Discrete Optimisation: Exercises Book

Thibaut Cuvelier

2016-2017

Version history

Version number	Date	Modifications
0.1	19 September 2016	Julia tutorial (Chapter 1) and basic modelling exercises (Section 2.1)
0.2	26 September 2016	Advanced modelling exercises (Sections 2.2, 2.3, and 2.4). Added exercise 2.1.6
0.3	4 October 2016	Branch-and-bound exercises (Chapter 3). Added exercise 2.1.7
0.3.1	7 October 2016	Typo in exercise 2.3.2
0.3.2	10 October 2016	Added exercises 2.3.4, 3.2.1, and 3.2.2
0.4	18 October 2016	Formulation comparison (Chapter 4). Structured the book in three parts. Added exercise 2.5.1
0.5	3 November 2016	Constraint programming exercises (Chapter 8). Added exercise 2.4.4
0.6	16 November 2016	Cutting planes and valid inequalities (Chapter 5). Added exercise 2.4.5
0.7	2 December 2016	Flow problems (Chapter 9)
0.8	16 December 2016	Dynamic programming (Chapter 7). Correct image for exercise 9.2.1. Added Section 9.4
0.9	21 December 2016	Added exercises 2.4.6, 2.4.7, 2.4.8, 3.1.4, 5.3.3, 5.3.4, 5.3.5, 5.3.6, 7.2.2, 9.4.2 from past exams; cross-linking between version history, exercises per session, and the exercises themselves
0.9.1	26 December 2016	Added exercise 2.5.2; more cross-linking
0.9.2	3 January 2017	Typo in exercise 9.4.1; restructured Chapters 3 and 7 (the list of exercises per session has the updated numbers); added exercises 9.1.3 and 9.1.4
0.10	24 April 2017	Lagrangian duality (Chapter 6); matching and assignment (Chapter 10); added exercises 2.4.9, 4.1.4, 9.3.2, and 3.3
0.11	28 June 2017	Description of graph notations (Section 2.5). Chapter 1 is updated for Julia 0.6. Solutions for many exercises in Parts I and II. Added exercises 7.1.2, 7.1.3, 7.1.4, 7.1.6

Typographical conventions

Depending on the size of the statement, exercises are either indicated with the **Exercise** prefix, or with a complete section. The **Variants** subsections indicate a series of supplementary exercises that share the same base as the main exercise.

Within a category, the most difficult exercises are indicated with a star *.

Exercises per session (2016-2017)

Exercise session	Date	Exercises
1	23 September 2016	Chapter 1 (all)
2	30 September 2016	2.1.1, 2.1.2, 2.1.3, 2.1.4, 2.1.5
3	7 October 2016	2.2.1, 2.2.2 (§1), 2.2.3 (§1), 2.2.4, 2.2.5; 2.3.1, 2.3.2, 2.3.3; 2.4.1, 2.4.3 (§1, §3)
4	14 October 2016	3.1.1 (§2), 3.1.2, 3.1.3
5	21 October 2016	4.1.1, 4.1.2, 4.2.1
6	28 October 2016	5.4.1
7	4 November 2016	8.0.1, 8.0.2, 8.0.3
8	18 November 2016	5.1.1(§1, §2, §3, §5), 5.1.2, 5.2.1 (§1, §4)
9	25 November 2016	5.2.2, 5.4.2
10	2 December	9.1.1, 9.1.2
11	16 December	9.2.1, 9.3.1, 9.4.1, 7.2.1

Chapter 1

Julia

Julia is a programming language that has the same principles as MATLAB: it is built for scientific purposes, i.e. writing mathematical operations should be easy. Its design goals also include dynamic, “modern” features — such a dynamic typing system (albeit it is possible to impose static types), closures, and macros — and extensibility — mathematical modelling is supported through an integrated domain-specific language.

Julia is often compared to MATLAB: their syntax is similar. However, Julia is much faster, even though it does not need compilation. Also, the interpreter is completely free and open-source (it is MIT-licensed).

