Introduction to course TTK16 Mixed integer optimization in energy and oil and gas systems

CONSTRAINED OPTIMIZATION

LARS IMSLAND

NTNU

TTK16 – overview

- This introduction lecture
- One week intensive course
  - Starting october 16th
  - Lecturer: Professor Eduardo Camponogara
  - Teaching assistant: Marco Aguiar
  - Lectures and practise sessions
- Implementation project
  - Oral presentation
- Exam: TBD?, Oral

1.4 CLASS SCHEDULE

Class	Date	Topic
1		Fundamentals
2		Introduction to modeling
3	Monday	Practice session
4		Review of linear programming
5		Introduction to integer programming
6		Practice session
7		Relaxations and bounding
8		Branch-and-bound algorithm
9	Tuesday	Practice Session
10		Valid inequalities
11		Cutting-plane algorithm
12		Practice session
13		Piecewise-linear approximation: one dimensional
14		Piecewise-linear approximation: multidimensional
15	Wednesday	Practice session
16		Gas-lift allocation problem
17		Introduction to MINLP
18		Practice session
19		Project presentations
20		Project presentations
21	Friday	Project presentations
22		Project presentations
23		Project presentations

NTNU

Outline this presentation

- General constrained optimization
  - KKT
  - Convexity
- Semi-definite programming, LMIs
- (Mixed) integer programming

Background material this presentation

General optimization

- Part of the material are from Prof. Moritz Diehl, Freiburg (and coworkers). He has also written a note:
  - http://www.syscop.de/files/2015ws/numopt/numopt_0.pdf
- Books:
  - J. Nocedal and S.J. Wright. **Numerical Optimization**. Springer, 2nd edition, 2006
  - S. Boyd and L. Vandenberghe. **Convex Optimization**. University Press, Cambridge, 2004

Semidefinite programming

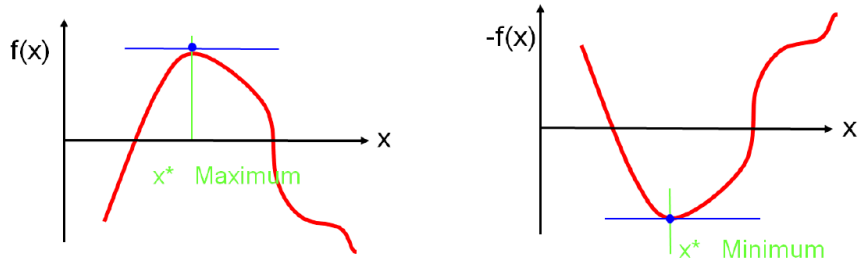
- Books:
  - S. Boyd et al., Linear Matrix Inequalities in System and Control Theory, SIAM, 1994
  - C. Scherer and S. Weiland. Lecture Notes DISC Course on Linear Matrix Inequalities in Control, 1999.

Mixed integer optimization

- Slides by Brage Knudsen, Eduardo Camponogara, …
  - Introductory example found many places
- Books&articles:
  - Wolsey. **Integer programming**. Wiley, 1998
  - Articles by I. Grossmann and co-authors

What is optimization?

- Optimization = search for the best solution
- In mathematical terms:
  - Minimization or maximization of an objective function $f(x)$ depending on variables x subject to constraints
  - Equivalence of maximization and minimization problems: (from now on only minimization):



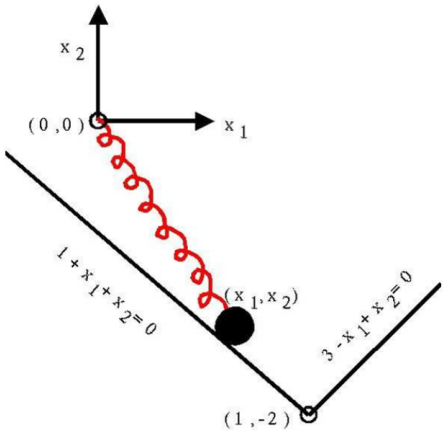
Constrained optimization

- Often variable x must satisfy certain constraints, e.g.:
  - $x \geq 0$
  - $x_1^2 + x_2^2 = C$
- General formulation:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g(x) = 0 \\ & h(x) \geq 0 \end{array}$$

  - f objective function / cost function
  - g equality constraints
  - h inequality constraints

Simple example: Ball hanging on a string

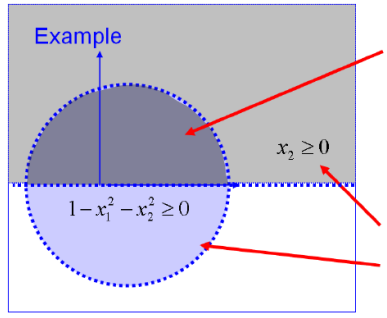


To find position at rest, minimize potential energy!

$$\begin{array}{ll} \text{minimize} & \underbrace{x_1^2 + x_2^2}_{\text{spring}} + \underbrace{m\,x_2}_{\text{gravity}} \\ \text{subject to} & 1 + x_1 + x_2 \geq 0 \\ & 3 - x_1 + x_2 \geq 0 \end{array}$$

Feasible set

Feasible set = collection of all points that satisfy all constraints:

