# **[WI000275] Management Science**

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 ( | Non-basic variables |
| = linear algebra concepts  In canonical form, BV = to right hand side | = linear algebra concepts |

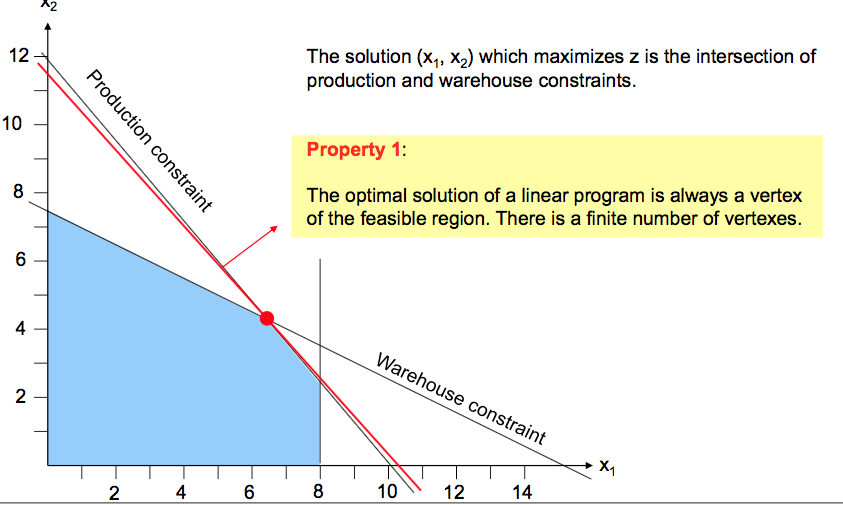
Constraints (production, demand, etc.)

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).

|  |  |
| --- | --- |
| ../../../../../Desktop/Screen%20Shot%202017-05-23%20at%2008 | ../../../../../Desktop/Screen%20Shot%202017-05-23%20at%2008 |



* 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. **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 | 1. All constraints, except non-negativity constraints of decision variables, are stated as equalities. 2. 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 → no production capacity left
2. If OF values are → if we have one additional unit of labor/energy, we can produce more
3. If OF values → 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**

1. Slack and surplus variables (

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

|  |  |
| --- | --- |
| \*the amount in the excess of minimum requirement | |
| Example |  |
| Steps | We need to minus one surplus variable and because of that we now should add artificial variable for M:    Change in OF: |

For inequalities, let the nonnegative slack variable () 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 |  |
| Steps | Add slack variable to obtain equality in each constraint: |

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.

1. Free variables ()

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 | |
| Example |  |
| Steps | Use  The variable represents positive inventory on hand and represents backorders (i.e., unfilled demand).  Whenever and when  For instance, can occur if we have negative change of inventory. |

1. Artificial variables, “Big M Method” (

|  |  |
| --- | --- |
| \*added to place the linear program in canonical form | |
| Example |  |
| Steps | 1. Create two types of equations: and then solve them separately.      1. We should solve it with surplus variable (use big M method)   max z = … - My1 |

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 |  | |
| Steps | Multiply by (-1) → change coefficient signs and rotate constraint sign and go further with necessary transformation. | |
| Minimization problem | | |
| Example | |  |
| Steps | | Multiply objective function by (-1) |

|  |  |
| --- | --- |
| If the optimality condition is not met | |
| Example |  |
| 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. **“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

1. 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
2. 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. **DUAL SIMPLEX METHOD**

Condition:

1. LP is dual feasible (=primal optimal, – if there is at least one OF coefficient is greater than 0, DS is not allowed)

AND

1. dual non-optimal (=primal infeasible, ).

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

* 1. **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  1)  while all corresponding  2) | In optimal tableau BV has .  In general, for BV with value 0, we have primal degeneracy. | In optimal tableau NBV, , 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  (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 (,

* dual problem is infeasible (.

(!) 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

* NBV of primal LP → Slack variable of associated constraint of the dual LP

|  |  |
| --- | --- |
| \*Primal feasible: | \*Dual optimal: |
| \*Primal optimal: | \*Dual feasible: |

**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**

|  |  |
| --- | --- |
| **../../../../../Desktop/Screen%20Shot%202017-06-12%20at%2017** | **../../../../../Desktop/Screen%20Shot%202017-06-12%20at%2017** |

* 1. **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 one data at a time, leaving all other data as it is (ceteris paribus)

Changing the OF coefficient of a BV (=shadow price)

Changing the OF coefficient of a NBV (=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**  Definition: 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 value of BV (slack variable) from initial to optimal tableau. The change = the shadow price.    **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: | ../../../../../Desktop/Screen%20Shot%202017-05-09%20at%2015 |
| **REDUCED COSTS**  Definition: 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).    – shadow price of constraint 𝑖  – objective function coefficient of variable 𝑗  – 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. **SENSATIVITY ANALYSES: VARIATION OF THE OF COEFFICIENT**

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

1. Add delta to the contribution margin (= 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
5. **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.
  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;
* P0 - At the beginning, only P0 is in K;
* Lower bound : The objective function value of a feasible solution of P0;
* Upper bound : The objective function of the LP-relaxation of P0

Cases for Eliminating a Subproblem

|  |  |
| --- | --- |
| 1. Integer solution 2. No feasible solution | 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