Its main disadvantage is that the language and its interpreter are not mature yet: they still have to reach version 1.0, with a well-defined and stable syntax. It was created in 2012, and the vast majority of the syntax is there to last, but there may still be some changes between minor versions. Also, Julia core developers tend to keep the base functionalities (i.e. without external packages) to a minimum: for example, there is no built-in plotting facility, the user must install some packages on the side; as a consequence, finding the documentation can be harder than with MATLAB.

Useful links:

- Julia website
- Julia documentation
- Plots.jl website and documentation
- JuliaOpt website
- JuMP documentation
- a tutorial (required reading)

1.1 Basic syntax and linear algebra

1.1.1 Matrix creation and reading

Exercise 1.1.1. Create a variable `a` containing the following matrix:

$$\mathbf{A} = \begin{pmatrix} 3 & 6 & 8 \\ 7 & 9 & 2 \\ -1 & 0 & 4 \end{pmatrix}.$$

Solution. `a = [[3 6 8]; [7 9 2]; [-1 0 4]]`

Exercise 1.1.2. Retrieve the first element of the matrix, i.e. A_{11}.

Solution. `a[1, 1]`

Exercise 1.1.3.

1. Select the first row of `a` (index `1`), i.e. the vector $\begin{pmatrix} 3 & 6 & 8 \end{pmatrix}$, and assign it to the variable `b`.
2. Select the last column of `a` (index `end`), i.e. the vector $\begin{pmatrix} 8 & 2 & 4 \end{pmatrix}$, and assign it to the variable `c`.
3. Are both variables vectors or matrices?
4. How can you convert the matrix `a` as a vector with nine components?

Solution.

1. `b = a[1, :]`
2. `c = a[:, end]` or `a[:, 3]`
3. Vectors.
4. Use the function `vec`.

Exercise 1.1.4. Select second element of both the first and the last row of `a`, i.e. the vector $\begin{pmatrix} 6 & 0 \end{pmatrix}$.

Solution. `a[[1, end], 2]` or `a[[1, 3], 2]`

Exercise 1.1.5.

1. Create a vector of integers starting at 6 and ending at 52.
2. Do the same with a step of 12.

Solution. Use the function `collect`.

Exercise 1.1.6. Create a vector containing three ones and another one containing three zeroes.

Solution. Use the functions `ones` and `zeros`.

Exercise 1.1.7. Create a 3×3 matrix of ones, a 3×3 matrix of zeroes, a 3×3 identity matrix.

Solution. Use the functions `ones`, `zeros`, and `eye`.

Exercise 1.1.8. Create a vector of 50 equally-spaced values between 6 and 52.

Solution. Use the function `linspace`.

1.1.2 Matrix operations

Exercise 1.1.9. Compute the dot product between `b` and `c`.

Solution. Use the function `dot` to get 68.

Exercise 1.1.10. Compute the sum of `b` and `c`.

Solution. `[11, 8, 12]`

Exercise 1.1.11.

1. Compute the product of `a` with itself.

2. Compute the square of each element in `a`.

Solution. Use the matrix operators `*` and `^`, and also their element-wise versions `.*` and `.^`.

Exercise 1.1.12. Compute the transpose of `a` with the `'` postfix operator.

Solution. Use the `'` postfix operator.

Exercise 1.1.13.

1. Compute the size of `a` as a tuple.

2. Compute the length of `a` in each dimension with two function calls (and no tuple indexing).

Solution. Use the function `size`.

Exercise 1.1.14.

1. Compute a matrix of the same size as `a` that indicates whether the corresponding element in `a` is negative or zero.

2. Use this to find the position of the negative values of `a`.

Solution. Use the broadcast comparison operator `.<=` and then the function `find`.

1.2 Flow control

Exercise 1.2.1. `n` and `x` being given, compute an approximation of the exponential function:

$$\exp x = \sum_{i=0}^{n} \frac{x^i}{i!}.$$

The exponential function is already available in Julia under the name `factorial`: for example, `factorial(2)` gives the value of 2!.