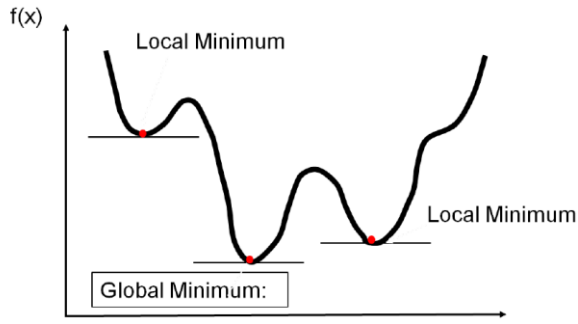


feasible set is intersection of grey and blue area

$$h_1(x) := \quad x_2 \geq 0$$
$$h_2(x) := \quad 1 - x_1^2 - x_2^2 \geq 0$$

The "feasible set" Ω is $\{x \in \mathbb{R}^n \| g(x) = 0, h(x) \geq 0\}$.

Local and global optima



- The point $x^* \in \mathbb{R}^n$ is a "global minimizer" iff $x^* \in \Omega$ and $\forall x \in \Omega : f(x) \geq f(x^*)$
- The point $x^* \in \mathbb{R}^n$ is a "local minimizer" iff $x^* \in \Omega$ and there exists a neighborhood $\mathcal{N}$ of x^* (e.g. an open ball around x^*) so that $\forall x \in \Omega \cap \mathcal{N} : f(x) \geq f(x^*)$.

Derivatives

- First and second derivatives of the objective function or the constraints play an important role in optimization
- The first order derivatives are called the gradient (of the resp. function) and the second order derivatives are called the Hessian matrix

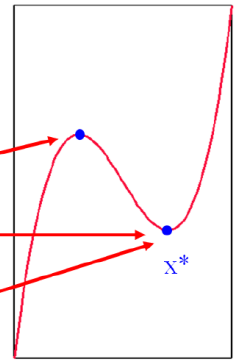
$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}, \qquad \nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Unconstrained optimality conditions

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad f(x)$$

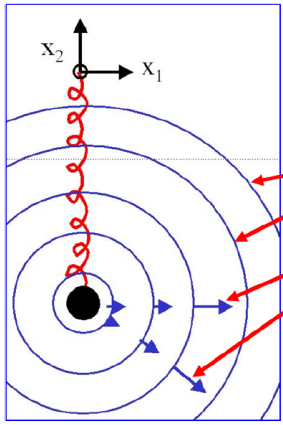
Assume that f is twice differentiable.
We want to test a point x^* for local optimality.

- *necessary condition:* $\nabla f(x^*)=0$ (*stationarity*)
- *sufficient condition:* x^* stationary and $\nabla^2 f(x^*)$ positive definite



Ball on a spring without constraints

$$\underset{x\in\mathbb{R}^2}{\text{minimize}} \quad x_1^2 + x_2^2 + m\,x_2$$



contour lines of $f(x)$

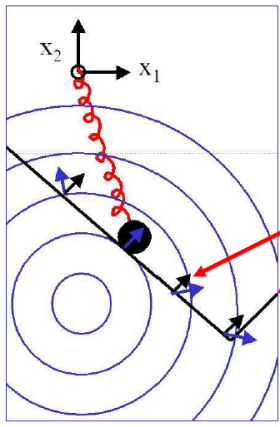
gradient vector

$$\nabla f(x) = (2x_1, 2x_2 + m)$$

unconstrained minimum:

$$0 = \nabla f(x^*) \Leftrightarrow (x_1^*, x_2^*) = (0, -\frac{m}{2})$$

Ball on a string with constraints



$$\begin{aligned} &\min f(x) \\ h_1(x) &:= 1 + x_1 + x_2 \geq 0 \\ h_2(x) &:= 3 - x_1 + x_2 \geq 0 \end{aligned}$$

gradient ∇h_1 of active constraint

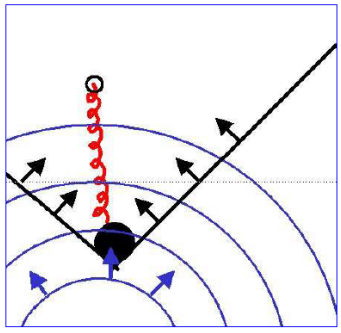
inactive constraint h_2

constrained minimum:

$$\nabla f(x^*) = \mu_1 \nabla h_1(x^*)$$

Lagrange multiplier

Ball on a string with two active constraints



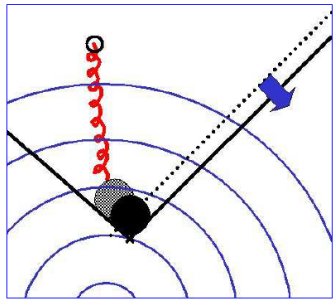
$$\begin{aligned} &\min f(x) \\ h_1(x) &:= 1 + x_1 + x_2 \geq 0 \\ h_2(x) &:= 3 - x_1 + x_2 \geq 0 \end{aligned}$$

„equilibrium of forces“

$$\nabla f(x^*) = \mu_1 \nabla h_1(x^*) + \mu_2 \nabla h_2(x^*) \qquad \mu_1, \mu_2 \geq 0$$

„constraint forces“

Multipliers are shadow prices



old constraint: $h(x) \geq 0$

new constraint: $h(x) + \varepsilon \geq 0$

What happens if we relax a constraint?
Feasible set becomes bigger,
so new minimum $f(x_\varepsilon^*)$ becomes smaller.
How much would we gain?

$$f(x_\varepsilon^*) \approx f(x^*) - \mu\varepsilon$$

Multipliers show the hidden cost of constraints.