1. MUB – Maximum upper bound: B&B tree grows horizontally, „breadth-first search “.

MLB – minimum lower bound

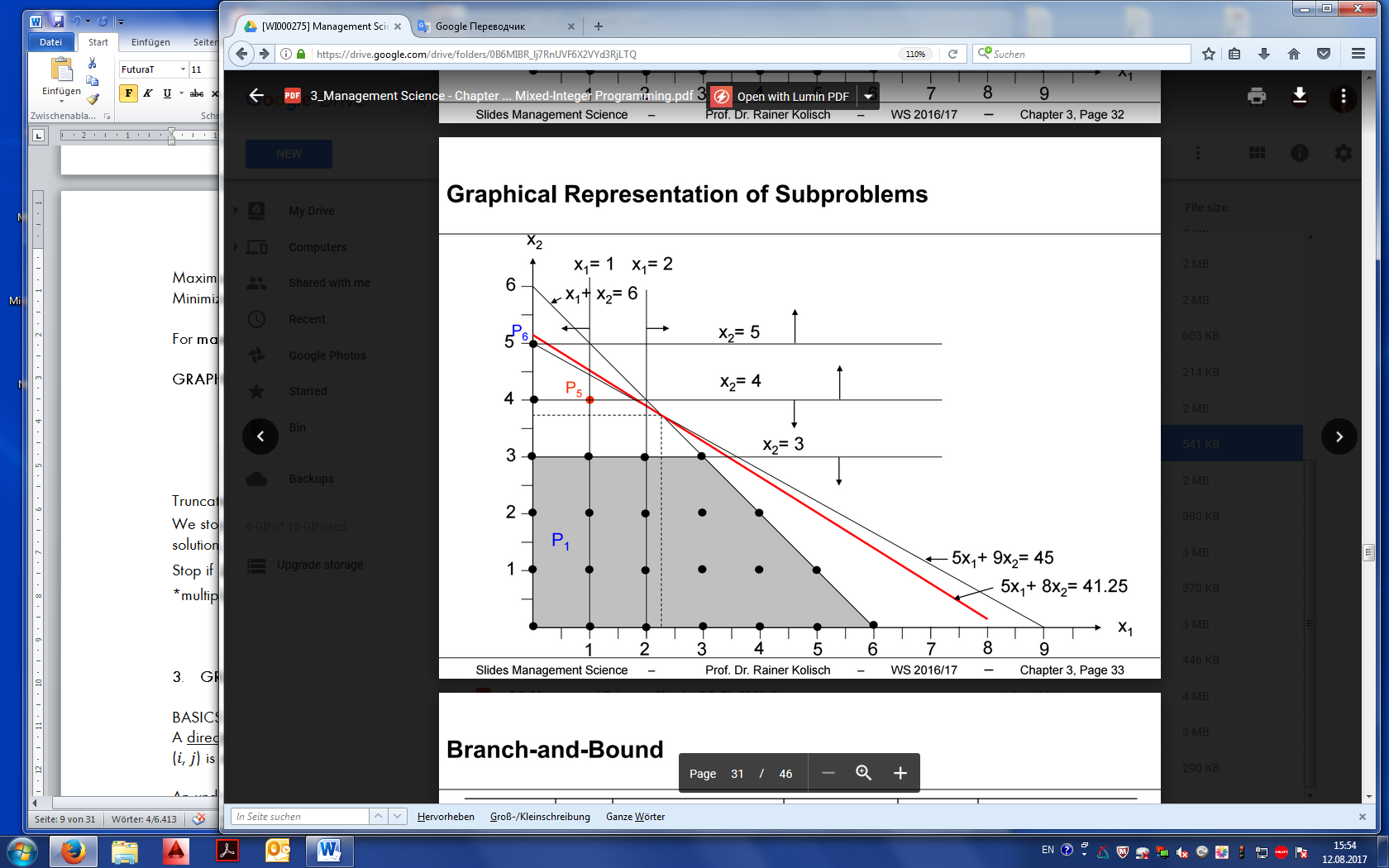
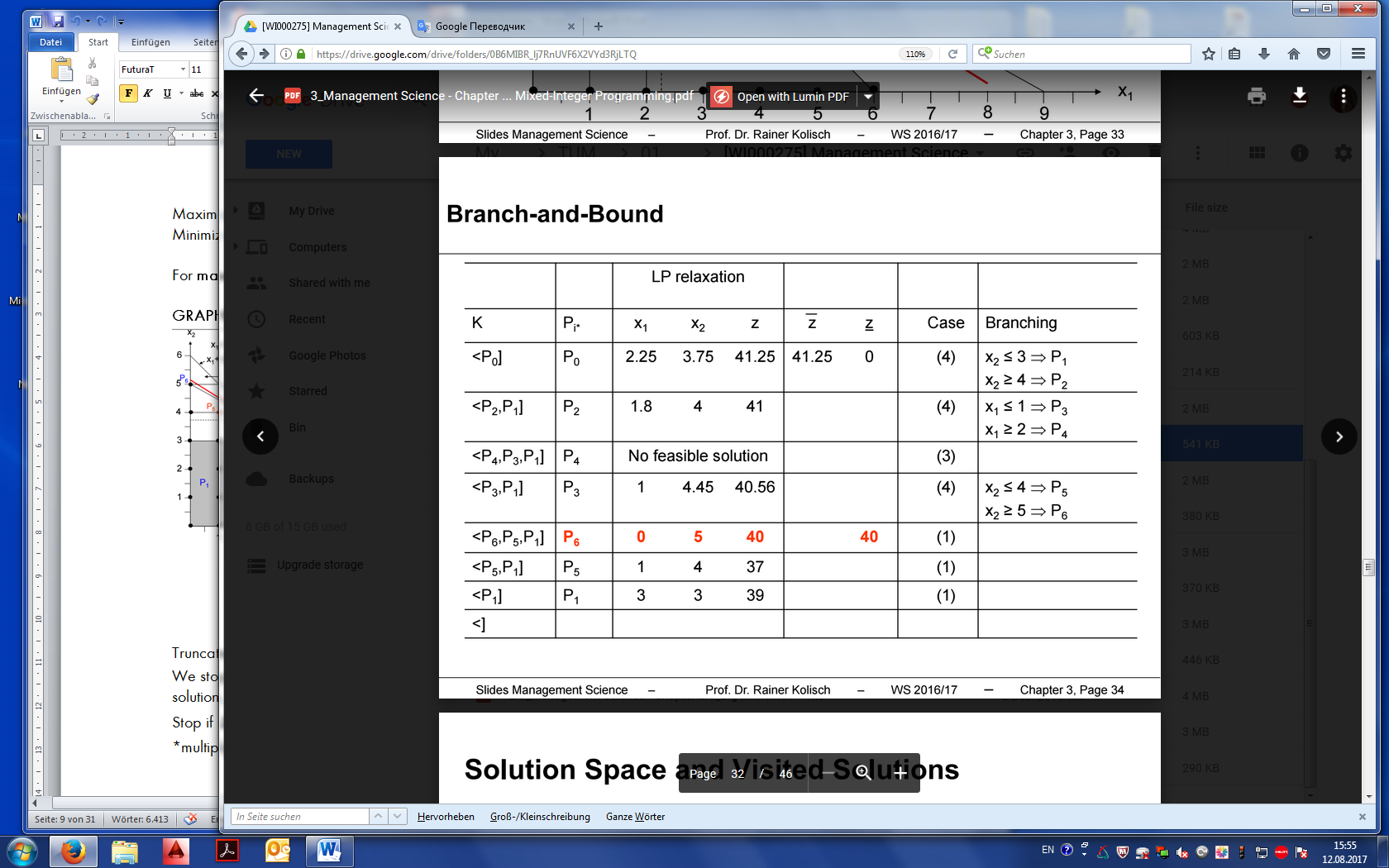
In case the priority rule cannot decide (when the difference between values is relative small e.g., we need **a tie breaking rule (index number rule)**. Smallest number rule, biggest number rule (index of a subproblem).

