Dario Monopoli

INTRO TO OR SUMMARY

Lecture 1 – Introduction

When building a model always have clearly stated the:

1) Decision variables: represent the decision to be made, examples: $x_1, x_2, \dots, x_n$.
2) Objective function: performance measure expressed as a function of the decision variables. It is what we either want to maximize or minimize (e.g., profit function, cost function, etc.).
3) Constraints: mathematical expressions for the restrictions (either equalities or inequalities). Constants as constraints in the equations are called parameters of the model.

Post-optimality analysis: also called what-if analysis, it is an analysis done after finding an optimal solution. What-if: what would happen if different assumptions were made?

Lecture 2 – Basics of Linear Programming

In linear programming, all mathematical functions that appear in the model (objective function and constraints) are linear.

Constraints can never be < or >, only large or smaller equal to a value.

Nonnegativity constraints: x1>=0 and x2>=0

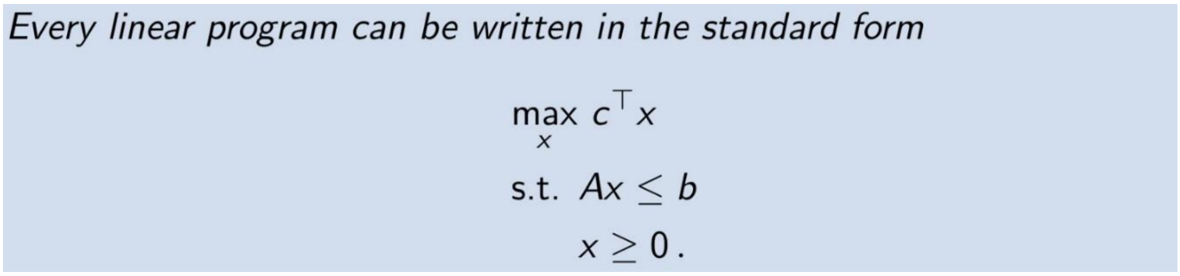
The ABC of optimization:

A) Adjust: decision variables
B) Best: objective function (quantity to maximize or minimize)
C) Constraints (e.g., resources)

Negative sales in a real-world context, e.g. in the manufacturing industry are possible if the consumer can return items/products. Leasing is also an example of negative sales.

Graphical solution - general summary and procedure:

1) Plot constraints (one line per constraint)
2) Identify the feasible region (there can also be cases where there is no feasible region)
3) Draw lines along which the objective function is constant ("iso-objective lines")
4) Find an optimal solution (if exists)



Every linear program can be written in the standard form

$$\begin{aligned} \max_x \ & c^\top x \\ \text{s.t.} \ & Ax \le b \\ & x \ge 0\,. \end{aligned}$$

Dario Monopoli

Linear program: standard form

- Min $2x_1 - 5x_2 \Rightarrow$ Max $-2x_1 + 5x_2$
- $-x_1 + 3x_2 \geq 10 \Rightarrow x_1 - 3x_2 \leq -10$
- $x_1 = 3 \Rightarrow x_1 \leq 3$ & $x_1 \geq 3 \Rightarrow x_1 \leq 3, -x_1 \leq -3$
- x_1: unrestricted in sign (it can also be negative) i will define a variable $x_1 \to x_1' - x_1''$
  $x_1' \geq 0$
  $x_1'' \geq 0$

- $\min_x \ c^\top x \Leftrightarrow \max_x -c^\top x$
- $a^\top x \geq b \Leftrightarrow -a^\top x \leq -b$
- $a^\top x = b \Leftrightarrow a^\top x \leq b, a^\top x \geq b \Leftrightarrow a^\top x \leq b, -a^\top x \leq -b$
- "unrestricted in sign" constraints for $x \Leftrightarrow x = \bar{x} - \bar{\bar{x}}, \bar{x} \geq 0, \bar{\bar{x}} \geq 0$

The most common application of linear programming: allocating resources to activities. Resources are what we want to make use of, and they are limited. Activities are processes that make use of *resources* needed to conduct the *activity* provided. Choose levels of activities that achieve the best possible value of your performance measure!

<u>Linear Programming: common notation</u>

f	$=$	value of overall measure of performance
x_j	$=$	level of activity j (for $j = 1, 2, \cdots, n$)
c_j	$=$	increase in f that results from each unit increase of x_j
b_i	$=$	available amount of resource i (for $i = 1, 2, ..., m$)
a_{ij}	$=$	amount of resource i consumed by each unit of activity j

Resource Usage per Unit of Activity

Resource	Activity 1	Activity 2	$\cdots$	Activity n	Amount Available (of Resource)
1	a_{11}	a_{12}	$\cdots$	a_{1n}	b_1
2	a_{21}	a_{22}	$\cdots$	a_{2n}	b_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
m	a_{m1}	a_{m2}	$\cdots$	a_{mn}	b_m
Contrib. to f	c_1	c_2	$\cdots$	c_n	

- $x_1, x_2, \ldots, x_n$ are called the decision variables
- the values of c_j, b_i, and a_{ij} (for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$) are the input parameters for the model

A feasible solution is a solution that satisfies all the constraints. An infeasible solution is a solution for which at least one constraint is violated.

The feasible region is the collection of all feasible solutions. It is the set of all points that satisfy all constraints.

It can be possible that a linear program has no feasible solution.

An optimal solution is a feasible solution that has the most favorable value of the objective function.

The most favorable value is the largest value if the objective function is to be maximized, whereas it is the smallest value if the objective function is to be minimized. Some problems will have just one optimal solution, others will have more than one. Any problem having multiple optimal solutions will have an infinite number of optimal solutions (since we suppose there is no integer constraint). If a problem has multiple optimal solutions, then it doesn't necessarily mean that feasible region is bounded, it can be unbounded.

In two cases there is no optimal solution:

1) No feasible solution exists (infeasible problem)
2) The value of the objective function can be improved indefinitely in the favorable direction (unbounded problem). This can happen where there are for example no constraints, so the optimal solution is approximated to infinity. $\rightarrow$ but a feasible solution exists (e.g. $-\infty$ or ∞)

A corner-point feasible (CPF) solution is a solution that lies at a corner of the feasible region (also called extreme points or vertices).

Consider a linear program where an optimal solution exists and there is a bounded nonempty feasible region. Then:

- the problem possesses CPF solutions and at least one optimal solution.

- the best CPF solution is an optimal solution (not when feasible region is unbounded)

- at least one optimal solution must lie on the boundary of the feasible region

- if a problem has exactly one optimal solution, then it is a CPF solution

- if a problem has multiple optimal solutions, then at least one must be a CPF solution

- if a problem has multiple optimal solutions and the feasible region is bounded, then at least two must be PCF solutions.

An optimal solution is not always a CPF solution, for example when there are an infinite amount of optimal solutions.

A CPF solution might not be the only optimal even if none of its adjacent CPFs are better (measured by the value of the objective function), this is the case when there are multiple optimal solutions and all adjacent CPFs have the same objective value.

If a feasible solution is optimal but is not a CPFs, then infinitely many solutions exist.

If the optimal value of the objective function is equal ~~to~~ at two different feasible points, then all points on the line connecting these two feasible points are feasible and are all optimal solutions.

If a problem has n variables, then the simultaneous solution of any set of n constraint boundary equations is a CPF solution only if the problem is feasible.

Fundamental Theorem of linear programming: if a linear program has an optimal solution, then it has an optimal solution at an extreme point of the feasible set.

If we increase the value of the objective function any further and the iso-objective line will no longer touch a single point in the feasible region we found an optimal solution, which lies at a corner point and is unique.

<u>Assumptions of linear programming</u>

Proportionality assumption: the contribution of each activity to the value of the objective function (or left-hand side of a functional constraint) is proportional to the level of the activity. If the assumption does not hold, one must use nonlinear programming

Additivity assumption: every function in a linear programming model is the sum of the individual contributions of the activities.

Divisibility assumption: decision variables in a linear programming model may have any value (linear values). It assumes that activities can be run at fractional values.

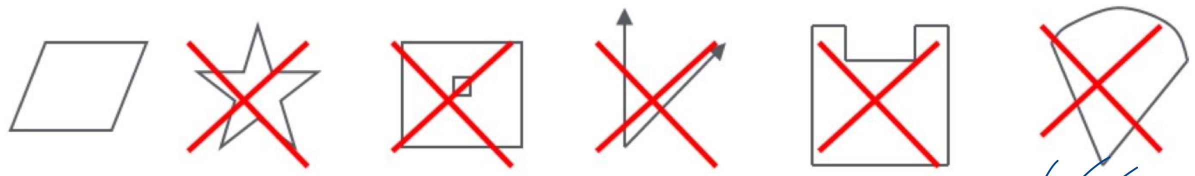
Certainty assumption: the value assigned to each parameter of a linear programming model is assumed to be a known constant (although it's not very realistic). Seldom satisfied precisely in real applications (sensitivity analysis used).

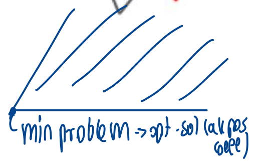
<u>Lecture 3 – Basics of Linear Programming (cont'd), feasible regions and potential outcomes</u>

A feasible region can be bounded, unbounded, or of a lower dimension than the number of variables (e.g., line segments and points).