The Lagrangian

For constrained problems, introduce modification of objective function:

$$L(x, \lambda, \mu) := f(x^*) - \sum \lambda_i g_i(x) - \sum \mu_i h_i(x)$$

- equality multipliers λ_i may have both signs in a solution
- inequality multipliers μ_i cannot be negative (cf. shadow prices)
- for inactive constraints, multipliers μ_i are zero

Constrained optimality conditions

Karush-Kuhn-Tucker necessary conditions (KKT-conditions):

- x^* feasible
- there exist λ^*, μ^* such that

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0$$

$$(\Leftrightarrow \text{"Equilibrium"} \nabla f = \sum \lambda_i \nabla g_i + \sum \mu_i \nabla h_i)$$

- $\mu^* \geq 0$ holds
- and it holds the complementarity condition

$$\mu^{*T} h(x^*) = 0$$

i.e. $\mu_i^* = 0$ or $h_i(x^*) = 0$ for each i

SQP algorithms for nonlinear programming

- Nonlinear programming problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$

- Idea in SQP: Solve sequential quadratic approximations:

$$\begin{aligned} \underset{\Delta x}{\text{minimize}} \quad & \nabla f(x^k)^\mathsf{T} \Delta x + \frac{1}{2} \Delta x^\mathsf{T} H(x_k) \Delta x \\ \text{subject to} \quad & g(x^k) + \nabla g(x^k)^\mathsf{T} \Delta x = 0 \\ & h(x^k) + \nabla h(x^k)^\mathsf{T} \Delta x \geq 0 \end{aligned}$$

- SQP = Sequential Quadratic Programming
  - (Major alternative to SQP: Interior point (IP) methods)

Properties of SQP method

- If we use the exact Hessian of the Lagrangian
$$H = \nabla^2 \mathcal{L}(x, \lambda, \mu)$$
the SQP method is a Newton-method for the KKT conditions
  - With quadratic convergence

- Often, we use approximate update-formulas for *H*, for example BFGS (called Quasi-Newton methods)
  - The right update-formulas leads to superlinear convergence

- Global convergence can be achieved by an appropriate stepsize selection strategy

Prototype SQP algorithm

0. Start with $k = 0$, start value x^0 and $H^0 = I$
1. Compute $f(x^k),\ g(x^k),\ h(x^k),\ \nabla f(x^k),\ \nabla g(x^k),\ \nabla h(x^k)$
2. If x^k feasible and
$$\|\nabla \mathcal{L}(x^k, \lambda^k, \mu^k)\| \leq \epsilon$$
then *stop* – convergence achieved!
3. Solve quadratic problem and get Δx^k
4. Perform line search and get stepsize α^k
5. Iterate
$$x^{k+1} = x^k + \alpha^k \Delta x^k$$
6. Update Hessian
7. $k = k + 1$, goto step 1

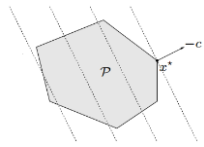
What characterize an optimization problem?

- Degrees of freedom *n*; the number of optimization variables
- Continuous or discrete (integer) search space
- Properties of the objective function
  - Linear, nonlinear, quadratic, ...
    - Convex?
  - Smoothness (continuity, differentiability)
- Properties of the constraints
  - Equalities or inequalities
  - Types: bounds, linear, nonlinear, cone (e.g. semi-definite), differential equations (optimal control), ...
    - Do they define a convex feasible set?
  - Smoothness (continuity, differentiability)

Types of optimization problems

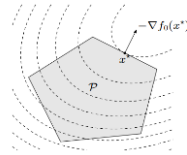
- Linear programming
  - Convex problem
  - Feasible set polyhedron

$$\begin{aligned} \text{minimize} \quad & c^\mathsf{T} x \\ \text{subject to} \quad & Ax \leq b \\ & Cx = d \end{aligned}$$



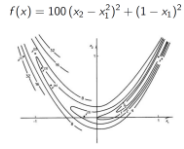
- Quadratic programming
  - Convex problem if $P \geq 0$
  - Feasible set polyhedron

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} x^\mathsf{T} P x + q^\mathsf{T} x \\ \text{subject to} \quad & Ax \leq b \\ & Cx = d \end{aligned}$$



- Nonlinear programming
  - In general non-convex!

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$



- But today's topics are two other types:
  - Semidefinite programming (briefly!)
  - Integer programming

When is an optimization problem difficult?

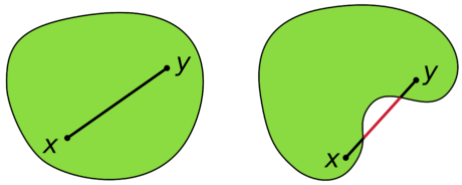
"*The great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity*"
– R. Tyrrell Rockafellar

For convex optimization problems, we can efficiently find global minima.
LP, QP, SDP, SOCP

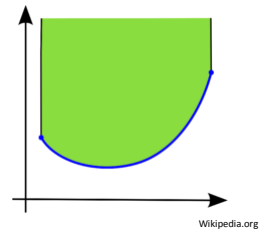
For non-convex, but smooth problems, we can efficiently find local minima.
Iterative use of (typically) QP, e.g. SQP

For integer programs that are convex (in particular linear) when integer variables are relaxed, we can (fairly) efficiently find global minima
MILP

Convexity



If the line segment between any two points within a set is inside the set, the set is **convex**.