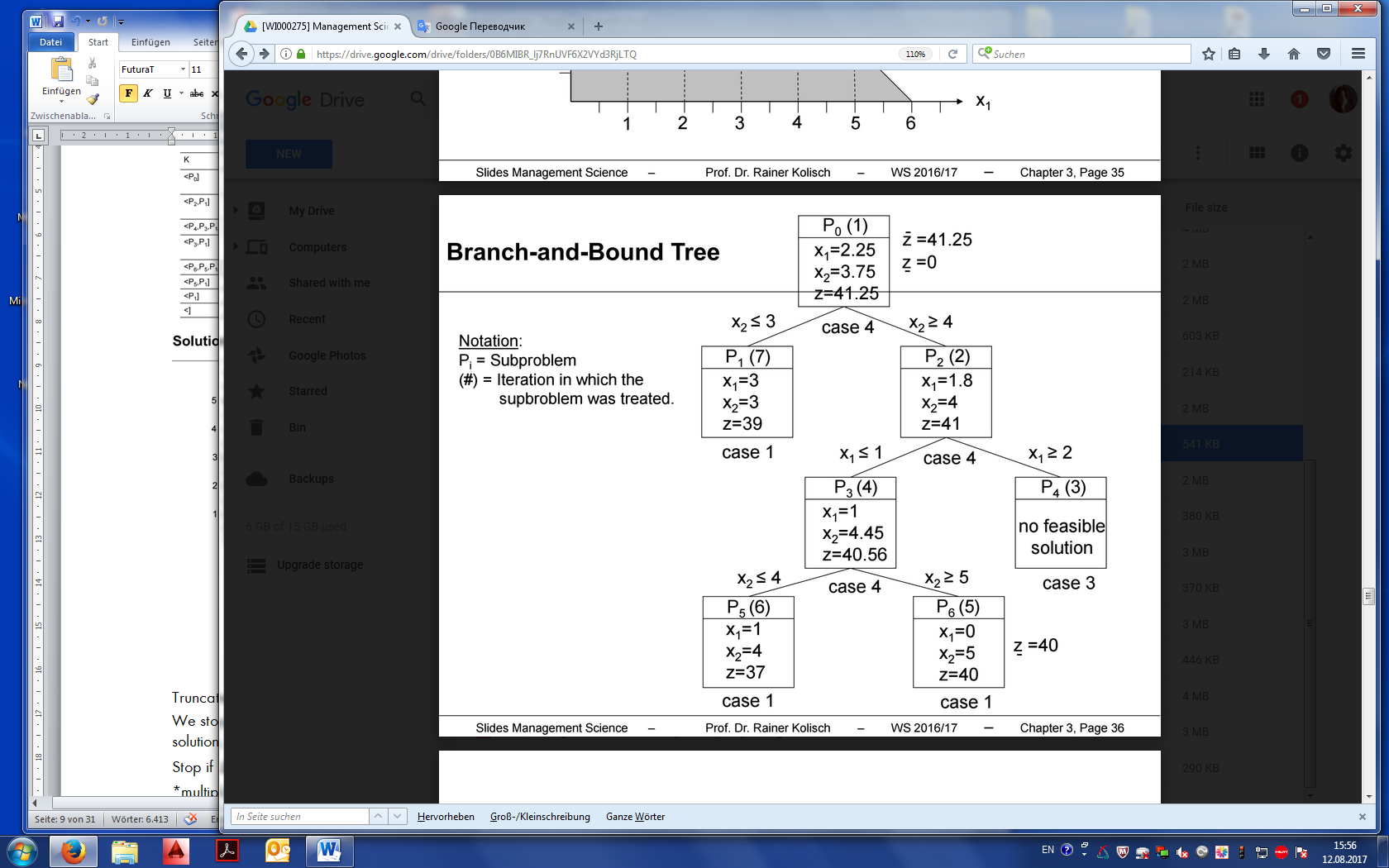
* Maximization problem: start with lower bound, update upper bound (when integer = case 1)
* Minimization problem: start with upper bound, update lower bound (when integer = case 1)

