1. LINEAR PROGRAMMING

BASICS

WHY?

1) Solves any linear program (it detects redundant constraints in the problem formulation; it identifies instances when the objective value is unbounded over the feasible region; and it solve s problems with one or more optimal solutions. The method is also self-initiating. It uses itself either to generate an appropriate feasible solution, as required, to start the method, or to show that the problem has no feasible solution)

2) Provides much more than just optimal solutions. As byproducts, it indicates how the optimal solution varies as a function of the problem data (cost coefficients, constraint coefficients, and right-hand-side data).

MODELLING

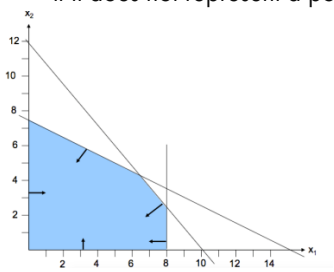
Decision variables (they value is not fixed, non-negative in canonical form)

Basic variables ($j-th\ basic\ variable$)	Non-basic variables
= linear algebra concepts In canonical form, BV = to right hand side	= linear algebra concepts

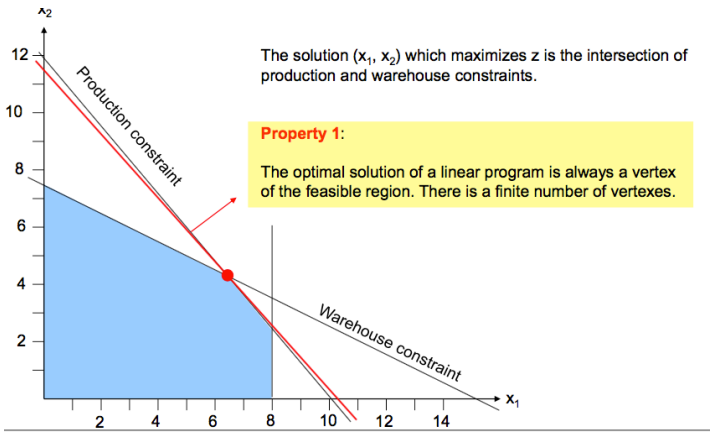
Constraints (production, demand, etc.)
Objective function (max, min)

GRAPHIC SOLUTION AND FEASIBLE REGION

- The optimal solution of a linear program is always a vertex (=top point) of the feasible region.
- There are a finite number of vertexes.
- The optimal solution is the intersection of two binding constrains. Solution is never optimal (production plan etc. if it does not represent a point in the solution space).



$$6x_1 + 5x_2 \le 60 \quad (2.2)$$
$$10x_1 + 20x_2 \le 150 \quad (2.3)$$
$$1x_1 \le 8 \quad (2.4)$$
$$1x_1 \ge 0 \quad (2.5)$$
$$1x_2 \ge 0 \quad (2.6)$$



1.1. SIMPLEX ALGORITHM: BASICS

Basic feasible solution
(Primal LP – vertical)

Unboundedness Criterion - Suppose that, in a maximization problem, some NBV has a positive coefficient in the objective function of a canonical form. If that variable has negative or zero coefficients in all constraints, then the objective function is unbounded from above over the feasible region. (OF line can be moved parallel to itself infinitely)

Optimal
(Primal LP – horizontal)

Optimality Criterion - Suppose that, in a maximization problem, every NBV has a no positive coefficient in the objective function of a canonical form. Then the basic feasible solution given by the canonical form maximizes the objective function over the feasible region.

- Set all non-basic variables to 0.
- The value of each basic variable equals the right-hand-side (the value right of the equal sign) where the basic variable has coefficient 1.
- The objective function value equals its constant.

- The solution is optimal if all non-basic variables have a negative or zero coefficient in the objective function.
- Note that this condition is for maximization problems.

Improvement Criterion - Suppose that, in a maximization problem, some NBV variable has a positive coefficient in the objective function of a canonical form. If that variable has a positive coefficient in some constraint, then a new basic feasible solution may be obtained by pivoting.

1.2. CANNONICAL FORM

VARIABLES

(1) Decision variables (DV) are constrained to be nonnegative.

(2) For each constraint, we have a decision variable with coefficient 1 in that constraint, but coefficients 0 in all other constraints and in the objective function.
   - These are basic variables (BV). All other variables are non-basic variables (NBV).
   - The number of BV equals the number of constraints (excluding non-negativity-constraints).
   - The set of BV is called basis.

CONSTRAINTS

(3) All constraints, except non-negativity constraints of decision variables, are stated as equalities.

(4) The right-hand-side of each constraint is nonnegative.

PIVOTING

By pivoting we are moving from one basic feasible solution to a neighbor basic feasible solution, both stated in a canonical form of the LP.

(1) Selecting the NBV which enters the basis (pivot column)
(2) Selecting the constraint which sets the value of the entering variable and thus selecting the BV which leaves the basis (pivot row)
(3) Transforming the pivot row such that the entering variable has a coefficient 1
(4) Retransforming the other constraints and the objective function value to canonical form

OF interpretation after simplex:

1. If all OF values are $= 0$ → no production capacity left
2. If OF values are < 0 → if we have one additional unit of labor/energy, we can produce more
3. If OF values > 0 → we don't produce this item at all (should continue with simplex, solution is not yet optimal).

FORMAL PRESENTATION OF THE SIMPLEX ALGORITHM (SEE TABLE IN ATTACHMENT)

TRANSFORMATION OF LP IN CANONICAL FORM

01. Slack and surplus variables ($\geq, \leq$)

For $\geq$ inequalities, let the nonnegative surplus variable (x_6) represent the amount by which the left-hand side exceeds the right-hand side.

*the amount in the excess of minimum requirement

Example	$40x_1 + 10x_2 + 6x_3 \geq 32.5$
Steps	We need to minus one surplus variable and because of that we now should add artificial variable for M: $40x_1 + 10x_2 + 6x_3 - x_4 + 1x_6 = 32.5$ Change in OF: $40x_1 + 10x_2 + 6x_3 - M * x_6 = 32.5$

For $\leq$ inequalities, let the nonnegative slack variable (x_5) represent the amount by which the right-hand side exceeds the left-hand side.

*the volume of capacity that isn't used, but can be added

Example	$40x_1 + 10x_2 + 6x_3 \leq 32.5$
Steps	Add slack variable to obtain equality in each constraint: $40x_1 + 10x_2 + 6x_3 + x_5 = 32.5$

Usually slack variables are basic variables at the beginning of transforming LP (start point: no production, no profit, etc.). We need to transform LP to get in the basis only decisions variables → maximize profit, minimize costs.

02. Free variables ($x_1 \in \mathbb{R}$)

Replace each DV unconstrained in sign by a difference between two nonnegative variables.
This replacement applies to all equations including the objective function.

*when we don't have a non-negativity constraint and we only state $x_1 \in \mathbb{R}$

Example	$x_1 \in \mathbb{R}$ $\Delta x_1 = x_1^+ - x_2^-$
Steps	Use $I_t = I_t^+ - I_t^-$ The variable I_t^+ represents positive inventory on hand and I_t^- represents backorders (i.e., unfilled demand). Whenever $I_t \geq 0, we\ set\ I_t^+ = I_t\ and\ I_t^- = 0$ and when $I_t < 0, we\ set\ I_t^+ = 0\ and\ I_t^- = -I_t$ For instance, can occur if we have negative change of inventory.

03. Artificial variables, "Big M Method" ($\geq, =$)

*added to place the linear program in canonical form

Example	1) $1x_1 + 2x_2 = 20$ 2) $1x_1 + 2x_2 \geq 20$
Steps	1. Create two types of equations: $\geq$ and $\leq$ then solve them separately. $1x_1 + 2x_2 \leq 20 \rightarrow add\ slack\ variable$ $1x_1 + 2x_2 \geq 20 \rightarrow minus\ surplus\ variable\ and\ add\ artificial\ (Big\ M)$ 2. We should solve it with surplus variable (use big M method) max z = ... - My_1 $1x_1 + 2x_2 - 1x_3 + 1x_4 = 20$ $\max z = \cdots M * 1x_4$

Artificial VS Slack variables

Slack variables have meaning in the problem formulation, artificial variables have no significance; they are merely a mathematical convenience useful for initiating the simplex algorithm.
Slacks are (implicitly) part of the original problem.

Negative right-hand side

Example	$1x_1 + 2x_2 \geq -20$
Steps	Multiply by (-1) → change coefficient signs and rotate constraint sign and go further with necessary transformation. $-1x_1 - 2x_2 \leq 20$ $-1x_1 - 2x_2 + 1x_3 = 20$

Minimization problem

Example	$\min z = 1x_1 - 0x_2 + 3x_3$
Steps	Multiply objective function by (-1) $\max z = -1x_1 + 0x_2 - 3x_3$

If the optimality condition is not met

Example	$\max z = 0x_1 + 0x_2 - 3x_3 + 1x_4 + 20$
Steps	Change the basis (here: x4 enters and x2 leaves the basis). A change of the basis takes place if we move from one basic feasible solution to another basic feasible solution by removing one former BV from the basis and introducing one former NBV to the basis → Simplex Algorithm.

