Linear Programming Course Summary

Generated by Scribe.AI

December 4, 2024

1 Key Terms

1.1 Linear Programming Fundamentals

- **Slack Variable**: A variable added to a less-than-or-equal-to inequality constraint to transform it into an equality constraint. The slack variable represents the difference between the left-hand side and the right-hand side of the inequality.
- **Feasible Region**: The set of all points that satisfy all the constraints of a linear programming problem.
- **Vertex**: A point where two or more constraint boundaries intersect in the feasible region of a linear programming problem. In higher dimensions, it is the intersection of multiple constraint hyperplanes.
- **Basic Variable**: In the Simplex method, a variable that is expressed explicitly in terms of non-basic variables in a dictionary.
- **Non-basic Variable**: In the Simplex method, a variable that is set to zero in a given iteration of the algorithm.
- **Primal Problem**: A linear programming problem in its standard maximization form, aiming to maximize a linear objective function subject to linear inequality constraints and non-negativity constraints.
- **Dual Problem**: A linear programming problem derived from the primal problem, typically in minimization form, with constraints and objective function related to the primal problem through matrix transposition and sign changes.
- **Weak Duality**: A theorem stating that the objective function value of any feasible solution to the primal problem is always less than or equal to the objective function value of any feasible solution to the dual problem.
- **Strong Duality**: A theorem stating that if a linear programming problem has an optimal solution, then its dual problem also has an optimal solution, and the optimal objective function values of both problems are equal.
- **Complementary Slackness**: A condition stating that for optimal primal and dual solutions, the product of each primal variable and its corresponding dual slack variable is zero, and the product of each dual variable and its corresponding primal slack variable is zero.
- **Negative Transpose Property**: The property that the coefficient matrix of the dual problem is the negative transpose of the coefficient matrix of the primal problem. This property is preserved throughout the simplex method iterations.
- **Shadow Price**: The rate of change in the optimal objective function value for a unit change in the right-hand side of a constraint.
- **Farkas Lemma**: A fundamental theorem in linear programming stating that a system of linear inequalities $Ax \leq b$ has no solution if and only if there exists a vector y such that $A^T y = 0$, $y \geq 0$, and $b^T y < 0$.

1.2 Linear Programming Algorithms

- **Pivot Operation**: The process of exchanging a basic and a non-basic variable in the Simplex method, updating the dictionary to move to a new vertex.
- **Dictionary**: A representation of a linear programming problem in a tabular format, where basic variables are expressed as functions of non-basic variables. It is used in the Simplex method to systematically move from one vertex to another.
- **Auxiliary Problem**: A modified linear programming problem introduced to find an initial feasible solution when the origin is not feasible in the original problem.
- **Largest-Coefficient Rule**: A pivot rule in the Simplex method that selects the entering variable with the largest coefficient in the objective function row of the dictionary.
- **Largest-Increase Rule**: A pivot rule in the Simplex method that selects the entering variable that results in the largest increase in the objective function value.
- **Klee-Minty Example**: A worst-case example for the Simplex method, demonstrating that the number of iterations can grow exponentially with the problem size.
- **Bland's Rule**: A pivot rule in the Simplex method designed to prevent cycling, ensuring termination.
- **Lexicographic Method**: A method for choosing the leaving variable in the Simplex method that ensures termination.
- **Interior-Point Method**: A class of algorithms for linear programming that traverse the interior of the feasible region, in contrast to the Simplex method which follows the boundary.
- **Dual Simplex Method**: A variant of the simplex method that starts with a dual feasible solution and iteratively improves it until a primal feasible and optimal solution is found.
- **Dual Based Phase I Algorithm**: A method used to find an initial feasible solution for the dual problem when the origin is not dual feasible, often as a precursor to the dual simplex method.
- **Entering Variable**: The non-basic variable selected to enter the basis in the next iteration of the simplex method, typically chosen based on the reduced costs.
- **Leaving Variable**: The basic variable selected to leave the basis in the next iteration of the simplex method, typically chosen based on the minimum ratio test.
- **Elementary Matrix**: A matrix obtained from an identity matrix by performing a single elementary row operation (e.g., swapping two rows, multiplying a row by a scalar, adding a multiple of one row to another).