**For maximization problem:**

**For minimization problem:**

**GRAPHICAL REPRESENTATION OF SUBPROBLEMS**



**LIMITED BRANCH-AND-BOUND**

We stop the algorithm when we know that the solution is not further away than from the optimal solution, i.e.

Stop if

\*multiply on 100 => get %

* 1. **BRANCH AND BOUND WITHOUT SIMPLEX**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project No | Capital value / npv | Budget demand /  expendit. |  | Selection |  |  |  |

**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

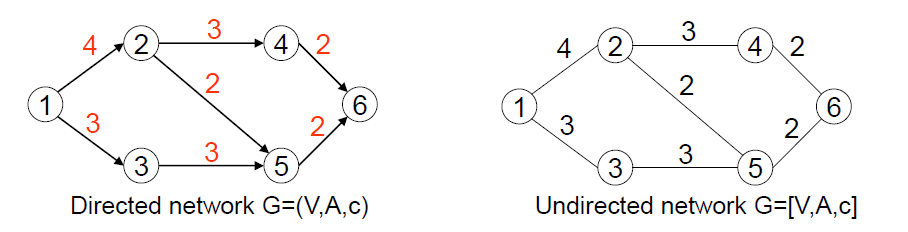
1. **GRAPH THEORY AND NETWORK FLOW PROBLEMS**

**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 (𝑐𝑖𝑗) associated with arcs or edges. A network with arcs is also called a digraph with cost weights.



A path in a directed or undirected graph is a sequence of 𝑘 ≥ 2 nodes (𝑛1, 𝑛2, ... , 𝑛𝑘) and associated 𝑘 – 1 arcs ((𝑛1, 𝑛2), (𝑛2, 𝑛3) ... , (𝑛𝑘−1, 𝑛𝑘)). The length of a path is the sum of the cost weights of the arcs (𝑛1, 𝑛2), (𝑛2, 𝑛3) ... , (𝑛𝑘−1, 𝑛𝑘).

A chain in a digraph 𝐺=(𝑉,𝐴) is a sequence of 𝑘 ≥ 2 nodes (𝑛1,𝑛2,...,𝑛𝑘) and associated 𝑘–1 arcs where the 𝑖-th arc is either traversed in forward direction, i.e. (𝑛𝑖,𝑛𝑖+1) or in backward direction, i.e. (𝑛𝑖+1,𝑛𝑖). 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. |

* 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 ≥ 0, or
* The graph has no negative cycles - an acyclic digraph 𝐺 = (𝑉, 𝐴, 𝑐) with cost weights 𝑐𝑖𝑗 ∈ R.

**Data**

|  |  |
| --- | --- |
| 𝑑[1, ... , 𝑛] | 𝑑[𝑖] provides the shortest distance from node 𝑎 to node 𝑖 |
| 𝑝[1, ... , 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  p[] no entry for all |  |
| **Iterations:**  If set M is not empty, then  Select node h ∈ M with smallest d[h]  Delete h from M  For all nodes with arch do  If then  Mark and add j in M  End if  End if  End if |  |

* 1. **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 :  Set  For all  If  Set  Else  Set | For all arcs we set and save 𝑖 as immediate predecessor of 𝑗 on the shortest path from 𝑖 to 𝑗. |
| **Iterations**:  For all  For all  For all  If  Set | For each triple we check if the length of the path is shorter than the path 𝑖 − 𝑗. |

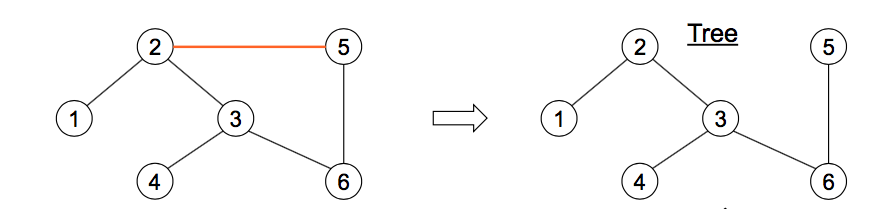
* 1. **KRUSKAL’S ALGORITHM**

Given graph 𝐺 = [𝑉, 𝐴, 𝑐] we want to determine a subset of edges 𝐴′ ⊆ 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] ∈ 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,′∪[𝑖,𝑗]] has no cycle set 𝐴′ = 𝐴′∪[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).