1.3. "BIG M-METHOD"

We use the 'Big M method' to obtain a feasible basic solution where the basic variables with M-coefficient are zero and can be deleted.

DRAWBACKS

- We don't know a priory how large M must be for a given problem, to make sure that all artificial variables are driven to zero.
- Using large numbers for M may lead to numerical difficulties on a computer. Alternative method - phase I–phase II procedure.

Basic idea:

1) We add M in OF, to do so – we add artificial variables in the necessary constraint and add M in OF multiplied by this artificial variable.
   For max problem max z =...−M*artificial variable
   For min problem min z =...+M*artificial variable
2) Afterwards we perform is simple algebraic transformation to get rid of M. Often we add to the OF row, the row with artificial variable multiplied on M. When we have no M rows left (M≤0), we can eliminate the row
3) After we solve the LP with simplex.

*If in final solution all artificial variables = 0 is feasible for the original problem, those with artificial variable > 0 are not feasible. The artificial variable should be driven to zero.

**If artificial variable >0 in the final tableau, then there is no solution to the original problem where the artificial variables have been removed; the problem is infeasible.

1.4. DUAL SIMPLEX METHOD

Condition:

(1) LP is dual feasible (=primal optimal, $\forall j: \overline{c_j} \leq 0$ – if there is at least one OF coefficient is greater than 0, DS is not allowed)
AND
(2) dual non-optimal (=primal infeasible, $\exists i: \overline{b_t} < 0$).

FORMAL PRESENTATION OF THE DUAL SIMPLEX METHOD (SEE TABLES IN ATTACHMENT)

1.5. SIMPLEX ALGORITHM: SPECIAL CASES

RHS - No feasible solution (2.8.1.)	Repeat - Redundant constraint	Cij and aij - Unbounded LP (unrestricted solution space)	BV – OF - Primal Degeneracy (basis variable has the 0 value)	NBV – OF - Dual degeneracy – (multiple optimal solution)
Look at right-hand side (negative values) If any artificial variable is positive in the optimal Big M tableau, the original LP has no feasible solution.	Look if constraints repeat each other The slack variable of the redundant constraint is always positive and thus BV.	Look at OF values and a_s 1) $c_j > 0$ while all corresponding 2) $a_s \leq 0$	In optimal tableau BV has $c_j = 0$, $b_{ij} = 0$. In general, for BV with value 0, we have primal degeneracy.	In optimal tableau NBV, $c_j = 0$, and then can therefore be moved into the basis without changing the objective function value.
Effects on dual simplex method: Dual simplex stops with the proof that no feasible solution exists. Therefore, the simplex is not performed.	Effects on dual simplex method: No influence on simplex algorithm.	Effects on simplex method: No pivot steps can be performed, since all $a_s < 0$ (If PLP is unbounded then DLP is infeasible)	Effects on dual simplex method: Performing basic steps of simplex without changing objective function.	Effects on dual simplex method: Simplex stops upon reaching the first optimal solution. All convex combinations between the two optimal solutions are optimal.

DUALITY

Why using the duality

(1) The number of iterations of the simplex to find an optimal solution = about 1.5 to 2 times the number of NBV of the linear program to be solved.
(2) The number of variables of the problem corresponds to the number of NBV of the dual problem.
(3) Optimal values of the OF in the primal and dual solutions are equal (strong duality property), i.e. It does not matter which of the two problems is solved.
(4) It is usually more efficient to solve the dual problem with the simplex if:
    - Number of NBV of the dual problem is less than the number of NBV of the primary problem.

The objective function value is:

Primal perspective: contributions of products (revenues)
Dual perspective: contributions of resources (shadow prices).

DUALITY PROPERTIES

(!) Weak Duality Property:
x of z(x) = feasible, but not optimal (primal problem)
y of v(y) = feasible but not optimal (dual problem)
⇨ $v(y) > z(x)$

(!) Strong Duality Property:
x* of z(x) = feasible and optimal (primal problem)
y* of v(y*) = feasible and optimal (dual problem)
⇨ $v(y*) = z(x*)$

(!) Unboundedness Property:
If the primal problem has an unbounded solution ($c_{ij} > 0, while\ a_{is} \leq 0$),
⇨ dual problem is infeasible ($c_{ij} > 0$).

(!) Complementary Slackness Property:
x* be a feasible and optimal solution of the primal problem
y* be a feasible and optimal solution of the dual problem, the following holds:

- BV of primal LP → Slack variable of associated constraint of the dual LP = 0
  $if\ x_j^* > 0, then\ \sum_{i=1}^{m} a_{ij} * y_i^* = c_j$
- NBV of primal LP → Slack variable of associated constraint of the dual LP ≥ 0
  $if\ x_j^* = 0, then\ \sum_{i=1}^{m} a_{ij} * y_i^* \geq c_j$

*Primal feasible: $b_{ij} \geq 0$	*Dual optimal: $b_{ij} \geq 0$
*Primal optimal: $c_{ij} \leq 0$	*Dual feasible: $c_{ij} \leq 0$

CORRESPONDENCE OF DUAL AND PRIMAL LP:

(!) In the Optimal tableau of the PLP the reduced costs of the slack variables (= shadow prices) corresponds to the optimal values of the decision variables of the dual.
(!) If we have a minimization problem in PLP we can change it to max in DLP directly.

OBTAINING DUAL LP

Rules for Obtaining the Dual LP

Primal LP

$$\max z = \sum_{j=1}^{n} c_j \cdot x_j$$

$$s.t. \quad \sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i \; (i = 1, 2, \dots, m)$$

$$x_j \geq 0 \; (j = 1, 2, \dots, n)$$

Dual LP

$$\min v = \sum_{i=1}^{m} b_i \cdot y_i$$

$$s.t. \quad \sum_{i=1}^{m} a_{ij} \cdot y_i \geq c_j \; (j = 1, 2, \dots, n)$$

$$y_i \geq 0 \; (i = 1, 2, \dots, m)$$

Correspondence between Primal and Dual Problem

Maximizing	Minimizing
i-th constraint ≤	i-th variable ≥ 0
i-th constraint ≥	i-th variable ≤ 0
i-th constraint =	i-th variable is real
j-th variable ≥ 0	j-th constraint ≥
j-th variable ≤ 0	j-th constraint ≤
j-th variable is real	j-th constraint =

1.6. SENSITIVITY ANALYSES: SHADOW PRICES AND REDUCED COSTS

WHY

- Data often is stochastic.
- Sensitivity analysis explores how the change of one data impacts the optimal solution.

ASSUMPTION

We always change <u>one data at a time</u>, leaving all other data as it is (ceteris paribus)
Changing the OF coefficient of a <u>BV</u> (=shadow price)
Changing the OF coefficient of a <u>NBV</u> (=reduced costs)

How much can we change the OF coefficient of a BV without changing the basis?
Note: the change of the coefficient will lead to a new optimal objective function value. Also, the value of the basis variables changes. However, the basis does not change.

SHADOW PRICE

<u>Definition</u>: The shadow prices associated with a particular constraint is the change in the optimal value of the objective function per unit increase in the right-hand side value for that constraint, all other problem data remaining unchanged.

Change of c_{ij} value of BV (slack variable) from initial to optimal tableau. The change = the shadow price.

$\sum_{i=1}^{m} a_{ij} \cdot y_i = opportunity\ costs\ of\ OF\ value$

Ausgangstableau

BV	Wert	x_1	x_2	x_3	x_4	
x_4	7	1	1	2	1	
x_5	10	2	1	3		
-z	0	8	6	5	0	

Optimaltableau

BV	Wert	x_1	x_2	x_3	x_4	
x_2	4		1	1	2	
x_1	3	1		1	-1	
-z	-48	0	0	-9	-4	

POSITIVE OR NEGATIVE SHADOW PRICE:

For an increase of the RHS by 1 the objective function changes by the absolute value of the coefficient in the objective function of the associated slack variable. The direction of the change is according to the following table:

Constraint	Objective function	
	Max	Min
≤	+	-
≥	-	+

REDUCED COSTS

<u>Definition</u>: The reduced cost associated with the non-negativity constraint for each decision variable is the shadow price of that constraint (i.e. the corresponding change in the objective function per unit increase in the lower bound of the variable).

$$Reduced\ costs$$
$$= contributed\ margin(objective\ function\ coefficien)$$
$$- opportunity\ costs$$
$$\bar{c_j} = c_j - \sum_{i=1}^{m} a_{ij} * y_i$$

Ausgangstableau

BV	Wert	x_1	x_2	x_3	x_4	x_5
x_4	7	1	1	2	1	
x_5	10	2	1	3		1
-z	0	8	6	5	0	0

Optimaltableau

BV	Wert	x_1	x_2	x_3	x_4	x_5
x_2	4		1	1	2	-1
x_1	3	1		1	-1	1
-z	-48	0	0	-9	-4	-2

y_i – shadow price of constraint i
c_j – objective function coefficient of variable j
a_{ij} – coefficient of variable j in constraint i

⇨ if positive, then worth introducing the product.

