

Transportation Simplex Method and Sensitivity Analyses

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

<http://www.stanford.edu/~yyye>

The Transportation Problem

$$\begin{aligned} \min \quad & \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^n x_{ij} = s_i, \quad \forall i \\ & \sum_{i=1}^m x_{ij} = d_j, \quad \forall j \\ & x_{ij} \geq 0, \quad \forall i, j. \end{aligned}$$

Assume that the total supply equal the total demand. Thus, exactly one equality constraint is redundant.

At each step the simplex method attempts to send units along a route that is **unused (non-basic)** in the current BFS, while eliminating one of the routes that is currently being **used (basic)**.

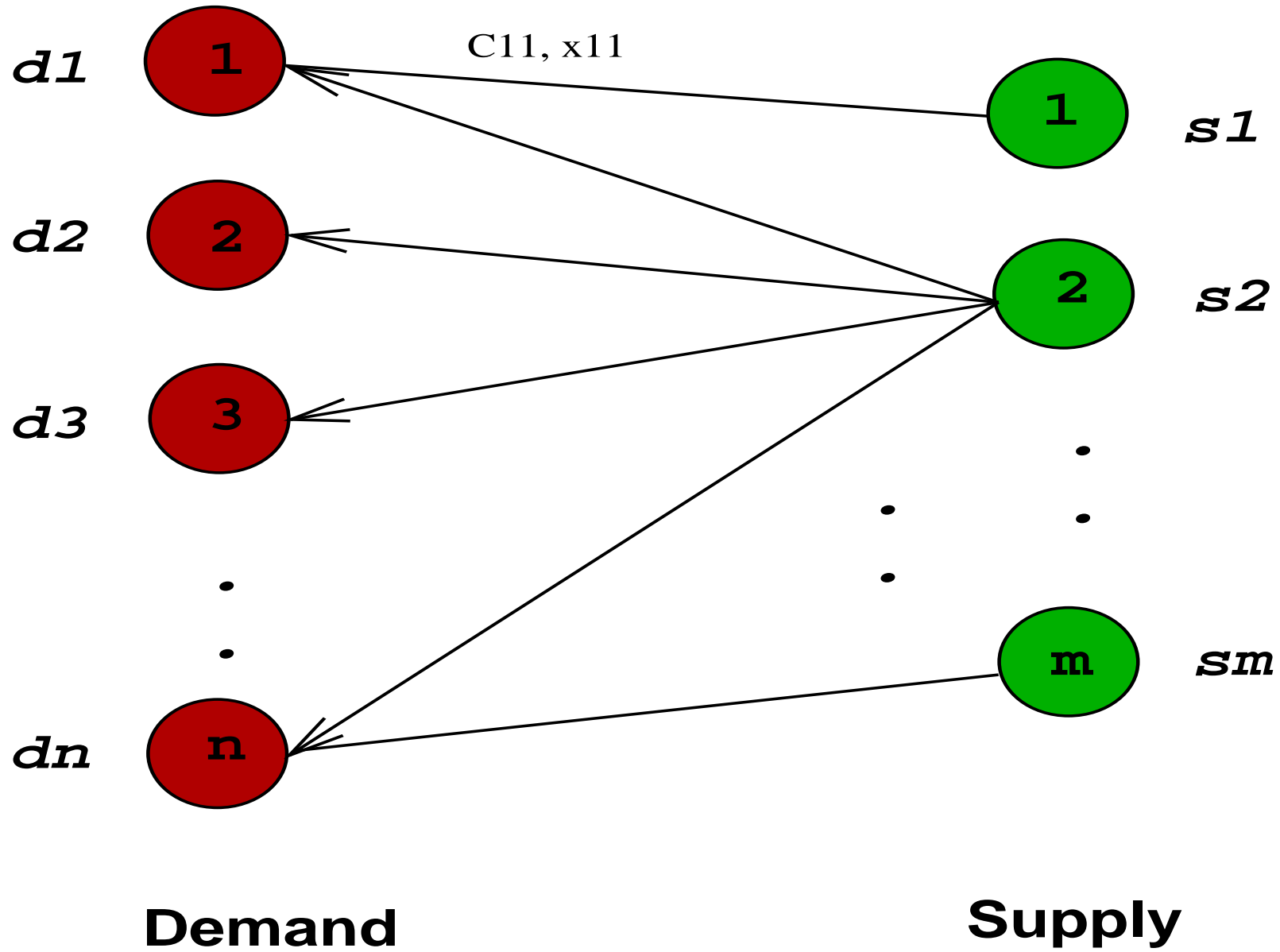


Figure 1: Supply chain network

Supply Chain Network

Warehouses	Retailers				Supply
	1	2	3	4	
1	12	13	4	6	500
2	6	4	10	11	700
3	10	9	12	4	800
Demand	400	900	200	500	20000