* 1. **FORD-FULKERSON ALGORITHM**
* Given digraph 𝐺 = (𝑉, 𝐴, 𝑐).
* For each node we have net flow (outflow –inflow)
  + equals supply (supply = outflow – inflow) of node
  + equals demand (demand = inflow – outflow) of node
  + equals transhipment of node

|  |  |
| --- | --- |
| * For each arc we have: * variable cost per transported unit * maximum flow * minimum flow | ../../../../../Desktop/Screen%20Shot%202017-06-21%20at%2017 |

We want to determine the flow for each arc .

**Prerequisites**:

Given digraph 𝐺 = (𝑉, 𝐴, 𝜆, 𝜅) 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

**Mathematical model LP**

* Adding artificial arc (s,q) with
* Objective function:

s.t.

1. Flow conservation constraint
2. Maximum capacity constraint, 𝜆𝑖,𝑗 – min flow
3. Minimum capacity constraint, 𝜅𝑖,𝑗 – max flow
4. Flow variable

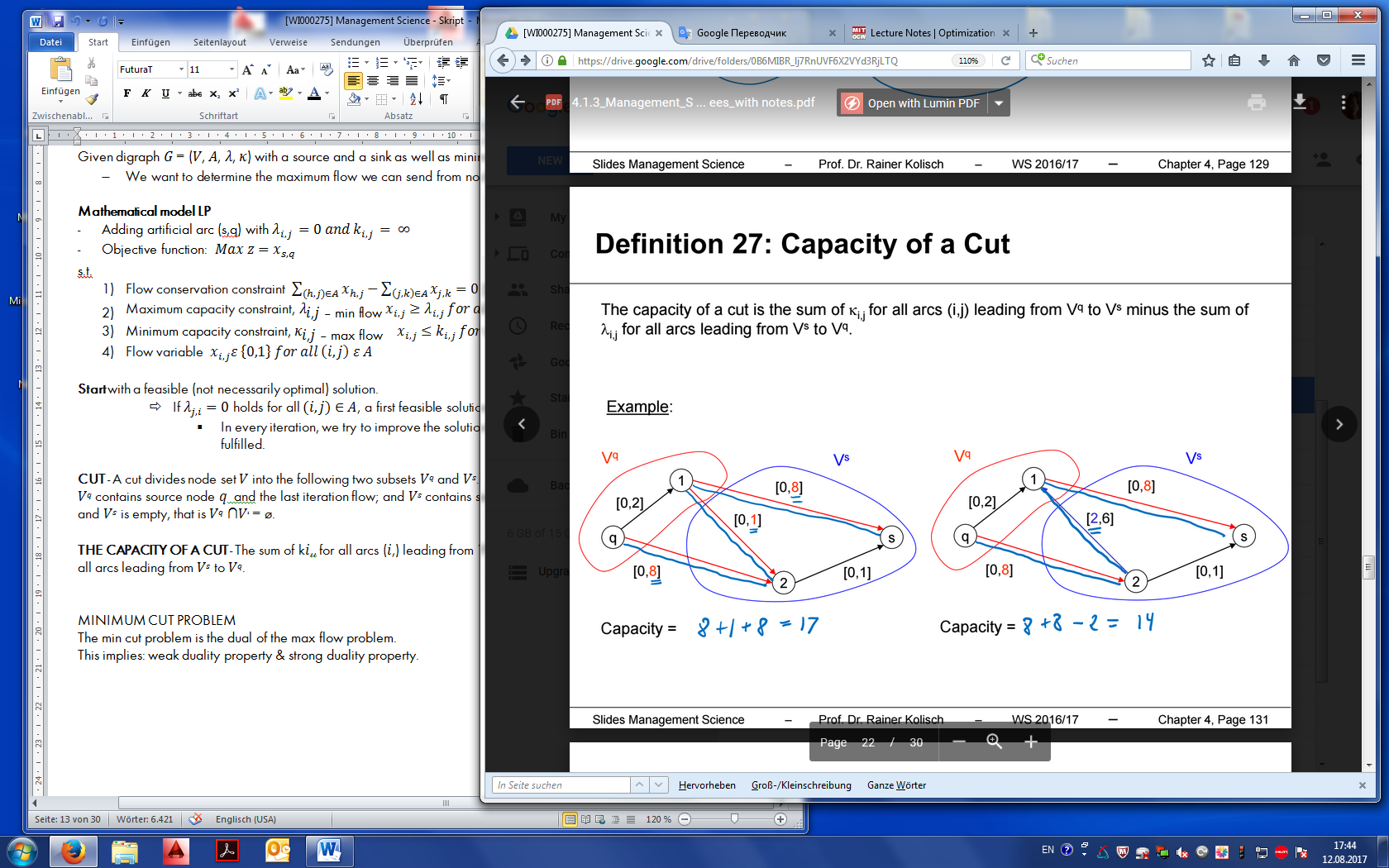
**Start** with a feasible (not necessarily optimal) solution.

* If holds for all , a first feasible solution is , for all
  + - 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 and .

contains source node 𝑞 and the last iteration flow (visited nodes); and contains sink node 𝑠 (and not visited nodes). The intersection of and is empty, that is .

