A Brief Review of

# Linear and Integer Optimization with Applications

MVE165/MMG631

Ruben Seyer <rubense@student.chalmers.se>

4th June 2020

## Contents

# Preface

A few short remarks about this document are in order. Of importance is the fact that it in no manner claims to be a complete treatise on the subjects contained within, but rather an overview of theory, results and methods for practical purposes. In many cases, a thorough understanding of the subject may already be required to make sense of the brief descriptions within. For that reason, frequent references to the course material that expand on mentioned subjects are contained within. Roman sans-serif numbers set in brackets (e.g. [I]) denote a particular lecture given. Arabic sans-serif numbers set in brackets (e.g. [1]) denote a particular section of *Optimization* by J. Lundgren, M. Rönnqvist, and P. Värbrand.

A short summary of the material covered at the oral exam and the rough structure of this document follows:

- *Mathematical modelling of optimization problems*: graphic solution
- *Linear programming*: BFSs, the simplex method, degeneracy, multiple optima, unbounded solution, infeasibility, starting solutions, LP duality, post-optimal and sensitivity analysis
- *Discrete and combinatorial optimization*: models of specific ILP problems, mathematical properties, complexity, algorithms, local/global optima, neighbourhoods, heuristics
- *Network flows*: shortest paths, dynamic programming, LP models of network flows, maximum flows, minimum cost network flows, unimodularity, integrality property
- *Multi-objective optimization*: Pareto optimality, (non-)convexity, solution methods, objective space representation
- *Non-linear optimization*: convexity, local/global optimality, mathematical properties, search methods

Some content pertaining to specific problems or models has been extracted into an appendix, but makes it no less important to the course material.

Finally, no author is perfect and the present one constitutes no exception. This document is prone to revision and the reader should ensure through the date that it is in fact the most recent version. From the author's website, the most recent version is always available. Questions, corrections and remarks are always welcome and will be thankfully credited should they lead to valuable changes. We hope you find this document useful.

# 1 Linear programming

## 1.1 Preliminaries

**Definition 1.1** (Linear program)**.** A *linear program* (LP, linear optimization problem) is an optimization problem where all relations are described by linear forms. In general, such a problem is formulated

$$\begin{aligned} \min / \max \quad & z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} && \boldsymbol{c}, \boldsymbol{x} \in \mathbb{R}^n \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} && A \in \mathbb{R}^{n \times m} \\ & \boldsymbol{x} \geq \boldsymbol{0} && \boldsymbol{b} \in \mathbb{R}^m. \end{aligned} \tag{1.1}$$

Any relation may be rewritten into an inequality as above. Non-negativity constraints are not necessary, but are usually assumed. [III]

The feasible region of a LP is defined as the intersection of half-spaces in $\mathbb{R}^n$ defined by its constraints.

$$X := \{\boldsymbol{x} \geq \boldsymbol{0} : A\boldsymbol{x} \leq \boldsymbol{b}\} \tag{1.2}$$

It can easily be shown by application of the definition that the feasible region of a LP forms a convex set.

**Definition 1.2** (Convex combinations)**.** A *convex combination* of the points $\boldsymbol{x}^p \in \mathbb{R}^n,\ p = 1, \ldots, P$ is any point $\boldsymbol{x} \in \mathbb{R}^n$ that can be expressed as

$$\boldsymbol{x} = \sum_{p=1}^{P} \lambda_p \boldsymbol{x}^p \tag{1.3}$$

where the *convexity weights* satisfy $\sum_{p=1}^{P} \lambda_p = 1,\ \lambda_p \geq 0\ \forall p$. [III] [4.1]

**Definition 1.3** (Convex set)**.** A set $X \in \mathbb{R}^n$ is a *convex set* if, for any two points $\boldsymbol{x}^1, \boldsymbol{x}^2 \in X$, and any $\lambda \in [0, 1]$, it holds that $\boldsymbol{x} := \lambda\boldsymbol{x}^1 + (1 - \lambda)\boldsymbol{x}^2 \in X$. [III] [2.4]

**Definition 1.4** (Extreme point)**.** The point $\boldsymbol{x^k} \in X$ is an *extreme point* of the polyhedron $X$ if it is *not* possible to express $\boldsymbol{x^k}$ as a *strict* convex combination (with non-integer weights) of two distinct points in $X$. [III] [4.1]

**Theorem 1.1** (Optimal solution in an extreme point)**.** *Suppose that the feasible region $X = \{\boldsymbol{x} \geq \boldsymbol{0} : A\boldsymbol{x} \leq \boldsymbol{b}\}$ is non-empty and bounded. Then, the minimum value of the objective $\boldsymbol{c}^\mathsf{T}\boldsymbol{x} =: z^*$ is attained at (at least) one extreme point $\boldsymbol{x}^k$ of $X$. [III] [4.1 Th. 2.4]*

*Proof.* Suppose that it were not so; that there exists a non-extreme point $\tilde{\boldsymbol{x}} \in X$ with a lower objective value than any of the extreme points, i.e. $\boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{x}} < \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^k$ for all extreme

points $\boldsymbol{x}^k \in X$. Since $X$ is a convex set, $\tilde{\boldsymbol{x}}$ may be expressed as a convex combination of the extreme points of $X$ with some weights $\lambda_k$. Then,

$$\boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{x}} = \boldsymbol{c}^\mathsf{T}\sum_k \lambda_k \boldsymbol{x}^k = \sum_k \lambda_k \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^k > \sum_k \lambda_k \boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{x}} = \boldsymbol{c}^\mathsf{T}\tilde{\boldsymbol{x}} \quad \lightning \tag{1.4}$$

which is a contradiction. □

The main method of solving LPs is to use the simplex method. To that end, we must reformulate our problems in the standard form. One may translate any LP to the standard form by some simple manipulations:

- in all inequalities, introduce *slack variables* (negative-signed surplus variables in the case of greater-than) and convert to an equality
- replace non-positive variables with the corresponding sign change
- replace unconstrained variables with the difference between two new non-negative variables

## 1.2 Basic solutions

**Definition 1.5** (Basic solution)**.** A *basic* solution to the $m \times n$ system of equations $A\boldsymbol{x} = b$ is obtained if $n - m$ of the variables are set to 0 and the remaining variables obtain unique values from the solution to the remaining $m \times m$ system of equations.

The variables set to 0 are called *non-basic variables* and the remaining are called *basic variables*. [III] [4.3 Def. 4.3]

A basic solution $\boldsymbol{x}$ corresponds to the intersection of $m$ hyperplanes in $\mathbb{R}^m$. The solution is feasible if $\boldsymbol{x} \geq \boldsymbol{0}$. However, each extreme point of the feasible set also satisfies these conditions.

### 1.2.1 Algebraic description

Consider a standard linear minimization problem. We partition $\boldsymbol{x}$ into $m$ basic variables $\boldsymbol{x}_B$ and $n - m$ non-basic variables $\boldsymbol{x}_N$, such that $\boldsymbol{x}^\mathsf{T} = (\boldsymbol{x}_B^\mathsf{T}, \boldsymbol{x}_N^\mathsf{T})$. Analogously, let $\boldsymbol{c}^\mathsf{T} = (\boldsymbol{c}_B^\mathsf{T}, \boldsymbol{c}_N^\mathsf{T})$ and $A = (A_B, A_N) =: (B, N)$. Assume that $B$ is non-singular, i.e. its inverse $B^{-1}$ exists. We may now rewrite the standard problem into

$$\begin{aligned} \min \quad & z = \boldsymbol{c}_B^\mathsf{T}\boldsymbol{x}_B + \boldsymbol{c}_N^\mathsf{T}\boldsymbol{x}_N \\ \text{s.t.} \quad & B\boldsymbol{x}_B + N\boldsymbol{x}_N = \boldsymbol{b} \\ & \boldsymbol{x}_B, \boldsymbol{x}_N \geq \boldsymbol{0}. \end{aligned} \tag{1.5}$$

The basic variables can be substituted away by rewriting the constraints

$$\boldsymbol{x}_B = B^{-1}(\boldsymbol{b} - N\boldsymbol{x}_N) \tag{1.6}$$

and finally

$$\begin{aligned} \min \quad & z = \boldsymbol{c}_B^\mathsf{T} B^{-1}\boldsymbol{b} + (\boldsymbol{c}_N^\mathsf{T} - \boldsymbol{c}_B^\mathsf{T} B^{-1} N)\boldsymbol{x}_N \\ \text{s.t.} \quad & B^{-1}\boldsymbol{b} - B^{-1}N\boldsymbol{x}_N \geq \mathbf{0} \\ & \boldsymbol{x}_N \geq \mathbf{0}. \end{aligned} \tag{1.7}$$

At any basic solution, the non-basic variables are zero, so $\boldsymbol{x}_B = B^{-1}\boldsymbol{b}$ and $z = c_B^\mathsf{T} B^{-1}\boldsymbol{b}$. The solution is feasible if $B^{-1}\boldsymbol{b} \geq \mathbf{0}$.