The feasible region is a *convex polyhedron*: convex: there are no holes or indentations in the graph, there cannot be drawn a line between two points inside the graph that goes out of the graph. Polyhedron: intersection of half-spaces (with flat sides). A half space (in 3D) is either one of the two parts into which a plane (flat, two-dimensional surface) divides the three-dimensional Euclidean space.

Polytope: nonempty bounded polyhedron.





An optimal solution can exist even though the feasible region is unbounded.

There are exactly four possible outcomes of an LP:

1) Infinitely many optimal solutions exist (for example the points on a line between two CPFs are all optimal solutions)
2) A unique optimal solution exists
3) No optimal solution exists, as the problem is unbounded
4) No optimal solution exists, as the problem is infeasible

For the set of all CPF solutions of an LP:

- there are only a finite number of CPF solutions

- If a CPF solution has no neighboring CPF solution that gives a better objective value, then there are no better CPF solutions anywhere. Therefore, such a CPF solution is an optimal solution.

The simplex method:

1) Start at some feasible corner point.
2) Check if any neighboring corner point improves the objective value: If yes move to that better point and repeat, else the current corner point is optimal.

If something can be met during later periods (e.g., demand), then we say that it can be backlogged.

<u>Lecture 4 – Simplex Method</u>

Two adjacent CPFS have the same constraint boundary. If we get points with the same optimal value, we continue to search further (but never go back).

In the simplex method, nonnegative constraints remain inequalities (treated separately). The other constraints are functions to be turned into constraints. The augmented solution is the solution for the original decision variables augmented by the slack variables. The original (without slack variables) and augmented forms represent the same problem. The augmented form allows to easier identify CPF solutions. A slack variable is equal to 0, if and only the solution lies on the constraint boundary. A slack variable is > 0, if and only if the solution lies on the feasible side of the constraint boundary. A slack variable is smaller than 0, if and only if the solution lies on the infeasible side of the constraint boundary. The constraint boundary equation for any constraint is obtained by replacing its <=, =, or >= sign with an = sign. Slack variables are added to a model to transform the inequality constraints into equality constraints.

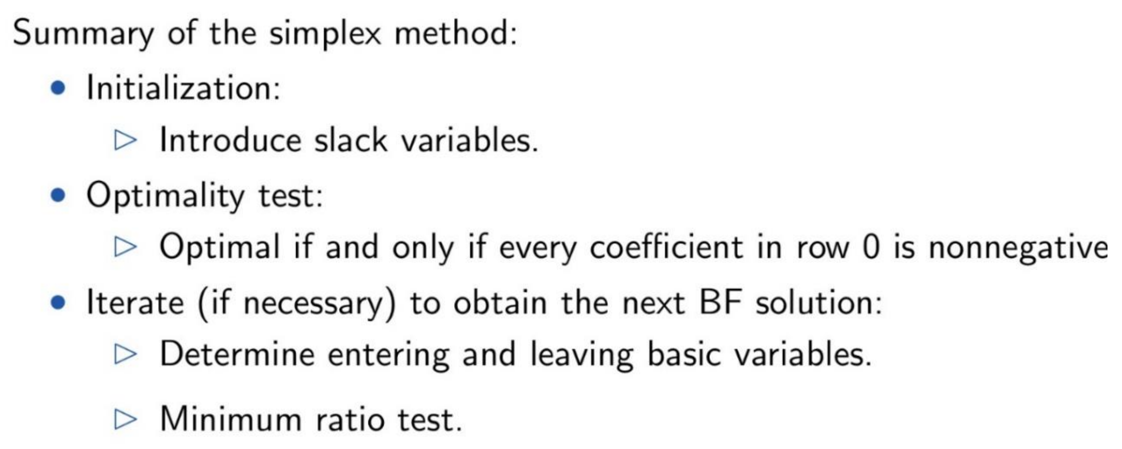
A basic solution is an augmented corner-point solution. There are as many basic solutions as the number of CPFs. A basic feasible (BF) solution is an augmented CPF solution. A unique solution happens when the number of constraints is equal to the number of decision variables. → i think it depends

In an augmented form of system of constraints, the number of degrees of freedom is equal to the number of variables minus the number of equations. N variables are set to zero, where n is the degree of freedom. The variables set to zero are called nonbasic variables, the others are called basic variables and they are the "remaining" decision variables.

Each variable is designated as either a nonbasic variable or a basic variable. The number of basic variables equals the number of functional constraints/equations. If all basic variables are nonnegative, the basic solution is a BF solution. Nonbasic variables must satisfy the constraints, see slide 13.

The tabular form records only the essential information, i.e., the coefficients of the variables, the constants of the right-hand sides of the equations and the basic variable appearing in each equation. The negative coefficients in the matrix or tabular form mean that the variables with those coefficients can be improved (under constraints) in the objective function.

Dario Monopoli


Summary of the simplex method:

- Initialization:
  - ▷ Introduce slack variables.
- Optimality test:
  - ▷ Optimal if and only if every coefficient in row 0 is nonnegative
- Iterate (if necessary) to obtain the next BF solution:
  - ▷ Determine entering and leaving basic variables.
  - ▷ Minimum ratio test.

For a linear programming problem with n decision variables:

- Each CPF solution lies at the intersection of n constraint boundaries.

- A system of n constraint boundary equations might have no solution or yield an infeasible one.

Adjacent CPF solutions lie on a line segment forming an edge of the feasible region.

Properties of CPF solutions (same holds for basic feasible solutions (BFs)):

- if there is exactly one optimal solution, then it must be a CPF solution.

- If there are multiple optimal solutions (bounded feasible region), then at least two must be adjacent CPF solutions.

- There are only a finite number of CPF solutions

- If a CPF solution has no adjacent CPF solutions that are better (measured by the objective function Z), then:

There are no better CPF solutions anywhere. The CPF solution must be an optimal solution.

Steps:

pay attention to negative coefficients

1) pick the variable with the highest negative coefficient. This is the entering variable.
2) choose the variable with the lowest MRT for that variable/column. This is the leaving variable.
2b) Divide row by value
3) Reduce the column and that basis variable to 1 and from there bring it to zero in the objective function.
4) Repeat the steps till all coefficients in the objective function are positive (which means that the solution is optimal)
5) Variables for which value in tableau in row 0 is 0 are basic variables

If there is a tie for the entering basic variable, we can choose. If there is a tie for the leaving basic variable it matters theoretically which one we choose, but rarely in practice, so we can choose arbitrarily. If there is a condition of no leaving basic variable, then the objective function Z (also denoted by f) is unbounded. For an unbounded problem there is at least a column in the simplex tableau for which every element is nonpositive.

If we have nonpositive column in the simplex tableau (all elements in that column including element in row zero are nonpositive), then the problem is unbounded. When we have a negative value in the simplex tableau, its MRT is infinitely large, since the values in the next tableau would be infeasible if we picked that variable as leaving variable.

If there are multiple solutions the simplex method stops after one optimal BF solution is found.

An unbounded solution in simplex has an unbounded MRT.

The constraint boundary equation is the equation obtained by replacing the inequality sign in the constraint equation with an equals sign. It defines a hyperplane in n-dimensional space. A system of n constraint boundary equations might have no solution or yield an infeasible solution. → repetition

Unrestricted in sign: variable's value can be positive/negative/zero. No sign restriction.

Adjacent CPF solutions lie on a line segment forming an edge of the feasible region.

Each basic solution has m basic variables (m is the number of constraints), the rest of the variables are nonbasic set to zero. In a linear problem the number of slack variables is equal to the number of constraints in the LP. also if there are = constraints in LP problem specification you have identity in simplex tableau/matrix

The vector C_B is the vector of coefficients for the basic variables.

Big M method (for helping to solve reformulated models): denote M as some really big number and revise the objective function by subtracting M times each artificial variable. Maximizing this objective function will force each artificial variable to be zero. The optimal solution is also optimal for the real problem.

Two-Phase Method (alternative to big m method): phase 1 revises the objective function to minimize the sum of the artificial variables. Applying the simplex method then yields a BF solution for the real problem, using this as the initial BF solution, phase 2 then applies the simplex method to solve the real problem.

Lecture 5 – Duality Theory and Sensitivity Analysis

Duality: every LP can be associated with another LP called its dual problem.

In an LP model in which the number of variables is smaller than the number of constraints, computational savings are realized by solving the dual problem.

Primal problem Maximize $Z = \sum_{j=1}^{n} c_j x_j$ subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{for} \quad i = 1, 2, \ldots, m$$

and

$$x_j \ge 0 \quad \text{for} \quad j = 1, 2, \ldots, n.$$

Dual problem Minimize $W = \sum_{i=1}^{m} b_i y_i$ subject to

$$\sum_{i=1}^{m} a_{ij} y_i \ge \overset{\substack{\text{vector of coefficients} \\ \downarrow}}{c_j} \quad \text{for} \quad j = 1, 2, \ldots, n$$

and

$$y_i \ge 0 \quad \text{for} \quad i = 1, 2, \ldots, m.$$

Primal problem: maximization, Dual problem: minimization

Minimum value of primal problem is same as maximum value of dual problem and viceversa.