(!) For an optimization problem reduced costs of BV = 0, and reduced costs of NBV ≤ 0.
Note: With the help of the reduced costs, you can only decide about the inclusion of one product in a production program.

1.7. SENSATIVITY ANALYSES: VARIATION OF THE OF COEFFICIENT

What is the contribution margin of x_3 so that it is in the basis?

(1) Add delta to the contribution margin (=c_j of the associated variable);
(2) Perform simplex;
(3) From optimal tableau calculate the value of delta.
(4) Sum up values of delta from optimal and from initial tableau = contribution margin

2. INTEGER AND MIXED-INTEGER PROGRAMS (MIP)

INTEGER PROGRAM / MIXED-INTEGER PROGRAM / BINARY PROGRAM

Definition: An integer program is a linear program where all variables are integer.

Integer program	Mixed-integer program	Binary program
An integer program is a linear program where all variables are integer.	A mixed-integer program is a linear program with integer and continuous variables.	A binary program is a linear program where all variables are binary.

LP-RELAXATION

For any IP, we can generate an LP (LP relaxation) from the IP by taking the same objective function and same constraints but with the requirement that variables are integer replaced by appropriate continuous constraints.

If we have variables taking fractional values at the LP optimal solution, then we can round these to the nearest integer value. If we do this then:

⇨ this may lead to certain constraints being violated (i.e. we have an infeasible solution) - this may, or may not, be important.

Optimal Solution of the LP-Relaxation and the Integer Program

- The optimal solution of the LP-relaxation is not integer.
- Rounding up does lead to an infeasible solution.
- Rounding down gives a feasible but not an optimal solution.

- OF value of the optimal LP-relaxation ≥ OF value of the optimal integer solution.
- For a max problem, the optimal OF value of the LP-relaxation is always ≥ to the optimal objective function value of the IP.

$$max: z^*_{LP} \leq z^*_{LP-relax}$$
$$min: z^*_{LP} \geq z^*_{LP-relax}$$

2.1. BRANCH-AND-BOUND WITH SIMPLEX

General procedure and components of B&B:

- K - A candidate list K contains all sorted problems which have to be investigated;
- P_0 - At the beginning, only P0 is in K;
- Lower bound $\underline{z}$: The objective function value of a feasible solution of P_0;
- Upper bound $\overline{z}$: The objective function of the LP-relaxation of P_0

Cases for Eliminating a Subproblem

1. Integer solution
2. $z(LP - relaxation) \leq \underline{z}$
3. No feasible solution
4. $z(LP - relaxation) > \underline{z}$

For cases (1) - (3) => stop

For case (4) => divide the problem into 2 new subproblems and add these to the candidate list

Creating New Subproblems for Case 4

In case of a non-integer variable (choice arbitrarily) we create two new subproblems emanating from the problem at hand by introducing one additional constraint for each sub problem.

- Rule: The "**left**" subproblem is created by the ≤-constraint and it receives the smaller number.

Sorting of Subproblems in the Candidate List

Many ways (priority rules), two common ones are:

(1) LIFO – Last in first out: B&B tree grows vertically, „depth-first search ".
FIFO – first-in-first out

(2) MUB – Maximum upper bound: B&B tree grows horizontally, „breadth-first search ".
MLB – minimum lower bound

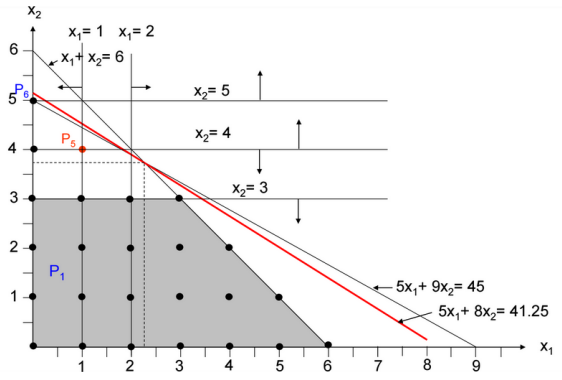
In case the priority rule cannot decide (when the difference between values is relative small e.g., we need **a tie breaking rule (index number rule)**. Smallest number rule, biggest number rule (index of a subproblem).

- Maximization problem: start with lower bound, update upper bound (when integer = case 1)
- Minimization problem: start with upper bound, update lower bound (when integer = case 1)

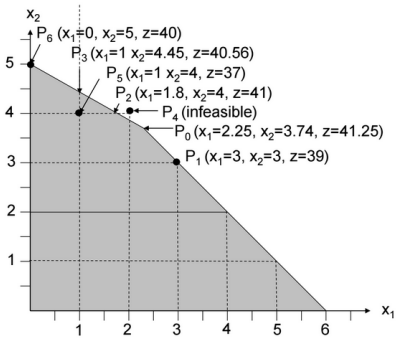
For maximization problem: $z^*(\boldsymbol{son\ problem}) \leq z^*(\boldsymbol{father\ problem})$
For minimization problem: $z^*(\boldsymbol{son\ problem}) \geq z^*(\boldsymbol{father\ problem})$

GRAPHICAL REPRESENTATION OF SUBPROBLEMS

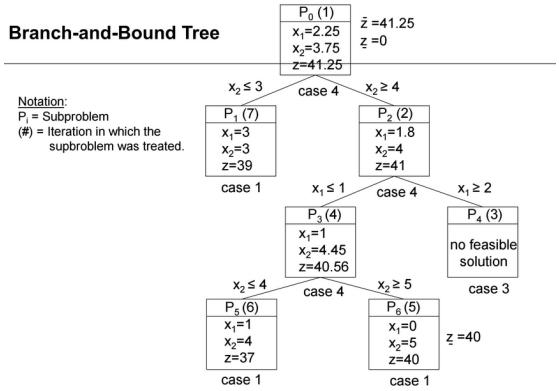
		LP relaxation						
K	P_{i^*}	x_1	x_2	z	$\bar{z}$	$\underline{z}$	Case	Branching
$<P_0]$	P_0	2.25	3.75	41.25	41.25	0	(4)	$x_2 \leq 3 \Rightarrow P_1$ $x_2 \geq 4 \Rightarrow P_2$
$<P_2,P_1]$	P_2	1.8	4	41			(4)	$x_1 \leq 1 \Rightarrow P_3$ $x_1 \geq 2 \Rightarrow P_4$
$<P_4,P_3,P_1]$	P_4	No feasible solution					(3)	
$<P_3,P_1]$	P_3	1	4.45	40.56			(4)	$x_2 \leq 4 \Rightarrow P_5$ $x_2 \geq 5 \Rightarrow P_6$
$<P_6,P_5,P_1]$	P_6	0	5	40		40	(1)	
$<P_5,P_1]$	P_5	1	4	37			(1)	
$<P_1]$	P_1	3	3	39			(1)	
<]								



Solution Space and Visited Solutions



Branch-and-Bound Tree



Notation:
P_i = Subproblem
(#) = Iteration in which the supbroblem was treated.

LIMITED BRANCH-AND-BOUND

We stop the algorithm when we know that the solution is not further away than $\bar{\Delta}$ % from the optimal solution, i.e.

Stop if $\Delta = \frac{\bar{z}-z}{\bar{z}} \leq \bar{\Delta}$

*multiply on 100 => get %

2.2. BRANCH AND BOUND WITHOUT SIMPLEX

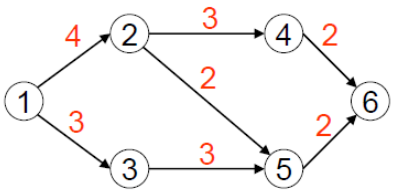
Project No	Capital value / npv (c_{ij})	Budget demand / expendit. (a_{ij})	$\frac{c_{ij}}{a_{ij}}$	Selection	x_{ij}	$\sum c_{ij} * x_{ij}$	$\sum a_{ij} * x_{ij}$

THE UPPER BOUND (GOOD UPPER BOUND) – realization of projects as budget permits.
SIMPLE UPPER BOUND – selection of all projects
THE LOWER BOUND – realization of projects until one cannot be selected.
SIMPLE LOWER BOUND – select zero projects

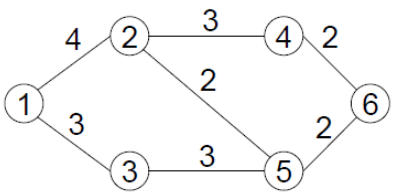
3. GRAPH THEORY AND NETWORK FLOW PROBLEMS

BASICS

A directed graph (digraph) $G = (V, A)$ is defined by a set of nodes V and a set of arcs A where each arc (i, j) is defined by an initial node i and a terminal node j.
An undirected graph $G = [V, A]$ is defined by a set of nodes V and a set of edges A where each edge $[i, j]$ is defined by nodes i and j.
A network is a graph with parameters (c_{ij}) associated with arcs or edges. A network with arcs is also called a digraph with cost weights.