## 1.3 The simplex method

**Algorithm 1.1** (The simplex method)**.** Consider a standard minimization LP. Iteratively find basic feasible solutions with better objective values. [IV] [4.4] [4.6]

1. Choose any feasible basis and construct the corresponding basic solution.
2. Select a non-basic variable $x_j$ to enter the basis using the optimality condition (largest negative reduced cost, i.e. fastest objective value change).

   $$j = \arg\min_{j \in N} [\boldsymbol{c}_N^\mathsf{T}]_j - \boldsymbol{c}_B^\mathsf{T} B^{-1} N_j \tag{1.8}$$

   If no value is negative, the basis is optimal and we are done. In a maximization problem, find instead the largest positive marginal value.
3. Select a basic variable $x_i$ to leave the basis using the feasibility condition (smallest non-negative quotient i.e. first violated constraint)

   $$i = \arg\min_{j \in B} \left\{ \frac{[B^{-1}\boldsymbol{b}]_i}{[B^{-1}N]_{ij}} : [B^{-1}N]_{ij} > 0 \right\} \tag{1.9}$$

4. Compute the new basic solution by performing the swap and matrix operations, then go to 2.

The simplex tableau at any particular iteration is displayed in table 1.

Table 1: Simplex tableau at BFS $(B, N)$. The slack column $\boldsymbol{s}$ contains copies of certain columns from $(\boldsymbol{x}_B, \boldsymbol{x}_N)$ (if applicable).

| basis | $z$ | $\boldsymbol{x}_B$ | $\boldsymbol{x}_N$ | $\boldsymbol{s}$ | RHS |
|---|---|---|---|---|---|
| $z$ | 1 | $\mathbf{0}$ | $-(\boldsymbol{c}_N^\mathsf{T} - \boldsymbol{c}_B^\mathsf{T} B^{-1} N)$ | $\boldsymbol{c}_B^\mathsf{T} B^{-1}$ | $\boldsymbol{c}_B^\mathsf{T} B^{-1}\boldsymbol{b}$ |
| $\boldsymbol{x}_B$ | $\mathbf{0}$ | I | $B^{-1}N$ | $B^{-1}$ | $B^{-1}\boldsymbol{b}$ |

When some non-basic variable $j$ is entering the basis, the search direction can be computed as

$$\boldsymbol{d} = (\boldsymbol{d}_B,\ \boldsymbol{d}_N)^\mathsf{T} = (-B^{-1}N_j,\ \boldsymbol{e}_j)^\mathsf{T} \tag{1.10}$$

where $\boldsymbol{e}_j$ is a standard basis vector such that the component corresponding to the variable is one and the rest zero and $N_j = N\boldsymbol{e}_j$ (the corresponding column).

### 1.3.1 Phase I problem

Suppose one could not easily find an initial basic feasible solution. Assuming that $\boldsymbol{b} \geq \mathbf{0}$, we introduce an *artificial variable* $a_i$ in each row lacking an unit column and obtain a phase I-problem to solve:

$$\begin{aligned} \min \quad & w = \mathbf{1}^\mathsf{T}\boldsymbol{a} \\ \text{s.t.} \quad & A\boldsymbol{x} + I\boldsymbol{a} = \boldsymbol{b} \\ & \boldsymbol{x}, \boldsymbol{a} \geq \mathbf{0}. \end{aligned} \tag{1.11}$$

If a feasible solution exists, we obtain the optimal value $w^* = 0$ and the corresponding optimal basic solution is feasible in the original problem. Otherwise, by construction, there exists no feasible solution to the original problem.

### 1.3.2 Degeneracy and convergence

If the smallest non-negative quotient in the feasibility condition is zero, the value of a basic variable will become zero in the next iteration and we obtain a *degenerate solution*. This occurs when a redundant constraint 'grazes' the feasible set. No improvement in objective value is made and we risk cycling around non-optimal bases. To avoid this, we may change the criteria (e.g. Bland's rule which sorts) or perturb the RHS (modern software).

*Remark.* If all of the basic feasible solutions are non-degenerate, then the simplex algorithm terminates after a finite number of iterations.

### 1.3.3 Multiple optimal solutions

If the entering variable has zero reduced cost, then there are multiple optimal extreme points. All points on the edge between the optimal extreme points are optimal.

### 1.3.4 Unbounded solutions

If all quotients in the feasibility condition are negative, the value of the entering variable may increase indefinitely. Thus, the feasible set is unbounded.

## 1.4 Duality

The *dual LP* is derived from the original *primal LP* such that their solutions both yield information about the problem. When then primal problem is interpreted as the production problem, the dual problem may be interpreted as the market problem. One can also see the dual variables as the weights of the constraints in the primal problem.

*Remark.* The dual of the dual is the primal.

### 1.4.1 Constructing the dual program

We invert the problem sense, take the transpose of the constraint coefficients, and swap objective coefficients and RHS. Then we apply the translation rules in table 2. For a standard problem, this implies

$$\begin{array}{lll} \min & z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} & \\ \text{s.t.} & A\boldsymbol{x} \leq \boldsymbol{b} & \iff \\ & \boldsymbol{x} \geq \boldsymbol{0} & \end{array} \qquad \begin{array}{lll} \max & w = \boldsymbol{b}^\mathsf{T}\boldsymbol{y} & \\ \text{s.t.} & A^\mathsf{T}\boldsymbol{y} \leq \boldsymbol{c} & \\ & \boldsymbol{y} \leq \boldsymbol{0} & \end{array} \tag{1.12}$$

Table 2: Translation rules for construction of the dual program [V] [6.2]

| maximization | constraints | | | variables | | |
|---|---|---|---|---|---|---|
| dual/primal | $\geq$ | $\leq$ | $=$ | $\geq 0$ | $\leq 0$ | free |
| **minimization** | **variables** | | | **constraints** | | |
| primal/dual | $\leq 0$ | $\geq 0$ | free | $\geq$ | $\leq$ | $=$ |

### 1.4.2 Properties

**Theorem 1.2** (Weak duality)**.** *Consider* (1.12)*. Let $\boldsymbol{x}$ be a feasible point in the primal and $\boldsymbol{y}$ be a feasible point in the dual. Then it holds that $z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \geq \boldsymbol{b}^\mathsf{T}\boldsymbol{y} = w$, i.e. the dual optimum gives an optimistic estimate of the primal optimum. [V] [6.3 Th. 6.1]*

*Proof.*

$$z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \geq \boldsymbol{y}^\mathsf{T}A\boldsymbol{x} \geq \boldsymbol{y}^\mathsf{T}\boldsymbol{b} = w. \tag{1.13}$$

□

**Corollary 1.3** (Optimality by weak duality)**.** *If $\overline{\boldsymbol{x}}$ is feasible in the primal and $\overline{\boldsymbol{y}}$ is feasible in the dual, and it holds that $\boldsymbol{c}^\mathsf{T}\overline{\boldsymbol{x}} = \boldsymbol{b}^\mathsf{T}\overline{\boldsymbol{y}}$, then $\overline{\boldsymbol{x}}$ is optimal in the primal and $\overline{\boldsymbol{y}}$ is optimal in the dual. [V] [6.3 Th. 6.2]*

**Theorem 1.4** (Strong duality)**.** *In a pair of primal and dual LPs, if one of them has a bounded optimal solution $\hat{\boldsymbol{x}}$ (or $\hat{\boldsymbol{y}}$), so does the other i.e. $\hat{\boldsymbol{y}}$ (or $\hat{\boldsymbol{x}}$), and their optimal values are equal: $\boldsymbol{c}^\mathsf{T}\hat{\boldsymbol{x}} = \boldsymbol{b}^\mathsf{T}\hat{\boldsymbol{y}}$. [V] [6.3 Th. 6.3]*

**Theorem 1.5** (Complementary slackness)**.** *If $\boldsymbol{x}$ is optimal in the primal and $\boldsymbol{y}$ is optimal in the dual, then it holds that*

$$\boldsymbol{x}^\mathsf{T}(\boldsymbol{c} - A^\mathsf{T}\boldsymbol{y}) = \boldsymbol{y}^\mathsf{T}(\boldsymbol{b} - A\boldsymbol{x}) = 0. \tag{1.14}$$

*If $\boldsymbol{x}$ and $\boldsymbol{y}$ are feasible in their respective problems, and the above holds, then $\boldsymbol{x}$ and $\boldsymbol{y}$ are optimal in their respective problems. [V] [6.3 Th. 6.5]*

This implies that if $y_i > 0$ for some $i$, then the slack $s_i$ in the corresponding primal constraint must be zero (the constraint is active).