Transportation Simplex Method: Phase I

1. Start with the cell in the **northwest corner cell**
2. Allocate as many units as possible, consistent with the **available** supply and demand.
3. Move one cell to **right** if there is remaining supply; otherwise, move one cell **down**.
4. goto Step 2.

				500
				700
				800
400	900	200	500	

Construct the initial BFS

400				100
				700
				800
0	900	200	500	

Construct the initial BFS

400	100			0
				700
				800
0	800	200	500	

Construct the initial BFS

400	100			0
	700			0
				800
0	100	200	500	

Construct the initial BFS

400	100			0
	700			0
	100			700
0	0	200	500	

Construct the initial BFS

400	100			0
	700			0
	100	200		500
0	0	0	500	

Construct the initial BFS

400	100			0
	700			0
	100	200	500	0
0	0	0	0	

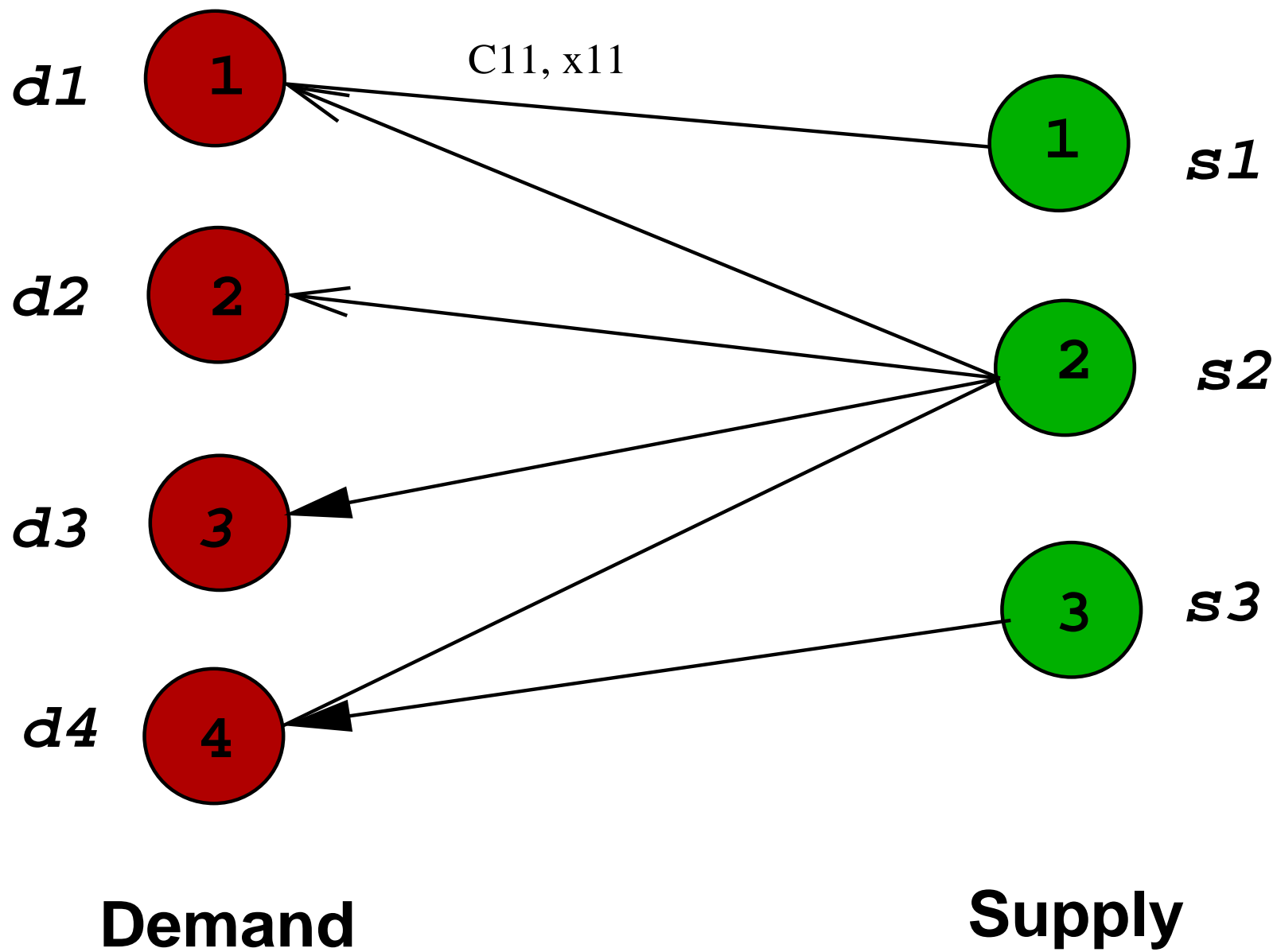


Figure 2: A BFS has a tree network structure

Transportation Simplex Method: Phase II

1. Determine the **shadow prices** for each supply side u_i and each demand side v_j such that

$$u_i + v_j = c_{ij}$$

for every **USED** cell (basic variable), that is, solving $A_B^T \mathbf{y} = \mathbf{c}_B$. One can always set $v_n = 0$.

2. Calculate the **reduced costs**

$$r_{ij} = c_{ij} - u_i - v_j$$

for the **UNUSED** cells (non-basic variable), that is, computing $\mathbf{r}_N = \mathbf{c}_N - A_N^T \mathbf{y}$. If the reduced cost for every unused cell is nonnegative, then STOP: declare **OPTIMAL**.

