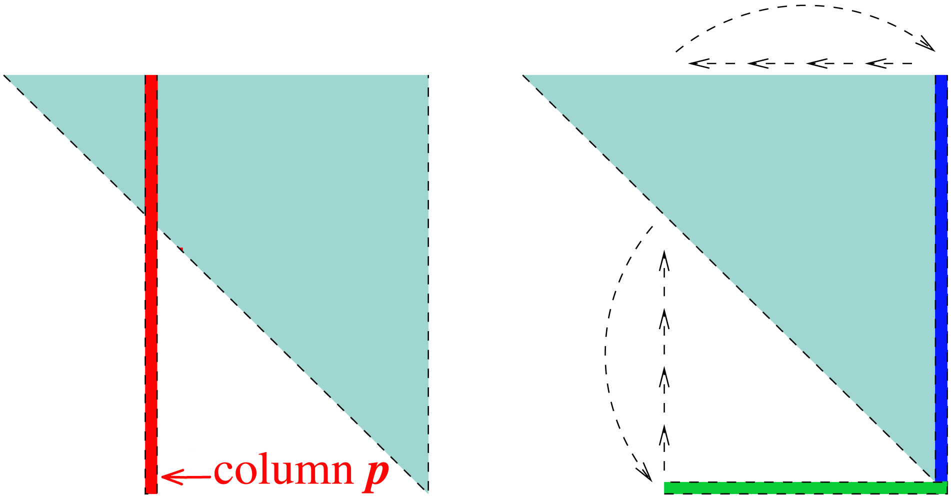


Figure: Left: $L^{-1}B^{+}$, which is upper triangular except for the column occupied by A_p. Right: After cyclic row and column permutation $P_{\downarrow}$, the non-upper triangular part of $P_{\downarrow}L^{-1}B^{+}P_{\downarrow}^{T}$ appears in the last row.

Comments

The role of cyclic permutation in the updating procedure is subtle but crucial. When the entering column replaces column p of the basis, the structure $L^{-1}B^{+}$ remains upper triangular except at that single column. If we were to attempt Gaussian elimination immediately, the disturbance would spread through the matrix, destroying sparsity and efficiency. Instead, we strategically reorder rows and columns so that the problematic column is shifted to the last position.

The permutation operator $P_{\downarrow}$ performs this rearrangement. Specifically, column p is moved to column m, while columns from $p+1$ through m are shifted one position to the left. At the same time, rows p through m are cyclically permuted, preserving consistency. The effect of this operation is that the non-triangular part, originally scattered, becomes localized entirely in the final row. This containment is what enables efficient elimination.

After applying $P_{\downarrow}$, the new matrix $P_{\downarrow}L^{-1}B^{+}P_{\downarrow}^{T}$ can be expressed as a nearly upper triangular form. Only the last row contains off-diagonal elements. Sparse Gaussian elimination then operates exclusively on this row, transforming it into a standard triangular shape. The resulting matrices, denoted $L_{\downarrow}$ and $U_{\downarrow}$, preserve triangularity while modifying only the necessary entries. Importantly, $L_{\downarrow}$ is equal to the identity matrix except in the last row, while $U_{\downarrow}$ matches the permuted structure except with a modified bottom-right element. This process highlights a fundamental principle of the revised simplex method: each update is localized, efficient, and exploits matrix structure to minimize computational effort.

Revised Simplex Method: LU Update Case (1)

Updating LU factorization for $m = 5$, $p = 2$ (replace column 2 with $L^{-1}A_q$).

- Given: $L^{-1}B = U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ & u_{22} & u_{23} & u_{24} & u_{25} \\ & & u_{33} & u_{34} & u_{35} \\ & & & u_{44} & u_{45} \\ & & & & u_{55} \end{bmatrix}$, $L^{-1}A_q = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix}$.
- Form: $L^{-1}B^+ = \begin{bmatrix} u_{11} & w_1 & u_{13} & u_{14} & u_{15} \\ & w_2 & u_{23} & u_{24} & u_{25} \\ & w_3 & u_{33} & u_{34} & u_{35} \\ & w_4 & & u_{44} & u_{45} \\ & w_5 & & & u_{55} \end{bmatrix}$.
- Apply permutation $P_\downarrow$: Move column 2 to position 5, shift columns 3–5 left, permute rows 2–5. Result: $P_\downarrow L^{-1}B^+P_\downarrow^T = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ & u_{23} & u_{24} & u_{25} & w_2 \end{bmatrix}$.