The dual problem uses the same parameters as the primal, but:

a) Coefficients in the objective function of the primal problem become the right-hand side for the dual problem
b) Right-hand side values of the primal problem become the coefficients in the objective function in the dual problem
c) Coefficients in constraints appear in both primal and dual problems.

The right-hand side values (parameters) of either problem are objective function coefficients for other problem.

Coefficients in the objective function of either problem are the right-hand side values for the other problem.

The dual problem of the dual problem is again the primal problem.

Shadow price: corresponds to the rate at which the value in the optimal objective value increases per unit of increase in available resource (RHS, constraint). The shadow price of the first unit is zero. The value of the variables for the optimal solution of the dual problem directly reflect shadow prices of the corresponding constraints in the primal problem and viceversa.

A shadow price of zero indicates that the constraint associated with that shadow price is not binding. If a constraint has a shadow price of zero, then increasing or decreasing the quantity specified in that constraint would not affect the value of the objective function. A shadow price different from zero means that constraint is binding. When the shadow price is negative, it means that relaxing the constraint will result in a decrease in the objective function. However, this does not necessarily mean that the constraint is binding.

Dario Monopoli

Strong Duality Theorem: if an optimal solution of the primal problem exists, then an optimal solution for the dual problem exists (and vice versa) and the values of the optimal solutions are always equal. If the dual has same value as primal, then the points used to come up with that solution are optimal for both the primal and the dual problem. The duality gap is zero.

Weak Duality Theorem: the optimal value of the dual problem is always greater than or equal to the optimal value of the primal problem (for any feasible solution x of the primal problem and any feasible solution y of the corresponding dual problem). If the optimal value of the dual problem is known, it can be used to determine an upper bound (for maximization, or lower for primal minimization problems) on the optimal value of the primal problem. This can be useful in cases where it is difficult to find the optimal solution to the primal problem, as it provides a way to determine if the current solution is close to optimal. The duality gap is greater than zero. Weak duality always holds for convex optimization problems.

Weak duality and strong duality cannot both hold at the same time, because they are mutually exclusive properties of convex optimization problems.

Duality gap: difference between the optimal value of the ~~dual~~ primal problem and the optimal value of the ~~primal~~ dual problem.
If the duality gap is zero, it means that the optimal solutions of the primal and dual problems are the same. If the duality gap is non-zero, it means that the optimal solutions of the primal and dual problems are different. In most cases, the duality gap is expected to be non-negative, meaning that the optimal value of the dual problem is equal to or greater than the optimal value of the primal problem. If the duality gap is negative, it can indicate that there is a problem with the optimization algorithm or with the formulation of the primal and dual problems. If the optimal value of the dual problem is larger than the optimal value of the primal problem, then the duality gap will be negative.

Duality theorem: The following are the only possible relationships between the primal and dual problems.

1. If one problem has feasible solutions and a bounded objective function (and so has an optimal solution), then so does the other problem, so both the weak and strong duality properties are applicable.

2. If one problem has feasible solutions and an unbounded objective function (and so no optimal solution), then the other problem has no feasible solutions (infeasible). Having both problems unbounded is possible when there is no upper or lower bound on the decision variables, or when there is no constraint on the variables in the problem. In this case, the objective function can take on arbitrarily large or small values, depending on the values of the decision variables.

3. If one problem has no feasible solutions, then the other problem has either no feasible solutions or an unbounded objective function.

P \ D	unbounded	has solution	not feasible
unbounded	no	no	possible
has solution	no	same values	no
not feasible	possible	no	possible

P \ D	feasible	not feasible
feasible	both have solutions; values equal	P unbounded
not feasible	D unbounded	possible

An optimal solution of the dual is a feasible solution of the primal. An optimal solution of the primal is a feasible solution for the dual.

The slack of a constraint is calculated as the difference between the rhs and lhs of that constraint. The slack tells how much room there is for the decision variables before they violate the constraint. A positive slack of = n means that the decision variables can increase up to n before the constraint is violated. If a slack is zero then it means that the constraint is binding, otherwise it is nonbinding.

Complementary slackness: if a slack variable is greater than 0, the shadow price (or value) of its corresponding constraint is equal to 0. If a slack variable is zero, then shadow price of its corresponding constraint is different than 0. The term complementary slackness refers to a relationship between the slackness in a primal constraint and the slackness (positivity) of the associated dual variable. It asserts that there cannot be slack in both a constraint and the associated dual variable. This means that if one of the variables in the primal problem has slack in its nonnegativity constraint (basic variable > 0), then the corresponding dual variable must have no slack (nonbasic variable = 0).

Theorem 2 ***Complementary Slackness*** *Assume problem (P) has a solution x^* and problem (D) has a solution y^*.*

1. *If $x_j^* > 0$, then the jth constraint in (D) is binding.*
2. *If the jth constraint in (D) is not binding, then $x_j^* = 0$.*
3. *If $y_i^* > 0$, then the ith constraint in (P) is binding.*
4. *If the ith constraint in (P) is not binding, then $y_i^* = 0$.*

Sensitivity analysis determines how the optimal solution of an LP reacts to: changes in the parameters of the objective function, and changes in the right-hand side of constraints. In sensitivity analysis, we identify which parameters are most sensitive, and for the non-sensitive ones, we determine the range over which the optimal solution remains unchanged.

It answers the question: How robust is the optimal solution to changes in the coefficients of the objective function?

In a Sensitivity report, when both the allowable increase and the allowable decrease are greater than zero for every objective function coefficient, then the optimal solution in the “final value” column of the sensitivity report in Excel is the only optimal solution.

Reduced cost: marginal rate at which the value in the objective function (objective value) changes if you force a variable to increase by 1. If the optimal value of a variable is strictly positive, then the reduced cost is zero. If the optimal value of a variable is zero and the reduced cost corresponding to the variable is also zero, then there exists at least one other CPF solution. The value of this variable will be positive at one of the other optimal corners. The reduced cost associated with a variable corresponds to the shadow price of the corresponding nonnegativity constraint. If the shadow price of a constraint is non-zero, the reduced cost of the decision variable associated with the constraint must be zero. On the other hand, if the shadow price of a constraint is zero, the reduced cost of the decision variable associated with the constraint can be non-zero.

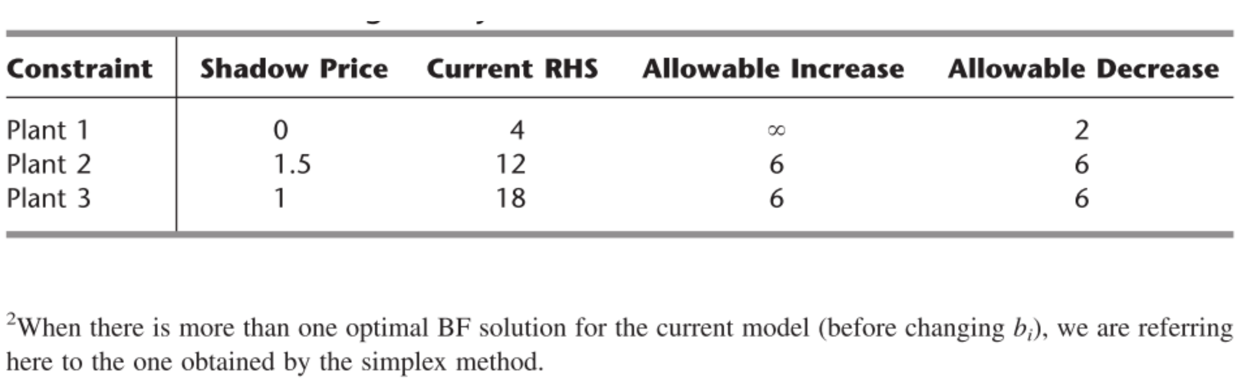
By changing the rhs (constraints), either increasing or decreasing the value of a constraint (e.g. budget), if the slope stays the same the basic variables (decision variables in basis) of the optimal solutions don't change, only their values change. Budget constraints can change in praxis due to external factors.

In the standard form (no negative coefficient) of a Linear Problem, with a larger rhs the value of the objective function is at least as good as the one before the increment.

The 100 Percent Rule for Simultaneous Changes in Right-Hand Sides: The shadow prices remain valid for predicting the effect of simultaneously changing the right-hand sides of some of the functional constraints as long as the changes are not too large. To check whether the changes are small enough, calculate for each change the percentage of the allowable change (increase or decrease) for that right-hand side to remain within its allowable range. If the sum of the percentage changes does not exceed 100 percent, the shadow prices will still be valid. (If the sum does exceed 100 percent, then we cannot be sure.)