A function is **convex** if the epigraph is a convex set.

- A convex optimization problem: Both *f(x)* and the feasible set convex
- Convex optimization problems are preferable!
  - For convex optimization problems, every local minimum is also a global minimum. **Sufficient to look for a local minimum!**
  - For many convex optimization problems, it is easy to find derivatives, exploit structure, etc. such that the optimization problem is well behaved.
  - They may have «guaranteed complexity».

Semidefinite programming

NTNU

Positive semidefiniteness

- A symmetric, square matrix *P* is positive semidefinite if

$$P \succeq 0 \quad \Longleftrightarrow \quad x^{\mathsf{T}} P x \geq 0, \quad \forall x \in \mathbb{R}^n \quad \Longleftrightarrow \quad P \in S^n$$

- The cone (set) of positive semidefinite matrices is convex

$$\begin{aligned} P, Q \in S^n &\Rightarrow x^{\mathsf{T}}(\alpha P + (1-\alpha)Q)x = \alpha x^{\mathsf{T}} P x + (1-\alpha) x^{\mathsf{T}} Q x \geq 0 \\ &\Rightarrow \alpha P + (1-\alpha) Q \in S^n \end{aligned}$$

NTNU

Semidefinite programming formulations

- The LMI formulation (common in control community)

  Find x such that

  $$F(x) := F_0 + x_1F_1 + x_2F_2 + \ldots + x_nF_n \succeq 0$$

- Also:

  $$\begin{aligned} \min_x \quad & c^\mathsf{T}x \\ \text{s.t.} \quad & F(x) := F_0 + x_1F_1 + x_2F_2 + \ldots + x_nF_n \succeq 0 \end{aligned}$$

- Mathematicians (SDP standard form):

  $$\begin{aligned} \min_X \quad & \langle C, X\rangle \\ \text{s.t.} \quad & \langle A_i, X\rangle = b_i, \quad i = 1, \ldots, K \\ & X \succeq 0 \end{aligned}$$

  $$\langle A, B\rangle = \mathrm{tr}A^\mathsf{T}B = \sum_i \sum_j A_{ij}B_{ij}$$

- LP and (convex) QP are special cases of SDP
  - Which again is a special case of an even more general class of convex problems called Second-Order Cone Programming (SOCP)

Example: Lyapunov stability

- Lyapunov:

  $$\dot{x} = Ax \text{ asymptotically stable} \iff \exists P \succ 0 \text{ such that } A^\mathsf{T}P + PA \prec 0$$

- Formulate as LMI

  $$\begin{pmatrix} P & 0 \\ 0 & -A^\mathsf{T}P - PA \end{pmatrix} \succ 0$$

- This is linear in elements of *P* (the variable), and is therefore an LMI

- Fortunately, we usually don't have to find/define the F_i matrices ourselves, modeling software does this for us!
  - Excellent Matlab toolbox: Yalmip

- Implementation detail: Numerical solvers don't handle strict inequalities
  - Related problem: 0 is a «marginal solution», must «dehomogize»
  - That is: Constrain such that 0 is not a solution (e.g. $P \succeq I$, $\mathrm{tr}P = 1$, etc.)

- (Better methods exists for this problem; e.g. check eigenvalues, or solve Lyapunov equation)

Example 2: State feedback synthesis

- Problem:
$$\text{Find matrix } K \text{ s.t. } \dot{x} = Ax + Bu \text{ with } u = Kx \text{ is a.s.}$$
- Lyapunov:
$$\exists P \succ 0, K \text{ such that } (A + BK)^{\mathsf{T}} P + P(A + BK) \prec 0$$
- Problem: Not linear!

- Trick #1: Congruence: Multiply with $Q = P^{-1}$ on both sides
$$\exists Q \succ 0, K \text{ such that } Q(A + BK)^{\mathsf{T}} + (A + BK)Q \prec 0$$
- Still not linear. Trick #2: Introduce $W = KQ$.
$$\exists Q \succ 0, W \text{ such that } QA^{\mathsf{T}} + W^{\mathsf{T}} B^{\mathsf{T}} + AQ + BW \prec 0$$
- Linear in Q and W! Solve as LMI. Find $P = Q^{-1}$ and $K = WQ^{-1}$
  - Note that Q is invertible since it is positive definite

- (In general: A number of different tricks and variable transformations exists)
  - Schur complements, *S*-procedure, ...

On solvers for SDPs

- Simplex-algorithm for LPs performs usually quite well, but have exponential worst-case performance

- Much research effort in 70s and 80s (Karmarkar) resulted in new class of methods for LPs called *interior point methods* (IPM) that have polynomial worst-case performance

- Nesterov and Nemirovskii (80s) shows that these methods can be used for more general classes of convex programs (including SDPs)

- Many classical control problems can be formulated as LMIs/SDPs, led to huge interest for these methods in control community in 90s and 00s
  - Lyapunov (1890s),Yakubovich (60s), Willems (70s), Boyd and others (90s)

- In parallel, numerical mathematicians developed efficient algorithms on basis of N&Ns work
  - Matlab's LMI toolbox (early, but not good!), Sedumi, SDPT3, MOSEK, ...