3. Select an unused cell with the **most negative** reduced cost. Using a **chain reaction cycle**, determine the **maximum number** of units (θ) that can be

allocated to the cell and adjust the allocation appropriately. Update the values of the **new set** of used cells (BFS).

4. Goto Step 1.

Determine the shadow prices

12	13			u_1
	4			u_2
	9	12	4	u_3
v_1	v_2	v_3	$v_4 = 0$	

Determine the shadow prices

12	13			u_1
	4			u_2
	9	12	4	$u_3 = 4$
v_1	v_2	v_3	$v_4 = 0$	

Determine the shadow prices

12	13			u_1
	4			u_2
	9	12	4	$u_3 = 4$
v_1	v_2	$v_3 = 8$	$v_4 = 0$	

Determine the shadow prices

12	13			u_1
	4			u_2
	9	12	4	$u_3 = 4$
v_1	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Determine the shadow prices

12	13			u_1
	4			$u_2 = -1$
	9	12	4	$u_3 = 4$
v_1	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Determine the shadow prices

12	13			$u_1 = 8$
	4			$u_2 = -1$
	9	12	4	$u_3 = 4$
v_1	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Determine the shadow prices

12	13			$u_1 = 8$
	4			$u_2 = -1$
	9	12	4	$u_3 = 4$
$v_1 = 4$	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Calculate reduced cost coefficients

12	13	4	6	$u_1 = 8$
6	4	10	11	$u_2 = -1$
10	9	12	4	$u_3 = 4$
$v_1 = 4$	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Calculate reduced cost coefficients

$12/0$	$13/0$	$4/ - 12$	$6/ - 2$	$u_1 = 8$
$6/3$	$4/0$	$10/3$	$11/12$	$u_2 = -1$
$10/2$	$9/0$	$12/0$	$4/0$	$u_3 = 4$
$v_1 = 4$	$v_2 = 5$	$v_3 = 8$	$v_4 = 0$	