Directed network G=(V,A,c)



Undirected network G=[V,A,c]

A path in a directed or undirected graph is a sequence of $k \geq 2$ nodes $(n_1, n_2, \ldots, n_k)$ and associated $k - 1$ arcs $((n_1, n_2), (n_2, n_3) \ldots, (n_{k-1}, n_k))$. The length of a path is the sum of the cost weights of the arcs (n_1, n_2), $(n_2, n_3) \ldots, (n_{k-1}, n_k)$.

A chain in a digraph $G=(V,A)$ is a sequence of $k \geq 2$ nodes $(n_1, n_2, \ldots, n_k)$ and associated $k-1$ arcs where the i-th arc is either traversed in forward direction, i.e. (n_i, n_{i+1}) or in backward direction, i.e. (n_{i+1}, n_i). The length of a path is the sum of the cost weights of the traversed arcs.

A cycle is a closed chain where the start node is the end node.

A graph is connected if it contains at least one path between each pair of nodes (if every node can be reached from every other node by a path).

A connected graph without cycles is a spanning tree.

GRAPH THEORY ALGORITHMS

Dijkstra's algorithm	The shortest path from an initial node to all nodes of a digraph. Required iterations: n	(1) Existing path from the source to all other nodes (2) All arc lengths are non-negative	Navigator, Internet routing, finding the way around
Floyd-Warshall algorithm	The shortest path between every pair of nodes of a digraph.	No prerequisites	Scheduling, Vehicle routing
Kruskal's algorithm	Minimum spanning tree Required iterations: n-1	(1) Graph is connected (2) Sum of the costs is minimal	Electricity networks, Water networks, Communication networks.
Ford-Fulkerson algorithm	Minimum cost flow (maximum flow problem)	(1) We need a dgraph (2) Source and sink (3) Capacities	Capacity of a network infrastructures, such as streets, railroads, pipeline.

3.1. DIJKSTRA'S ALGORITHM

Calculates for shortest path from an initial node a to all nodes of a digraph with cost weights (V, A, c).
In a graph defined by a set of **n** nodes, it is requiring making n iterations to find the shortest way, since the method of Dijkstra determines a shortest path from the output node to a further node of the graph in each iteration.

Prerequisites

- Any graph (directed or non-directional) - a digraph $G = (V, A, c)$ with non-negative cost weights, i.e. $cij \geq 0$, or
- The graph has no negative cycles - an acyclic digraph $G = (V, A, c)$ with cost weights $cij \in$ R.

Data

$d[1, \dots, n]$ $d[i]$ provides the shortest distance from node a to node i
$p[1, \dots, n]$ $p[i]$ provides the immediate predecessor node on the shortest path from node a to node i

M Set of marked nodes (the last nodes in the paths so far).

A node i enters M when the first distance path from a to i has been found. When i leaves M, the shortest distance from a to i has been established.

Initialize

M={a}
d[a] = 0
d[i] = ∞ for all $i \in V\{a\}$
p[] no entry for all $i \in V$

Iterations:

If set M is not empty, then
Select node h $\in$ M with smallest d[h]
Delete h from M
For all nodes $j \in V$ with arch $(h, j) \in A$ do
If $d[j] > d[h] + c_{hj}$ then
$d[j] = d[h] + c_{hj}$
$p[j] = h$
Mark and add j in M
End if
End if
End if

3.2. FLOYD-WARSHALL ALGORITHM

- Calculates shortest paths between every pair of nodes of a digraph.
- No prerequisites.

Data:

d $d[i, j]$ length of the shortest path from node i to j.
p $p[i, j]$ immediate predecessor node on the shortest path from i to j.

Initialize:

For all $i \in V$:
Set $d_{i,j} = 0, p_{i,j} = i$
For all $j \in V\{i\}$
If $(i, j) \in A$
Set $d_{i,j} = c_{i,j} and\ p_{i,j} = i$
Else
Set $d_{i,j} = \infty\ and\ p_{i,j} = 0$

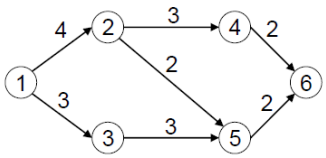
For all arcs (i, j) we set $d_{i,j} = c_{i,j}$ and save i as immediate predecessor of j on the shortest path from i to j.

Iterations:

For all $v \in V$
For all $i \in V$
For all $j \in V$
If $d_{i,j} > d_{i,v} + d_{v,j}, then$

For each triple i, v, j we check if the length of the path $i - v - j$ is shorter than the path $i - j$.

Set $d_{i,j} = d_{i,v} + d_{v,j}, and\ p_{i,j} = p_{v,j}$



	1	2	3	4	5	6
	$d_{i,1}, p_{i,1}$	$d_{i,2}, p_{i,2}$	$d_{i,3}, p_{i,3}$	$d_{i,4}, p_{i,4}$	$d_{i,5}, p_{i,5}$	$d_{i,6}, p_{i,6}$
i=1	0, 1	4, 1	3, 1	7, 2	6, 2	8, 5
2	∞, 0	0, 2	∞, 0	3, 2	2, 2	4, 5
3	∞, 0	∞, 0	0, 3	∞, 0	3, 3	5, 5
4	∞, 0	∞, 0	∞, 0	0, 4	∞, 0	2, 4
5	∞, 0	∞, 0	∞, 0	∞, 0	0, 5	2, 5
6	∞, 0	∞, 0	∞, 0	∞, 0	∞, 0	0, 6

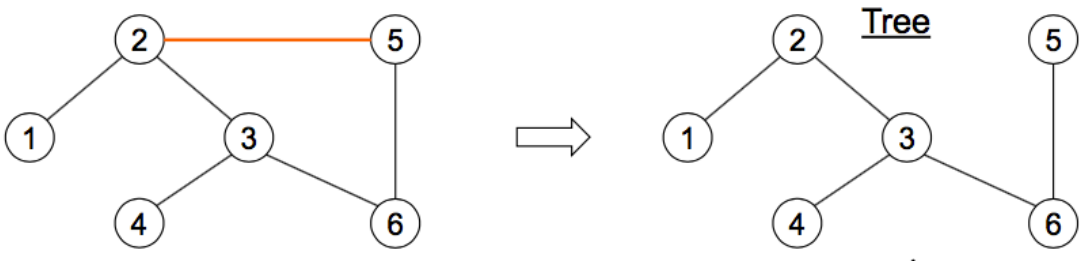
3.3. KRUSKAL'S ALGORITHM

Given graph $G = [V, A, c]$ we want to determine a subset of edges $A' \subseteq A$ such that graph $G' = [V,A',c]$:

- is connected and
- sum of the cost of the selected edges is minimal.

Observation: a minimum spanning tree is cycle free.

Applications: electricity networks, water networks or communication networks.



Start

Sort edges $[i, j] \in A$ in list L per non-decreasing cij (in case of ties sort according to increasing i and j).

Set A – all edges

Set A' – selected edges

Set A' = ⌀

Iteration

⇨ Select the edge $[i, j]$ from the list L.

- If graph $G = [V,' \cup [i,j]]$ has no cycle set $A' = A' \cup [i,j]$.
  - Delete $[i, j]$ from L.

⇨ Repeat until L = ⌀

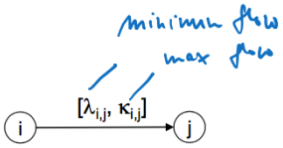
! Don't include the edges, which create cycles!

Sometimes the algorithm is stopped because a minimally existing tree contains exactly **n-1** edges for a graph with n nodes (the graph has 5 nodes and the method is aborted after the fourth edge has been added to the graph).

3.4. FORD-FULKERSON ALGORITHM

- Given digraph $G = (V, A, c)$.
- For each node $j \in V$ we have net flow (outflow –inflow) f_j
  - $f_j > 0$ equals supply (supply = outflow – inflow) f_j of node j
  - $f_j < 0$ equals demand (demand = inflow – outflow) $|f_j|$ of node j
  - $f_j = 0$ equals transhipment of node j

- For each arc $(i,j) \in A$ we have:
  - variable cost $c_{j,i}$ per transported unit
  - maximum flow $k_{j,i}$
  - minimum flow $\lambda_{j,i}$



We want to determine the flow $x_{i,j}$ for each arc $(i,j) \in A$.