- Methods also discovered by other communities (operations research, combinatorics, ...) in 90s, 00s, ...

(Mixed) Integer Programming

Example

- Consider

$$\begin{aligned} \max \quad & 3x + 4y \\ \text{subject to} \quad & 5x + 8y \leq 24 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

What type of optimization problem is this?

- Consider now

$$\begin{aligned} \max \quad & 3x + 4y \\ \text{subject to} \quad & 5x + 8y \leq 24 \\ & x \geq 0,\ y \geq 0, \quad x \text{ and } y \text{ integers} \end{aligned}$$

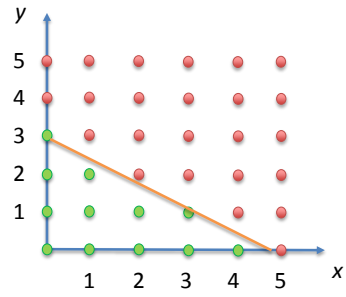
What type of optimization problem is this?

(Linear) Integer Programming, example

- Consider

$$\begin{aligned} \max \quad & 3x + 4y \\ \text{subject to} \quad & 5x + 8y \leq 24 \\ & x \geq 0,\ y \geq 0, \quad x \text{ and } y \text{ integers} \end{aligned}$$

- Q1: What is the optimal integer solution?
- Q2: What is the optimal relaxed solution?
  - That is, the linear programming solution?
- Q3: Can we use linear programming to solve the integer program?



(Linear) Integer Programming, example
Solving via LP and rounding?

- Relax IP, solve LP to get

$$x = \frac{24}{5}, \quad y = 0$$

value: $14\frac{2}{5}$

- Round to get

$$x = 5, \quad y = 0$$

Infeasible!

- Truncate to get

$$x = 4, \quad y = 0$$

value: 12

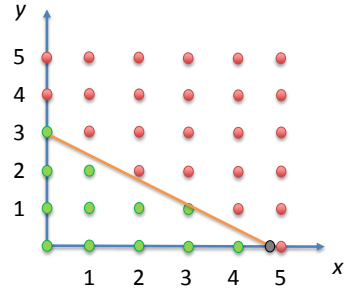
Same solution at (0,3)

- Optimal solution is

$$x = 3, \quad y = 1$$

value: 13

$$\begin{aligned} \max \quad & 3x + 4y \\ \text{s.t.} \quad & 5x + 8y \leq 24 \\ & x \geq 0,\ y \geq 0, \quad x \text{ and } y \text{ integers} \end{aligned}$$



Types of integer programs

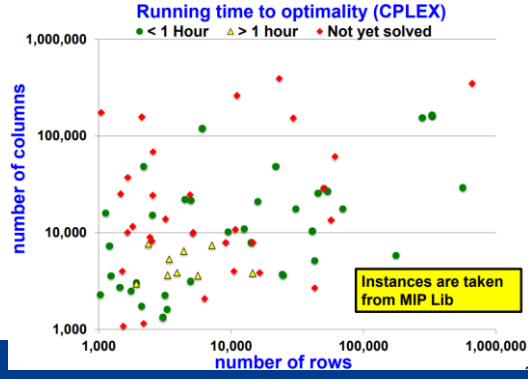
- (Pure) integer programs: All variables are integers
- Binary integer programs: variables are 0 or 1
  - However,
    $$x \in \{0,1\} \quad \Longleftrightarrow \quad x \text{ integer}, \ 0 \leq x \leq 1$$
  - That is, binary programs special case of integer programs, and usually not treated separately
  - But binary variables are common; often used to model logics (Boolean)
- Mixed integer programs
  - Only some of the variables are integer
- Linear, quadratic, nonlinear, semi-definite (mixed) integer programs: As before, only with (some) variables integer
  - MILP, MIQP, MINLP, MISDP, ...

Why (mixed) integer programs?

- Advantages of restricting (some) variables to take on integer values
  - Flexibility! You can model (and solve) more realistic problems
- Disadvantages
  - More difficult to model?
    - But modelling software helps: AMPL, GAMS, Yalmip, …
  - Can be much more difficult to solve!

On computation for (mixed) integer (linear) programs

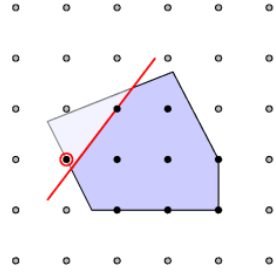
- Much, much harder to solve than LPs
- But very good solvers can solve (some) very large problems
  - Like 50 000 variables, 2 million non-zeros
  - Good solvers: CPLEX, Gurobi, ...?
- Hard to predict if a problem will be solved quickly, in a long time, or not at all (before the universe dies)



NTNU

What is fundamentaly different from continuous optimization?

- The feasible region consists of a set of disconnected integer points.
- Gradient-based algorithms cannot be *directly* applied.
- No conditions similar to the KKT conditions to prove first order optimality.