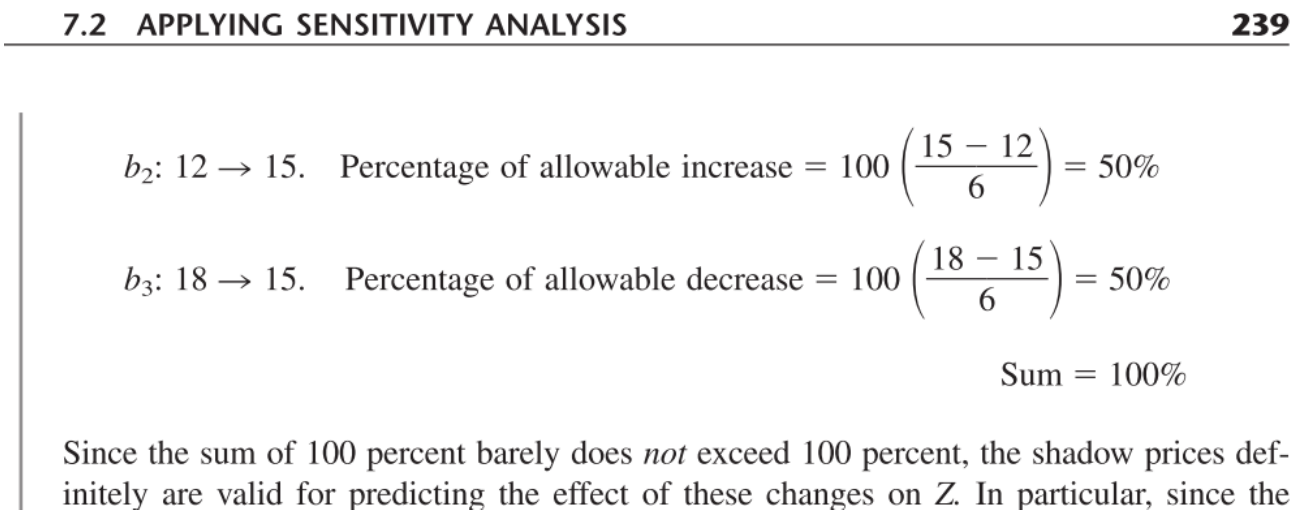
To find the decision variable that has the smallest margin for error we have to find the ones that have the least flexibility in allowable increase and decrease (infinite values are not to be considered).

Allowable range for decision variables: if allowable increase not infinity: allowable range <= objective coefficient + allowable increase. If allowable decrease not infinity: allowable range >= objective coefficient - allowable decrease.



Constraint	Shadow Price	Current RHS	Allowable Increase	Allowable Decrease
Plant 1	0	4	∞	2
Plant 2	1.5	12	6	6
Plant 3	1	18	6	6

[2]When there is more than one optimal BF solution for the current model (before changing b_i), we are referring here to the one obtained by the simplex method.



7.2 APPLYING SENSITIVITY ANALYSIS 239

$$b_2: 12 \rightarrow 15. \quad \text{Percentage of allowable increase} = 100\left(\frac{15-12}{6}\right) = 50\%$$

$$b_3: 18 \rightarrow 15. \quad \text{Percentage of allowable decrease} = 100\left(\frac{18-15}{6}\right) = 50\%$$

$$\text{Sum} = 100\%$$

Since the sum of 100 percent barely does *not* exceed 100 percent, the shadow prices definitely are valid for predicting the effect of these changes on Z. In particular, since the

<u>Lecture 6 – Transportation Problems and Assignment Problems</u>

Transportation problems solve how to optimally transport goods. Assignment problems solve how to assign people to tasks. Both problems are cases of the minimum cost flow problem (MCFP).

The concern of transportation problems is distributing any commodity from sources to destinations. The requirements assumption of transportation problems states:

a) Each source has a fixed supply
b) Entire supply must be distributed to the destinations
c) Each destination has a fixed demand
d) Entire demand must be received from the sources

The cost assumption: The cost of distributing units from any source to any destination is directly proportional to the number of units distributed. Therefore, this cost is just the unit cost of distribution times the number of units distributed -> costs are linear.

The number of sources doesn't have to be equal to the number of destinations.

Feasible solutions property: a transportation problem will have feasible solutions if and only if:

Dario Monopoli

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$$

Balanced transportation problem: supply and demand are equal.

A balanced transportation problem always has a feasible solution because the total supply of goods is equal to the total demand for goods. This means that it is possible to satisfy the demand at all of the destinations using the available goods from the sources.

Transportation simplex method is more efficient than the general simplex method. For transportation problems with m sources and n destinations the number of basic variables is m + n – 1. In general simplex method number of basic variables is m + n.

General Procedure of Transportation simplex method:

1) If the problem is unbalanced (demand not equal supply), balance it (introduce dummy variable)
2) Find basic feasible solution by using one of the following methods:
   a) Northwest corner method: begin in the upper left/northwest corner of the transportation tableau
   b) Vogel's approximation method: begin by computing for each row and column a "penalty" equal to the difference between the two smallest costs in the row (resp. column) (abs if negative). Next find the row or column with largest penalty and allocate all least value between demand and supply for those coordinates, cancel out either row or column depending on which quantity has been satisfied and then repeat the process by recomputing the penalties again.
   c) Minimum-cost methods: use the shipping costs to produce a bfs that has a lower total cost.
3) Use the fact that u1 = 0 and ui + vj = cij for all basic variables to find the [u1, …um, v1, …vn] for the current basic feasible solution (bfs).
4) If ui + vj – cij <= 0 for all nonbasic variables, then the current bfs is optimal. If this is not the case, then we insert the variable with the highest positive ui + vj – cij into the basis using the pivoting procedure. This yields a new bfs.
5) Using the new bfs, return to steps 3 and 4.

In a particular iteration of the simplex method, if there is a tie for which variable should be the leaving basic variable, then the next BF solution must have at least one basic variable equal to zero. This is true since the ratio test tells how far the entering basic variable can be increased before one of the current basic variables drops below zero. If there is a tie then both variables drop to zero at the same value of the entering basic variable, but since only one variable can become nonbasic in any iteration, the other will remain in the basis even though it will be zero.

If there is no leaving variable at some iteration, then the problem is unbounded and the entering basic variable can be increased indefinitely, but the problem remains feasible.

All the basic variables in the row 0 of the final tableau of the Simplex method have a coefficient of zero.

The assignment problem is a special case of transportation problems where assignees are assigned to perform tasks (employees, machines, vehicles, plants, time slots). Assignment and transportation problems can be solved with a solver for integer programming and also with the simplex method.

Major assumptions of assignment problem:

1) The number of assignees and the number of tasks is the same (this number is denoted by n).
2) Each assignee is assigned to exactly one task.
3) Each task is to be performed by exactly one assignee.
4) There is a cost cij associated with assignee i (i = 1,2,…,n) performing task j (j=1,2,…,n).
5) The objective is to determine how all n assignments should be made to minimize the total cost.

Procedure (heuristic approach):

1) Consider the highest score, match the resource with activity of highest score, eliminate the resource and the activity (job) from the table.
2) Choose the next highest score. If no scores are available stop, all jobs have been assigned to all resources. Otherwise, return to step 1.

The objective function is to maximize the total matching score of the assignments that satisfy the job and the resource constraints.

The decision variable xr,j = 1 in the constraints represents that resource r is assigned to job j, otherwise 0.

Assignment problem model:

$$\text{Minimize} \qquad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1, 2, \ldots, n,$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for } j = 1, 2, \ldots, n,$$

and

$$x_{ij} \geq 0, \qquad \text{for all } i \text{ and } j$$
$$(x_{ij} \text{ binary}, \qquad \text{for all } i \text{ and } j).$$

Assignment problems can be seen as transportation problems: sources are assignees, destinations are tasks, number of sources m is number of destinations n. Every supply si = 1, every demand dj = 1.

Special algorithm for the assignment problem: Hungarian method, steps:

1. Subtract the smallest number in each row from every number in the row. Enter the results in a new table.

2. Subtract the smallest number in each column of the new table from every number in the column. Enter the results in another table.

3. Test whether an optimal set of assignments can be made. To do this, determine the minimum number of lines needed to cross out all zeros. If the minimum number of lines equals the number of rows, an optimal set of assignments is possible. Proceed with step 6. If not, proceed with step 4.

4. If the number of lines is less than the number of rows, modify the table as follows:

▪ Subtract the smallest uncovered number from every uncovered number in the table.

▪ Add the smallest uncovered number to the numbers at intersections of covering lines.

▪ Numbers crossed out but not at intersections of cross-out lines carry over unchanged to the next table.

5. Repeat steps 3 and 4 until an optimal set of assignments is possible.

6. Make assignments one at a time in positions that have zero elements. Begin with rows or columns with only one zero. Cross out both rows and columns after each assignment is made. Move on, with preference given to any row with only one zero not crossed out. Continue until every row and column has exactly one assignment and so has been crossed out. This will be an optimal solution for the problem.

Distribution channel: path through which all goods travel from the original vendor to the end consumer.

Transshipment problem: special case of the transportation problem in which shipping paths can include intermediate points.