Prerequisites:
Given digraph $G = (V, A, \lambda, \kappa)$ with a source and a sink as well as minimum and/or maximum capacities.
- We want to determine the maximum flow we can send from node q to node $s, q, s \in V, q \neq s$

Mathematical model LP
- Adding artificial arc (s,q) with $\lambda_{i,j} = 0 \; and \; k_{i,j} = \infty$
- Objective function: $Max\ z = x_{s,q}$

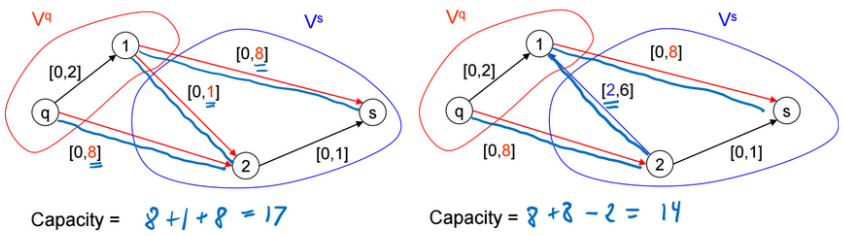
s.t.
1) Flow conservation constraint $\sum_{(h,j)\in A} x_{h,j} - \sum_{(j,k)\in A} x_{j,k} = 0 \; for\ all\ j \in V$
2) Maximum capacity constraint, $\lambda_{i,j}$ – min flow $x_{i,j} \geq \lambda_{i,j} \; for\ all\ (i,j) \in A$
3) Minimum capacity constraint, $\kappa_{i,j}$ – max flow $x_{i,j} \leq k_{i,j} \; for\ all\ (i,j) \in A$
4) Flow variable $x_{i,j} \varepsilon \{0,1\} \; for\ all\ (i,j)\ \varepsilon\ A$

Start with a feasible (not necessarily optimal) solution.
⇨ If $\lambda_{j,i} = 0$ holds for all $(i,j) \in A$, a first feasible solution is $x_{i,j} = 0$, for all $(i,j) \in A$.
- In every iteration, we try to improve the solution until the optimality criterion is fulfilled.

CUT - A cut divides node set V into the following two subsets V^q and V^s.
V^q contains source node q and the last iteration flow (visited nodes); and V^s contains sink node s (and not visited nodes). The intersection of V^q and V^s is empty, that is $V^q \cap V^s = \emptyset$.

THE CAPACITY OF A CUT - The sum of $k_{j,i}$ for all arcs $(i,)$ leading from V^q to V^s minus the sum of λi, for all arcs leading from V^s to V^q.

Example:



MINIMUM CUT PROBLEM
For a digraph $G = (V, A, \lambda, \kappa)$ divide the nodes into disjoint subsets V^s and V^q such that the capacity of the cut is minimal.

The min cut problem is the dual of the max flow problem:
⇨ This implies: weak duality property & strong duality property.
The max flow decreases if maximum capacities of arches (which are used by its maximum) reduced or minimum capacity of another arches (which are used by minimum) are increased.

Feasible but not optimal objective function value
Optimal objective function value

4. DYNAMIC PROGRAMING

Dynamic Programming is an approach to optimize a problem sequentially in multiple stages, each of which solves for an optimal single decision (or variable). The sequence of these single decisions constitutes the overall optimal

solution.

DIFFICULTIES:

(1) To appropriately model a problem to be solved by dynamic programming is usually difficult and requires experience.
(2) There is no general algorithm for solving a dynamic program. Rather the problem must be specifically formulated based on the principals of DP.

POLICIES:

1. single order in period 1,
2. order the demand for each period,
3. mixed policy.

PROPERTIES OF OPTIMAL POLICIES:

For an optimal policy, the following holds for each period i = 1… n:

- An order for period i only takes place if the inventory at the end of period $i-1$ is 0.
- An order $x_i > 0$ in period i equals the demand of periods i to τ with τ = i,…,.

STAGES AND DECISIONS

i Number of period

d_i Demand in specific period

n Number of stages

X_i Set of possible order quantities for period i.

$$X_i = \{0, \sum_{\tau=i}^{u} d_\tau \ \forall u = i, \dots, n\}$$

x_i Quantity ordered (and delivered) at the beginning of period i (i = 1,… , n).

$$x_i \in X_i = \{0, \sum_{\tau=i}^{u} d_\tau \ \forall u = i, \dots, n\}$$

z_i State variable: Inventory at the end of period i (i = 0, … , n)

$$Z_i = \{0, \sum_{\tau=i+1}^{u} d_\tau \ \forall u = i+1, \dots, n\}$$

Z_i Set of states in period i=1,…,n.

Dynamic Inventory Balance Constraint: $z_i = z_{i-1} + x_i - d_i$

State Dependent cost function:

$$f_i(x_i, z_i) = FC \cdot \left\{ \begin{matrix} 1, \\ 0, \end{matrix} \middle| \begin{matrix} if\ x_i > 0 \\ otherwise \end{matrix} \right\} + k \cdot z_{i,j} = 1$$

- Where FC – fixed cost per order,
- $k \cdot z_i$ – Variable inventory cost per unit and period.

Backward Recursive Function:

$$F_i(z_i, x_{i+1}) = f_{i+1}(z_{i+1}, x_{i+1}) + F^*_{i+1}(z_{i+1}),\ i = n, \dots, 0, z_i \in Z_i, x_{i+1} \in X_{i+1}$$

- Cost in stage i depending on state z_i and decision x_{i+1}.

Optimality Principle of Bellman:

$$F^*_i(z_i) = min\{F^*_i(z_i, x_{i+1}) | x_{i+1} \in X_{i+1}(z_i)\},\ i = n, \dots, 0, z_i \in Z_i$$

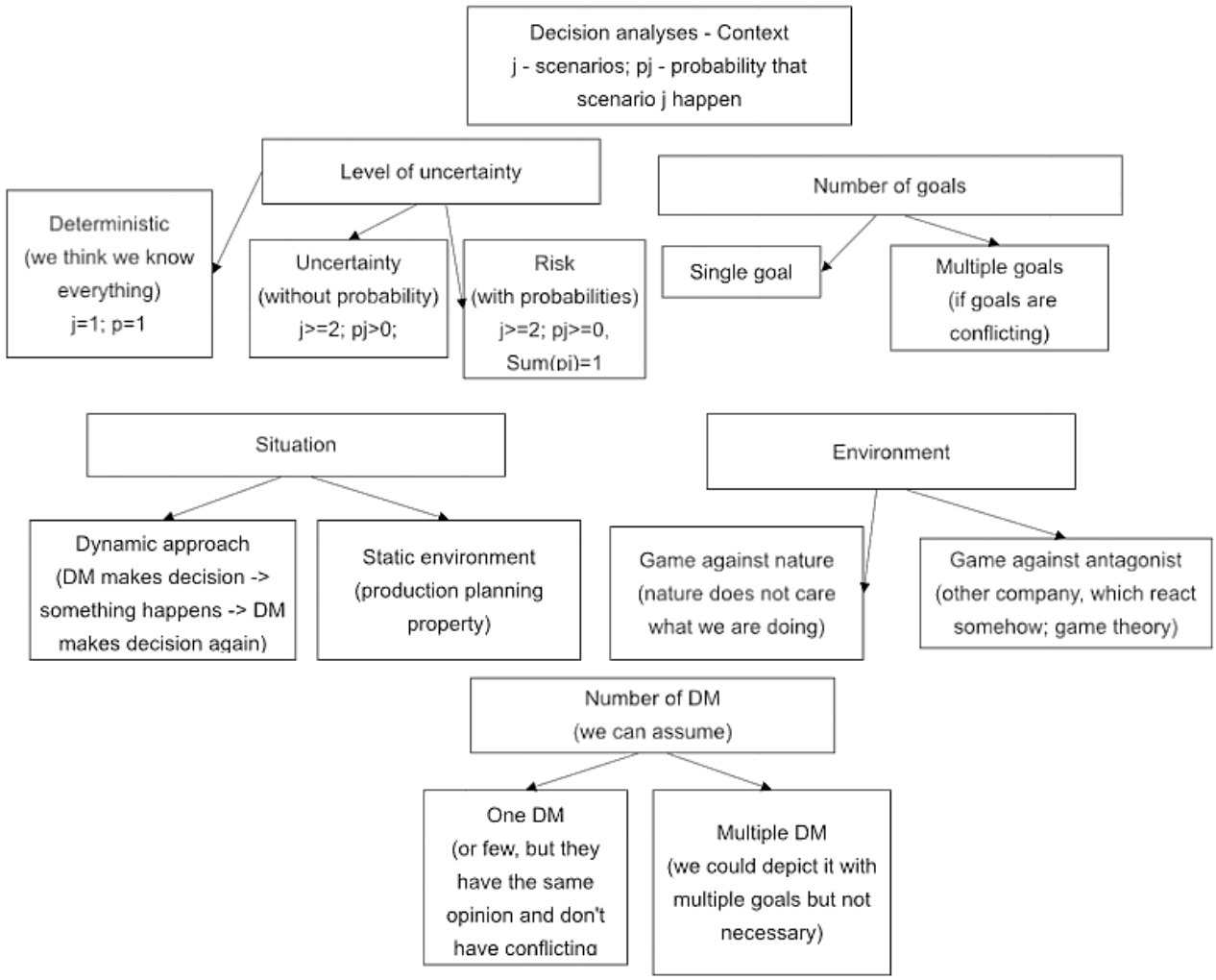
Recursion in Table Format

z_3 = state	x_4 =contribution	z_4 =change after contribution(total)	$f_4(z_4, x_4)$ = probability	$F^*_4(z_4)$ = the best probability from last stage	$F_3(z_3)$ = conditional probabilities (costs to go)

- Maximum capacities
- Minimal capacities
- Sizing

5. DECISION THEORY – DECISION ANALYSES

5.1. GENERAL



5.2. DECISION UNDER UNCERTAINTY

DECISION CONTEXT CHARACTERISTICS

- Uncertainty (multiple scenarios without probabilities)
- Single goal
- Single DM
- Static environment
- Game with a nature

DECISION MATRIX

Sequence of actions and events:

Alternative → Scenario → Outcome

$a_1, a_2, a_3, \dots, a_m - m$ alternatives. One of them must be chosen.