Comments

To illustrate the mechanics of the LU update, let us consider a concrete case where the dimension m equals five and the replaced column index p equals two. Initially, we have the factorization $L^{-1}B = U$, where U is a five-by-five upper triangular matrix with diagonal entries u_{11} through u_{55}. The entering column A_q, after being premultiplied by L^{-1}, yields the vector w with components w_1 through w_5.

Forming the updated product $L^{-1}B^+$ amounts to substituting the second column of U with this vector w. As a result, the new matrix preserves triangular structure everywhere except in column two, which now contains the entries w_1 through w_5. This breaks the upper triangularity because entries below the diagonal in that column are generally nonzero.

To prepare for elimination, we apply the cyclic permutation operator $P_\downarrow$. In this case, column two is moved to position five, and columns three through five shift one place to the left. Simultaneously, rows two through five undergo the same cyclic shift. The transformed matrix $P_\downarrow L^{-1}B^+P_\downarrow^T$ now exhibits the desired structure: the disturbance is entirely relocated to the last row. The upper left block, consisting of the first four rows and columns, remains perfectly triangular. This sets the stage for applying sparse elimination to restore full triangularity with minimal work, highlighting how theory translates into efficient computational practice.

Revised Simplex Method: LU Update Case (2)

Continuing LU factorization update for $m = 5$, $p = 2$.

- Gaussian elimination on $P_\downarrow L^{-1} B^+ P_\downarrow^T$ gives: $L_1 = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 0 & l_{52} & l_{53} & l_{54} & 1 \end{bmatrix}$,

  $U_1 = \begin{bmatrix} u_{11} & u_{13} & u_{14} & u_{15} & w_1 \\ & u_{33} & u_{34} & u_{35} & w_3 \\ & & u_{44} & u_{45} & w_4 \\ & & & u_{55} & w_5 \\ & & & & \hat{w}_2 \end{bmatrix}$, for certain values of l_{52}, l_{53}, l_{54} and $\hat{w}_2$.
- L_1: Identity except last row; U_1: Matches $P_\downarrow L^{-1} B^+ P_\downarrow^T$ except modified $(5, 5)$ element and zeroed off-diagonal last row elements.
- Result: $B^+ = L^+ U^+$, where $L^+ = L P_\downarrow^T L_1$, $U^+ = U_1 P_\downarrow$.

Comments

Continuing with the same example, we now perform Gaussian elimination on the permuted matrix. The result is a factorization into L_1 and U_1. The matrix L_1 is identical to the identity except for its last row, where entries l_{52}, l_{53}, and l_{54} appear. These coefficients record the multipliers used in the elimination process. The matrix U_1 retains the triangular form: its upper left block matches the permuted structure, while the last row has been cleared of off-diagonal elements, leaving only the modified entry $\hat{w}_2$ in the bottom-right corner.

This operation produces a valid LU decomposition of the updated basis matrix B^+. Explicitly, we can write $B^+ = L^+ U^+$, where $L^+ = L P_\downarrow^T L_1$, and $U^+ = U_1 P_\downarrow$. Thus, both factors are obtained without recomputing the entire decomposition from scratch. The efficiency gain is significant: each pivot requires only localized row operations, rather than a global recomputation.

From a computational perspective, this method ensures that the revised simplex algorithm scales gracefully to large problems. Because only a single column update is needed at each iteration, the work per pivot remains manageable. At the same time, numerical stability is maintained by careful use of triangular structure. This combination of localized updates, sparsity preservation, and stability makes the LU updating procedure a cornerstone of modern simplex implementations.

Revised Simplex Method: Entering Index Selection

Selecting entering index q from negative $s_{\mathcal{N}}$ components in simplex method for $\min c^T x$, s.t. $Ax = b$, $x \geq 0$.

- **Goal**: Choose q to reduce $c^T x$, ideally minimizing steps to optimal x^*. Practical strategies focus on current iteration.
- **Dantzig's Rule**: Select q with most negative s_q in $s_{\mathcal{N}} = c_{\mathcal{N}} - N^T \lambda$. Maximizes $c^T x$ decrease per unit x_q, but x_q^+ may be small.
- **Partial Pricing**: Compute subvector of $s_{\mathcal{N}}$ (less costly than full N^T multiplication). Choose q from negative entries, cycling nonbasic indices.
- **Multiple Pricing**: For a subset of indices, compute s_q, $d = B^{-1} A_q$, max x_q^+ (feasible), and $s_q x_q^+$. Reuse subset until $s_q \geq 0$, then update $s_{\mathcal{N}}$. Reduces memory access.
- **Heuristics**: Combine partial and multiple pricing for efficiency.