1.3 Network Optimization

- **Network Flow**: A mathematical model representing the flow of commodities through a network, often formulated as a linear program.
- **Nodes**: Points in a network representing sources, sinks, or intermediate points.
- **Arcs**: Directed edges in a network connecting nodes, representing the flow paths.
- b_i: Net flow at node i; positive values represent supply, negative values represent demand.
- X_{ij}: Amount of flow transported from node i to node j.
- C_{ij}: Cost of transporting one unit of flow from node i to node j.
- **Balanced Equation**: An equation stating that the total flow into a node equals the total flow out of the node, plus any supply or demand at that node.

- **Incidence Matrix**: A matrix representing the network structure, where rows correspond to nodes and columns correspond to arcs.
- **Root Node**: A node selected as a reference point for simplifying the network flow equations.
- **Tree Solution**: A feasible solution to a network flow problem where the flow along arcs in a spanning tree ~~is zero~~.
- **Spanning Tree**: A tree that connects all nodes in a network.
- **Reduced Cost**: The cost of increasing the flow along a non-basic arc by one unit.
- **Primal Flow**: The flow values assigned to the arcs in a network flow problem that satisfy the balance equations at each node. It ~~may be~~ is infeasible if some flow values are negative.
- **Dual Flow**: The values of the dual variables associated with the nodes in a network flow problem. They satisfy the dual constraints, and ~~may be~~ is infeasible if some dual slack variables are ~~positive~~ negative.
- **Dual Slack**: The difference between the cost of an arc and the difference in dual variables between its endpoints. It represents the reduced cost of the arc in the dual problem.
- **Leaf Node**: A node in a tree with only one edge connected to it.
- **Transportation Problem**: A network flow problem where the objective is to minimize the cost of transporting goods from multiple sources to multiple destinations, subject to supply and demand constraints. The constraints may be equalities or inequalities. (bipartite graph)
- **Dummy Node**: A node added to a network flow model to handle inequality constraints in supply and demand. It allows for representing excess supply or unmet demand.
- **Upper Bounded Transshipment Problem**: A network flow problem where the flow along each arc is constrained by an upper bound, in addition to supply and demand constraints.
- **Shortest Path Problem**: A graph problem where the objective is to find the path of minimum cost or distance between a source node and a destination node in a weighted graph.
- **Bellman's Equation**: A recursive equation used to solve the shortest path problem. It defines the shortest distance to a node as the minimum of the distances to its neighbors plus the cost of the edge connecting them.
- **Dijkstra's Algorithm**: A greedy algorithm for finding the shortest paths from a single source node to all other nodes in a graph with non-negative edge weights.
- **Max Flow**: The maximum amount of flow that can be sent from a source node to a sink node in a network, subject to capacity constraints on the edges.
- **Min Cut**: A partition of the nodes in a network into two sets, one containing the source and the other containing the sink, such that the sum of the capacities of the edges crossing from the source set to the sink set is minimized.
- **Max-Flow Min-Cut Theorem**: A theorem stating that the maximum flow in a network is equal to the minimum capacity of any cut in the network.
- **Augmenting Path**: A path in a network from the source to the sink along which the flow can be increased without violating capacity constraints.
- **Ford-Fulkerson Algorithm**: An algorithm for finding the maximum flow in a network by iteratively finding augmenting paths and increasing the flow along them.
- **Graphical Method**: A method for solving linear programming problems with two variables by visualizing the feasible region and objective function on a graph. The optimal solution is found at a vertex of the feasible region.
- Minimum Spanning Tree – Greedy algorithm

1.4 Integer Programming

- **Integer Programming**: A mathematical optimization problem where the objective function and constraints are linear, but the variables are restricted to integer values.
- **Relaxed Problem**: The linear programming problem obtained by removing the integer constraints from an integer programming problem.
- **Branch-and-Bound**: A method for solving integer programming problems by recursively partitioning the feasible region into smaller subproblems and bounding the optimal solution within each subproblem.
- **Enumeration Tree**: A tree structure used in the Branch-and-Bound algorithm to represent the partitioning of the feasible region and the exploration of subproblems.
- **Pruning**: The process of eliminating branches in the enumeration tree that cannot lead to a better solution than the best integer solution found so far.
- **Gomory Cuts**: Additional linear constraints added to an integer programming problem to cut off non-integer solutions from the feasible region of the relaxed problem.