Formulation:

$$Min\ \ z = \sum_i \sum_j c_{ij} x_{ij}$$

$$subject\ to:$$

$$\sum_j x_{ij} \leq S_i \qquad \forall i \in S$$

$$\sum_i x_{ij} \geq D_j \qquad \forall j \in D$$

$$\sum_i x_{ij} - \sum_i x_{ji} = 0\ \ \forall\ j \notin D, \notin S$$

Location consideration:

$$Min\, z = \sum_i \sum_j c_{ij} x_{ij} + \sum_i f_i y_i$$

$$\sum_j x_{ij} \leq S_i \quad \forall i \in S$$

$$\sum_i x_{ij} \geq D_j \quad \forall j \in D$$

$$\sum_j x_{ij} \leq M * y_i \quad \forall i \in S$$

Yi is a variable with only binary values, either 1 (open) or 0 (close). M has to be greater equal to the sum of all demands (the best number is to be equal to that) and has no upper bound. We could also take a different M for each cannery equation where M is greater equal to the capacity of the cannery in the corresponding equation.

Network problems consist of shortest-path problems, minimum spanning tree problems, maximum flow problems, minimum cost flow problems, CPM method of time-cost-trade-offs problems.

A minimum spanning tree algorithm is used to find a tree that connects all the vertices in a graph, while the shortest path algorithm is used to find the shortest path between two specific vertices in a graph. Every spanning tree has exactly n-1 arcs (edges).

Maximum flow problems: problems where the question is how to route various tram trips from the start node (source) to the finish node (sink) to maximize the flow. The constraints are the arc/edge capacities.

In a residual network, the edges of the original graph are replaced with new edges that represent the possible flow that can still be sent through the network. Each edge in the residual network has a capacity, which is the maximum amount of flow that can be sent through the edge, and a flow, which is the amount of flow that is currently being sent through the edge.

An augmenting path is a directed path from the source to the sink in the residual network such that every arc on this path has strictly positive residual capacity. The minimum of these residual capacities is called the residual capacity of the augmenting path because it represents the amount of flow that can feasibly be added to the entire path.

In the minimum cost flow problem, the network is directed and connected. At least one of the nodes is a supply node that generates a specific amount of flow and at least one of the other nodes is a demand node that absorbs a specific amount of flow. All the remaining nodes are transshipment nodes. The cost of the flow through each arc is proportional to the amount of that flow, where the cost per unit flow is known. The objective is to minimize the total cost of sending the available supply through the network to satisfy the given demand.

Formulation:

$$x_{ij} = \text{flow through arc } i \to j$$
$$c_{ij} = \text{cost per unit flow through arc } i \to j$$
$$u_{ij} = \text{arc capacity for arc } i \to j$$
$$b_i = \text{net flow generated at node } i$$

Minimize $Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

s.t.

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i \;\forall\, i$$

$$0 \le x_{ij} \le u_{ij} \quad \text{for each } i \to j.$$

The value of b_i depends on the nature of node i, where

$b_i > 0$ **if node i is a supply node**

$b_i < 0$ **if node i is a demand node**

$b_i = 0$ **if node i is a transshipment node**

Supply nodes: out flow > in flow

Demand nodes: in flow > out flow

Transshipment nodes: in flow = out flow

Net flow (out flow – in flow) for origin (supply node) has to be positive, for transshipment nodes it has to be zero and for demand nodes it has to be negative.

One type is the activity-on-arc (AOA) project network, where each activity is represented by an arc. A node is used to separate an activity (an outgoing arc) from each of its immediate predecessors (an incoming arc). The second type is the activity-on-node (AON) project network, where each activity is represented by a node. Then the arcs are used just to show the precedence relationships that exist between the activities. In particular, the node for each activity with immediate predecessors has an arc coming in from each of these predecessors.

<u>Forward pass</u>

ES = max {EF of all immediate predecessors}

EF = ES + activity time

<u>Backward pass</u>

LF = min {LS of all immediate following activities}

LS = LF – activity time

For the terminating node, the LS = its EF. The last calculated EF is the completion time (in days) of the project.

The activities for which EF – LF = 0 or LS – ES = 0 are called critical activities -> they cannot be delayed for completing the project at the same time. Critical activities have a slack of zero, meaning that they must be completed on time in order for the project to be completed on schedule. Slack, also known as float, is the amount of time that an activity can be delayed without affecting the completion time of the project.

Crashing an activity refers to taking special costly measures to reduce the duration of an activity below its normal value. Crash time: absolute minimum time the activity can take. Crash cost: cost for reaching absolute minimum time for the activity.

Crashing the project refers to crashing a number of activities in order to reduce the duration of the project below its normal value (reducing the duration of activities implies costs, e.g., for more workers).

Critical path method: can be used to determine the length of time required to complete a project. CP M also can be used to determine how long each activity in the project can be delayed without delayi ng the completion of the project.

A critical path is a path from the start node to the finish node which only consists of critical activities. If an activity is a critical activity, then any delay in the start of the activity will delay the completion time of the project. Critical activities have a shadow price of 1 (useful when sensitivity report is given). They are the activities we should invest in and crash for reducing the completion time of the projec t and not delay it.
The smallest margin of error for a connection between two nodes is min(allowable increase, allowabl e decrease). The greatest effort should be put ion the connection(s) with the smallest margin of error

Lecture 7 – Integer Programming

Variables only take integer values.

A (pure) integer (linear) programming problem (IP) is a linear program with all variables being integer constrained.

A mixed-integer programming problem (MIP) is a linear programming problem with some variables integer-constrained and some continuous (linear). If we relax an integer constraint, then the optimal objective value increases or remains the same (in a maximization problem, and it decreases or stays the same for a minimization problem).

Binary variables: yes-or-no decision, value either 1 (if decision is yes) or 0 (decision is no). Binary integer programming (BIP) problems are IP problems with only binary variables. N optimization problem with n variables which are all binary has 2^n feasible solutions. Binary problems are usually easier to solve than integer problems. Integer programming problems are generally harder to solve than linear programming problems. For IP problems, the number of integer variables is generally more important in determining the computational effort than is the number of functional constraints.

To solve IP problems with an approximate procedure you cannot apply simplex method with relaxed integer constraints and then round each noninteger value to the nearest integer since the results can be infeasible.

Integer variables are harder to handle than binary variables, since there are many more alternative ways to solve a problem, while with binary either one or the other.

Auxiliary binary variables: newly inserted variables that either take value 0 or 1. They are inserted when there are continuous decision variables. Continuous variables can take any value.

If we want to define auxiliary that takes a value of 0 it's better not to define them at all.

Integer programming problems can be solved using solvers that only solve linear programs.

The feasible region for an IP problem is a subset of the feasible region for its LP relaxation. If an optimal solution for the LP relaxation is an integer solution, then the optimal value of the objective function is the same for the LP relaxation problem and the IP problem.

If a noninteger solution is feasible for the LP relaxation, then the nearest integer solution is not automatically a feasible solution for the IP problem, it can be infeasible.

Groups of yes-or-no decisions often constitute groups of mutually exclusive alternatives: only one decision within a group can be yes.

Constraints:

a) Sum of corresponding binary variables = 1 (exactly one yes decision)
b) Sum of corresponding binary variables <=1 (at most one yes decision, can also be 0)

Contingent decisions: decisions that depend on other decisions. A decision is allowed to be yes if and only if the other decision is yes.

Multiple-choice constraint: constraint requiring that the sum of two or more binary variables is less or equal to one.

Alternative constraints: constraints for which one, but not necessarily both, must be satisfied.

If-then constraints: if f1(x1, x2,…,xn) > b1 => (= then) f2(x1,x2,…,xn) <= b2. This is logically equivalent to an alternative constraint: f1(x1, x2,…,xn) <= b1 and/or f2(x1,x2,…,xn) <= b2, where at least one must be satisfied.

k-Fold alternatives: problem where we must satisfy at least k of the constraints. Mj is chosen so that the ignored constraints will not be binding (binding constraint: left hand side is equal to the right hand side, the value of the decision variable associated with the binding constraint is optimal and cannot be improved without violating the constraint.).

Problem formulation:

$$f_j(x_1, x_2, \cdots, x_n) - M_j(1 - y_j) \le b_j \quad (j = 1, 2, \cdots, p)$$
$$\sum_{j=1}^{p} y_j \ge k$$
$$y_j = 0 \quad \text{or} \quad 1 \quad (j = 1, 2, \cdots, p)$$

That is, $y_j = 1$ if the jth constraint is to be satisfied, and at least k of the constraints must be satisfied.

Set covering problem: it involves several potential activities (such as flight sequences) and characteristics (such as flights). Each activity possesses some but not all the characteristics and the objective is to determine the least costly combination of activities that collectively covers each characteristic at least once.

at least once. Thus, let S_i be the set of all activities that possess characteristic i. At least one member of the set S_i must be included among the chosen activities, so a constraint,

$$\sum_{j \in S_i} x_j \geq 1,$$

is included for each characteristic i.

Set partitioning problems are problems for which exactly one member of each set S_i of all activities that possess characteristic I must be included among the chosen activities.

A related class of problems, called **set partitioning problems,** changes each such constraint to

$$\sum_{j \in S_i} x_j = 1,$$

so now *exactly* one member of each set S_i must be included among the chosen activities.