**THE CAPACITY OF A CUT** - The sum of for all arcs (𝑖,) leading from 𝑉𝑞 to 𝑉𝑠 minus the sum of 𝜆𝑖, for all arcs leading from to .



**MINIMUM CUT PROBLEM**

For a digraph 𝐺 = (𝑉, 𝐴, 𝜆, 𝜅) divide the nodes into disjoint subsets 𝑉𝑠 and 𝑉𝑞 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

1. **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 𝑥𝑖 > 0 in period 𝑖 equals the demand of periods 𝑖 to 𝜏 with 𝜏 = 𝑖,...,.

**STAGES AND DECISIONS**

|  |  |  |
| --- | --- | --- |
|  | Number of period |  |
|  | Demand in specific period |  |
|  | Number of stages |  |
|  | Set of possible order quantities for period 𝑖. |  |
|  | Quantity ordered (and delivered) at the beginning of period 𝑖 (𝑖 = 1,... , n). |  |
|  | State variable: Inventory at the end of period 𝑖 (𝑖 = 0, ... , 𝑛) |  |
|  | Set of states in period 𝑖=1,…,n. |  |

**Dynamic Inventory Balance Constraint:**

**State Dependent cost function:**

* Where 𝐹𝐶 – fixed cost per order,
* – Variable inventory cost per unit and period.

**Backward Recursive Function:**

,

* Cost in stage 𝑖 depending on state and decision .

**Optimality Principle of Bellman:**

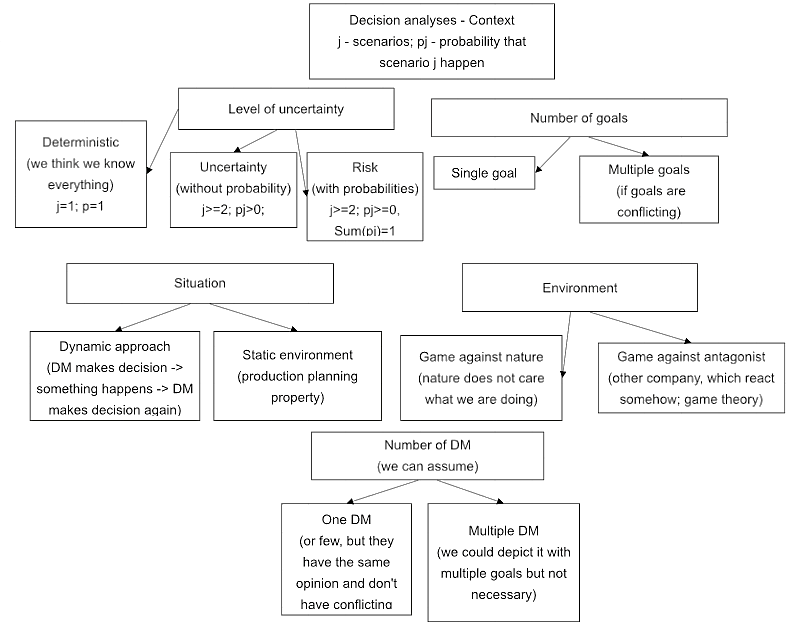
,

**Recursion in Table Format**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| = state | =contribution | =change after contribution(total) | = probability | = the best probability from last stage | = conditional probabilities (costs to go) |

* + Maximum capacities
  + Minimal capacities
  + Sizing

1. **DECISION THEORY – DECISION ANALYSES**
   1. **GENERAL**



* 1. **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  alternatives. One of them must be chosen.  scenarios. One will occur.  outcomes. | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | … |  |  |  | |  |  |  |  | |  |  |  |  | |

**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-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. |  |
| **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 ≤** | **λ** | **≤ 1** | | | | pessimist |  | | optimist | | (max-min rule) |  | | (max-max rule) | | |
| **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. |  | |
| **Laplace** – Risk neutral DM  I assume that all scenarios have the same probability.  Selects the alternative for which the average profit is maximum |  | |

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

* 1. **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  A column 1 ≤ i ≤ m of the decision matrix can be a lottery:  Expected value of the lottery :  probability x outcome: | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | | … |  |  |  | |  |  |  |  | |  |  |  |  | |

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.

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

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

2. 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: | 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.

**RISK PREMIUM (RP(L))**

Cost of getting rid of the risk: expected value of the lottery – certainty equivalent:

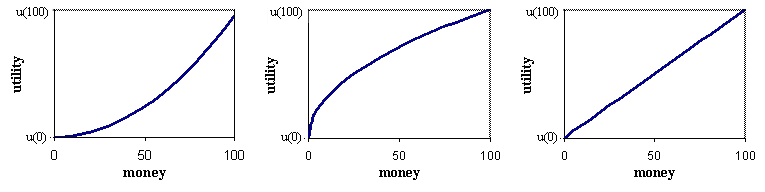
+ Positive Risk-premium: OUT of risk

- Negative Risk premium: SEEKING for risk

**ARROW-PRATT MEASURE**

Reveals the risk attitude of the DM at x:

Prerequisites: **u has first and second derivative:**



|  |  |  |
| --- | --- | --- |
| Convex - seeking      => AP(x) < 0  for all  CE(L)>E(L | Concave - averse      => AP(x) > 0  for all  The more the line away from the straight line => the more risk averse; CE(L)<E(L) | Linear - neutral      => AP(x) = 0  for all   * CE(L)=E(L) * RP(L)=0 |