Chain reaction cycle

400	100(-)	(+)		0
	700			0
	100(+)	200(-)	500	0
0	0	0	0	

$$\theta = 100.$$

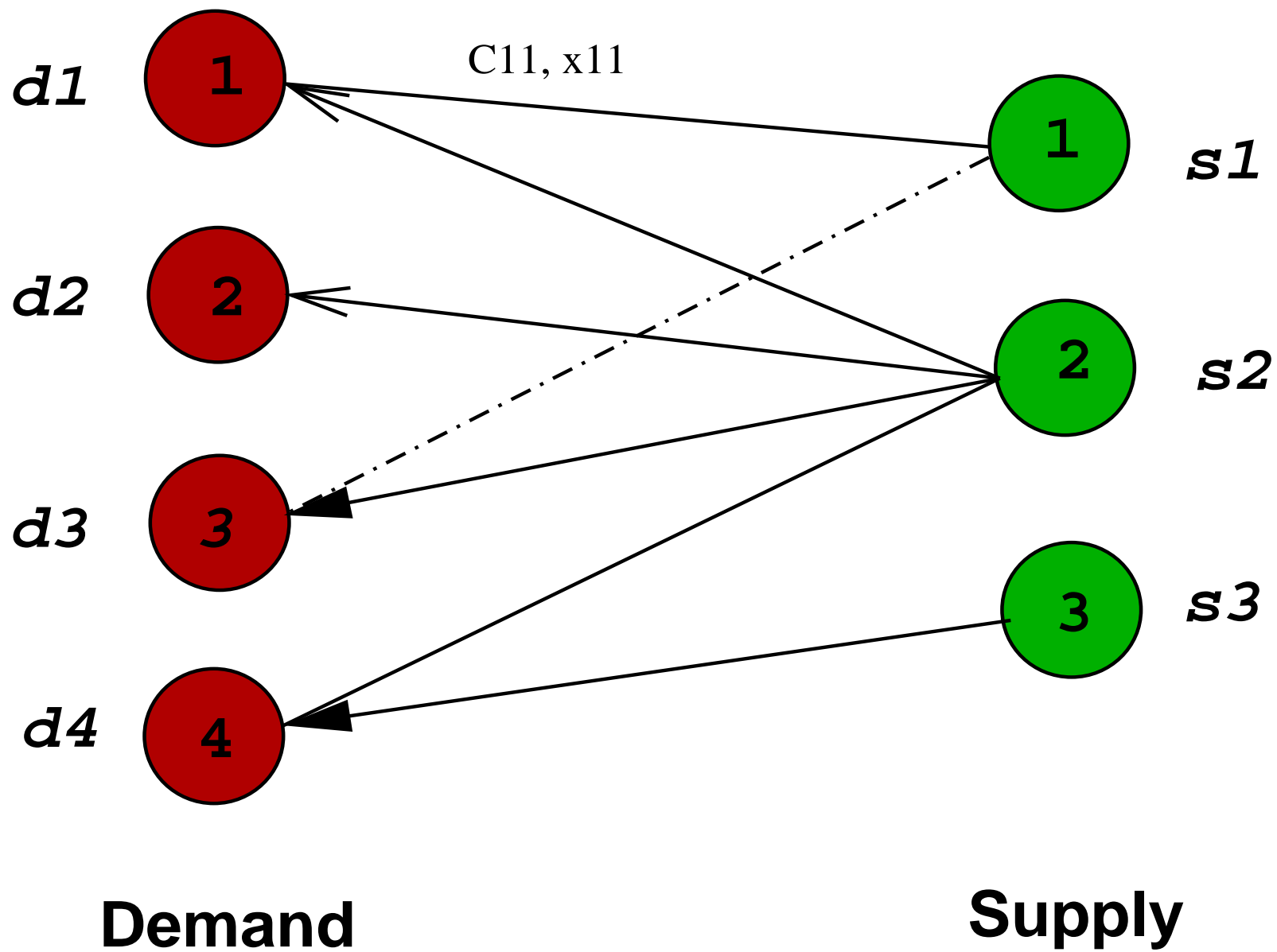


Figure 3: The newly selected route and the BFS tree will form a unique cycle

Update the new BFS

400		100		0
	700			0
	200	100	500	0
0	0	0	0	

The total supply cost is reduced by **\$1200** at the new BFS.

Two-Phase Simplex Method for General LP

How to determine a **starting feasible basis** for more general LP?

One technique is again constructing a so-called **Phase I Problem**, and uses the Simplex Method itself to solve the Phase I LP problem for which a starting BFS is known, and for which an optimal basic solution is a BFS for the original LP problem if it's feasible. If Phase I results in the discovery of a BFS for the original problem, then we can initiate **Phase II** wherein the Simplex Method is applied to the solving the original problem. The combination of Phases I and II gives rise to the **Two-Phase Simplex Method**. Since there are two different linear programs being solved in these phases, it is advantageous to have a "**smooth transition**" between them.