**Corollary 1.6** (Duality theorem)**.** *Suppose that $\boldsymbol{x}_B = B^{-1}\boldsymbol{b}$ is an optimal BFS to the primal problem* (1.5)*. Then $\boldsymbol{y}^\mathsf{T} = \boldsymbol{c}_B^\mathsf{T} B^{-1}$ is an optimal solution to the dual problem and $z^* = w^*$.* *[V] [6.3 Th. 6.4]*

Note that the optimal solutions to the dual variables correspond to the cost coefficients in the simplex tableau for the slack variables in the optimal BFS. Some relations between corresponding primal and dual optimal solutions are listed in table 3.

Table 3: Relations between primal and dual optimal solutions [V]

| primal (dual) problem | $\iff$ | dual (primal) problem |
|---|---|---|
| unique, non-degenerate solution | $\iff$ | unique, non-degenerate solution |
| unbounded solution | $\implies$ | no feasible solution |
| no feasible solutions | $\implies$ | unbounded/no feasible solution |
| degenerate solutions | $\iff$ | alternative solutions |

## 1.5 Post-optimal sensitivity analysis

Assume that the basis in (1.7) is optimal.

### 1.5.1 Changes in right-hand-side coefficients

**Definition 1.6** (Shadow price)**.** The *shadow price* of a constraint is defined as the change in the optimal value as a function of the (marginal) change in the RHS. It equals the optimal value of the corresponding dual variable $\boldsymbol{y}^\mathsf{T} = \boldsymbol{c}_B^\mathsf{T} B^{-1}$. [VI] [5 Def. 5.3]

Suppose $\boldsymbol{b}$ changes to $\boldsymbol{b} + \boldsymbol{\Delta b}$. The new optimal value becomes

$$z^{\text{new}} = \boldsymbol{c}_B^\mathsf{T} B^{-1}(\boldsymbol{b} + \boldsymbol{\Delta b}) = z + \boldsymbol{c}_B^\mathsf{T} B^{-1}\boldsymbol{\Delta b} \tag{1.15}$$

and the current basis is still feasible if $B^{-1}(\boldsymbol{b} + \boldsymbol{\Delta b}) \geq 0$. Otherwise, we find a new solution using the *dual simplex method*. [7.3]

### 1.5.2 Changes in the objective coefficients

**Definition 1.7** (Reduced cost)**.** The *reduced cost* of a non-basic variable defines the change in the objective value when the value of the corresponding variable is (marginally) increased. The basis $B$ is optimal if the reduced costs are non-negative, i.e. $\boldsymbol{c}_N^\mathsf{T} - \boldsymbol{c}_B^\mathsf{T} B^{-1} N \geq \boldsymbol{0}$. [VI]

Suppose $\boldsymbol{c}$ changes to $\boldsymbol{c} + \boldsymbol{\Delta c}$. The new optimal value becomes

$$z^{\text{new}} = (\boldsymbol{c}_B + \boldsymbol{\Delta c}_B)^\mathsf{T} B^{-1}\boldsymbol{b} = z + \boldsymbol{\Delta c}_B^\mathsf{T} B^{-1}\boldsymbol{b} \tag{1.16}$$

and the current basis is optimal if

$$(\boldsymbol{c}_N + \boldsymbol{\Delta c}_N)^\mathsf{T} - (\boldsymbol{c}_B + \boldsymbol{\Delta c}_B)^\mathsf{T} B^{-1} N \geq \mathbf{0}. \tag{1.17}$$

# 2 Discrete and combinatorial optimization

Consider the standard minimization ILP

$$\begin{aligned} \min \quad & z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} && \boldsymbol{c}, \boldsymbol{x} \in \mathbb{R}^n \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} && A \in \mathbb{R}^{n\times m} \\ & \boldsymbol{x} \geq \mathbf{0}, \text{ integer} && \boldsymbol{b} \in \mathbb{R}^m. \end{aligned} \tag{2.1}$$

Contrary to (1.1), adding integrality requirements results in a non-convex feasible set $X := \{\boldsymbol{x} \in \mathbb{Z}_+^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$. One can no longer prove optimality using strong duality/complementarity as used in solving LP problems.

Instead we find bounds on the optimal value $z^*$ and try to reconcile them. We may obtain an optimistic estimate $\underline{z} \leq z^*$ from a relaxation of (2.1), and a pessimistic estimate $\overline{z} \geq z^*$ from any feasible solution to (2.1). If $\overline{z} - \underline{z} \leq \varepsilon$ we can conclude that the value of our solution candidate $\overline{\boldsymbol{x}}$ is at most $\varepsilon$ from $z^*$.

We cover several different methods used in the solution of ILPs.

## 2.1 Relaxations

To relax an ILP, one can enlarge the feasible set $X$ by removing constraints, replace the objective function $\boldsymbol{c}^\mathsf{T}\boldsymbol{x}$ by an underestimating (in the minimization case) function $f$ such that $f(\boldsymbol{x}) \leq c^\mathsf{T}\boldsymbol{x}$ for all $\boldsymbol{x} \in X$, or do both. [VIIIa] [14.2]

### 2.1.1 LP-relaxation

A common and simple method of relaxation is to simply consider the problem without integrality. If $X := \{\boldsymbol{x} \in \mathbb{Z}_+^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$, and $X^{\text{LP}} := \{\boldsymbol{x} \geq 0 : A\boldsymbol{x} \leq \boldsymbol{b}\}$, then clearly $X \subseteq X^{\text{LP}}$ and we have enlarged the feasible set. It must then hold that $\min_{\boldsymbol{x}\in X^{\text{LP}}} \boldsymbol{c}^\mathsf{T}\boldsymbol{x} =: z^{\text{LP}} \leq z^*$ (or the corresponding relation for a maximization).

Such a relaxation is called the *LP-relaxation*. It has the added benefit of being relatively easy to solve compared to the original problem, as we now have a convex feasible set.

### 2.1.2 Combinatorial relaxation

We may enlarge the feasible set by neglecting one or more constraints in the problem. Such a relaxation is called a *combinatorial relaxation* and may be of interest in particular problems.

### 2.1.3 Lagrangean relaxation

*Lagrangean relaxation* allows us to obtain optimistic estimates of the optimal objective value $z^*$ in the face of complicating constraints in the problem. Consider an ILP formulated as

$$\begin{aligned} \min \quad & z = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \leq \boldsymbol{b} \\ & D\boldsymbol{x} \leq \boldsymbol{d} \\ & \boldsymbol{x} \geq \boldsymbol{0}, \text{ integer} \end{aligned} \tag{2.2}$$

wherein the constraints $A\boldsymbol{x} \leq \boldsymbol{b}$ are considered complicating. Let $X := \{\boldsymbol{x} \in \mathbb{Z}_+^n : D\boldsymbol{x} \leq \boldsymbol{d}\}$ be the feasible set w.r.t. the remaining constraints, and allow violating the complicating constraints by paying a penalty in the *dual function*

$$h(\boldsymbol{v}) = \min_{\boldsymbol{x} \in X} \{\boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{v}^\mathsf{T}(A\boldsymbol{x} - \boldsymbol{b})\} \tag{2.3}$$

(pay attention to the signs of the constraints and $\boldsymbol{v}$). For each value of $\boldsymbol{v}$ we thus obtain a *subproblem*. [IX] [17.1–2]

**Theorem 2.1** (Weak duality of Lagrangean relaxation)**.** *For arbitrary $\boldsymbol{v} \geq 0$ it holds that $h(\boldsymbol{v}) \leq z^*$.*

*Proof.* Let $\overline{\boldsymbol{x}}$ be feasible in (2.2), i.e. both $\overline{\boldsymbol{x}} \in X$ and $A\overline{\boldsymbol{x}} \leq \boldsymbol{b}$ hold. Then

$$h(\boldsymbol{v}) \leq \boldsymbol{c}^\mathsf{T}\overline{\boldsymbol{x}} + \boldsymbol{v}^\mathsf{T}(A\overline{\boldsymbol{x}} - \boldsymbol{b}) \leq \boldsymbol{c}^\mathsf{T}\overline{\boldsymbol{x}}. \tag{2.4}$$

Since an optimal solution $\boldsymbol{x}^*$ to (2.2) also is feasible it holds that $h(\boldsymbol{v}) \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^* = z^*$. □

Such a lower bound may be used to e.g. cut branches or compare solutions. The best lower bound is provided by solving the *dual problem* $h^* = \max_{\boldsymbol{v} \geq 0} h(\boldsymbol{v})$. There exist special algorithms for this purpose (e.g. subgradient optimization). The dual function $h$ is always concave but typically non-differentiable (it is piecewise linear).