Comments

When applying the revised simplex method, the choice of the entering index plays a central role. Recall that our aim is to reduce the objective value, namely the inner product of c and x, while maintaining feasibility. Among all nonbasic variables, we look at their reduced costs. Only those with negative reduced costs are eligible to enter, since moving them from zero upward can potentially reduce the objective. The most straightforward approach is Dantzig's rule: simply pick the index with the most negative reduced cost. This maximizes the decrease in the objective per unit increase of that particular nonbasic variable. However, the step length for that variable may be very small, which limits the actual improvement achieved in one iteration.

Because evaluating all reduced costs can be computationally demanding, especially for large-scale problems, several variants were proposed. Partial pricing considers only a subset of the nonbasic variables at each iteration, rotating through different subsets over time. This lowers the cost of the reduced-cost computation. Multiple pricing takes a different approach: it evaluates reduced costs and candidate directions for a chosen set of variables, and reuses that information for multiple iterations until it becomes outdated. This reduces memory traffic, which is often a bottleneck on modern machines.

Finally, heuristics often combine partial and multiple pricing. They do not guarantee the mathematically best choice at each step, but they balance progress toward optimality with computational efficiency. The critical message is that, in practice, rules for entering variables are shaped not only by theoretical optimality but also by considerations of speed and scalability.

Revised Simplex Method: Steepest-Edge Rule

Selecting entering index q using steepest-edge rule for $\min c^T x$, s.t. $Ax = b$, $x \geq 0$.

- **Steepest-Edge Rule**: Chooses $q \in \mathcal{N}$ for largest decrease in $c^T x$ per unit distance along edge (most downhill), unlike Dantzig's rule, which maximizes decrease per unit x_q^+. A small x_q^+ change can mean a large edge distance.
- **Pivot Step**: Change in x is

$$x^+ = \begin{bmatrix} x_{\mathcal{B}}^+ \\ x_{\mathcal{N}}^+ \end{bmatrix} = \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{bmatrix} + \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} x_q^+ = x + \eta_q x_q^+, \text{ where}$$

$$\eta_q = \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix} = \begin{bmatrix} -d \\ e_q \end{bmatrix}, \ e_q \text{ is unit vector with 1 at q.}$$

- **Decrease Measure**: Change in $c^T x$ per unit step along η_q is $\frac{c^T \eta_q}{\|\eta_q\|}$. Choose q to minimize this.
- **Efficient Update**: Computing η_i for all $i \in \mathcal{N}$ (via $Bd = A_i$) is costly. Goldfarb and Reid's method updates $c^T \eta_i$ and $\|\eta_i\|$ economically each iteration.

Comments

Dantzig's rule focuses on the decrease per unit of the entering variable, but that can be misleading. Imagine a situation where the variable's allowable increase is tiny, but the reduced cost is very negative. The predicted gain is large per unit of that variable, yet the total step is negligible. The steepest-edge rule addresses this by measuring progress differently. Instead of considering a unit increase of the variable, it evaluates the decrease in the objective per unit length of movement along the actual edge of the feasible polytope. In other words, it asks: if we step in this direction, how steep is the descent of the objective function?

Formally, the update vector is written as $\eta_q = \begin{bmatrix} -B^{-1}A_q \\ e_q \end{bmatrix}$. The change in the basic variables is $-B^{-1}A_q$, while the entering variable increases in the unit direction. Thus, the entire step is represented as $x^+ = x + \eta_q x_q^+$. The rate of decrease per unit distance in this direction is given by the ratio $(c^T \eta_q)/\|\eta_q\|$ Choosing the entering index now means selecting the variable q that minimizes this ratio.

The challenge is computational. For each candidate, one needs both the numerator and the norm of η_q. Computing these from scratch is costly. This is where the work of Goldfarb and Reid becomes essential: they developed efficient recursive formulas to update the required quantities after each iteration. Thanks to this, the steepest-edge rule became practical and has shown strong empirical performance, often outperforming Dantzig's rule on difficult problems.