Integer programming problems have far fewer solutions to be considered compared to linear programming problems. Pure iP problems with a bounded feasible region are guaranteed to have just a finite number of feasible solutions, but this does not imply that a problem is solvable (finite numbers can be very large).

In the case of a BIP problem with n variables, the number of solutions to be considered is 2^n, where some of these solutions can be discarded because they violate the functional constraints. If n is increased by 1, the number of solutions is doubled (exponential growth of the size of the problem).

Depending on their characteristics, certain relatively small problems can be much more difficult to solve than some much larger ones. A problem with a smaller feasible region doesn't imply that it is easier to solve. Linear programming problems are considered in general easier to solve than IP problems. ~~Solutions to linear programming problems cannot be found with the simplex method nor the graphical one~~.

Rounding off is a good technique with large integer numbers, but not with small ones since a round-off could make a big difference in the result.

Branch and bound method: method to solve integer programs, for which the optimal solution is obtained by solving a sequence of linear programs. The algorithm starts by neglecting all integer constraints (linear relaxation). From the optimal solutions we found, if we have multiple linear variables that should take integer values, we pick the one with the largest decimal value and branch it, i.e., we consider two cases: nearest smaller integer and nearest larger integer -> branch step (with tree); we add two new integer constraints to the problem. The optimal objective value of the LP at some node is never greater than optimal objective value of the LP at parent node.
Stopping criteria:
- Solution of LP is feasible for Ip
- Opt. obj. val. Worse than best value found so far
- LP infeasible
If no stopping criteria is satisfied, the tree generates two more LPs to solve, each one with one additional constraint.

Depth-first strategy: always choose the deepest LP to solve next, this yields to a feasible solution quicker than breadth-first. We may stop searching when we found a feasible solution to the LP or the optimal objective value of the LP at current node is worse than obj. value of best feasible solution found so far somewhere else in the tree (since child nodes will only have worse objective function value since we add new would add new constraints when branching further). If there is only one variable that has the decimal portion, then we do the same but branch that one (we don't have to look at largest decimal portion since only one variable has it).

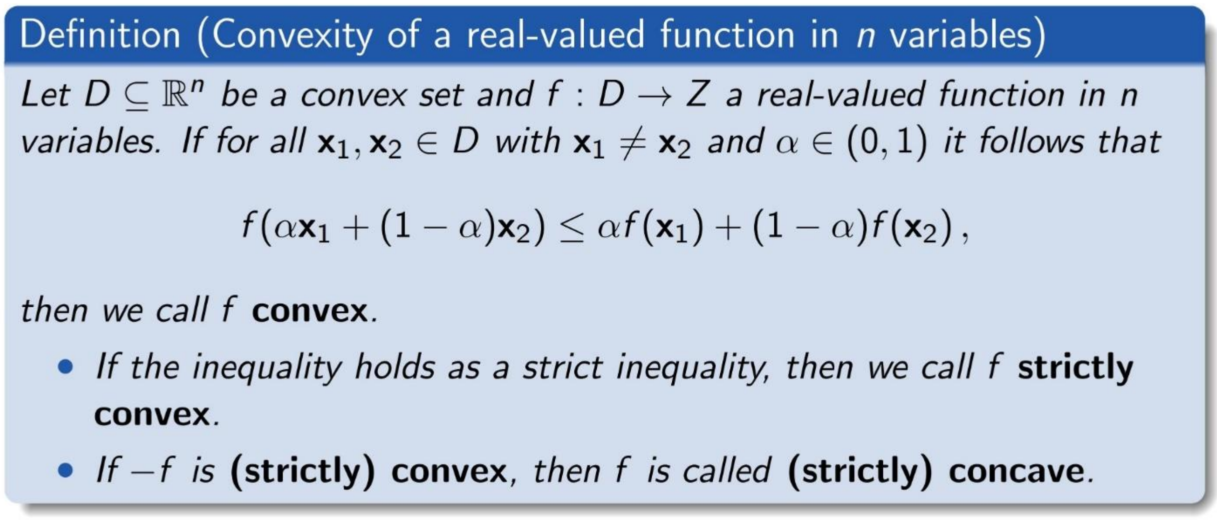
The optimal solution can only be determined at the end on the entire algorithm, this means when all the nodes in the tree are completed.

Lecture 8 – Nonlinear Programming

General optimization problem formulation:

$$\begin{aligned} \min_x \ & f(x) \\ \text{s.t.} \ & g_i(x) \leq 0 \quad \forall i = 1, \dots, m \\ & h_j(x) = 0, \quad \forall j = 1, \dots, p \end{aligned}$$

gi(x): inequality constraints; hj(x): equality constraints
If f and all gi are convex functions and all hj(x) = aj + bj^{T}x are affine functions, it is a convex program. The equality constraints don't need to be convex, but the inequality and the objective function must be. A convex problem has a convex objective function and convex constraints. To check if a problem is convex, make Hesse matrix of problem and check that it is positive definite (all subdeterminants > 0). If a problem has functions with odd power then they are most likely not convex.
If at least one of the conditions of the general problem formulation to be a convex problem is not satisfied, then we call it a general nonlinear program.
Nonlinear problems cannot be solved with the standard version of the simplex algorithm.
Some general nonlinear programs can be reformulated as equivalent convex programs. Otherwise, they are nonconvex programs.



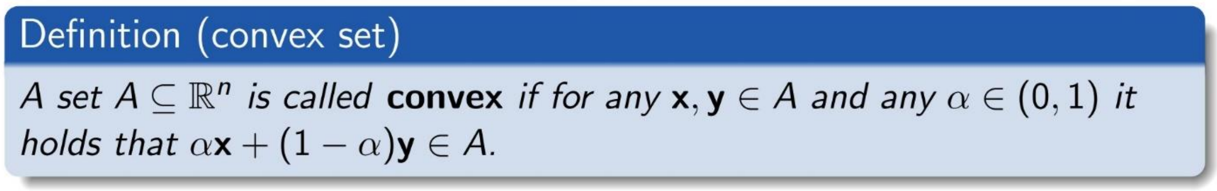
Definition (Convexity of a real-valued function in n variables)

Let $D \subseteq \mathbb{R}^n$ be a convex set and $f : D \to Z$ a real-valued function in n variables. If for all $\mathbf{x}_1, \mathbf{x}_2 \in D$ with $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\alpha \in (0,1)$ it follows that

$$f(\alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1-\alpha) f(\mathbf{x}_2)\,,$$

then we call f **convex**.

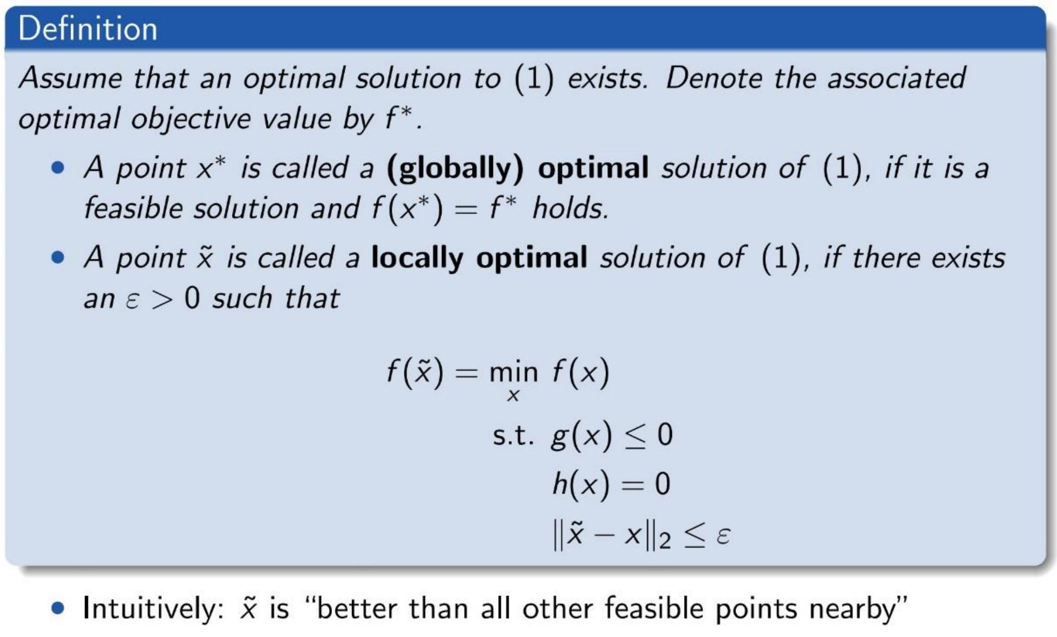
- *If the inequality holds as a strict inequality, then we call f* **strictly convex**.
- *If $-f$ is* **(strictly) convex**, *then f is called* **(strictly) concave**.



Definition (convex set)

A set $A \subseteq \mathbb{R}^n$ is called **convex** *if for any $\mathbf{x}, \mathbf{y} \in A$ and any $\alpha \in (0,1)$ it holds that $\alpha \mathbf{x} + (1-\alpha)\mathbf{y} \in A$.*

The feasible set of a convex optimization problem is a convex set. In a convex optimization problem, we minimize a convex objective function over a convex set.
Concave maximization problems can be solved by minimizing the convex objective function -f(x).