1.5 Advanced Topics

- **Smoothed Complexity**: A framework for analyzing the complexity of algorithms by considering the average-case performance after small random perturbations to the input.
- **Lagrange Multiplier**: A variable introduced in the Lagrangian function to incorporate constraints into an optimization problem.
- **Lagrangian Function**: A function that combines the objective function and constraints of an optimization problem using Lagrange multipliers.
- **Weak Duality Theorem**: A theorem stating that the optimal value of the primal problem is always less than or equal to the optimal value of the dual problem.
- **Strong Duality Theorem**: A theorem stating that if the primal problem has a finite optimal solution, then the dual problem also has a finite optimal solution, and their optimal values are equal.
- **Fredholm Alternatives**: A statement of the conditions under which the system $Ax = b$ either has a solution or there exists a vector y such that $y^T A = 0$ and $y^T b \neq 0$.

2 Problem Types

2.1 Linear Programming Fundamentals

- **Formulating the Diet Problem as a Linear Program**: This problem involves translating the description of a diet problem (minimizing cost while meeting nutritional requirements) into a mathematical linear program with an objective function and constraints.
- **Representing the Diet Problem using Tables**: This problem involves representing the data of the diet problem (nutrient content of foods, minimum nutrient requirements, and food costs) in a tabular format to facilitate the formulation of the linear program.
- **Transforming the Diet Problem into a Pill-Selling Problem**: This problem involves reinterpreting the diet problem as a profit maximization problem by considering the sale of nutrient pills instead of food consumption. Dual Problem
- **Representing Linear Programs in Matrix Notation**: This problem focuses on expressing linear programs using matrices and vectors, facilitating the application of the simplex method.

- **Solving a Linear Program Graphically**: This problem involves using a graphical method to find the feasible region and optimal solution of a linear program.
- **Handling Unrestricted Variables in Linear Programming**: This problem addresses how to handle linear programs where variables are not restricted to be non-negative. A technique of splitting each unrestricted variable into the difference of two non-negative variables is presented.

LP in general form

2.2 Simplex Method Algorithms and Analysis

- **Finding an Initial Feasible Solution**: The Simplex method requires an initial feasible solution. If the origin is not feasible, an auxiliary problem is introduced to find a feasible solution to the original problem.
- **Handling Infeasible Origins**: The origin $(0, 0)$ may not satisfy all constraints in a linear programming problem. Techniques like introducing an auxiliary problem are used to find a feasible starting point.
- **Using Dictionaries to Represent Solutions**: The Simplex method uses dictionaries to represent the current solution and to perform pivot operations efficiently. Each dictionary corresponds to a vertex of the feasible region.
- **Worst-Case Analysis of the Simplex Method**: The Klee-Minty example demonstrates that the Simplex method can require an exponential number of iterations in the worst case, depending on the pivot rule used.
- **Impact of Pivot Rules on Simplex Method Efficiency**: Different pivot rules (e.g., largest-coefficient, largest-increase) significantly affect the number of iterations required by the Simplex method, with some rules exhibiting exponential worst-case behavior while others perform better in practice.
- **Empirical Analysis of Simplex Method Performance**: Empirical studies show that the number of iterations in the Simplex method tends to grow polynomially with the problem size in practice, even though worst-case examples exist.
- **Theoretical Bounds on Simplex Method Iterations**: Theoretical analysis of randomized pivot rules provides bounds on the expected number of iterations, which are better than exponential but not polynomial.
- **Smoothed Complexity of the Simplex Method**: The smoothed complexity analysis suggests that small random perturbations to the input data can significantly improve the performance of the Simplex method, leading to a polynomial number of iterations with high probability.
- **Partitioning Variables into Basic and Non-Basic Sets**: This problem involves separating variables into basic and non-basic sets, a crucial step in the simplex algorithm for iterative solution updates.
- **Expressing the Objective Function and Constraints using Basic and Non-Basic Variables**: This problem describes the process of rewriting the objective function and constraints in terms of basic and non-basic variables to simplify calculations within the simplex method.
- **Updating the Simplex Tableau using Matrix Operations**: This problem involves using matrix operations to update the simplex tableau during each iteration of the simplex method.
- **Solving a Linear Program using the Simplex Method**: This problem involves applying the simplex method to a specific linear program, iteratively updating the tableau until an optimal solution is found.
- **Determining the range of objective function coefficient changes that maintain optimality**: This problem investigates how much the coefficients in the objective function can change before the optimal solution changes. The analysis uses the reduced costs of the non-basic variables to determine the allowable range of changes.