Source: http://www.zib.de/en/optimization/mip/projects-long/exactip.html

NTNU

Operations research

Source: Wikipedia

- Designing airline crew schedules:
  - Pair (assign duty periods) airline crews that cover every flight leg at the least cost.
  - Must satisfy legal rules such as limited total flying time, minimum rest time, etc.
- Train scheduling:
  - Find a feasible train schedule that secures sufficient transit time for passengers with connections, assigning trains to single tracks such that train collisions are avoided (hard constraint!), and minimize excessive wait time for trains.
- Production planning:
  - Given a set of X products to be produced in Y factories, with final shipment to Z sales areas.
  - Products are produced in batches, with both fixed and marginal costs.
  - Maximize profit/ minimize cost with respect to seasonal demands.
- Allocating lecture halls at NTNU:

Electric power production

Source: powerop.co.uk

The hydro-thermal unit-commitment (UC) dispatch problem:

Given a set of electric-power generating units with different characteristics:

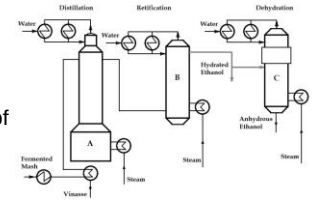
- Maximum output power (e.g. 400 MW).
- Efficiency curves.
- Start-up cost, start-up time and minimum up/down times.
- Emission level constraints.

Given a certain planning horizon (e.g. 24 hours): Select units such that

- The power demand d_t is satisfied for all time periods t.
- Fuel costs or emissions are minimized, or profit is maximized.
- The generating units have a certain excess reserve capacity r_t due to demand uncertainty.
- The unit schedule must satisfy a certain security level.

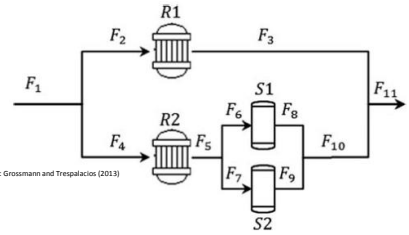
Chemical engineering

- Optimal design of distillation columns: Separation of components in a mixture passed through distillation units. Decision variables can be selecting the number of trays and feed locations, and the location of output streams (products).



Source: intechopen.com

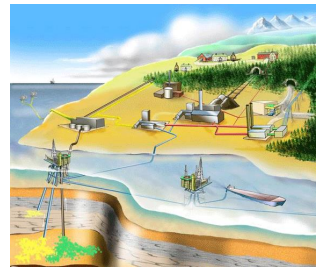
- Used extensively in process design and synthesis, e.g. Optimal reactor selection and configuration:



Source: Grossmann and Trespalacios (2013)

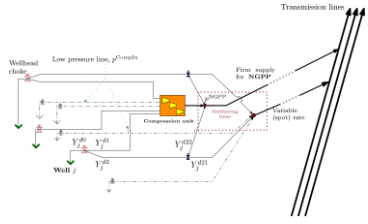
Petroleum production optimization

- Optimization of gas flow and routing in the natural-gas value chain: Meet seasonal varying gas demands, contractual obligations, minimize fuel consumption of compressors, etc.



Source: sintef.no

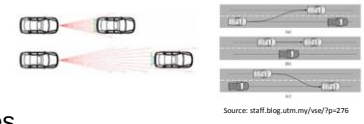
- Maximize revenues of oil and gas subject to constraints in the reservoir and wells, and the gathering system, for instance the capacity of separators and compressors.



Source: Knudsen et. al 2014

Motion control and hybrid systems

- Collision avoidance in trajectory-planning for aircrafts, UAVs and vehicles:
  - Avoid multiple vehicles colliding.
  - Obstacle avoidance for single vehicles.



Source: staff.blog.utm.my/vse/?p=276

- General hybrid model *predictive* control: MPC with discrete variables. Numerous applications in chemical, mechanical and electrical engineering. See: http://cse.lab.imtlucca.it/~bemporad/teaching/mpc/imt/6-hybrid-examples.pdf

Definitions of problems with discrete variables

Comments on notation:

$\mathbb{R}^n_+$ is the n-dimensional space of all *non-negative* real numbers:

$$\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$$

$\mathbb{Z}^p_+$ is the p-dimensional space of all *non-negative* integers:

$$\mathbb{Z}^n_+ = \{y \in \mathbb{Z}^p : y \geq 0\}$$

$\mathbb{B}^q$ is the q-dimensional space of all binary variabls:

$$\mathbb{B}^q = \{y : y \in \{0,1\}^q\}$$

- The expression mixed integer *program* and mixed integer *problem* is used interchangeably, both referring to a mathematical problem with continuous and discrete variables.