In general, ILPs typically show a non-zero duality gap $z^* - h^* > 0$. Furthermore, the bound is never worse than the LP relaxation bound $z^{\text{LP}} \leq h^* \leq z^*$. If $X$ has the integrality property, they are equal.

## 2.2 Branch-and-bound — Enumeration

A *branch-and-bound* algorithm (also known as *implicit enumeration* and *tree search*) is a divide and conquer approach to finding optimal solutions to optimization problems with integrality requirements. The general idea is to efficiently enumerate feasible solutions by successive partitioning of the feasible set in a search tree, and pruning branches using approximations provided by relaxations of the problem.

We may construct an algorithm by deciding on the relaxation, the branching strategy (e.g. fractional values, subtours), the tree search strategy (e.g. depth-, breadth-, best-first), and the pruning criteria. [VIIIa] [15]

**Algorithm 2.1** (Land-Doig-Dakin B&B)**.** Consider a minimization ILP. Let $P_k$ denote the $k$th subproblem. We leave the branching strategy (on which variable) and the search strategy undetermined.

0. Let $n = 0,\ k = 0$ and initialize the tree with a single node $P_0$ containing the original problem. Set the pessimistic estimate $\overline{z}$ with some heuristic, e.g. any feasible solution, or simply $\overline{z} = +\infty$.

1. Solve the LP relaxation of $P_k$. The solution $\boldsymbol{x}^k$ yields a local optimistic estimate $z_k$ in the subtree.

2. If $P_k$ has no feasible solution; go to step 6.

3. If $z_k > \overline{z}$ we cannot improve our bounds; go to step 6.

4. If $\boldsymbol{x}^k$ is integer we cannot improve our bounds. If $z_k < \overline{z}$ we update the global pessimistic estimate to $z_k$ and $\boldsymbol{x}^k$ becomes our new candidate $\hat{\boldsymbol{x}}$. Go to step 6.

5. Choose a non-integer variable $x_j$ and construct the two new subproblems

$$P_{n+1} : P_k \text{ with } x_j \leq \left\lfloor x_j^k \right\rfloor, \qquad P_{n+2} : P_k \text{ with } x_j \geq \left\lceil x_j^k \right\rceil,$$

   letting $n := n + 2$. Node $P_k$ is now visited.

6. If no unvisited nodes remain, we are done and the optimal solution is $\hat{\boldsymbol{x}}$ with optimal value $\overline{z}$. Else, select the next unvisited node $P_k$ and go to step 1.

## 2.3 Cutting plane algorithms

The ideal formulation of an ILP is one where the set defined by the constraints in the LP relaxation is exactly the convex hull of the feasible set in the ILP. Such an ideal linear program has integer extreme points and can thus be solved quickly using the simplex algorithm. [IX] [14.3]

By adding *valid inequalities*, i.e. constraints satisifed by the feasible set in the ILP but eliminate some fractional solutions, we can approach an ideal formulation of the problem. Unfortunately, very many cuts may be needed. For faster convergence we should ensure that each added cut passes through at least one integer point. Pure cutting plane algorithms are usually less efficient than branch-and-bound, but are used with good results in presolve stages by solvers. [IX] [14.4]

**Algorithm 2.2** (Gomory's cutting plane algorithm)**.** [14.5.1]

1. Solve the LP relaxation.
2. If the LP solution is integer, we have found an optimal solution to the ILP.
3. Consider the optimal basis $B$:

$$\boldsymbol{x}_B + B^{-1}N\boldsymbol{x}_N = B^{-1}\boldsymbol{b}. \tag{2.5}$$

For all $i \in B$, let $\overline{a}_{ij} = (B^{-1}N)_{ij}$ and $\overline{b}_i = (B^{-1}\boldsymbol{b})_i$, then

$$x_i + \sum_{j \in N} \overline{a}_{ij} x_j = \overline{b}_i. \tag{2.6}$$

Consider an $i \in B$ such that $\overline{b}_i$ is non-integer and define the fractions

$$\tilde{b}_i := \overline{b}_i - \lfloor \overline{b}_i \rfloor \in (0, 1) \tag{2.7}$$

$$\tilde{a}_{ij} := \overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor \in [0, 1),\ j \in N. \tag{2.8}$$

From (2.5) it follows

$$x_i + \sum_{j \in N} \lfloor \overline{a}_{ij} \rfloor x_j - \lfloor \overline{b}_i \rfloor = \tilde{b}_i - \sum_{j \in N} \tilde{a}_{ij} x_j. \tag{2.9}$$

By construction, the LHS of (2.9) is integer, so the RHS must also be integer. Since $\tilde{b}_i < 1$, $\tilde{a}_{ij} \geq 0$, and $x_j \geq 0$, $j \in N$, it then follows that

$$\tilde{b}_i - \sum_{j \in N} \tilde{a}_{ij} x_j < 1 \implies \tilde{b}_i - \sum_{j \in N} \tilde{a}_{ij} x_j \leq 0 \tag{2.10}$$

and we have derived the valid inequality $\sum_{j \in N} \tilde{a}_{ij} x_j \geq \tilde{b}_i$. The current basic solution will become infeasible.

4. Resolve the new problem and go to step 2.

## 2.4 Heuristic algorithms

We preface this section with a few definitions. Consider, with the necessary definitions implied, a minimization problem $\min_{\boldsymbol{x} \in X} \boldsymbol{c}^\mathsf{T}\boldsymbol{x}$.

**Definition 2.1** (Global minimum)**.** An $\boldsymbol{x}^* \in X$ such that $\forall \boldsymbol{x} \in X : \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^* \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x}$.

**Definition 2.2** ($\varepsilon$-neighbourhood of $\overline{\boldsymbol{x}}$)**.** $N_\varepsilon(\overline{\boldsymbol{x}}) = \{\boldsymbol{x} \in X : \|\boldsymbol{x} - \overline{\boldsymbol{x}}\| \leq \varepsilon\}$ where $\varepsilon \geq 0$, and the distance measure $\|\cdot\|$ may be freely defined, e.g. Hamming, Euclidean, Manhattan, TSP interchanges...

**Definition 2.3** (Local minimum)**.** An $\overline{\boldsymbol{x}} \in X$ such that $\forall \boldsymbol{x} \in N_\varepsilon(\overline{\boldsymbol{x}}) : \boldsymbol{c}^\mathsf{T}\overline{\boldsymbol{x}} \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x}$, for some $\varepsilon > 0$.

**Theorem 2.2** (Optima in convex problems)**.** *For a convex problem, any local optimum is also a global optimum.*

*Heuristic algorithms* are an opportunity to allow acceptable locally optimal solutions to otherwise intractable problems. While LP problems are convex (since the feasible set is convex and the objective function is linear and *a fortiori* convex) general ILP/BLPs are non-convex. Non-convex optimization problems do not fulfil strong duality, which means that one cannot in general construct (efficient) algorithms to fulfil optimality conditions. [X] [16]

### 2.4.1 Constructive heuristics

The algorithm begins with an 'empty set' and 'adds' elements according to some (simple) rule. Any greedy algorithm falls into this category. There may be no guarantee that even a feasible solution will be found, and no available measure of how 'close' to a global optimum the solution is. [X] [16.3]

Structured problems may possess special appropriate rules and may even have constructive heuristics that generate provably optimal solutions (e.g. MST).

### 2.4.2 Local search

**Algorithm 2.3** (Local search)**.** Iteratively improve a feasible solution to find a local optimum.

0. Initialize with a feasible solution $\boldsymbol{x}^0 \in X$. Let $k := 0$.
1. Find all feasible points in an $\varepsilon$-neighbourhood of $\boldsymbol{x}^k$ $N_\varepsilon(\boldsymbol{x}^k)$.
2. If $\boldsymbol{c}^\mathsf{T}\boldsymbol{x}^k \leq \boldsymbol{c}^\mathsf{T}\boldsymbol{x}\ \forall \boldsymbol{x} \in N_\varepsilon(\boldsymbol{x}^k)$; stop, we have found a local optimum.
3. Choose $\boldsymbol{x}^{k+1} \in N_\varepsilon(\boldsymbol{x}^k)$ such that $\boldsymbol{c}^\mathsf{T}\boldsymbol{x}^{k+1} < \boldsymbol{c}^\mathsf{T}\boldsymbol{x}^k$.
4. Let $k := k + 1$ and go to step 1.