- **Determining the range of constraint value changes that maintain optimality**: This problem investigates how much the right-hand side values of the constraints can change before the optimal solution changes. The analysis uses the optimal values of the basic variables and the inverse of the basis matrix to determine the allowable range of changes.
- **Sensitivity Analysis**: Determining how changes in constraint values or objective function coefficients affect the optimal solution and objective function value.
- **Solving Linear Programs using the Simplex Method and Matrix Operations**: Utilizing matrix operations and the simplex tableau to solve linear programs.

2.3 Duality and Optimality Conditions

- **Verifying Optimality using Complementary Slackness**: This problem focuses on using the complementary slackness conditions ($x_j z_j = 0$ and $w_i y_i = 0$) to verify whether a given primal and dual feasible solution pair is optimal.
- **Demonstrating the Negative Transpose Property in Simplex Iterations**: This problem involves showing that the negative transpose relationship between the primal and dual tableaux is maintained throughout the simplex method iterations.
- **Proving the Strong Duality Theorem**: This problem involves proving that if a primal linear program has an optimal solution, then its dual also has an optimal solution, and their objective function values are equal. The proof utilizes the structure of the simplex tableau at optimality and the complementary slackness conditions.
- **Relating Primal and Dual Objective Functions**: This problem focuses on demonstrating the relationship between the primal and dual objective functions, showing how the primal objective function is always less than or equal to the dual objective function, and how they are equal at optimality.
- **Formulating the Lagrangian for Equality Constraints**: This problem involves constructing the Lagrangian function $L(x, \mu) = f_0(x) - \sum_{j=1}^{n} \mu_j g_j(x)$ to incorporate equality constraints $g_j(x) = 0$ into the optimization problem.
- **Formulating the Lagrangian for Inequality and Equality Constraints**: This problem involves constructing the Lagrangian function $L(x, \lambda, \mu) = f_0(x) - \sum_{i=1}^{m} \lambda_i f_i(x) - \sum_{j=1}^{n} \mu_j g_j(x)$ to incorporate both inequality constraints $f_i(x) \leq 0$ and equality constraints $g_j(x) = 0$ into the optimization problem.
- **Establishing Weak Duality**: This problem involves proving that the optimal value of the primal problem is less than or equal to the optimal value of the dual problem, i.e., $\max_{x \in \mathcal{G}} \xi(x) \leq \min_{\lambda \geq 0, \mu} \xi(\lambda, \mu)$.
- **Establishing Strong Duality**: This problem involves proving that under certain conditions (e.g., the primal problem having a finite optimal solution), the optimal values of the primal and dual problems are equal, i.e., $\xi(x^) = \xi(\lambda^)$.
- **Deriving the Dual Problem from the Lagrangian**: This problem involves deriving the dual problem by analyzing the maximum of the Lagrangian function with respect to the primal variables, leading to the dual objective function and constraints.
- **Applying Complementary Slackness**: This problem involves using the complementary slackness condition ($x^{i\lambda i=0}$ for all i) to verify the optimality of primal and dual solutions.
- **Formulating the Dual of the Diet Problem**: This problem involves deriving the dual linear program from the primal linear program representing the diet problem, interpreting the dual variables in the context of the problem.
- **Illustrating Complementary Slackness through Variable Relationships**: This problem demonstrates the relationship between primal and dual variables, specifically how their entry and exit from the basis are complementary.

- **Deriving the Dual Problem from the Primal Problem**: This problem focuses on deriving the dual linear program from a given primal linear program. The derivation involves using Lagrange multipliers and manipulating inequalities to obtain the dual objective function and constraints.
- **Proving Weak Duality**: This problem involves demonstrating that the objective function value of any feasible solution to the primal problem is less than or equal to the objective function value of any feasible solution to the dual problem.
- **Deriving and Utilizing the Dual Problem**: Formulating the dual problem from the primal problem and using it to verify optimality and perform sensitivity analysis.