$s_1, s_2, s_3, \dots, s_n - n$ scenarios. One will occur.

$e_{1,1}, e_{1,2}, e_{1,3}, \dots, e_{m,n} - m \times n$ outcomes.

	s_1	… s_j	… s_n
a_1…	$e_{1,1}$	… $e_{1,j}$	… $e_{1,n}$
a_i …	$e_{i,1}$	… $e_{i,j}$	… $e_{i,n}$
a_m	$e_{m,1}$	… $e_{m,j}$	… $e_{m,n}$

DECISION RULES FOR DIFFERENT DM

Max-Min - Risk-afraid DM

- "I am unlucky" (nature will always be against me).
- Selects the alternative for which the "guaranteed" minimum profit is maximum.

$$\max_i \{\min_j \{e_{ij}\}\}$$

	s_1	s_2	s_3	$\min_j\{e_{ij}\}$	
a_1	1	-1	3	-1	
a_2	4	0	2	0	
a_3	3	-1	2	-1	
a_4	2	2	2	2	← $\max_i$

Max-Max - Risk-seeking DM

- "I am lucky" (nature will help me to get the best outcome).
- Selects the alternative for which the maximum gain is maximum.

$$\max_i \{\max_j \{e_{ij}\}\}$$

	s_1	s_2	s_3	$\max_j\{e_{ij}\}$	
a_1	1	-1	3	3	
a_2	4	0	2	4	← $\max_i$
a_3	3	-1	2	3	
a_4	2	2	2	2	

Hurwicz – Both, depend on lambda

- Selects the alternative that maximizes the combination from the maximum and minimum results.
- λ – optimistic parameter

for λ = 1, the Max-max rule (risk-afraid DM)
for λ = 0 the Max-min rule (risk-seeking DM)

$$0 \leq \lambda \leq 1$$

pessimist	optimist
(max-min rule)	(max-max rule)

$$\max_i \{\lambda \cdot \max_j \{e_{ij}\} + (1-\lambda) \cdot \min_j \{e_{ij}\}\}$$

Beispiel

$\lambda = 0{,}3$

	s_1	s_2	s_3	$\max_j\{e_{ij}\}$	$\min_j\{e_{ij}\}$	e_i	
a_1	1	-1	3	3	-1	0,2	
a_2	4	0	2	4	0	1,2	
a_3	3	-1	2	3	-1	0,2	
a_4	2	2	2	2	2	2	← $\max_i$

Min-Max Regret - Risk-afraid DM

- The DM determines the maximum gain that can be achieved for every environmental situation.
- He calculates how much profit he loses when infortune situation occurs instead of fortune.
- Selects the alternative for which the maximum loses are lowest.

*Regret = difference in result for chosen action vs. ex post optimal action.

Auswahlregel

$$\min_i \{\max_j \{r_{ij}\}\}$$

Beispiel

	s_1	s_2	s_3
a_1	1	-1	3
a_2	4	0	2
a_3	3	-1	2
a_4	2	2	2
$\max_i\{e_{ij}\}$	4	2	3

Auswahlregel

$$\min_i \{\max_j \{r_{ij}\}\}$$

Beispiel

2-(-1)

	s_1	s_2	s_3
a_1	1	-1	3
a_2	4	0	2
a_3	3	-1	2
a_4	2	2	2
$\max_i\{e_{ij}\}$	4	2	3

	s_1	s_2	s_3	$\max_j\{r_{ij}\}$	
a_1	3	3	0	3	
a_2	0	2	1	2	← $\min_i$
a_3	1	3	1	3	
a_4	2	0	1	2	← $\min_i$

Laplace – Risk neutral DM

I assume that all scenarios have the same probability.
Selects the alternative for which the average profit is maximum

Auswahlregel

$$\max_i \{\sum_{k=1}^{n} e_{ik} \cdot \frac{1}{n}\}$$

Beispiel

	s_1	s_2	s_3	e_i	
a_1	1	-1	3	1	
a_2	4	0	2	2	← $\max_i$
a_3	3	-1	2	1,33	
a_4	2	2	2	2	← $\max_i$

Note: There is no optimal rule. The choice of the rule reflects the DM attitude towards risk

5.3. DECISION UNDER RISK

EXPECTED VALUE THEORY

DECISION CONTEXT CHARACTERISTICS

- Risk
- Single goal
- Single DM
- Static environment
- Game with a nature

DECISION MATRIX

Sequence of actions and events:

Alternative → Probability → Scenario → Outcome

	p_1	… p_j	… p_n
	s_1	… s_j	… s_n
a_1…	$e_{1,1}$	… $e_{1,j}$	… $e_{1,n}$
a_i …	$e_{i,1}$	… $e_{i,j}$	… $e_{i,n}$
a_m	$e_{m,1}$	… $e_{m,j}$	… $e_{m,n}$

A column 1 ≤ i ≤ m of the decision matrix can be a lottery:

$$L_i = (p_1, e_{i1}; p_n, e_{in})$$

Expected value of the lottery L_i:
probability x outcome:

$$EV(L_i) = \sum_{k=1}^{n} p_k \cdot e_{ij}$$

Decision matrix is multiplied by probabilities p, …, p for each (environmental) situation s, …,s

EXPECTED UTILITY THEORY

DECISION CONTEXT CHARACTERISTICS

- Risk
- Single goal
- Single DM
- Static environment
- Game with a nature

Value of the result (e.g., payout) depends on the DM and his attitude to risk -> we make utility function = risk-taking function.

$$u(x): x \in [e^-, e^+] \rightarrow [0,1]$$

Expected utility function (maximum amount which risk-neutral DM pay to participate the lottery)

$$EU(L_i) = \sum_{k=1}^{n} p_k \cdot u(e_{i,j})$$

Characteristics of utility function, scaling to 0-worst outcome and 1-best outcome:

a) $u(e^-) = 0 \rightarrow \min\{e_{ij}\}$
b) $u(e^+) = 1 \rightarrow \max\{e_{ij}\}$
c) Ordering of alternatives: for a > b, u(a) > u(b)
for a = b, u(a) = u(b)

Axioms, which should be satisfied by the DM if he wants to use the expected utility criterion:

(1) Completeness: for two results e1 and e2, from the viewpoint of the DM e1 ≻ e2 or e2 ≻ e1 or e1 ~ e2 must apply.
(2) Continuity: for the DM e1 ≻ e2 ≻ e3 holds, then there is p (probability), which is 0<p<1 for the L1 = (1, e2) ~L2 = (p, e1; 1-p, e3).
(3) Independence: If for the DM e1 ~ e2, then there is a p with 0 <p <1: L1 = (p, e1; 1-p, e3) ~L2 = (p, e2; 1-p, e3)
(4) Compound lotteries: the DM is indifferent between a combined lottery and a non-combined lottery if the probabilities and results do not change.

(5) Unequal probabilities: if the DM considers e1 > e2, then must apply for p1> p2: L1 = (p1, e1; 1 - p1, e2) > L2 = (p2, e1; 1-p2, e2).

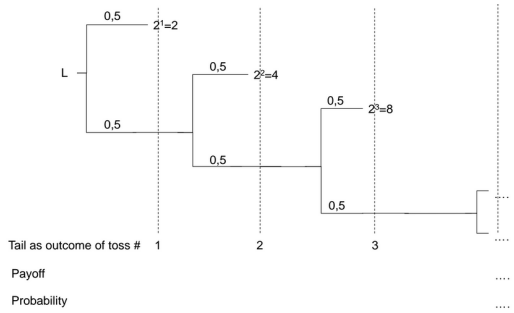
ST. PETERSBURG PARADOX

Utility of playing the game:

$u(game) = \sum_{n=0}^{\infty} u(n,x)P(n)$
$EU(game) = \infty$

n – number of heads, which you observe before the first tail;
P(n) – probability that exactly this number of heads will occur before the first tail.