**BUILDING THE UTILITY FUNCTION**

1. Building utility function
2. Mapping the of the decision matrix into utilities
3. Calculating for each action the expected utility
4. Choosing the action with maximum

* 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

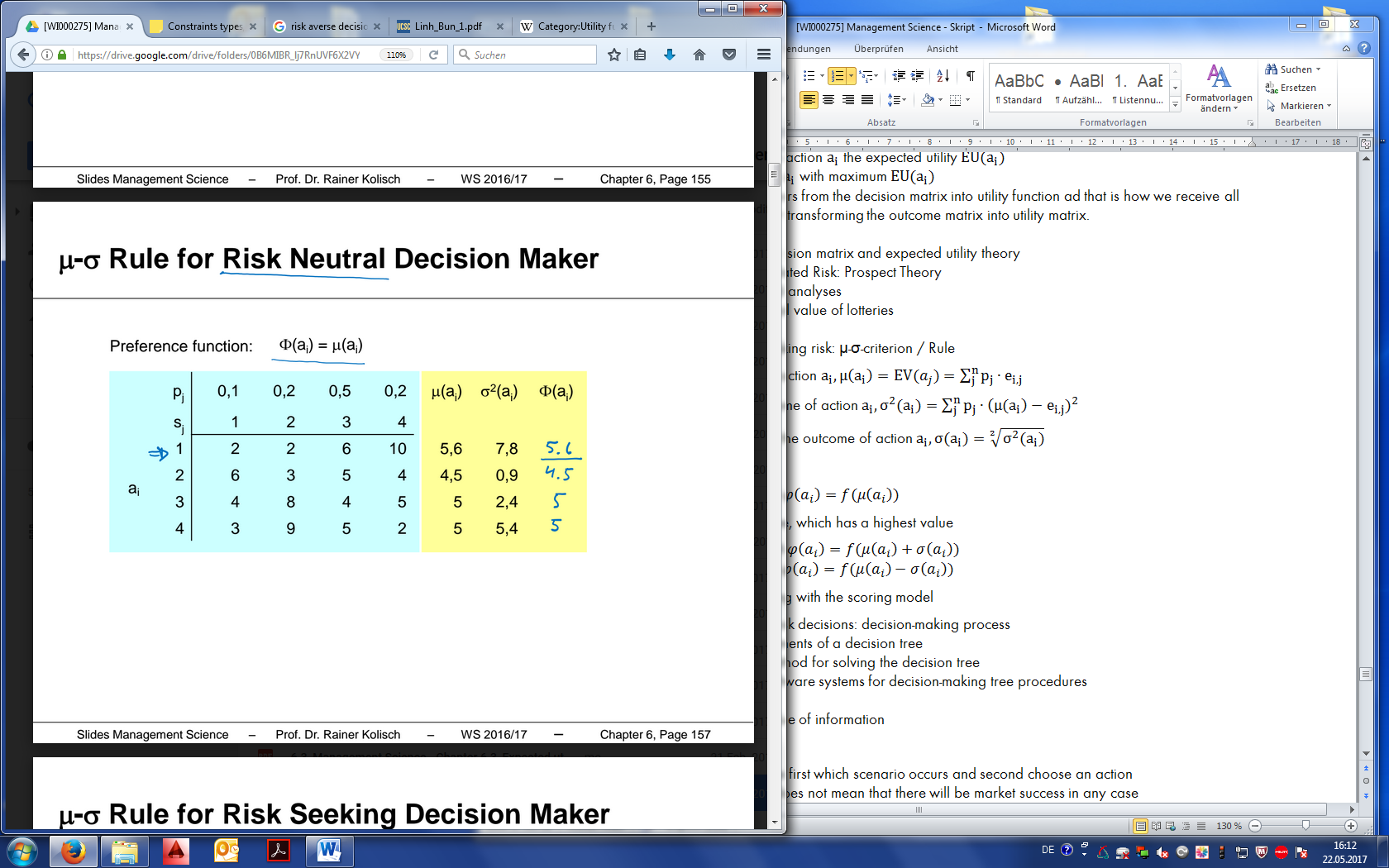
Expected outcome of action

Variance of the outcome of action

Standard deviation of the outcome of action

Preference function

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



* + Risk neutral DM
  + Risk seeking DM
  + Risk averse DM

**EXPECTED UTILITY VS μ-σ RULE**

Bothe methods provide with equal solutions if:

* Risk-neutral DM
* Quadratic utility function and appropriate calibration of preference function
* 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.

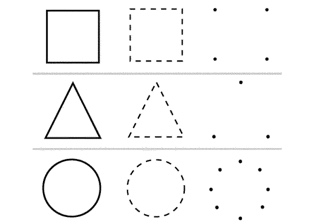
* 1. **MULTI-STAGE DECISION MAKING**
* Risk
* Single goal
* Single DM
* Dynamic environment
* Game with a nature

**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

* 1. **MULTI-CRITERIA DECISION MAKING (UTILITY ANALYSES)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * Deterministic * Multiple goals * Single DM * Static environment   **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 4. Determine the utility function for each target: 5. Find best and worst 6. Utility function 7. and 8. Determination of expected utility for each alternative 9. Alternative with the highest S = optimal solution   **Methods:**   * Scoring model * Analytical hierarchy process (AHP) * Multi-attribute utility theory | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |

– goal (ziel)