2.4 Network Flow Problems

- **Formulating Network Flow Problems as Linear Programs**: This problem involves representing a network flow problem, characterized by nodes, arcs, supply/demand at nodes, and arc costs, as a linear program with an objective function minimizing total transportation cost and constraints ensuring flow conservation at each node.
- **Analyzing the Incidence Matrix of a Network**: This problem focuses on understanding the properties of the incidence matrix, including its rank and the relationship between its submatrices and spanning trees of the network. The goal is to leverage these properties to efficiently solve the network flow problem.
- **Deriving and Utilizing the Dual Problem for Network Flows**: This problem involves formulating the dual linear program for a given network flow problem and interpreting its variables and constraints in the context of the network. The goal is to use the dual problem to gain insights into the primal problem and its solution.
- **Finding Tree Solutions and Verifying Optimality**: This problem involves finding a tree solution (a feasible solution where flow is zero along arcs in a spanning tree) and verifying its optimality using complementary slackness conditions. The goal is to efficiently find an optimal solution by leveraging the structure of the network.
- **Applying the Primal Simplex Method to Network Flows**: This problem involves using the primal simplex method to iteratively improve a tree solution until an optimal solution is found. The steps involve identifying entering and leaving variables, updating the spanning tree, and updating primal and dual variables.
- **Finding a spanning tree in a network**: This problem involves finding a connected, acyclic subgraph that spans all nodes in a given network.
- **Finding a primal feasible flow**: This problem focuses on determining a flow in a network that satisfies all flow conservation constraints, potentially with some flow values being negative.
- **Finding a dual feasible solution**: This problem involves finding a set of dual variables that satisfy all dual constraints, potentially with some dual variables being positive.
- **Selecting entering and leaving arcs in the primal network simplex method**: This problem involves choosing an entering arc with a negative reduced cost and a leaving arc to maintain a spanning tree structure, ensuring cost reduction.
- **Selecting the entering arc in the dual network simplex method**: This problem involves choosing an entering arc that reconnects two disconnected subtrees while maintaining dual feasibility.
- **Updating primal and dual flows**: This problem involves updating the primal flow values and dual variables after selecting entering and leaving arcs to maintain feasibility and optimality.
- **Handling infeasible primal or dual solutions**: This problem addresses situations where initial solutions are not feasible, requiring modifications to the problem or the use of two-phase methods.

- **Finding an optimal solution using primal and dual network simplex methods**: This problem involves iteratively improving a feasible solution using primal or dual pivot steps until an optimal solution is reached.
- **Developing a two-phased network simplex method**: This problem involves designing a method that first finds a feasible solution (using artificial costs or supplies) and then optimizes it using the primal network simplex method.
- **Developing a parametric self-dual network simplex method**: This problem involves creating a method that interweaves primal and dual pivots to reduce a perturbation parameter and find an optimal solution.
- **Formulating Transportation Problems with Inequality Constraints**: This problem involves formulating a transportation problem where supply at sources may exceed demand at destinations, or vice versa, requiring the use of dummy nodes and slack variables to balance the network flow.
- **Formulating General Network Flow Problems with Inequality Constraints**: This problem extends the transportation problem to more general network flow scenarios with inequality constraints, using incidence matrices and slack variables to represent excess supply or unmet demand.
- **Solving the Upper Bounded Transshipment Problem**: This problem focuses on finding optimal flows in a network where each arc has an upper bound on the flow it can carry. This requires the introduction of slack variables to handle the upper bound constraints.
- **Solving the Shortest Path Problem using Linear Programming**: This problem involves formulating the shortest path problem as a linear program, using flow variables and constraints to ensure that exactly one unit of flow reaches the destination node from the source node.
- **Solving the Shortest Path Problem using Bellman's Equation**: This problem focuses on finding the shortest path using Bellman's dynamic programming approach, iteratively approximating the shortest distances from each node to the destination node.
- **Solving the Shortest Path Problem using Dijkstra's Algorithm**: This problem involves using Dijkstra's algorithm to find the shortest path, iteratively updating shortest distance estimates and tracking the path.
- **Finding Fair Prices in Network Simplex**: This problem involves determining a set of prices y_i at each node i in a network such that $y_i + c_{ij} = y_j$ for each arc ij in a spanning tree T, representing fair market prices consistent with shipping costs.
- **Identifying Profitable Shipping Opportunities for Competitors**: This problem focuses on finding an arc ij in a network where the cost of buying at node i and shipping to node j ($y_i + c_{ij}$) is less than the selling price at node j (y_j), indicating a profitable opportunity for a competitor.
- **Adjusting Shipments to Maintain Feasibility**: This problem involves adjusting shipments along the arcs of a spanning tree T in a network to maintain feasibility after introducing flow along a new arc, ensuring the balance of flow is preserved.