Utility of not playing the game: 0



METHOD FOR DETERMINING THE UTILITY FUNCTION

CERTAINTY EQUIVALENT (CE(L))

The number between min and max which is indifferent for DM – the maximum amount which DM (out of risk) would pay for participating in the lottery.

$$1, CE(L) \sim L$$
$$CE(L) = u^{-1}(EU)$$

RISK PREMIUM (RP(L))

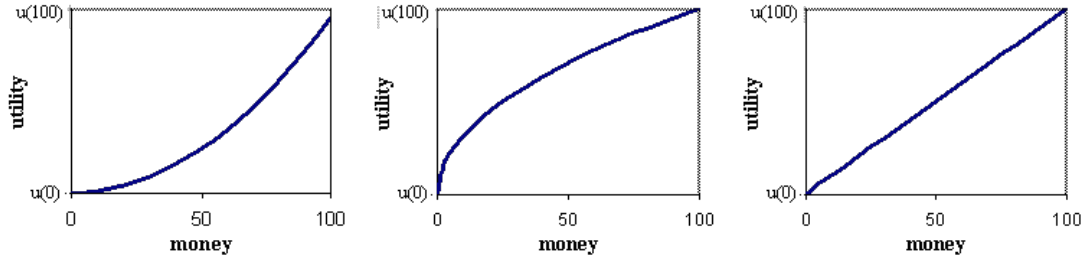
Cost of getting rid of the risk: expected value of the lottery – certainty equivalent:
$RP(L) = EV(L) - CE(L)$

+ Positive Risk-premium: OUT of risk
- Negative Risk premium: SEEKING for risk

ARROW-PRATT MEASURE

Reveals the risk attitude of the DM at x: $AP(x) = -\frac{u''(x)}{u'(x)}$

Prerequisites: **u has first and second derivative:** $u'(x) \neq 0$



Convex - seeking
$u'(x) > 0$
$u''(x) > 0$
=> AP(x) < 0
for all $x \geq e^-, x \leq e^+$
CE(L)>E(L

Concave - averse
$u'(x) > 0$
$u''(x) < 0$
=> AP(x) > 0
for all $x \geq e^-, x \leq e^+$
The more the line away from the straight line => the more risk averse; CE(L)<E(L)

Linear - neutral
$u'(x) > 0$
$u''(x) = 0$
=> AP(x) = 0
for all $x \geq e^-, x \leq e^+$

- CE(L)=E(L)
- RP(L)=0

BUILDING THE UTILITY FUNCTION

1) Building utility function
2) Mapping the $e_{i,j}$ of the decision matrix into utilities $u(e_{i,j})$
3) Calculating for each action a_i the expected utility $EU(a_i)$
4) Choosing the action a_i with maximum $EU(a_i)$

- Plug in all numbers from the decision matrix into utility function ad that is how we receive all the outcomes => transforming the outcome matrix into utility matrix.

μ-σ RULE

DECISION CONTEXT CHARACTERISTICS

- Risk
- Single goal
- Single DM
- Static environment
- Game with a nature

$\mu(a_i)$ Expected outcome of action a_i, $\mu(a_i) = EV(a_j) = \sum_j^n p_j \cdot e_{i,j}$

$\sigma^2(a_i)$ Variance of the outcome of action a_i, $\sigma^2(a_i) = \sum_j^n p_j \cdot (\mu(a_i) - e_{i,j})^2$

$\sigma(a_i)$ Standard deviation of the outcome of action a_i, $\sigma(a_i) = \sqrt[2]{\sigma^2(a_i)}$

$\varphi(a_i)$ Preference function

DM will pick up an alternative, which has a highest value

	p_j	0,1	0,2	0,5	0,2	$\mu(a_i)$	$\sigma^2(a_i)$	$\Phi(a_i)$
	s_j	1	2	3	4			
⇒	1				10	5,6	7,8	5.6
	2				4	4,5	0,9	4.5
a_i	3				5	5	2,4	5
	4				2	5	5,4	5

- Risk neutral DM $\varphi(a_i) = f(\mu(a_i))$
- Risk seeking DM $\varphi(a_i) = f(\mu(a_i) + \sigma(a_i))$
- Risk averse DM $\varphi(a_i) = f(\mu(a_i) - \sigma(a_i))$

EXPECTED UTILITY VS μ-σ RULE

Bothe methods provide with equal solutions if:

- Risk-neutral DM $\varphi(a_i) = f(\mu(a_i))$
- Quadratic utility function and appropriate calibration of preference function $\varphi(a_i)$
- Normally distributed outcomes

There is no help with coming up to preference function, μ-σ rule does not show a risk attitude of the DM.

5.4. MULTI-STAGE DECISION MAKING

- Risk
- Single goal
- Single DM
- Dynamic environment
- Game with a nature

DECISION TREE ANALYSES

Power of utility function: <1 → risk-afraid DM; >1 →risk-seeking

Calculating utility for different options:

(1) Minus costs of each option from outcome
(2) Insert the values in the utility function
(3) Multiply the utility on probability for each brunch

Decision | Outcome | Nature acts

5.5. MULTI-CRITERIA DECISION MAKING (UTILITY ANALYSES)

- Deterministic
- Multiple goals
- Single DM
- Static environment

	z_1	z_j	z_n
	w_1	w_j	w_n
a			
a_1	e_{11}	e_{1j}	e_{1n}
a_i	e_{i1}	e_{ij}	e_{in}
a_m	e_{m1}	e_{mj}	e_{mn}

Steps of utility analyses:

1. Determine the objectives: low costs, high market share and etc.
2. Determinate and standardize the target weights
3. Determination of the result values e_{ij}
4. Determine the utility function for each target:
   a) Find best e_j^+ and worst e_j^-
   b) Utility function $u_{ij}(e_{ij}) = \frac{e_{ij}-e_j^-}{e_j^+-e_j^-}$
   c) $u_{ij}(e_j^+) = 1$ and $e_j^- = 0$
5. Determination of expected utility for each alternative

$$S(a_i) = \sum_{j=1}^{m} w_j \cdot v_j(e_{ij})$$

6. Alternative with the highest S = optimal solution

Methods:

- Scoring model
- Analytical hierarchy process (AHP)
- Multi-attribute utility theory

z_n – goal (ziel)
w_n – weight of the goal, the higher the weight – the more important the goal

SCORING MODEL

1) We want to minimize costs
2) We want to be successful

$0 < w_j < 1 \mid \sum w_j = 1$

Step 1. Criteria weights

(1) The value g_j of each goal z_j is determined according to the following scale:
- Very important 5
- Important 4
- Average important 3
- Less important 3
- Marginally important 1

(2) The relative importance w_j of each goal is determined per:

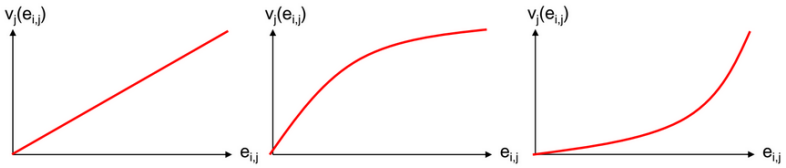
$$w_j = \frac{g_i}{\sum g_i}$$

Step 2. Transforming qualitative into quantitative values

Step 3. Normalizing scales

- Outcomes e_{ij} of actions $a_1, \dots, a_m$ with respect to goal z_j are normalized to range [0,1] by value function $v_j(e_{i,j})$
- Characteristics of the value function:
  - Monotonically increasing (decreasing)
  - Worst outcome has value 0
  - Best outcome has value 1

*The same as utility function, but without a risk



⇨ For the remainder we are using the linear value function

VALUE OF INFORMATION

Assumptions:

- The DM can observe first which scenario occurs and second choose an action
- Perfect information does not mean that there will be market success in any case
- The probability for the market success (55%) does not change
- The sequence of action and occurrence of scenario is reversed in the decision tree

SENSITIVITY ANALYSIS

How does the change of the success probability impact the decision?
Let p be a probability of a market success in case of the success of the test market.

- P=success
- 1-p=failure

1) plug in p everywhere, except %
2) create an equation, solve

PRIMAL SIMPLEX ALGORITHM

Prerequisite:
LP in canonical form with max objective.

(1) $if\ \overline{c_j} \leq 0\ for\ j = 1, \dots n$
➔ Stop its optimal solution.

Otherwise, continue only if there is at least one $\overline{c_j} > 0$.

(2) Choose pivot column s: $\overline{c_s} = max\{\overline{c_j} | \overline{c_j} > 0, j = 1, \dots, n\}$
$if\ \overline{a_{i,s}} \leq 0\ for\ i = 1, \dots m$
➔ Stop its unbounded solution.

Continue only if there is at least one $\overline{a_{i,s}} > 0$

(3) Choose pivot row r: $\frac{\overline{b_r}}{\overline{a_{i,s}}} = min\left\{\frac{\overline{b_i}}{\overline{a_{i,s}}} | \overline{a_{i,s}} > 0, i = 1, \dots, m\right\}$, 0 also ok.
(4) Change basis variable and re-establish a canonical form.

(5) Go to step (1).

(!) Negative coefficient in Pivot Column
- When selecting the pivot row, we do not need to take into account row(s) with negative coefficient(s) in the pivot column.

(!) The Simplex ends with one of the following results:
- an optimal solution
- unbounded solution

Optimal solution of Primal simplex algorithm
Unbounded solution of primal simplex algorithm

Notation:
n - number of variables
j - variable index
$\overline{c_j}$ – current coefficient of objective function (j)
m – number of coefficients
i – constraint index
$\overline{a_{i,s}}$ - coefficient of elements (i, s) in the current tableau
b_i – current right-hand-side

DUAL SIMPLEX ALGORITHM

Condition:
LP is dual feasible (=primal optimal, $\forall j: \overline{c_j} \leq 0$)
and
dual non-optimal (=primal infeasible, $\exists i: \overline{b_i} < 0$).