There is no measure on how close to a global optimum the resulting solution is. [X] [16.4]

### 2.4.3 Approximation algorithms

A category of specialized algorithms for specific problems that come with performance guarantees. Let $\overline{z} := \boldsymbol{c}^\mathsf{T}\overline{\boldsymbol{x}}$ for some $\overline{\boldsymbol{x}} \in X$ be computed by an approximation algorithm. Then, for some $\alpha \in (0, 1]$ pertaining to the algorithm

$$\frac{\overline{z} - z^*}{z^*} \leq \alpha \tag{2.11}$$

should hold. [X] [16.6]

### 2.4.4 Metaheuristics

Metaheuristics intend to be more efficient than plain local search by guiding in a systematic and efficient way. Some examples are [X] [16.5]

- Tabu search — allow worse solutions and prohibit returning steps [16.5]
- Simulated annealing — probability of movement

One might (perhaps facetiously) remark that these methods are usually used by people ignorant of the mathematics behind optimization.

General combinations of heuristics also fall under this category. A common useful method is to start using a constructive heuristic to obtain a feasible solution, and then continue using local search (with a model-specific neighbourhood definition) to obtain a locally optimal solution. In general, there is however no guarantee of global optimality.

# 3 Network flows

Such problems are concerned with flows represented on some graph. There exist supply nodes that generate flow and demand nodes that sink flow with storage nodes in between. The links have limited capacities. One might wish for example to minimize costs for transport and storage, or maximize the flow rate through the network. [XI] [8.6.1]

Many problems can be represented as network flows: pipeline capacity, project scheduling, logistics, maintenance windows... We will later see that there exists specialized algorithms and special properties of these problems that help us solve them.

## 3.1 Shortest path

Let $G = (N, A, \boldsymbol{d})$ be a directed graph. We wish to find the shortest path from node $s \in N$ to $t \in N$ with respect to the edge weights in $\boldsymbol{d}$. See also specialized algorithms in A.3.2, which are more efficient than solving using a general LP method.

### 3.1.1 'Stretch' model (dual)

Let $y_t$ be the length of the shortest path from node $s$ to $t$. The idea is to 'stretch the arcs' in the network between $s$ and $t$, i.e. maximize the difference between their two potentials $y_t - y_s$. Since the arcs are not elastic, their lengths are constraining, and physical intuition says the constraining arcs are candidates for inclusion in the shortest path. All involved functions are potential differences, as seen in the LP formulation

$$\begin{aligned} \max \quad & y_t - y_s \\ \text{s.t.} \quad & y_j - y_i \leq d_{ij} \qquad (i,j) \in A \\ & y_k \text{ free} \qquad\qquad k \in N. \end{aligned} \tag{3.1}$$

### 3.1.2 Flow model (primal)

We send one unit of flow along the shortest path from node $s$ to $t$. Let

$$x_{ij} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is in the shortest path,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

The goal is then to minimize the cost (length) of the chosen links, respecting the node balance; any entering flow must also leave (except at source and destination).

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} d_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i:(i,k)\in A} x_{ik} - \sum_{j:(k,j)\in A} x_{kj} = \begin{cases} -1, & k = s, \\ +1, & k = t, \\ 0, & k \in N \setminus \{s,t\} \end{cases} \\ & x_{ij} \geq 0 \qquad\qquad (i,j) \in A. \end{aligned} \tag{3.3}$$

Due to the particular structure of the constraint matrix, binary constraints are unnecessary for this problem. As the reader has already surmised, problems (3.3) and (3.1) are a primal-dual pair.

## 3.2 Maximum flow

Let $G = (N, A, K)$ be a directed graph, where $K_{ij}$ denotes the capacity on arc $(i,j) \in A$. Let $x_{ij}$ denote the amount of flow through the directed arc $(i.j)$ and $v$ denote the total flow from source $s \in N$ to sink $t \in N$. We define the model

$$\begin{aligned} \max_{x,v} \quad & v & & & \text{max flow} \\ \text{s.t.} \quad & -\sum_{j\in N:(s,j)\in A} x_{sj} = & -v & & \text{balance } s \\ & \sum_{i\in N:(i,t)\in A} x_{it} = & v & & \text{balance } t \\ & \sum_{i\in N:(i,k)\in A} x_{ik} - \sum_{j\in N:(k,j)\in A} x_{kj} = & 0 & k \in N \setminus \{s,t\} & \text{balance } k \\ & x_{ij} \leq & K_{ij} & (i.j) \in A & \text{capacity} \\ & x_{ij} \geq & 0 & (i.j) \in A & \end{aligned} \tag{3.4}$$

Compare the shortest path problem; we obtain a similar structure in the equality constraints. If we have multiple sources/sinks we can replace them with a single artificial source/sink with outward capacities to the sources/sink equalling their respective supply/demand. We can also consider this a circulating network by defining $x_{ts} := v$, since their flows are balanced. [XII]

### 3.2.1 Edmonds-Karp algorithm

**Algorithm 3.1** (Edmonds-Karp)**.** Solution method for maximum flow problems (although not the most efficient). [XII]

1. Let $k := 0$, $v^0 := 0$, $x_{ij}^0 := 0$, and $u_{ij}^0 := K_{ij}\ \forall (i,j) \in A$.

2. Find a maximum capacity path $P^k \subset A$ from $s$ to $t$ (modified shortest path algorithm). The capacity of $P^k$ is

$$\hat{u}^k := \min\{\min\{u_{ij}^k : (i,j) \in P^k\}; \min\{x_{ij}^k : (j,i) \in P^k\}\}. \tag{3.5}$$

   If $\hat{u}^k = 0$ go to step 4.

3. Update the flows, capacities and total flow:

$$x_{ij}^{k+1} = \begin{cases} x_{ij}^k + \hat{u}^k & \text{if } (i,j) \in P^k \\ x_{ij}^k - \hat{u}^k & \text{if } (j,i) \in P^k \\ x_{ij}^k & \text{otherwise} \end{cases} \qquad u_{ij}^{k+1} = \begin{cases} u_{ij}^k - \hat{u}^k & \text{if } (i,j) \in P^k \\ u_{ij}^k + \hat{u}^k & \text{if } (j,i) \in P^k \\ u_{ij}^k & \text{otherwise} \end{cases}$$
$$v^{k+1} := v^k + \hat{u}^k. \tag{3.6}$$

   Let $k := k + 1$ and go to step 2.

4. The maximum total flow equals $v^k$ with solution $x_{ij}^k,\ (i,j) \in A$.

### 3.2.2 Minimum cut — LP dual

An $(s,t)$-cut is a set of arcs which, when deleted, interrupt all flow in the network between the source $s$ and the sink $t$. The cut capacity equals the capacities on all the forward arcs through the $(s,t)$-cut. Finding the minimum $(s,t)$-cut is equivalent to solviong the dual of the maximum flow problem.

Let $\gamma_{ij}$ indicate whether arc $(i,j)$ passes through the minimum cut (in a forward direction), and $\pi_k$ indicate whether node $k$ can be reached by more flow from the source.

$$\begin{aligned} \min_{\pi,\gamma} \quad & \sum_{(i,j)\in A} K_{ij}\gamma_{ij} && && \text{min cut} \\ \text{s.t.} \quad & -\pi_i + \pi_j + \gamma_{ij} \geq && 0 \quad && (i,j) \in A \\ & -\pi_t + \pi_s = && 1 \quad && \\ & \pi_k && \text{free} \quad && k \in N \\ & \gamma_{ij} \geq && 0 \quad && (i.j) \in A \end{aligned} \tag{3.7}$$

The optimal solution must have binary variables.

**Corollary 3.1** (Weak duality — max flow/min cut)**.** *Each feasible flow $x_{ij},\ (i,j) \in A$ yields a lower bound on $v^*$; the capacity of each $(s,t)$-cut yields an upper bound on $v^*$.*

**Corollary 3.2** (Strong duality — max flow/min cut)**.** *The value of the maximum flow equals the capacity of the minimum cut.*

### 3.2.3 General minimum cost network flow problems

Consider a general network $G = (N, A)$ consisting of a set of nodes $N$ linked by a set $A$ of arcs. Each node $i$ in the network has a net demand $d_i$ (a supply is described by negative demand). We associate a distance/cost $c_{ij}$ with each arc $(i,j) \in A$. Each arc then carries an (to be determined) amount of flow $x_{ij}$ restricted by a maximum capacity $u_{ij} \in [0, \infty]$ and a minimum capacity $\ell_{ij} \in [0, u_{ij}]$. The flow through each node must be balanced. [XII]

$$\begin{aligned} \min_{x} \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{i\in N:(i,k)\in A} x_{ik} - \sum_{j\in N:(k,j)\in A} x_{kj} = \quad d_k \quad k \in N \\ & x_{ij} \leq \quad u_{ij} \quad (i.j) \in A \\ & x_{ij} \geq \quad \ell_{ij} \quad (i.j) \in A \end{aligned} \tag{3.8}$$

As long as every parameter involved is integer, all extreme points of the feasible set are integral due to the unimodularity of the (equality) constraint matrix. [8.6.3]

The corresponding dual problem is

$$\begin{aligned} \max_{\pi,\alpha,\beta} \quad & \sum_{k\in N} d_k\pi_k + \sum_{(i,j)\in A} (\ell_{ij}\alpha_{ij} - u_{ij}\beta_{ij}) \\ \text{s.t.} \quad & \pi_j - \pi_i + \alpha_{ij} - \beta_{ij} = \quad c_{ij} \quad (i.j) \in A \\ & \alpha_{ij}, \beta_{ij} \geq \quad 0 \quad (i.j) \in A \end{aligned} \tag{3.9}$$

# 4 Multi-objective optimization

Many practical problems have several conflicting objectives. Some goals cannot be reduced to a common scale of cost/profit and we must make a trade-off.