Definitions:

General MILP

$$J^* = \min_{(x,y)} \quad c^T x + d^T y$$
$$\text{s.t.}$$
$$Ax + By \geq b$$
$$(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+$$

- A is an $m \times n$ matrix
- B is an $m \times p$ matrix
- b is an m-dimensional vector
- c is and n-dimensional vector
- d is an p-dimensional vector

We define X as the set of feasible solutions:

$$X = \left\{(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+ : Ax + By \geq b\right\}$$

Mixed binary linear program:

$$J^* = \min_{(x,y)} \quad c^T x + d^T y$$
$$\text{s.t.}$$
$$Ax + By \geq b$$
$$x \in \mathbb{R}^n_+$$
$$y \in \{0, 1\}^p$$

(Linear) Integer program (IP):

$$J^* = \min_{y} \quad d^T y$$
$$\text{s.t.} \quad By \geq b$$
$$y \in \mathbb{Z}^p_+$$

Examples on formulating integer programs (IPs):

The generalized assignment problem (GAP): Given n assignments/tasks and m agents/servers/vehicles to carry out the tasks:

$i = 1, \ldots, n$	: index of tasks
$j = 1, \ldots, m$	: index of available agents
d_{ij}	: cost of assigning task i to agent j
b_j	: resource available from agent j
a_{ij}	: resource required by agent j to do task i
y_{ij}	: a binary variable equal to 1 if agent j is assigned to do task

Problem can be formulated as the linear integer program (IP):

$$\min_y \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}y_{ij}$$

s.t.

$$\sum_{j=1}^{m} y_{ij} = 1, \qquad i = 1 \ldots n \qquad : \text{Each task is assigned to exactly one agent}$$

$$\sum_{i=1}^{n} a_{ij}y_{ij} \leq b_j, \quad j = 1 \ldots m \qquad : \text{Total assignment for agent } j \text{ cannot exceed its capacity}$$

$$y_{ij} \in \{0, 1\}$$

Numerical example: GAP

Construct and solve the GAP with following specifications:

$n = 3$ tasks and $m = 2$ machines

$$\text{Available resources for machines } j: b_j = \begin{bmatrix} 13 \\ 11 \end{bmatrix}$$

$$\text{Costs}: d_{ij} = \begin{Bmatrix} & j_1 & j_2 \\ i_1 & 9 & 2 \\ i_2 & 1 & 2 \\ i_3 & 3 & 8 \end{Bmatrix}$$

$$\text{Assigment costs for task } i \text{ to machine } j: a_{ij} = \begin{Bmatrix} & j_1 & j_2 \\ i_1 & 6 & 8 \\ i_2 & 7 & 5 \\ i_3 & 9 & 6 \end{Bmatrix}$$

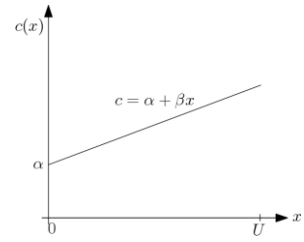
Minimize total costs, assigning each task to one machine.

- Find input matrices for intlinprog in Matlab!

General model formulations requiring integer variables

Fixed costs:
A fixed cost α, only present if $x > 0$.
The cost increases with βx:

$$c(x) = \begin{cases} \alpha + \beta x & \text{if } 0 < x \leq U \\ 0 & \text{otherwise} \end{cases}$$



By introducing a binary $y \in \{0, 1\}$, we can model this as

$$c = \alpha y + \beta x$$
$$0 \leq x \leq Uy$$

Similarly for variables that are zero or only in a certain range:

$$Ly \leq x \leq Uy$$
$$y \in \{0, 1\}$$

Implications and conditions:

Conditions and constraints given by a Boolean Y:

- Condition 1, modeled by $Y_1 = True$
- Condition 2, modeled by $Y_2 = True$

Given expression of the type:

*	$Y_1 \Rightarrow Y_2$	(if Y_1 then Y_2)
**	$Y_1 \vee Y_2$	(Y_1 or Y_2)
***	$Y_1 \Leftrightarrow Y_2$	(if and only if)

Replace Boolean Y with binary $y \in \{0, 1\}$. The given logical conditions can be defined by the constraints

*	$y_1 \leq y_2$,
**	$y_1 + y_2 \geq 1$,
***	$y_1 = y_2$,

respectively. General rules for reformulations logical conditions exist, see
Raman, R., & Grossmann, I. E. (1994). Modelling and computational techniques for logic based integer programming. *Computers and Chemical Engineering*, 18(7), 563–578.

Systematical derivation of linear inequalities from logic propositions

Goal: convert logical expressions to

$$Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \tag{1}$$

where each logical clause consists of expressions

$$Q_i \ : \ Y_1 \vee Y_2 \vee \ldots \vee Y_u \tag{2}$$

Steps:

1. Replace implication by disjunction:

$$Y_1 \Rightarrow Y_2 \ \Leftrightarrow \ \neg Y_1 \vee Y_2$$

2. If necessary, (particularly with several logical terms), apply DeMorgan's rules to move negation inward

$$\neg(Y_1 \vee Y_2) \ \Leftrightarrow \neg Y_1 \wedge \neg Y_2$$
$$\neg(Y_1 \wedge Y_2) \ \Leftrightarrow \neg Y_1 \vee \neg Y_2$$

3. If more than two Booleans, recursively distribute OR operator over AND to get expressions of the form (1)–(2)

$$(Y_1 \wedge Y_2) \vee Y_3 \ \Leftrightarrow (Y_1 \vee Y_3) \wedge (Y_2 \vee Y_3) \tag{3}$$

4. Replace Booleans Y_i with binary y_i. Each clause with only OR operators defines linear inequalities. An AND operator as in (3) gives an additional inequality, i.e., (3) results in the constraints

$$y_1 + y_3 \geq 1$$
$$y_2 + y_3 \geq 1$$

Example

If product A is chosen, product B cannot be chosen while product C have to be chosen. Define Booleans Y_i, for $i = A, B, C$.