(1) $if\ \overline{b_i} \geq 0\ for\ i = 1, \dots, m\ and\ \overline{c_j} \leq 0 for\ all\ j = 1, \dots, n$
➔ Stop its optimal solution
If there exist some $\overline{b_i} < 0$, then continue

(2) Chose pivot row
$r: \overline{b_r} = min\{\overline{b_i} | \overline{b_i} < 0, j = 1, \dots, m\}$
$if\ \overline{a_{r,j}} > 0\ for\ j = 1, \dots n$
➔ Stop there is no feasible solution
Continue only if there is at least one $\overline{a_{r,j}} < 0$

(3) Choose pivot column
$$s: \frac{\overline{c_s}}{\overline{a_{r,s}}} = min\left\{\frac{\overline{c_j}}{\overline{a_{r,j}}} | \overline{a_{r,j}} < 0, i = 1, \dots, n\right\}$$
(4) Replace the basic variable and re-establish canonical form.
(5) Go to step (1).

(!) The dual Simplex ends with one of the following results:
- an optimal solution
- there is no feasible solution

Notation:
n – number of variables
j – variable index
$\overline{c_j}$ – current coefficient of objective function (j)
m – number of coefficients (constraints)
i – constraint index
$\overline{a_{r,j}}$ – coefficient of elements (i, s) in the current tableau
b_i – current right-hand-side

Application of Dual Simplex Method:
- When the primal problem has many constraints.
- When an additional constraint is added to the optimal primal LP and makes the current optimal solution infeasible.

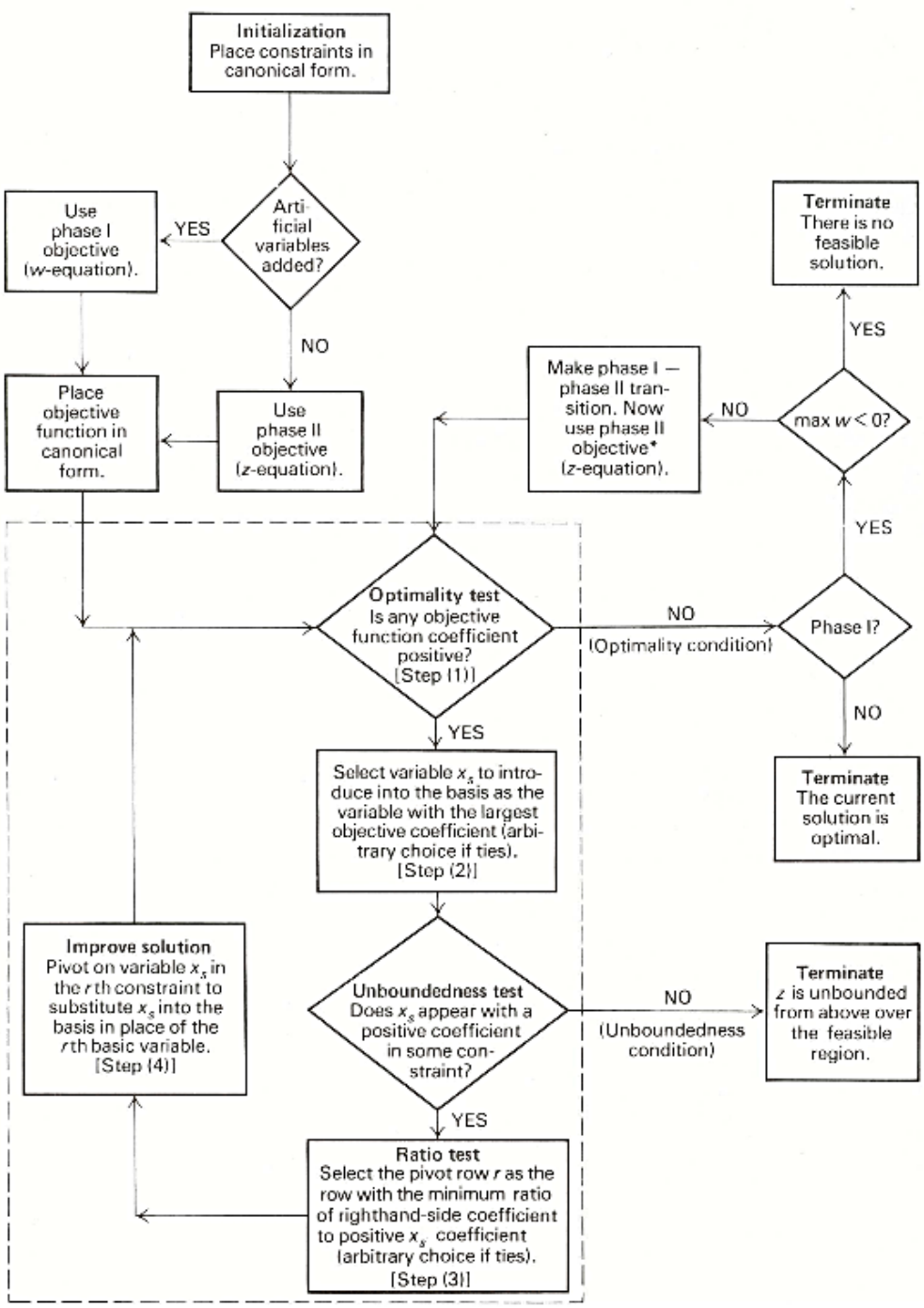


Figure 2.1 Simplex phase I–phase II maximization procedure.

Problem

Maximize $z = -3y_1 + 2y_2 - y_3 + 4y_4,$

subject to:

$$\begin{aligned} y_1 + y_2 - 4y_3 + 2y_4 &\geqq 4, \\ -3y_1 + y_2 - 2y_3 &\leqq 6, \\ y_2 - y_4 &= -1, \\ y_1 + y_2 - y_3 &= 0, \\ y_3 \geqq 0, \quad y_4 &\geqq 0. \end{aligned}$$

STEP 1 REDUCTION

COMMENTS

Maximize $z = -3x_1 + 3x_2 + 2x_3 - 2x_4 - x_5 + 4x_6,$

subject to:

$$\begin{aligned} x_1 - x_2 + x_3 - x_4 - 4x_5 + 2x_6 &\geqq 4, \\ -3x_1 + 3x_2 + x_3 - x_4 - 2x_5 &\leqq 6, \\ x_3 - x_4 - x_6 &= -1, \\ x_1 - x_2 + x_3 - x_4 - x_5 &= 0, \\ x_j \geqq 0 \quad (j = 1, 2, \ldots, 6). & \end{aligned}$$

Substitute

$x_1 - x_2 = y_1, \quad x_5 = y_3$

$x_3 - x_4 = y_2, \quad x_6 = y_4.$

STEP 2 REDUCTION

Maximize $z = -3x_1 + 3x_2 + 2x_3 - 2x_4 - x_5 + 4x_6,$

subject to:

$$\begin{aligned} x_1 - x_2 + x_3 - x_4 - 4x_5 + 2x_6 - x_7 &= 4, \\ -3x_1 + 3x_2 + x_3 - x_4 - 2x_5 + x_8 &= 6, \\ x_3 - x_4 - x_6 &= -1, \\ x_1 - x_2 + x_3 - x_4 - x_5 &= 0, \\ x_j \geqq 0 \quad (j = 1, 2, \ldots, 8). & \end{aligned}$$

Introduce surplus variable, x_7.

Introduce slack variable, x_8.

STEP 3 REDUCTION

Maximize $z = -3x_1 + 3x_2 + 2x_3 - 2x_4 - x_5 + 4x_6,$

subject to:

$$\begin{aligned} x_1 - x_2 + x_3 - x_4 - 4x_5 + 2x_6 - x_7 &= 4, \\ -3x_1 + 3x_2 + x_3 - x_4 - 2x_5 + x_8 &= 6, \\ -x_3 + x_4 + x_6 &= 1, \\ x_1 - x_2 + x_3 - x_4 - x_5 &= 0, \\ x_j \geqq 0 \quad (j = 1, 2, \ldots, 8). & \end{aligned}$$

The third constraint was multiplied by -1.

STEP 4 REDUCTION

Maximize $z = -3x_1 + 3x_2 + 2x_3 - 2x_4 - x_5 + 4x_6,$

subject to:

$$\begin{aligned} x_1 - x_2 + x_3 - x_4 - 4x_5 + 2x_6 - x_7 + x_9 &= 4, \\ -3x_1 + 3x_2 + x_3 - x_4 - 2x_5 + x_8 &= 6, \\ -x_3 + x_4 + x_6 + x_{10} &= 1, \\ x_1 - x_2 + x_3 - x_4 - x_5 + x_{11} &= 0, \\ x_j \geqq 0 \quad (j = 1, 2, \ldots, 11). & \end{aligned}$$

Artificial variables, x_9, x_{10}, x_{11}, added.

PHASE I OBJECTIVE

Maximize $w = -x_9 - x_{10} - x_{11}$

$$= +2x_1 - 2x_2 + x_3 - x_4 - 5x_5 + 3x_6 - x_7 - 5.$$

Constraints 1, 3, and 4 added to w objective.

Fig. 2.2 Reduction to canonical form.