The Phase I Problem

it is not restrictive to assume that $\mathbf{b} \geq \mathbf{0}$ for this condition can be brought about by multiplying -1 to the both sides of an equation.

$$\begin{aligned} \text{(Phase I)} \quad & \text{minimize} \quad \mathbf{e}^T \mathbf{u} \\ & \text{subject to} \quad A\mathbf{x} + \mathbf{u} = \mathbf{b}, \\ & \quad \quad \quad (\mathbf{x}, \mathbf{u}) \geq \mathbf{0}, \end{aligned}$$

where \mathbf{e} is the vector of all ones.

The variables u_1, \dots, u_m of which \mathbf{u} is comprised are called **artificial variables**.

Infeasibility error function

$$w := \mathbf{e}^T \mathbf{u} = \mathbf{e}^T (\mathbf{b} - A\mathbf{x}) = \mathbf{e}^T \mathbf{b} - \mathbf{e}^T A\mathbf{x}.$$

- The Phase I Problem is in feasible canonical form with respect to the basis associated with the artificial variables.
- For feasible (\mathbf{x}, \mathbf{u}) , the value of w is nonnegative.
- The Simplex Algorithm is applicable to Phase I.
- The Phase I Problem has an optimal solution $(\mathbf{x}^*, \mathbf{u}^*)$, and (LP) has a feasible solution if and only if $w^* = 0$ (Fakar's lemma).

Sensitivity Analyses: Parametric LP

The the **right-hand-side** vector becomes $\mathbf{b} + \lambda\mathbf{d}$ or the **objective coefficient** vector becomes $\mathbf{c} + \lambda\mathbf{g}$, where the parameter λ belongs to an **interval**.

Denote this problem by $\text{LP}(\lambda)$:

$$\begin{aligned} \text{LP}(\lambda) \quad & \text{minimize} && (\mathbf{c} + \lambda\mathbf{g})^T \mathbf{x} \\ & \text{subject to} && A\mathbf{x} = \mathbf{b} + \lambda\mathbf{d}, \\ & && \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

Geometrical Observations

1. If \mathbf{b} is replaced by $\mathbf{b} + \lambda\mathbf{d}$, as λ varying the point $\mathbf{b} + \lambda\mathbf{d}$ moves away from \mathbf{b} in the direction \mathbf{d} (depending on the sign of λ). This raises the question of whether or not the point $\mathbf{b} + \lambda\mathbf{d}$ lies in the cone generated by A_B or even A ?
2. We know that for the function $\mathbf{c}^T \mathbf{x}$, the vector \mathbf{c} denotes the direction of steepest ascent. Thus, parameterizing the cost function according to the rule $\mathbf{c} + \lambda\mathbf{g}$ changes the gradient/slope, the normal direction of the objective hyperplane.

Getting Started

Let us consider λ around 0.

A key question in these parametric problems is: how much can the parameter λ be changed before the current **optimal basic solution** of LP(0) is lost?

Theorem 1 *The optimal basis of LP(0) remains optimal for LP(λ) if and only if*

$$A_B^{-1}(\mathbf{b} + \lambda\mathbf{d}) \geq \mathbf{0} \quad \text{and} \quad (\mathbf{c} + \lambda\mathbf{g}) - \mathbf{A}^T(\mathbf{A}_B^T)^{-1}(\mathbf{c} + \lambda\mathbf{g})_B \geq \mathbf{0}.$$

This will establish an interval on λ in which the optimal basis of LP(0) remains optimal.

Ceteris Paribus Sensitivity Analysis: Right-Hand-Side

The problem before us is to find (for each $i = 1, \dots, m$) the **range of values** of the scalar λ for which the basis B remains **optimal** for the new RHS $\mathbf{b} + \lambda \mathbf{e}_i$, where \mathbf{e}_i is the vector of all zero except 1 in the i th position.