2.5 Integer Programming

- **Integer Programming**: A linear programming problem where some or all variables are restricted to integer values.
- **Solving Integer Programs using Branch and Bound**: A method for solving integer programs by recursively partitioning the feasible region and solving relaxed linear programs.
- **Applying Gomory Cuts**: A method to iteratively add constraints (cuts) to a relaxed linear program to eliminate non-integer solutions, eventually leading to an integer optimal solution.

2.6 Advanced Topics

- **Optimizing in Higher Dimensions**: The Simplex method is extended to linear programming problems with more than two variables, requiring iterative pivoting operations to find the optimal solution.
- **Existence of Polynomial-Time Algorithms for Linear Programming**: Polynomial-time algorithms for linear programming exist, such as interior-point methods, which contrast with the Simplex method's potential for exponential worst-case behavior.

3 Algorithm Solutions

3.1 Linear Programming Fundamentals

- **Simplex Method**: Iteratively move from one vertex of the feasible region to another by setting non-basic variables to zero and choosing a new set of (non)basic variables to improve the objective function. This corresponds to moving from vertex to vertex in the feasible region.
- **Auxiliary Problem**: If the origin is not feasible, introduce a new variable x_0 and modify the constraints to create an auxiliary problem that is feasible. Solve the auxiliary problem using the Simplex method. If the optimal solution has $x_0 = 0$, then the solution is feasible for the original problem.
- **Largest-Coefficient Rule**: Choose the variable with the largest coefficient in the objective function row to enter the basis.
- **Largest-Increase Rule**: Choose the variable whose entrance into the basis brings about the largest increase in the objective function.
- **Bland's Rule**: Choose the variable with the smallest index among those eligible to enter the basis.
- **Randomized Pivot Rule**: Randomly permute the indices of the variables; then use Bland's rule for entering and the lexicographic method for leaving variables.
- **Weak Duality**: $c^T x \leq b^T y$
- **Strong Duality**: If (P) has an optimal solution x, then (D) has an optimal solution y, and $c^T x = b^T y$.
- **Complementary Slackness**: Primal feasible x and dual feasible y are optimal if and only if $x_j z_j = 0$ for all j and $w_i y_i = 0$ for all i.
- **Lagrangian Function**: $L(x, \lambda, \mu) = f_0(x) - \sum_{i=1}^{m} \lambda_i f_i(x) - \sum_{j=1}^{n} \mu_j g_j(x)$
- **Simplex Method in Matrix Form**: The simplex method iteratively improves a feasible solution by exchanging basic and non-basic variables until an optimal solution is found. This involves updating the objective function and constraints using matrix operations, specifically involving the basis matrix B and its inverse B^{-1}, and the non-basic variable matrix N. The objective function is expressed as $\mathcal{Z} = \mathbf{c}_B^T B^{-1}\mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T B^{-1} N)\mathbf{x}_N$, where $\mathbf{c}_B$ and $\mathbf{c}_N$ are the cost vectors for basic and non-basic variables, respectively, and $\mathbf{b}$ is the right-hand side vector of the constraints.
- **Derivation of the Dual Problem**: The dual problem is derived by taking a linear combination of the primal problem's constraints using Lagrange multipliers, leading to the dual objective function and constraints.
- **Dual Simplex Method**: This algorithm solves linear programming problems by iteratively improving a dual-feasible solution until primal feasibility and optimality are achieved.
- **Dual Based Phase I Algorithm**: This algorithm finds an initial dual-feasible solution for the dual simplex method when the origin is not feasible for both the primal and dual problems.