**Definition 4.1** (Pareto optimal)**.** A solution is *Pareto optimal* if no other feasible solution has a better value in all objectives.

The set of Pareto optima is also called the non-dominated points or the efficient frontier, which stems from the fact that solutions are usually visualized in the objective space (at least for small numbers of objectives). In that case, the non-dominated points

have no neighbours in the *optimality cone*. Note that the set of Pareto optima is not necessarily convex; in fact, if the problem has integrality constraints, the set may not even be connected. [XIII]

## 4.1 Solution methods

### 4.1.1 $\varepsilon$-constraints method

Construct the efficient frontier by treating all but one objective as a constraint, optimizing for the remaining one. We enforce that that the secondary objective $k$ must be greater than or equal to some $\varepsilon_k$, for a suitable range of such right-hand sides. Usually the worst and best values are chosen as endpoints for the range. The solutions to these problems are points on the Pareto front. [XIII]

However, adding constraints might make the problem significantly harder. Also, the number of problems to solve grows exponentially with the number of objectives, so for large sets of objectives this method might not be tractable.

### 4.1.2 Weighted objectives

Give each maximization (minimization) objective a positive (negative) weight and then solve a single-objective maximization problem. This always yields an efficient solution, with comparatively few extra computations. [XIII]

However, spread out weights do not necessarily produce solutions that are evenly distributed on the efficient frontier. Furthermore, if the objectives are non-concave (max), or if the feasible set is non-convex (e.g. integral), then not all points on the efficient frontier may be possible to detect using weighted sums of objectives.

### 4.1.3 Soft constraints

Consider the multiobjective optimization problem

$$\max_{\boldsymbol{x}} [f_k(\boldsymbol{x})]_{k=1}^K \text{ s.t. } \boldsymbol{x} \in X. \tag{4.1}$$

Define a target value $t_k$ and a deficiency variable $d_k \geq 0$ for each objective $f_k$, and from them construct a soft constraint

$$f_k(\boldsymbol{x}) + d_k \geq t_k. \tag{4.2}$$

The new problem is to minimize the sum of deficiencies $\sum_{k=1}^K d_k$ such that the soft constraints hold and $\boldsymbol{x} \in X$. [XIII]

The target values should be set too optimistic. We will then find a point on the Pareto front such that we minimize the Manhattan distance to the target values. It is important that the objectives have a common scale for such a problem to make sense.

### 4.2 Normalization

Consider the multiobjective optimization problem (4.1). Define for $k = 1, \ldots, K$ the rescaled objectives $\tilde{f}_k(\boldsymbol{x}) \in [0, 1]$:

$$\tilde{f}_k(\boldsymbol{x}) = \frac{f_k(\boldsymbol{x}) - f_k^{\min}}{f_k^{\max} - f_k^{\min}} \tag{4.3}$$

where $f_k^{\min} := \min_{\boldsymbol{x} \in X} f_k(\boldsymbol{x})$ and $f_k^{\max} := \max_{\boldsymbol{x} \in X} f_k(\boldsymbol{x})$. The new unitless objectives have a common scale and can be directly compared.

## 5 Non-linear optimization

Non-linear programming has several areas of application: vehicle design, architecture, traffic networks, least squares approximation etc. Function evaluations may be computationally intensive due to time consuming simulations, and surrogate models may be required. [XIV] [9.1]

A general non-linear program is described as follows:

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^n} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & g_i(\boldsymbol{x}) \leq 0, \qquad i \in \mathcal{L} \\ & h_j(\boldsymbol{x}) = 0, \qquad j \in \mathcal{E}. \end{aligned} \tag{5.1}$$

Special cases of these are

- Unconstrained problems ($\mathcal{L} = \mathcal{E} = \varnothing$)
- Convex programming ($f$ convex, $g_i,\ i \in \mathcal{L}$ convex, $h_j,\ j \in \mathcal{E}$ linear)
- Linear constraints ($g_i,\ i \in \mathcal{L}$ and $h_j,\ j \in \mathcal{E}$ linear)
  - Quadratic programming ($f(\boldsymbol{x}) = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^\mathsf{T}Q\boldsymbol{x}$)
  - Linear programming ($f(\boldsymbol{x}) = \boldsymbol{c}^\mathsf{T}\boldsymbol{x}$)

The general mathematical properties of nonlinear optimization may be very different. No one-size-fits-all algorithm exists. An optimal solution is not necessarily located at an extreme point. Non-linear programs can be unconstrained (such an LP would be unbounded). The objective function $f$ may be non-differentiable (e.g. the piecewise linear Lagrangean dual objective function). Local optima may exist that are not global optima (compare integer linear optimization).

## 5.1 Definitions

Solely by considering where possible extrema are located in the problem $\min_{\boldsymbol{x}\in S} f(\boldsymbol{x})$, i.e. boundary points of $S$, stationary points of $f$, and discontinuities in $f$ or $f'$ (which can be modelled using integer variables), we motivate several definitions.

**Definition 5.1** (Boundary point)**.** The point $\overline{\boldsymbol{x}}$ is a *boundary point* to the feasible set $S = \{\boldsymbol{x} \in \mathbb{R}^n : g_i(\boldsymbol{x}) \leq 0,\ i \in \mathcal{L}\}$ if $g_i(\overline{\boldsymbol{x}}) \leq 0,\ i \in \mathcal{L}$ and $g_i(\overline{\boldsymbol{x}}) = 0$ for at least one index $i \in \mathcal{L}$. [XIV] [10.0]

**Definition 5.2** (Stationary point)**.** The point $\overline{\boldsymbol{x}}$ is a *stationary point* to $f$ if $\nabla f(\overline{\boldsymbol{x}}) = \mathbf{0}$. [XIV] [10.0]

Consider the non-linear problem $\min_{\boldsymbol{x}\in S} f(\boldsymbol{x})$. The following are generalizations of the previous definitions 2.3, 2.1.

**Definition 5.3** (Local minimum, general)**.** The point $\overline{\boldsymbol{x}}$ is a *local minimum* if $\overline{\boldsymbol{x}} \in S$ and $\exists \varepsilon > 0$ such that $f(\overline{\boldsymbol{x}}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{\boldsymbol{y} \in S : \|\boldsymbol{y} - \overline{\boldsymbol{x}}\| \leq \varepsilon\}$. [XIV] [2.4]

**Definition 5.4** (Global minimum, general)**.** The point $\overline{\boldsymbol{x}}$ is a *global minimum* if $\overline{\boldsymbol{x}} \in S$ and $f(\overline{\boldsymbol{x}}) \leq f(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$. [XIV] [2.4]

## 5.2 Convex optimization problems

**Definition 5.5** (Convex function)**.** A function $f$ is *convex* on $S$ if, for any $\boldsymbol{x}, \boldsymbol{y} \in S$ it holds that $f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y})\ \forall \alpha \in [0,1]$.

Hence, a function $f$ is *strictly convex* if, for any $\boldsymbol{x}, \boldsymbol{y} \in S$ such that $\boldsymbol{x} \neq \boldsymbol{y}$ it holds that $f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) < \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y})\ \forall \alpha \in (0,1)$.

Recall the definition of convex set in 1.3. If all functions involved are convex or concave (min or max respectively, differing by a sign) and all sets are convex the optimization problem becomes convex. [XIV] [9.3]

**Definition 5.6** (Convex optimization problem)**.** If $f$ and $g_i,\ i \in \mathcal{L}$ are convex functions, then the problem to minimize $f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0,\ i \in \mathcal{L}$ is said to be a *convex* optimization problem.