Revised Simplex Method: Steepest-Edge Update

Updating $c^T\eta_i$ and $\|\eta_i\|$ for steepest-edge rule, where $\eta_i = \begin{bmatrix} -B^{-1}A_i \\ e_i \end{bmatrix}$ picks direction for index i.

- **Numerator**: $c^T\eta_i = s_i$ (from $c^Tx^+ = c^Tx + s_q x_q^+$ (was shown earlier) and $c^Tx^+ = c^Tx + c^T\eta_q x_q^+$). No need to compute η_i; s_i measures objective change rate.
- **Denominator** (we define $\gamma_i = \|\eta_i\|^2$):
  $$\gamma_i = \|\eta_i\|^2 = \|B^{-1}A_i\|^2 + 1 \text{ (current step)};$$
  $$\gamma_i^+ = \|\eta_i^+\|^2 = \|(B^+)^{-1}A_i\|^2 + 1 \text{ (next step)}.$$
  Norms adjust steepness.
- **Basis Update**: Assume, e.g., that we replace column A_t (corresponds to colomn p in B) with A_q:
  $B^+ = B + (A_q - Be_p)e_p^T$, $e_p = (0, \dots, 0, p, 0, \dots, 0)^T$. Updates basis matrix.
- **Applying Sherman-Morrison formula obtain**: $(B^+)^{-1} = B^{-1} - \frac{(d-e_p)e_p^T B^{-1}}{e_p^T d}$, where $d = B^{-1}A_q$.
  Then, $(B^+)^{-1}A_i = B^{-1}A_i - \dfrac{e_p^T B^{-1}A_i}{e_p^T d}(d - e_p)$. Adjusts inverse efficiently.

Comments

To apply the steepest-edge rule efficiently, we need a systematic way to maintain the quantities that determine the choice of entering index. Recall that the numerator in our ratio is simply the reduced cost s_i. This is convenient: it means we do not need to explicitly compute η_i. The reduced cost already encodes the rate of change of the objective in the direction associated with variable i.

The denominator is more challenging. We define γ_i as the squared norm of the step direction, i.e. as the squared norm of η_i. For the current basis, this is the squared norm of $B^{-1}A_i$ plus one. After a pivot, the basis changes, and thus γ_i must be updated. If we had to recompute these norms from scratch every iteration, the computational cost would be prohibitive.

Here enters the Sherman–Morrison formula. When the basis matrix is updated by replacing one column with a new one, the inverse can be updated cheaply without recomputation. Specifically, if B^+ is the new basis matrix, then its inverse is given by B^{-1} minus (the quantity $d - e_p$) times $e_p^T B^{-1}$, all over $e_p^T d$, where $d = B^{-1}A_q$. This allows us to update $(B^+)^{-1}A_i$ efficiently. Once we have this, we can update the squared norms γ_i^+.

This mechanism is what makes the steepest-edge method competitive. Instead of recalculating large matrix inverses, we adjust only the necessary vectors. In practice, this reduces the computational load drastically, while keeping the selection criterion accurate.

Revised Simplex Method: Steepest-Edge Update (Cont)

Continuing update of $\gamma_i = \|\eta_i\|^2$ for steepest-edge rule, where η_i defines direction for index $i \in \mathcal{N}$.

- **Update Formula**: Substitute $(B^+)^{-1}A_i$ into $\gamma_i^+ = \|(B^+)^{-1}A_i\|^2 + 1$ to get:

$$\gamma_i^+ = \gamma_i - 2\left(\frac{e_p^T B^{-1} A_i}{e_p^T d}\right) A_i^T (B^{-1})^T d + \left(\frac{e_p^T B^{-1} A_i}{e_p^T d}\right)^2 \gamma_q, \text{ where } d = B^{-1}A_q.$$

- **Simplified Form**: Using r, $\hat{d}$ (solving $B^T r = e_p$, $B^T \hat{d} = d$),

$$\gamma_i^+ = \gamma_i - 2\left(\frac{r^T A_i}{r^T A_q}\right) \hat{d}^T A_i + \left(\frac{r^T A_i}{r^T A_q}\right)^2 \gamma_q.$$

  Computes new norms efficiently.
- **Computation**: Solve $B^T \hat{d} = d$, $B^T r = e_p$, then evaluate $r^T A_i$, $\hat{d}^T A_i$ for $i \in \mathcal{N}$, $i \neq q$, to update all γ_i^+.
- **Practical Note**: Steepest-edge doesn't ensure long steps to next vertex but is highly effective in practice.