3.2 Linear Programming Sensitivity Analysis

- **Sensitivity Analysis of Objective Function Coefficients**: Determine $\Delta C_N = (B^{-1}N)^T \Delta C_B - \Delta C_N$ such that $\tilde{Z}_N^+ \Delta \tilde{Z}_N \geq 0$, where $\tilde{Z}_N^= (B^{-1}N)^T C_B - C_N$ is the vector of reduced costs at the optimal solution.
- **Sensitivity Analysis of Constraint Values**: Determine $\Delta X_B^= B^{-1} \Delta b$ such that $X_B^+ \Delta X_B^\geq 0$, where $X_B^= B^{-1} b$ is the vector of basic variable values at the optimal solution.
- **Sensitivity Analysis**: This technique analyzes how changes in the objective function coefficients or constraint values affect the optimal solution and objective function value of a linear programming problem. It uses the optimal dual variables (shadow prices) to quantify these effects.

3.3 Advanced Linear Programming Techniques

- **Branch and Bound**: This algorithm solves integer programming problems by recursively partitioning the feasible region into subproblems, solving the relaxed linear program for each subproblem, and using the objective function values to prune branches that cannot lead to a better integer solution. The algorithm continues until an optimal integer solution is found.
- **Gomory Cuts**: This algorithm solves integer programming problems by iteratively adding new constraints (Gomory cuts) to the relaxed linear program. These cuts are derived from the fractional parts of the basic variables in the optimal dictionary of the relaxed problem and are designed to eliminate non-integer solutions while maintaining the integer feasible region. The process is repeated until an integer optimal solution is found.

3.4 Network Flow Problems

- **Network Flow Problem Formulation**: The network flow problem is formulated as a linear program with an objective function to minimize the total cost of transportation $min \sum_{(i,j)\in\mathcal{A}} C_{ij} X_{ij}$ subject to flow conservation constraints at each node $\sum_i x_{ik} - \sum_j x_{kj} = -b_k$ and non-negativity constraints $X \geq 0$.
- **Primal Simplex Method for Network Flows**: The primal simplex method is adapted to solve network flow problems by iteratively updating a spanning tree. The algorithm identifies an entering variable (arc with the most negative reduced cost), a leaving variable (arc in the cycle formed by the entering variable and the spanning tree), updates the spanning tree, and updates primal and dual variables until all dual slacks are non-negative.
- **Finding a Spanning Tree Basis**: A square submatrix of the modified incidence matrix $\tilde{A}$ (incidence matrix with a root node row removed) is a basis if and only if the arcs corresponding to its columns form a spanning tree. This allows for efficient representation and solution of the network flow problem using the simplex method.
- **Dual Variable Calculation from Spanning Tree**: Given a spanning tree, dual variables are calculated iteratively starting from a root node using the dual flow equation $y_j - y_i + z_{ij} = c_{ij}$ for arcs in the tree ($z_{ij} = 0$). Dual slacks for arcs not in the tree are then computed using $z_{ij} = y_j - y_i + c_{ij}$.
- **Primal Network Simplex**: Iteratively choose an entering arc with $z_{ij} < 0$, find a cycle including the entering arc, select a leaving arc with the smallest flow oriented in the reverse direction of the entering arc within the cycle, and update primal and dual flows along the cycle.
- **Dual Network Simplex**: Iteratively choose a leaving arc with $x_{ij} < 0$, find two subtrees resulting from removing the leaving arc, select an entering arc with the smallest dual slack bridging the two subtrees in the opposite direction of the leaving arc, and update primal and dual flows.

3.5 Shortest Path Algorithms

- **Bellman's Equation**: $v_i = \min_j \{c_{ij} + v_j\}$, $v_r = 0$

3.6 Graphical Methods

- **Graphical Method**: Solve a linear program with two variables by plotting the constraints to define the feasible region and then finding the vertex that maximizes (or minimizes) the objective function.

3.7 Farkas' Lemma

- **Farkas Lemma**: The system $Ax \leq b$ has no solutions if and only if there is a y such that $A^T y = 0$, $y \geq 0$, and $b^T y < 0$.