B remains optimal if

$$\mathbf{0} \leq A_B^{-1}(\mathbf{b} + \lambda \mathbf{e}_i) = \bar{\mathbf{b}} + \lambda(A_B^{-1} \mathbf{e}_i).$$

Then the **new optimal objective value** is changed from the old one by $\lambda \cdot y_i^*$ where \mathbf{y}^* is the optimal dual solution of LP(0):

$$\mathbf{c}_B^T A_B^{-1}(\mathbf{b} + \lambda \mathbf{e}_i) = (\mathbf{y}^*)^T (\mathbf{b} + \lambda \mathbf{e}_i) = (\mathbf{y}^*)^T \mathbf{b} + \lambda \cdot (\mathbf{y}^*)^T \mathbf{e}_i = (\mathbf{y}^*)^T \mathbf{b} + \lambda \cdot y_i^*.$$

Ceteris Paribus Sensitivity Analysis: Objective Coefficient

The problem before us is to find (for each $j = 1, \dots, n$) the **range of values** of the scalar λ for which the basis B remains **optimal** for the new $\mathbf{c} + \lambda \mathbf{e}_j$, where \mathbf{e}_j is the vector of all zero except 1 in the j th position.

B remains optimal if

$$\mathbf{0} \leq (\mathbf{c} + \lambda \mathbf{e}_j)_N - A_N^T (A_B^T)^{-1} (\mathbf{c} + \lambda \mathbf{e}_j)_B = \begin{cases} \mathbf{r}_N - \lambda (\mathbf{e}_j^T \bar{A})_N^T \geq \mathbf{0} & \text{if } j \in B \\ \mathbf{r}_N + \lambda \mathbf{e}_j \geq \mathbf{0} & \text{otherwise} \end{cases} .$$

Then the **new optimal objective value** is changed from the old one by $\lambda \cdot x_j^*$ where \mathbf{x}^* is the optimal primal solution of LP(0):

$$\begin{aligned} (\mathbf{c} + \lambda \mathbf{e}_j)_B^T A_B^{-1} \mathbf{b} &= (\mathbf{c} + \lambda \mathbf{e}_j)_B^T \mathbf{x}_B^* \\ &= (\mathbf{c} + \lambda \mathbf{e}_j)^T \mathbf{x}^* = \mathbf{c}_B^T \mathbf{x}_B^* + \lambda \mathbf{e}_j^T \mathbf{x}^* = \mathbf{c}_B^T \mathbf{x}_B^* + \lambda \cdot x_j^* . \end{aligned}$$

Where to find information in the (production problem) tableau

We start with a form below with $\mathbf{b} \geq \mathbf{0}$ and basis I .

$$\left[\begin{array}{ccc|c} -\mathbf{c}^T & \mathbf{0}^T & 0 & \\ \hline A & I & \mathbf{b} & \end{array} \right]$$

When the simplex method **terminates**:

$$\left[\begin{array}{ccc|c} \mathbf{r}^T & -\mathbf{y}^T & -\mathbf{y}^T \mathbf{b} & \\ \hline A_B^{-1} A & A_B^{-1} & A_B^{-1} \mathbf{b} & \end{array} \right]$$

where A_B is an optimal **basis** selected from matrix (A, I) , and I is replaced by A_B^{-1} at the final tableau.

Properties of the dual (shadow) prices y

- All **inactive** constraint have **zero dual price** from complementarity.
- In general, the dual price on a given active constraint is the **rate of change** in the OV as the **RHS** of the constraint **increases**, ceteris paribus.
- The constraint RHS **ranges** give the ranges of the constraint RHS over which no change in the **optimal basis** will occur.
- One of the **allowable increase and decrease** for an **inactive** constraint is infinite and the other equals to the slack or surplus.
- In general, when the RHS of an **active** constraint changes, both the OV and OS will change.

Properties of the dual slacks (reduced costs) r ?

- All **basic** variables have **zero dual slack value** from complementarity.
- In general, the dual slack value of any **non-basic** variable is the amount the objective coefficient of that variable would have to change, with all other data held fixed, in order for it to become a **basic** variable at optimality.
- If the OS is **degenerate**, the objective coefficient of a **non-basic** variable would have to change by at least the slack value in order to become a **basic** variable.
- The objective coefficient **ranges** give the ranges of the objective function over which no change in the **optimal basis** will occur.
- One of the **allowable increase and decrease** for a **non-basic** variable is infinite and the other is the slack value.
- If a **non-basic** variable has **zero** slack value, then there may exist an **alternative** optimal solution.