Definition

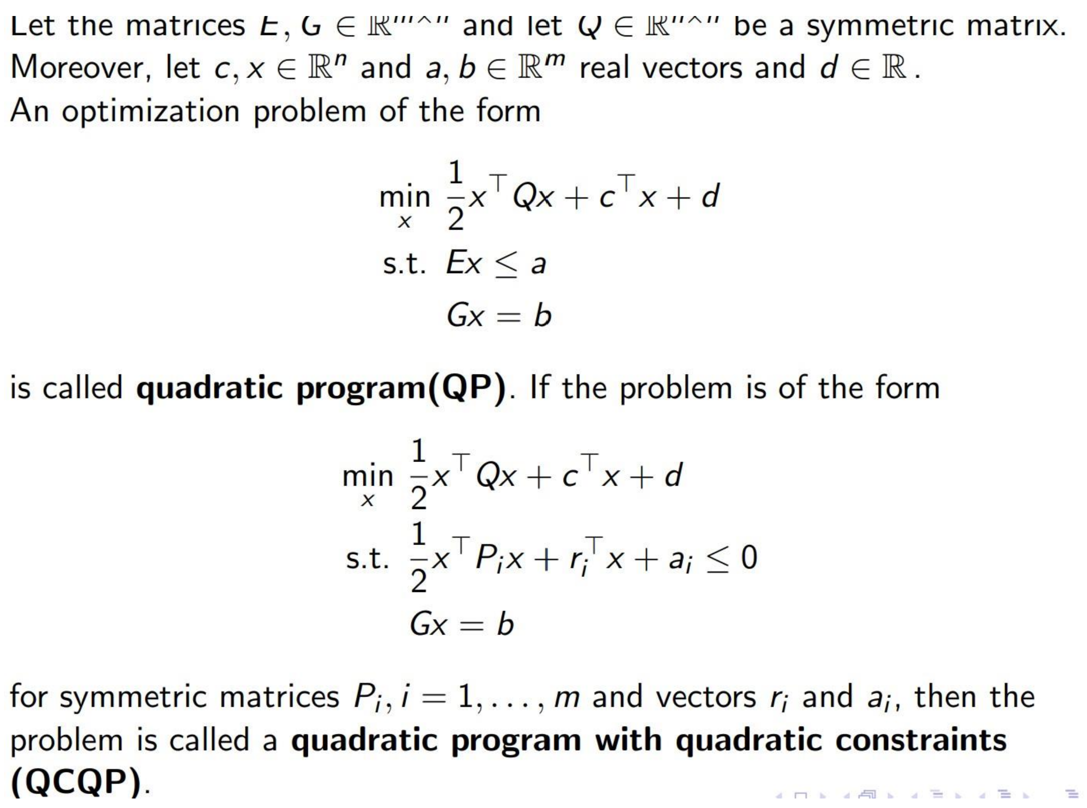
Assume that an optimal solution to (1) exists. Denote the associated optimal objective value by f^*.

- *A point x^* is called a* **(globally) optimal** *solution of (1), if it is a feasible solution and $f(x^*) = f^*$ holds.*
- *A point $\tilde{x}$ is called a* **locally optimal** *solution of (1), if there exists an $\varepsilon > 0$ such that*

$$\begin{aligned} f(\tilde{x}) = \min_{x}\ & f(x) \\ \text{s.t.}\ & g(x) \leq 0 \\ & h(x) = 0 \\ & \|\tilde{x} - x\|_2 \leq \varepsilon \end{aligned}$$

- Intuitively: $\tilde{x}$ is "better than all other feasible points nearby"

Fundamental property of convex optimization problems: For convex optimization problems, any locally optimal point is also globally optimal.



Let the matrices $E, G \in \mathbb{R}^{m \times n}$ and let $Q \in \mathbb{R}^{n \times n}$ be a symmetric matrix.
Moreover, let $c, x \in \mathbb{R}^n$ and $a, b \in \mathbb{R}^m$ real vectors and $d \in \mathbb{R}$.
An optimization problem of the form

$$\begin{aligned} \min_x \ & \frac{1}{2} x^\top Q x + c^\top x + d \\ \text{s.t. } & Ex \le a \\ & Gx = b \end{aligned}$$

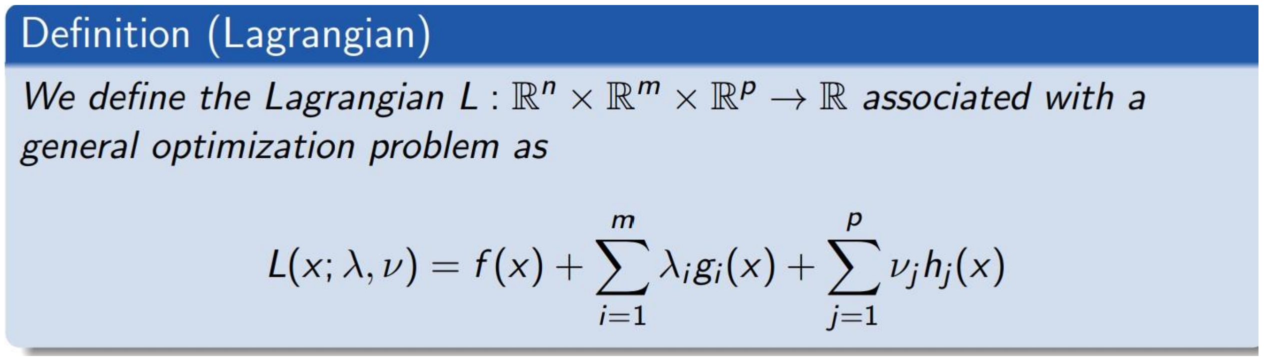
is called **quadratic program(QP)**. If the problem is of the form

$$\begin{aligned} \min_x \ & \frac{1}{2} x^\top Q x + c^\top x + d \\ \text{s.t. } & \frac{1}{2} x^\top P_i x + r_i^\top x + a_i \le 0 \\ & Gx = b \end{aligned}$$

for symmetric matrices $P_i, i = 1, \ldots, m$ and vectors r_i and a_i, then the problem is called a **quadratic program with quadratic constraints (QCQP)**.

A QP has a quadratic objective function and linear constraints, whereas in QCPC both the objective function and the constraints are quadratic.

Basic idea in Lagrangian duality: take constraints into account by augmenting the objective function with a weighted sum of the constraint functions.



Definition (Lagrangian)

We define the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with a general optimization problem as

$$L(x; \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \nu_j h_j(x)$$

Meaning of lambda: penalize for violation of constraints, high lambda = high obj. val. = high penalization

- We call λ_i the *Lagrange multiplier* associated with the i-th inequality constraint $g_i(x) \leq 0$
- We call ν_j the Lagrange multiplier associated with the j-th equality constraint $h_j(x) = 0$
- Vectors λ, ν: *Lagrange multiplier vectors* or *dual variables*

Definition

We define the Lagrange dual function (or just dual function) $q : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ *as the minimum value of the Lagrangian over* x*:*

$$q(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = \inf_x \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \nu_j h_j(x) \right)$$

Weak duality: lambdas greater equal 0.
Strong duality: lambdas greater equal zero. Duality gap greater equal zero.
Lagrange dual problem is a concave problem (regardless of whether or not primal problem is convex).
The primal problem is the original problem.
F* - q(lambda*, v*) is called duality gap.
Convex problems: additional regularity conditions required, so-called constraint qualifications, for strong duality.
Slater conditions: f convex, nicht affine hj: konvex, affine hj, gi: affin.
Slater's condition: strong duality holds for a convex problem if there exists a strictly feasible point (i.e., if all inequality constraints are strict inequalities in that point).
Saddle point: minimizes the function over all possible values of x over all lambdas and maximizes x over all possible values of lambda. It minimizes over the decision variables x and maximizes over lagrange multipliers.