Comments

Let us now complete the update for the squared norms. Substituting the expression for $B^+)^{-1}A_i$ into the formula for γ_i^+, we obtain an explicit update. The key observation is that the new norm can be expressed in terms of the old norm, together with inner products involving a small number of auxiliary vectors. Specifically, one introduces two vectors, denoted r and $\hat{d}$, which solve the linear systems $B^T r = e_p$ and $B^T \hat{d} = d$, respectively. Using these, the update formula becomes much more compact:

$$\gamma_i^+ = \gamma_i - 2\left(\frac{r^T A_i}{r^T A_q}\right) \hat{d}^T A_i + \left(\frac{r^T A_i}{r^T A_q}\right)^2 \gamma_q.$$

This formula highlights the efficiency of the approach: rather than recomputing norms from scratch, one solves two auxiliary systems once per iteration, and then applies inexpensive vector operations to update all γ_i.

From a practical perspective, this makes the steepest-edge rule computationally viable. Importantly, while the method aims to choose directions that correspond to steepest descent, it does not necessarily ensure long steps in the primal space. Nevertheless, empirical studies have consistently shown that it tends to accelerate convergence and to avoid the stalling that can occur under Dantzig's rule. In fact, many state-of-the-art simplex implementations rely on steepest-edge or variants of it as their default pricing strategy.

Two-Phase Simplex: Finding a Starting Point

Sensitivity Analysis
Fundamental Theorem
SIMPLEX METHOD
Steepest-Edge Rule
Two-Phase Simplex

Key Idea: Simplex method for $\min c^T x$, s.t. $Ax = b$, $x \geq 0$ needs a feasible starting point x with basis $\mathcal{B}$. Use two-phase approach to find it.

Problem Context: Finding initial x with basis $\mathcal{B}$ (m indices), where basis matrix B is nonsingular and $x_{\mathcal{B}} = B^{-1}b \geq 0$, $x_{\mathcal{N}} = 0$, is as hard as solving the linear program.

Two-Phase Strategy:

- Phase I: Solve auxiliary problem to get feasible point.
- Phase II: Use Phase I solution to solve original problem.

Phase I designs an easy-to-start problem:

$$\min \sum z_j, \text{ s.t. } Ax + Ez = b, \ (x, z) \geq 0,$$

with $z \in \mathbb{R}^m$, E is a diagonal matrix:

$$E_{jj} = \begin{cases} 1, & \text{if } b_j \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

Starting Point: Set $x = 0$, $z_j = |b_j|$. Basis $B = E$ is invertible, ensuring a simple feasible solution to launch Phase I

Why It Works: The point $(x = 0, z_j = |b_j|)$ satisfies $Ax + Ez = b$, and E's diagonal structure guarantees an invertible basis for Phase I, enabling Phase II to extract the original solution.

Comments

In order for the simplex method can work, it is necessary to select an initial feasible point with a valid basis. Without such a point, the algorithm cannot even begin, because every iteration of the simplex relies on moving between adjacent basic feasible solutions. The challenge is that finding this first feasible solution can be just as difficult as solving the entire problem itself. To overcome this, the two-phase strategy was developed.

The essence of Phase One is to construct an auxiliary problem that is guaranteed to have a simple starting solution. Artificial variables are introduced to absorb infeasibilities in the original system. Suppose we want to solve the system $Ax = b$ with $x \geq 0$. For each constraint, we add an artificial variable z_j. Then we minimize the sum of all artificial variables. In words, the auxiliary problem becomes: minimize the sum over all z_j, subject to $Ax + Ez = b$, with nonnegativity on both x and z. The matrix E is chosen as a diagonal matrix where each diagonal entry is either plus one or minus one, depending on the sign of the corresponding right-hand side b_j. This construction guarantees feasibility.

The starting point is straightforward: set all x variables to zero, and assign each artificial variable z_j to be the absolute value of b_j. This ensures the equation is satisfied, because then $Ax + Ez = b$. At the same time, the artificial basis matrix B=E is invertible by design. Thus, Phase One always begins with a valid basis and a feasible point. Once Phase One concludes, if the artificial variables can be driven to zero, then the solution obtained will also satisfy the original system and will provide the required feasible starting point for Phase Two.

Two-Phase Simplex: Checking Feasibility

Phase I minimizes constraint violations to find a feasible point for the original problem $\min c^T x$, s.t. $Ax = b$, $x \geq 0$, solved in Phase II.