**Theorem 5.1** (Convex constraints yield convex sets)**.** *If all the functions $g_i,\ i \in \mathcal{L}$ are convex on $\mathbb{R}^n$, then the set $S = \{\boldsymbol{x} \in \mathbb{R}^n : g_i(\boldsymbol{x}) \leq 0,\ i \in \mathcal{L}\}$ is convex.*

We collect some useful, well-known results:

**Proposition 5.2** (Intersection of convex sets)**.** *If $X_i$ are convex $\forall i$, then $\cap_i X_i$ is convex.*

**Proposition 5.3** (Hessian test)**.** *A function $f$ is convex if its* Hessian $\nabla^2 f$ *is positive semidefinite.*

Algorithms (typically) converge to an optimal solution on such problems. Recall a very important property of convex problems: *a local optimum is a global optimum* (theorem 2.2).

## 5.3 Karush-Kuhn-Tucker (KKT) conditions

Let $S = \{\boldsymbol{x} \in \mathbb{R}^n : g_i(\boldsymbol{x}) \leq 0,\ i \in \mathcal{L}\}$.. Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, $g_i : \mathbb{R}^n \to \mathbb{R},\ i \in \mathcal{L}$ are convex and differentiable, and there exists a point $\overline{\boldsymbol{x}} \in S$ such that $g_i(\overline{\boldsymbol{x}}) < 0,\ i \in \mathcal{L}$.

If $\boldsymbol{x}^* \in S$ is a local minimum of $f$ over $S$, then there exists a vector $\boldsymbol{\mu} \in \mathbb{R}^m$ (where $m = |\mathcal{L}|$ such that

$$\begin{aligned} \nabla f(\boldsymbol{x}^*) + \sum_{i \in \mathcal{L}} \mu_i \nabla g_i(\boldsymbol{x}^*) &= \mathbf{0} \\ \mu_i g_i(\boldsymbol{x}^*) &= 0 \quad i \in \mathcal{L} \\ \boldsymbol{\mu} &\geq \mathbf{0}. \end{aligned} \tag{5.2}$$

As a special case, note that if all functions are convex and differentiable the conditions are *sufficient* for $\boldsymbol{x}^*$ to be a global minimum of $f$ over $S$. [p. 289]

For unconstrained problems, the conditions decompose to $\nabla f(\boldsymbol{x}) = \mathbf{0}$. They can also be stated for problems with equality constraints. For a quadratic program they form a system of linear (in)equalities plus the complementarity constraints, which is used in specialized algorithms.

The conditions allow us to verify (local) optimal solutions, solve certain special cases, construct more general algorithms and derive properties of a solution to a non-linear program. [XIV]

## 5.4 General iterative search method

**Algorithm 5.1** (Unconstrained iterative search)**.** Minimization. Many different choices are possible for subalgorithms in steps 2, 3 and 4. [XIV] [2.5.1]

1. Choose a starting solution $\boldsymbol{x}^0 \in \mathbb{R}^n$. Let $k := 0$.
2. Determine (somehow) a search direction $\boldsymbol{d}^k$.
3. If (some chosen) termination criterion is fulfilled, stop.
4. Determine a step length $t_k$ by (somehow) solving $\arg\min_{t \geq 0} f(\boldsymbol{x}^k + t\boldsymbol{d}^k)$.
5. Let $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + t\boldsymbol{d}^k$ be the new iteration point, $k := k + 1$, and go to step 2.

### 5.4.1 Search direction

If we are minimizing, the goal is to ensure $f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k)$. Consider the Taylor expansion of $f$ along a direction $\boldsymbol{d}^k$ [XIV] [9.2]

$$f(\boldsymbol{x}^k + t\boldsymbol{d}^k) = f(\boldsymbol{x}^k) + t\nabla f(\boldsymbol{x}^k)^\mathsf{T}\boldsymbol{d}^k + \mathcal{O}(t^2) \tag{5.3}$$

which implies for sufficiently small $t > 0$ that

$$f(\boldsymbol{x}^k + t\boldsymbol{d}^k) < f(\boldsymbol{x}^k) \implies \nabla f(\boldsymbol{x}^k)^\mathsf{T} \boldsymbol{d}^k < 0. \tag{5.4}$$

Thus we make the following definition:

**Definition 5.7** (Descent direction)**.** If $\nabla f(\boldsymbol{x}^k)^\mathsf{T} \boldsymbol{d}^k < 0$ (or $> 0$) then $\boldsymbol{d}^k$ is a descent (or an ascent, respectively) direction for $f$ at $\boldsymbol{x}^k$.

To minimize, we thus choose $\boldsymbol{d}^k$ as a descent direction from $\boldsymbol{x}^k$. [XIV] [10]

### 5.4.2 Step length — line search

Once a direction is determined, the remaining problem to find the step length simplifies to a one-dimension problem. Although analytically solvable in theory, it is rarely possible in practice. Several different numerical algorithms exist, e.g.

- The golden section method (reduce interval of uncertainty),
- The bi-section method (reduce interval of uncertainty),
- Newton-Raphson's method,
- Armijo's method.

These should be recognized by any reader who has completed a numerical analysis course. Again, in practice the exact minimum is not obtained but rather a sufficient improvement of the function value is made. [XIV] [10.4]

### 5.4.3 Termination criterion

Since $\nabla f(\boldsymbol{x}^k) = \boldsymbol{0}$ will not be exactly attained, termination criteria are required. Typical examples are

- $\left\|\nabla f(\boldsymbol{x}^k)\right\| < \varepsilon_1$
- $|f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^k)| < \varepsilon_2$
- $\left\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\right\| < \varepsilon_3$
- $t_k < \varepsilon_4$

where the $\varepsilon_j > 0$ are selected tolerance parameters. Several of the criteria ore often combined. The search method only guarantees a stationary solution, the properties of which are determined by $f$. [XIV]

# A Particular problems

## A.1 Standard ILP models

### A.1.1 Common 0-1 constraints

A short table of common 0-1 modelling situations is collected in the table 4. For further reference on formulating models, the author recommends the work by G. Brown and R. Dell, 'Formulating Integer Linear Programs: A Rogues' Gallery', *INFORMS Transactions on Education*, vol. 7, no. 2, pp. 153-159, 2007. [VII] [13.1]

Table 4: 0-1 constraint examples, where $y_i \in \{0,1\}$ and $x_j \geq 0$ continuous/integer. Note that depending on problem sense, some constraints may be unneeded.

| | constraints | notes |
|---|---|---|
| $y_1 \implies y_2$ | $y_1 \leq y_2$ | |
| $y_1 \wedge y_2 \implies y_3$ | $y_1 + y_2 \leq 1 + y_3$<br>$y_3 \leq y_1,\ y_3 \leq y_2$ | AND. Avoid non-linearity! |
| $y_1 \vee y_2 \implies y$ | $y_1 + y_2 \geq y_3$<br>$y_3 \geq y_1,\ y_3 \geq y_2$ | (Inclusive) OR. |
| $x_1 > 0 \implies y_1$ | $x_1 \leq M y_1$ | $M$ larger than maximum $x_1$. |
| $y_1 \implies g(x_1) \leq b$ | $g(x_1) \leq b + M(1 - y_1)$ | $M$ large enough to relax. |
| $x_1 \in \{a_1, \ldots, a_n\}$ | $x_1 = \sum_i a_i y_i$<br>$\sum_i y_i \leq 1$ | |

### A.1.2 Knapsack problem

Given a budget $b$, a collection of objects $\mathcal{J}$ with corresponding costs $a_j$ and benefits $c_j$ we wish to select an optimal subset of objects. Let the variables $x_j,\ j \in \mathcal{J}$ indicate whether $j$ is chosen. Then we formulate the ILP problem

$$\begin{aligned} \max \quad & \sum_{j \in \mathcal{J}} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in \mathcal{J}} a_j x_j \leq b \\ & x_j \in \{0,1\} \qquad j \in \mathcal{J}. \end{aligned} \tag{A.1}$$

The binary knapsack problem is $\mathcal{O}(2^{|\mathcal{J}|})$ in the worst case. To solve the LP relaxation one only needs to sort the choices by utility $c_j/a_j$ which is $\mathcal{O}(|\mathcal{J}| \log |\mathcal{J}|)$. [VIIb]

Knapsack problems also have a specific type of cuts called *minimal covers*.

**Definition A.1** (Minimal cover)**.** We call the set $S$ a *cover* if it holds that $\sum_{j \in S} a_j > b$. Moreover, if for all $k \in S$, $S \setminus \{k\}$ is not a cover, then $S$ is a *minimal cover*.