1. Replace implication by disjunction:

$$Y_1 \Rightarrow Y_2 \ \Leftrightarrow \ \neg Y_1 \vee Y_2$$

2. If necessary, (particularly with several logical terms), apply DeMorgan's rules to move negation inward

$$\neg(Y_1 \vee Y_2) \ \Leftrightarrow \neg Y_1 \wedge \neg Y_2$$
$$\neg(Y_1 \wedge Y_2) \ \Leftrightarrow \neg Y_1 \vee \neg Y_2$$

3. If more than two Booleans, recursively distribute OR operator over AND to get expressions of the form (1)–(2)

$$(Y_1 \wedge Y_2) \vee Y_3 \ \Leftrightarrow (Y_1 \vee Y_3) \wedge (Y_2 \vee Y_3) \tag{1}$$

4. Replace Booleans Y_i with binary y_i. Each clause with only OR operators defines linear inequalities. An AND operator as in (3) gives an additional inequality, i.e., (3) results in the constraints

$$y_1 + y_3 \geq 1$$
$$y_2 + y_3 \geq 1$$

$$Y_A \Rightarrow \neg Y_B \wedge Y_C$$
$$\Updownarrow$$
$$\neg Y_A \vee (\neg Y_B \wedge Y_C)$$
$$\Updownarrow$$
$$(\neg Y_A \vee \neg Y_B) \wedge (\neg Y_A \vee Y_C)$$

Replace Y_i's with binaries and rewrite as linear constraints:

$$1 - y_A + 1 - y_B \geq 1$$
$$1 - y_A + y_C \geq 1$$

$$1 - y_A \geq y_B$$
$$y_C \geq y_A$$

Assignment: modeling logical conditions with binaries

(From Wolsey (1998)). Suppose you are interested in choosing a set of investments $\{1, \ldots, 7\}$. Model the following constraints:

1. You cannot invest in all of them.
2. You must choose at least one of them.
3. Investment 1 cannot be chosen if investment 3 is chosen.
4. Investment 4 can be chosen only if investment 2 is also chosen.
5. You must choose either both investments 1 and 5 or neither.

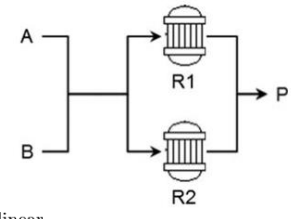
Disjunctive constraints:

Given $x \in \mathbb{R}$ with lower and upper bound, $0 \leq x \leq U$, and two linear constraints:

$$\underbrace{[ax \leq b]}_{R1} \veebar \underbrace{[dx \leq e]}_{R2}$$

where only one must hold:

$$\begin{aligned} ax &\leq b + M(1 - y_1) \\ dx &\leq e + M(1 - y_2) \\ y_1 + y_2 &= 1 \\ y_1, y_2 &\in \{0, 1\} \end{aligned}$$



where M is a sufficiently large big-M parameter, $M \geq \max(b, e)$.

Alternatively, use the extended, but tighter, convex hull reformulation of linear disjuncitons (more on this later):

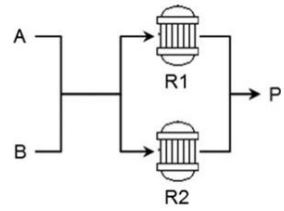
Source: Grossmann and Trespalacios (2013)

$$\begin{aligned} x &= z_1 + z_2, \\ az_1 &\leq by_1, \\ dz_2 &\leq ey_2, \\ y_1 + y_2 &= 1, \\ 0 \leq z_i &\leq Uy_i, \quad i = 1, 2 \\ y_1, y_2 &\in \{0, 1\} \end{aligned}$$

Example with disjunctive constraints:

Given the structure of a reactor and raw material selection with the following specifications:

- Objective: maximize profit of selling product P with price 10.
- To produce P, the options are:
  1. Buy reactor R1 with cost C = 5*F (flow), and with 90% conversion of material A and 70% of B.
  2. Buy reactor R2 with cost C = 4.6*F (flow), and with 85% conversion of material A and 80% of B.
- The cost of raw material A is 1.1, and available feed rate is 5.
- The cost of raw material B is 1, and available feed rate is 7.



Source: Grossmann and Trespalacios (2013)

Assignment:

1. Formulate the optimization problem using linear disjunctions.
2. Formulate the corresponding MILP using big-M reformulation.

How to solve?

$$J^* = \min_{(x,y)} \quad c^T x + d^T y$$
$$\text{s.t.} \quad Ax + By \geq b$$
$$(x, y) \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+$$

- Total enumeration (if possible):
  - Solve LP in *x* for all possible *y*'s, select solution with smallest optimum
  - Possible for (very) small problems, but quickly becomes infeasible

#binary variables (n)	2	10	20	30
#LPs to solve (2^n)	4	1024	> 1 million	> 1 billion

  - This is always «worst case» solution, illustrates «NP-hardness» of problem

- Can do much better if we use problem geometry to iteratively cut away parts of the feasible set
  - Methods: «branch-and-bound», «cutting-plane», «branch-and-cut», ...
  - There has been a tremendous development in these methods last ~20 years
  - Solvers: CPLEX, Gurobi, ...