– 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

|

**Step 1. Criteria weights**

1. The value of each goal is determined according to the following scale:

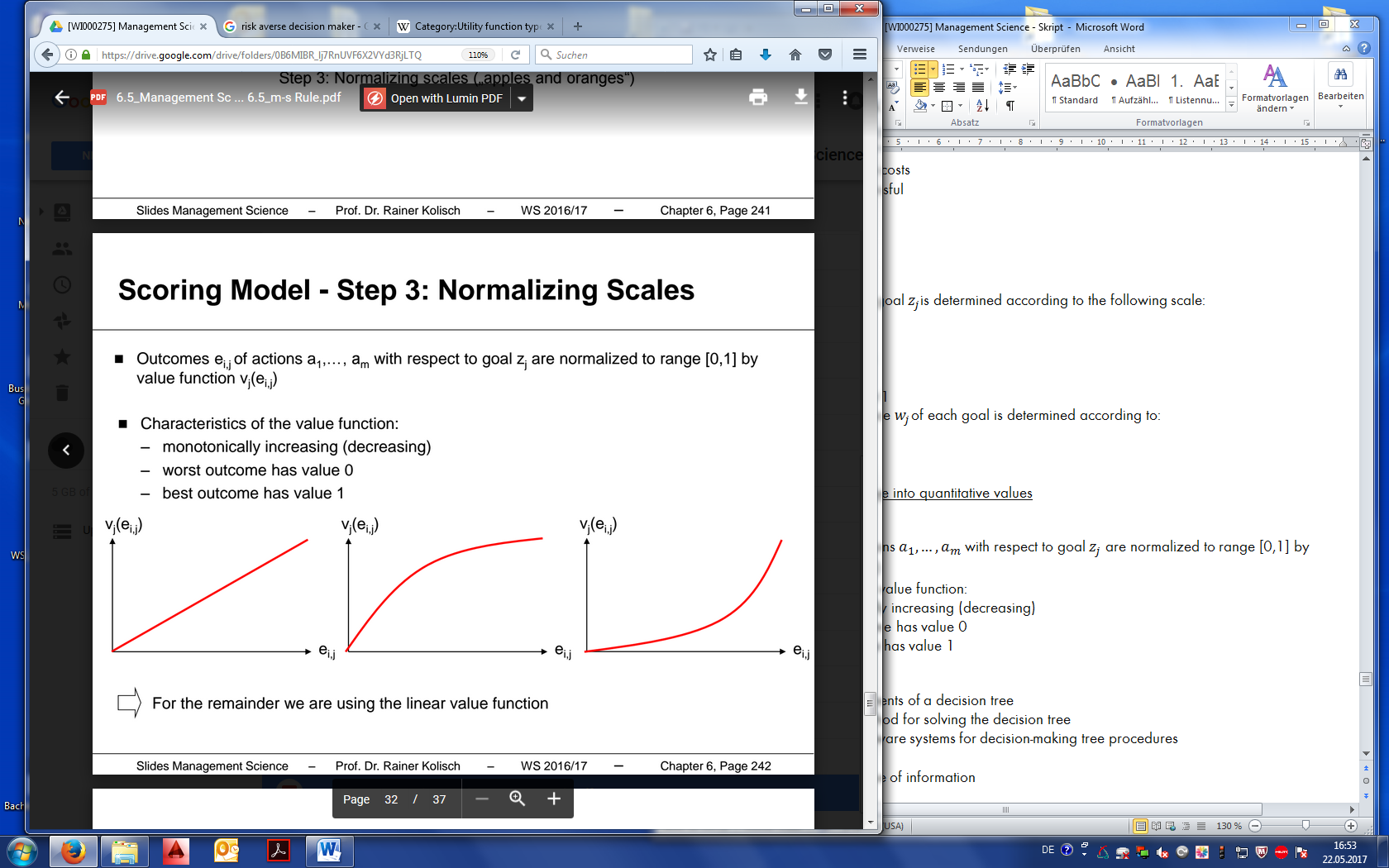
* Very important 5
* Important 4
* Average important 3
* Less important 3
* Marginally important 1

1. The relative importance of each goal is determined per:

**Step 2. Transforming qualitative into quantitative values**

**Step 3. Normalizing scales**

* Outcomes of actions with respect to goal are normalized to range [0,1] by value function
* 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 

**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.   * Stop its optimal solution.   Otherwise, continue only if there is at least one .   1. Choose pivot column      * Stop its unbounded solution.   Continue only if there is at least one   1. Choose pivot row , 0 also ok. 2. Change basis variable and re-establish a canonical form. 3. Go to step (1). | Notation:  - number of variables  - variable index  – current coefficient of objective function (j)  – number of coefficients  – constraint index  - coefficient of elements (i, s) in the current tableau  – current right-hand-side |
| (!) 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

**DUAL SIMPLEX ALGORITHM**

|  |  |
| --- | --- |
| Condition:  LP is dual feasible (=primal optimal, )  and  dual non-optimal (=primal infeasible, ).   * Stop its optimal solution   If there exist some , then continue     1. Chose pivot row      * Stop there is no feasible solution   Continue only if there is at least one     1. Choose pivot column 2. Replace the basic variable and re-establish canonical form. 3. Go to step (1). | Notation:  – number of variables  – variable index  – current coefficient of objective function (j)  – number of coefficients (constraints)  – constraint index  – coefficient of elements (i, s) in the current tableau  – current right-hand-side |
| (!) The dual Simplex ends with one of the following results:   * an optimal solution * there is no feasible solution | 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. |