Phase I Role: For

$$\min \sum z_j, \text{ s.t. } Ax + Ez = b, \ (x, z) \geq 0,$$

z measures violations of $Ax = b$.

- Minimizing $\sum z_j$ seeks x feasible for original constraints.
- $\sum z_j = 0$ if and only if original problem is feasible (then $\tilde{z} = 0$, $A\tilde{x} = b$, $\tilde{x} \geq 0$). If $\sum z_j > 0$, problem is infeasible.

Start simplex at $x = 0$, $z_j = |b_j|$. Bounded below by 0, so terminates.

Phase II Role: If $\sum \tilde{z}_j = 0$, use $(\tilde{x}, \tilde{z})$ to solve:

$$\min c^T x, \text{ s.t. } Ax+z = b, \ x \geq 0, \ 0 \geq z \geq 0.$$

- Matches original problem, as $z = 0$ for feasibility.
- Retains z in basis from Phase I (with $\tilde{z}_j = 0$).

Solution gives original x.

Comments

The crucial purpose of Phase One is to determine whether the original linear program is feasible. The artificial problem introduces slack in the form of variables z_j that capture violations of the equality constraints. If we denote the auxiliary objective as minimizing the sum of these violations, the interpretation becomes clear: if the minimum sum equals zero, then we have found an exact feasible point for the original system. If the minimum remains strictly positive, then no feasible solution exists.

To start Phase One, the simplex algorithm uses the artificially constructed feasible solution, where all original variables are set to zero and the artificial variables equal the absolute values of the right-hand sides. Since the objective is nonnegative and bounded below by zero, the procedure is guaranteed to terminate.

When the outcome of Phase One yields a zero objective value, the artificial variables vanish, and the corresponding vector of original variables satisfies $Ax = b$ with $x \geq 0$. At that point, Phase Two can begin. In Phase Two, the artificial variables are formally kept in the system, but they are forced to equal zero. Their role is essentially transitional: they remain in the basis initially, ensuring continuity of the simplex process, but they do not affect the optimization since their coefficients vanish.

If, instead, the Phase One optimization concludes with a strictly positive objective value, then it is impossible to satisfy the original equalities with nonnegative variables. In that case, we must declare the original problem infeasible, and no further optimization is possible. Thus, Phase One not only generates an initial feasible basis when possible, but it also acts as a test of feasibility for the entire model.

Two-Phase Simplex: Phase II Details

Finalizing Phase II for $\min c^T x$, s.t. $Ax = b$, $x \geq 0$, and optimizing Phase I.

- **Phase II Adjustments:** For $\min c^T x$, s.t. $Ax + z = b$, $x \geq 0$, $0 \geq z \geq 0$, modify simplex to handle z's two-sided bounds. Remove z_j from problem once it leaves basis to avoid redundant iterations.
- Solution Properties: Basic solution (x^*, z^*) has $z^* = 0$, so x^* solves original problem. But basis $\mathcal{B}$ may include z^*-components, not ideal for original problem.

Postprocessing: Extract z-components from $\mathcal{B}$, replace with nonbasic x-components to keep basis matrix invertible, yielding optimal basis for original problem.

- **Phase I Optimization:** No need for m artificial variables if slack/surplus variables (from inequality constraints) can act as artificial ones, reducing complexity.

Comments

Once Phase One provides a feasible point for the original system, the algorithm transitions into Phase Two, where the true objective function is optimized. The formulation at this stage can include artificial variables constrained between zero and zero, but these variables are superfluous. As soon as any artificial variable leaves the basis, it is eliminated from further consideration, since it no longer plays any role in the optimization.

The key property of the Phase Two solution is that the final basic feasible solution will have all artificial variables equal to zero. The remaining vector of decision variables then constitutes a feasible solution to the original problem. However, the basis at termination may still formally include some artificial variables. In such cases, a postprocessing step is necessary: one replaces artificial components in the basis with genuine decision variables that are nonbasic but can enter without losing invertibility. This yields a proper basis for the original problem, ensuring that the final tableau fully corresponds to the actual decision space.

Another practical refinement concerns the construction of Phase One itself. In many problems that originate with inequality constraints, slack or surplus variables already exist. These can often serve as artificial variables, reducing the number of additional variables that must be introduced. This significantly lowers the size of the auxiliary problem and simplifies computations. Thus, the two-phase method is not only theoretically sound but also efficient in practice, as it exploits the problem's structure to minimize unnecessary complexity.