Definition (Saddle point)

A point (x^*, λ^*, ν^*) *is called a saddle point of the Lagrange function, if for any* (x, λ, ν) *with* $\lambda \geq 0$ *it holds that*

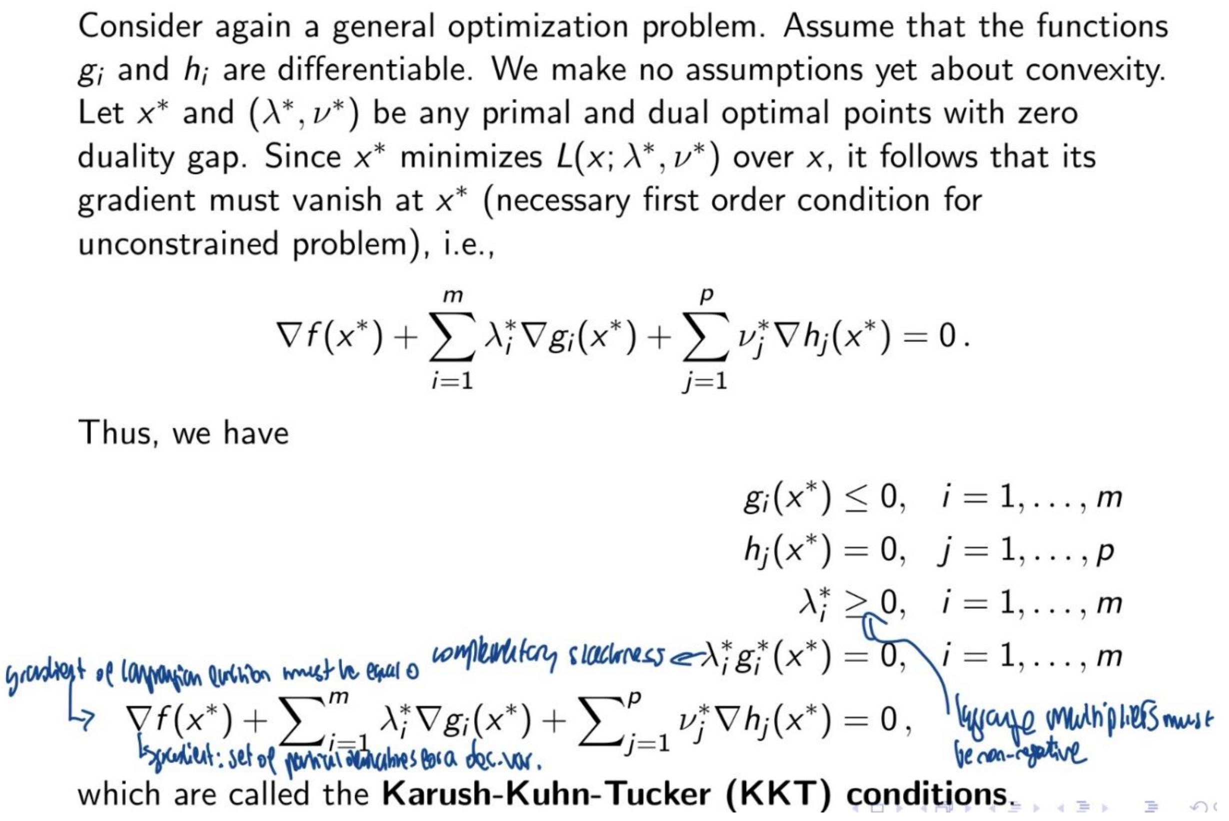
$$L(x^*; \lambda, \nu) \leq L(x^*; \lambda^*, \nu^*) \leq L(x; \lambda^*, \nu^*) \,.$$

Proposition

A point (x^*, λ^*, ν^*) *is a saddle point of the Lagrangian function if and only if* x^* *is primal optimal,* (λ^*, ν^*) *is dual optimal, and strong duality holds.*

Complementary slackness: sum of lambda*gi(x*) = 0, since each term in the sum is nonpositive). This condition holds for any primal solution x* and any dual optimal (lambda*, v*) (when strong duality holds).

Roughly speaking, complementary slackness means that the i-th optimal Lagrange multiplier is zero unless the i-th constraint is active (i.e., satisfied with equality) at the optimum.



Consider again a general optimization problem. Assume that the functions g_i and h_i are differentiable. We make no assumptions yet about convexity. Let x^* and (λ^*, ν^*) be any primal and dual optimal points with zero duality gap. Since x^* minimizes $L(x; \lambda^*, \nu^*)$ over x, it follows that its gradient must vanish at x^* (necessary first order condition for unconstrained problem), i.e.,

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \nu_j^* \nabla h_j(x^*) = 0 .$$

Thus, we have

$$g_i(x^*) \leq 0, \quad i = 1, \ldots, m$$

$$h_j(x^*) = 0, \quad j = 1, \ldots, p$$

$$\lambda_i^* \geq 0, \quad i = 1, \ldots, m$$

complementary slackness $\leftarrow$ $$\lambda_i^* g_i^*(x^*) = 0, \quad i = 1, \ldots, m$$

lagrange multipliers must be non-negative

gradient of Lagrangian function must be equal 0

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} \nu_j^* \nabla h_j(x^*) = 0 ,$$

gradient: set of partial derivatives for a dec. var.

which are called the **Karush-Kuhn-Tucker (KKT) conditions.**

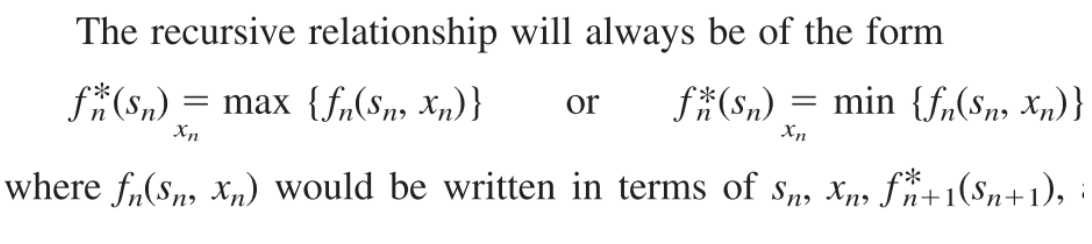
For any optimization problem with different objective and constraint functions for which strong duality holds, any pair of primal and dual optimal points must satisfy the KKT conditions.

In convex primal problems, the KTT conditions are also sufficient for (primal and dual) optimality. This means that if KTT is satisfied we have found the optimal solution.

<u>Lecture 10 – Dynamic Programming</u>

The dynamic programming approach starts with a small portion of the original problem, finds the optimal solution for it and then it gradually enlarges the problem and finds the current optimal solution from the preceding one.

A recursive relationship can be defined as one that identifies the optimal solution for stage n, given the optimal solution for stage n + 1. Using the recursive relationship, the solution procedure starts at the end and works backward.



The recursive relationship will always be of the form

$$f_n^*(s_n) = \max_{x_n} \{f_n(s_n, x_n)\} \qquad \text{or} \qquad f_n^*(s_n) = \min_{x_n} \{f_n(s_n, x_n)\}$$

where $f_n(s_n, x_n)$ would be written in terms of s_n, x_n, $f_{n+1}^*(s_{n+1})$,

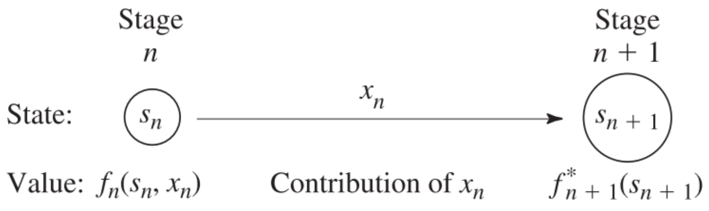
Deterministic Dynamic Programming problems: the state at the next stage is completely determined by the current stage and the policy decision at that stage.

The objective function in Dynamic Programming problems is either to minimize the sum of contributions of the individual stages (e.g., stagecoach problem) or to maximize a sum or minimize a product of the terms.

Nature of the states: discrete or continuous or might be represented by a state vector.

Nature of the decision variables: discrete or continuous.

FIGURE 11.3
The basic structure for deterministic dynamic programming.



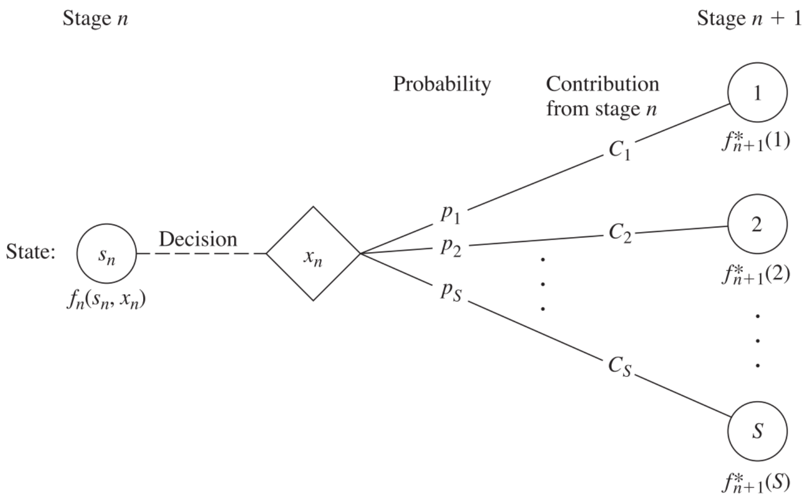
Differences between Dynamic Programming and linear programming:

-involves only one resource (one functional constraint), whereas linear programming can deal with thousands of resources (constraints).

The only assumption needed for Dynamic Programming is additivity.

In probabilistic dynamic programming the next state is not completely determined by state and policy decisions at current state, but instead there is a probability distribution that describes what the next state will be.

Basic structure of probabilistic dynamic programming problems:



Dynamic programming can be used to solve nonlinear problems with few decision variables. Have each stage has decision variable.

If a nonoptimal decision is made by mistake at some stage, the solution procedure will not need to be reapplied since an optimal policy decision only depends on the state itself and not on the past ones.

Once an optimal policy has been found for the overall problem, the information needed to specify the optimal decision at a particular stage only depends on the state itself and not on the state at that stage and the decisions made at preceding stages.