**Proposition A.1** (Knapsack minimal covers yield VIs)**.** *If $S$ is a minimal cover, then*

$$\sum_{j \in S} x_j \leq |S| - 1 \tag{A.2}$$

*is a valid inequality for the knapsack problem.*

### A.1.3 Assignment problem

Given $n$ tasks, $n$ resources, and corresponding assignment costs $c_{ij}$, we wish to pair tasks and resources with each other such that we minimize the total cost. Let $x_{ij}$ indicate whether task $i$ is assigned to resource $j$.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, \ldots, n \\ & x_{ij} \geq 0 \qquad i, j = 1, \ldots, n. \end{aligned} \tag{A.3}$$

Note that binary constraints are not required, since this ILP has integral extreme points due to its formulation as a network flow problem. It can efficiently be solved using specialized LP techniques, or using even more efficient specialized algorithms (primal-dual-graph-based) in $\mathcal{O}(n^4)$. [VIIb]

### A.1.4 Set covering problem

Given $n$ items, their costs $\boldsymbol{c}$, and $m$ subsets of the items (with a matrix $A \in \{0,1\}^{m \times n}$ indicating membership of the subsets), select items such that each subset contains at least one selected item while minimizing total cost. Let $\boldsymbol{x}$ be a vector of indicators whether items are selected.

$$\begin{aligned} \min \quad & \boldsymbol{c}^\mathsf{T} \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \geq \mathbf{1} \\ & \boldsymbol{x} \text{ binary.} \end{aligned} \tag{A.4}$$

One can also consider the related problems of set partitioning $A\boldsymbol{x} = \mathbf{1}$ and set packing $A\boldsymbol{x} \leq \mathbf{1}$. [VIIb]

## A.2 Travelling salesman — TSP

Given a set $V$ of connected cities and distances $d_{ij}$ between them (which may be infinite, i.e. no link, and directed such that $d_{ij} \neq d_{ji}$) find the shortest tour passing through all cities. A very famous problem with many variants (Euclidean, symmetric etc.). Larger problems computationally intractable due to combinatorial explosion. [VIIIa] [13.10]

### A.2.1 ILP formulation

Let $X_{ij}$ indicate whether the tour passes through the link between cities $i$ and $j$.

$$\min \quad \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{A.5}$$

$$\text{s.t.} \quad \sum_{j \in V} x_{ij} = 1 \qquad i \in V \tag{A.6}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{A.7}$$

$$\sum_{i \in U,\, j \in V \setminus U} x_{ij} \geq 1 \qquad \forall U \subset V : 2 \leq |U| \leq |V| - 2 \tag{A.8}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V. \tag{A.9}$$

Compare the assignment problem; the formulation is similar, except we add the *subtour eliminiation* constraints (A.8). These are the complicating constraints behind the combinatorial explosion, as a very large number of such subsets $U$ exist. [VIIIa]

Alternatively, one may formulate (A.8) as

$$\sum_{(i,j) \in U} x_{ij} \leq |U| - 1 \qquad \forall U \subset V : 2 \leq |U| \leq |V| - 2 \tag{A.10}$$

### A.2.2 Heuristics

TSP, being a structured problem, has several specialized heuristics. For constructive solutions a greedy algorithm using either nearest neighbour, cheapest insertion, or farthest insertion is commonly used. For local search $k$-interchange (usually $k = 2, 3$) can be used, which swaps around $k$ links in the tour such that the tour length locally improves.

### A.2.3 Specialized Branch-and-bound

Relaxing integrality in the TSP does not in itself yield a tractable problem. If the subtour elimination constraints are relaxed the problem decomposes into an assignment problem with the integrality property that is easily solved. However, the relaxed solution typically contains subtours. Instead of branching on lots of fractional variables we can branch on these subtours, such that each node in the tree becomes an assignment problem to solve.

### A.2.4 MST approximation

Consider a TSP on an undirected graph $G = (N, E, \boldsymbol{c})$. Assume $G$ complete and that the triangle inequality $c_{ij} \leq c_{ik} + c_{kj}$ holds $\forall i, j, k \in N$ (e.g. considering Euclidean distances as weights). Then one can find a MST $T \subset G$, create a multigraph $G'$ using two copies of $T$, and find an Eulerian walk of $G'$ containing an embedded TSP tour. This algorithm guarantees the the approximate tour is no longer than double the optimal tour length. [X]

## A.3 Graph problems

### A.3.1 Minimum spanning tree — MST

Given an undirected graph $G = (N, E, \boldsymbol{d})$ find a subset of the edges that connects all nodes at minimum total distance. A spanning tree necessarily consists of $|N| - 1$ edges and no cycles. The MST problem is a simple 'matroid' problem that is solved by greedy algorithms.

**Algorithm A.1** (Kruskal's algorithm)**.** Suitable for sparse graphs, $\mathcal{O}(|E| \log |N|)$.

1. Sort edges by increasing distance.
2. Choose edges from the beginning of the list; skip any that would result in a cycle.
3. Stop when all nodes are connected.

**Algorithm A.2** (Prim's algorithm)**.** Suitable for dense graphs, $\mathcal{O}(|N|^2)$.

1. Start at an arbitrary node.
2. Among the nodes that are not yet connected, choose the one that can be connected at minimum cost.
3. Stop when all nodes are connected.

### A.3.2 Shortest path

Recall the shortest path problem: given a directed graph $G = (N, A, \boldsymbol{d})$, find the shortest path from source $s \in N$ to target $t \in N$ w.r.t. edge weights $\boldsymbol{d}$. The different formulations of the problem in 3.1 give rise to specialized solution algorithms.

**Definition A.2** (Negative cycle)**.** A *negative cycle* $\{i_1, \ldots, i_k\} \subseteq N$ is a path such that

$$\sum_{\ell=1}^{k-1} d_{i_\ell i_{\ell+1}} + d_{i_k i_1} < 0. \tag{A.11}$$

Intuitively, any graph with a negative cycle will have an unbounded shortest path.

*Remark.* In a graph with no negative cycles, optimal paths will have optimal subpaths, i.e. a shortest path from $s$ to $t$ passing through $k$ contains a shortest path from $s$ to $k$.

**Algorithm A.3** (Bellman's equations)**.** Suppose that the graph is acyclic. In that case one may solve *Bellman's equations* in topological order (i.e. downwards with the flow):

$$\begin{cases} y_s = 0 \\ y_j = \min_{i \in N}\{y_i + d_{ij} : (i,j) \in A; \infty\} \quad \forall j \in N \setminus \{s\} \end{cases} \tag{A.12}$$

The result is a tree of shortest paths from $s$ to all other nodes. [XI] [8.4.1]

**Algorithm A.4** (Dijkstra's algorithm)**.** Suppose $\boldsymbol{d} \geq 0$, letting $d_{ij} := \infty$ if $(i,j) \notin A$.

0. Let $S := \{s\}$, $\overline{S} := N \setminus S$, and $\forall i \in N : y_i := d_{si}$, $\text{pred}(i) := s$ if $d_{si} < \infty$.
1. If $\overline{S} = \varnothing$ stop; else, find $j \in \overline{S}$ such that $y_j = \min_{i \in \overline{S}}\{y_i\}$. Set $S := S \cup \{j\}$ and $\overline{S} := \overline{S} \setminus \{j\}$.
2. For all $k \in \overline{S}$ and $i \in S$: If $y_k > y_i + d_{ik}$ set $y_k := y_i + d_{ik}$ and $\text{pred}(k) := i$. Go to step 2.

The predecessor links in pred describe the shortest path from $s$ to all other nodes.

Dijkstra's algorithm can be extended to support negative arc lengths by moving nodes back from $S$ to $\overline{S}$ systematically (*Ford's algorithm*), with detection of negative cycles.

The *Floyd-Warshall* algorithm finds the shortest path between each pair of nodes. It is based on the idea of inserting shortcuts into pairs of nodes: $i \to k \to j$ is a shortcut for $i \to j$ if $d_{ik} + d_{kj} < d_{ij}$. In each iteration one improves the distances by shortcuts, storing matrices $D[k]$ for lengths and pred$[k]$ for predecessors. [8.4.2]

There exist non-linear variants of the shortest path problem, where weights combine in products (e.g. probabilities) or with other functions. Such problems cannot be formulated as a LP (at least directly). One might consider the *most reliable path* (with probability $p_{ij}$ on arc $(i,j) \in A$)

$$\begin{cases} y_s = 1 \\ y_j = \max_{i \in N}\{y_i \cdot p_{ij} : (i,j) \in A; 0\} \quad \forall j \in N \setminus \{s\} \end{cases} \tag{A.13}$$

(for which the trick is to instead consider the negative logarithm of the probabilities as shortest path weights) or the *highest capacity path* (with arc capacity $K_{ij}$ on arc $(i,j) \in A$)

$$\begin{cases} y_s = \infty \\ y_j = \max_{i \in N}\{\min\{y_i, K_{ij}\} : (i,j) \in A; 0\} \quad \forall j \in N \setminus \{s\} \end{cases} \tag{A.14}$$

which does possess a direct LP formulation.

# List of Theorems and Algorithms