A Production Example

$$\begin{array}{llllllll} \text{max} & 8x_1 & +14x_2 & +30x_3 & +50x_4 & & & \\ \text{subject to} & x_1 & +2x_2 & +10x_3 & +16x_4 & +x_5 & & = 800 \\ & 1.5x_1 & +2x_2 & +4x_3 & +5x_4 & & +x_6 & = 1000 \\ & 0.5x_1 & +0.6x_2 & +x_3 & +2x_4 & & +x_7 & = 340 \\ & \mathbf{x} & & & & & & \geq \mathbf{0}. \end{array}$$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	1
B	-8	-14	-30	-50	0	0	0	0
5	1	2	10	16	1	0	0	800
6	1.5	2	4	5	0	1	0	1000
7	0.5	0.6	1	2	0	0	1	340

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	1
B	0	0	28	40	5	2	0	6000
2	0	1	11	19	1.5	-1	0	200
1	1	0	-12	-22	-2	2	0	400
7	0	0	-4	1.6	0.1	-0.4	1	20

Resolving Cycling in the Simplex Algorithm

In a system of rank m , a (basic) solution that uses fewer than m columns to represent the right-hand side vector is said to be **degenerate**. Otherwise, it is called **nondegenerate**.

A basic feasible solution will be nondegenerate if and only if its m **basic variables are positive**.

Why is degeneracy a problem? The Simplex Algorithm can **cycling** (an infinite repetition of a finite sequence of bases) when a degenerate basic feasible solution crops up in the course of executing the algorithm, unless a suitable rule is employed to break the ties. Fortunately, there are rules to overcome this problem.

Cycling Example

$$\begin{array}{ll}
 \min & -2x_1 - 3x_2 + x_3 + 12x_4 \\
 \text{s.t.} & -2x_1 - 9x_2 + x_3 + 9x_4 + x_5 = 0 \\
 & \frac{1}{3}x_1 + x_2 - \frac{1}{3}x_3 - 2x_4 + x_6 = 0 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
 \end{array}$$

Initially, the basic variables are $\{x_5, x_6\}$ and it is in the canonical form. The pivot sequence shown in the table below leads back to the original system after 6 pivots.

Pivot number	1	2	3	4	5	6
Basic var. out	x_6	x_5	x_2	x_1	x_4	x_3
Basic var. in	x_2	x_1	x_4	x_3	x_6	x_5

Methods for Resolving Cycling

There are several methods for resolving degeneracy in linear programming.

Among these are:

1. Perturbation of the **right-hand side**.
2. **Lexicographic ordering**.
3. Application of **Bland's pivot** selection rule.

Bland's Rule

It is a **double least-index rule** consisting of the following two parts:

- (i) Among all candidates for the entering column (i.e., those with $r_j < 0$), choose the one with the **smallest index**, say e .
- (ii) Among all rows i for which the minimum ratio test results in a tie, choose the row r for which the corresponding basic variable has the **smallest index**, j_r .

Theorem 2 Under *Bland's pivot selection rule*, the Simplex Algorithm cannot *cycle*.

Sketch of Proof

We think of the initial data as being expressed in a tableau of $m + 1$ rows and $n + 2$ columns (indexed from 0 to $n + 1$) which we write as

$$\mathcal{A} = \begin{bmatrix} 1 & \mathbf{c}^T & 0 \\ 0 & A & \mathbf{b} \end{bmatrix}.$$

One row (the 0^{th} , $\mathcal{A}_{0\cdot}$) and column (the 0^{th} , $\mathcal{A}_{\cdot 0}$) pertain the variable x_0 we wish to optimize. The column indexed by $n + 1$ is the right-hand side of the system of equations (augmented by an equation for the objective function).

Let $\bar{\mathcal{A}}$ denote the first $n + 1$ columns of \mathcal{A} , i.e., with column $\mathcal{A}_{\cdot n+1}$ deleted. Analogous notations will be used for pivotal transforms of $\bar{\mathcal{A}}$.

Now if cycling occurs, there is a set τ of indices $j \in \{1, \dots, n\}$ such that x_j becomes basic during cycling. Clearly τ has only a finite number of elements, so it has a largest element which we denote by q . Let \mathcal{A}' denote the tableau that first

specifies q as the pivot column. This means that x_q becomes a basic variable in the next tableau.

Let $\mathbf{y} = (y_0, y_1, \dots, y_n) = \bar{\mathcal{A}}'_0$. By virtue of the definition of q and the rule that results in the choice of q , we have

$$y_0 = 1, \quad y_j \geq 0 \quad 1 \leq j < q, \quad y_q < 0. \quad (1)$$

Note that the $(n + 1)$ -vector \mathbf{y} belongs to the row space of $\bar{\mathcal{A}}$. Now x_q must also leave the basis at some tableau \mathcal{A}'' . Let $x_q = x_{j_r}$, and let t denote the pivot column when x_q becomes nonbasic. Define the $(n + 1)$ -vector $\mathbf{v} = (v_0, v_1, \dots, v_n)$ as follows:

$$v_{j_i} = \bar{a}''_{it} \quad i = 0, 1, \dots, m, \quad v_t = -1, \quad v_j = 0 \quad \text{else.} \quad (2)$$

Note that $v_0 = v_{j_0} = \bar{a}''_{0t} < 0$, $v_q = \bar{a}''_{rt} > 0$, and \mathbf{v} is in the null space of $\bar{\mathcal{A}}$. Thus, $\mathbf{y} \cdot \mathbf{v} = 0$, and by construction $y_0 v_0 < 0$. Hence $y_j v_j > 0$ for some $j \geq 1$. Since $y_j \neq 0$, x_j must be nonbasic in \mathcal{A}' ; since $v_j \neq 0$, then either x_j is basic in \mathcal{A}'' or else $j = t$. Accordingly, $j \in \tau$, and hence $j \leq q$. By the

construction again, $y_q < 0 < v_q$ which implies that $y_q v_q < 0$ hence $j < q$.

Furthermore, (1) implies that $y_j > 0$, so $v_j > 0$. Next we observe that $v_t = -1$ implies $j \neq t$. All these lead to the conclusion that x_j is currently basic in \mathcal{A}'' .

Let $j = j_p$ for some p . Then $v_j = \bar{a}_{pt}'' > 0$.

Note that during the cycling the right-hand-side vector $\bar{\mathbf{b}}$ does not change and the values of all variables in τ are fixed at 0. This implies $\bar{b}_p'' = 0$. We have established that $j = j_p$, $\bar{a}_{pt}'' > 0$ and $\bar{b}_p'' = 0$. But this contradicts the assumption that x_q is removed from the basic set corresponding to tableau \mathcal{A}'' , since $j < q$ and by Bland's rule j should be removed. This means that cycling cannot occur when Bland's Rule is applied.

Remark. This elegant degeneracy resolution rule has the drawback that it may result in pivot choices that do not *significantly* improve the objective function value. It may also force the selection of dangerously small pivot elements.

The Dual Simplex Method^a

In the process of the **Simplex Method**:

1. The vector r_N is nonnegative.
2. The \bar{b} might **not** be nonnegative.

That is: the **dual vector** y is feasible for the dual, but the basic solution for the primal is **infeasible**.

Methodological Philosophy

Recall that the **Primal** Simplex Algorithm maintains the **primal feasibility and complementary** slackness conditions while working toward **dual feasibility**. By contrast, the **Dual** Simplex Algorithm maintains **dual feasibility and complementary** slackness while working toward **primal feasibility**. In a sense, it is the primal Simplex Algorithm applied to the dual problem, but carried out in the format of the primal problem.

Test for Termination

The algorithm works with **pivot** steps like those of the Primal Simplex Algorithm but uses different criteria for **pivot selection and termination**.

Once the problem is put into the canonical form, the method checks whether it is time to terminate. This will be the case if either of the following conditions is met by the current system:

1. $\bar{b} \geq 0$;
2. $b_o < 0$ and $\bar{A}_{oj} \geq 0$ for all $j \in \{1, \dots, n\}$.

In the first case, the current basic solution is **primal feasible**. In the second case, the **infeasibility of the primal** is revealed.

Outgoing and Entering Variables

If neither of these conditions obtains, we have $\bar{b}_o < 0$ and $\bar{A}_{oj} < 0$ for some j .

One of these **negative** numbers a_{rj} will be chosen as the pivot element.

Let x_o , $o \in B$ be the **outgoing** variable.

Under the rules of the algorithm, it is necessary to maintain the dual feasibility (nonnegativity of the coefficients in the objective function row). This is accomplished by choosing the pivot column e according to the rule

$$e \in \operatorname{argmin}_{j \in N} \left\{ \frac{r_j}{-\bar{A}_{oj}} : \bar{A}_{oj} < 0 \right\}.$$

Then, x_e is the **entering** variable.