Solution. Compute the sum within a `for` loop. You should use an accumulator and the operator `+=`.

Exercise 1.2.2. Write *a function* that computes the same approximation of the exponential function, `n` and `x` being arguments to this function; do not use the built-in `factorial` function, but write your own using recursion.

Solution. A function is defined with the keyword `function`. `if` denotes a condition.

Exercise 1.2.3. Write *a function* that computes the same approximation of the exponential function, `n` and `x` being arguments to this function; do not use the built-in `factorial` function, but write your own using recursion *without* the `function` keyword (you can use the ternary operator to replace the `if` condition).

1.3 Plots with Plots.jl

Julia has no built-in plotting facility. Before making plots, you must thus install Plots.jl and a rendering engine (such as GR). To this end, type the following commands:

```
Pkg.add("Plots")
Pkg.add("GR")
```

Then, before attempting to plot, import the Plots.jl package with `using Plots`.

Exercise 1.3.1. Plot the following functions separately, using the shortest syntax possible, between 0 and 2 with an appropriate step:

$$f_1(x) = \sin x, \qquad f_2(x) = \sin x^2.$$

Solution. Use the function `plot`.

Exercise 1.3.2. Compare the two previous functions on the same graph.

Solution. Use the function `plot!` for the second plot instead of `plot`.

1.4 Mathematical optimisation models with JuMP

Just like plotting, mathematical optimisation is not included in Julia. You must thus install a modelling layer (JuMP) and a solver (like Cbc). To this end, type the following commands:

```
Pkg.add("JuMP")
Pkg.add("Cbc")
```

Then, before attempting to plot, import the JuMP package with a solver with `using JuMP; using Cbc`.

Exercise 1.4.1.

1. Implement the following model with JuMP:

$$\begin{array}{rc} \min & x + y \\ \text{s.t.} & x \geq 8.2\,y, \\ & y \geq 2 \\ & x \in \mathbb{R}, \\ & y \in \mathbb{R}^+. \end{array}$$

2. Solve it.

3. What is the optimal objective function?

4. What is the value of x?

5. What is the value of $x + 4\,y$?

Solution.

1. Create a model with `Model`. Create a variable with `@variable`. Create a constraint with `@constraint`. Set the objective with `@objective`.
   You must provide an instance of the solver `CbcSolver()` with the `solver` keyword argument when calling `Model`.

2. Use the function `solve`.

3. Use the function `getobjectivevalue`.

4. Use the function `getvalue` with a variable.

5. Use the function `getvalue` with a more complex expression.

Exercise 1.4.2.

1. Implement the following model with JuMP ($\mathbb{N}$ denotes the set of natural numbers, i.e. nonnegative integers; $\mathbb{Z}$ denotes the set of all integers, be they positive, negative, or zero):

$$\begin{array}{ll} \min & x + y \\ \text{s.t.} & x \geq 8.2\, y, \\ & y \geq 2 \\ & x \in \mathbb{Z}, \\ & y \in \mathbb{N}. \end{array}$$

2. Solve it.

3. What is the optimal objective function? Why is it different from the one of the previous exercise?

4. What is the value of x? Why is it different from the one of the previous exercise?

5. What is the value of $x + 4\,y$? Why is it different from the one of the previous exercise?

Exercise 1.4.3.

1. Implement the following model with JuMP ($\mathbb{B}$ denotes the set $\{0, 1\}$):

$$\begin{array}{lll} \min & \sum_{t\in\mathcal{T}} (c_t\, y_t + d_t\, x_t) & \\ \text{s.t.} & x_t \leq 50\, y_t, & \forall t \in \mathcal{T}, \\ & z_1 = x_1 - e_1, & \\ & z_t = z_{t-1} + x_t - e_t, & \forall t \in \mathcal{T} \setminus \{1\}, \\ & \mathbf{x} \in \mathbb{R}_+^{|\mathcal{T}|}, & \\ & \mathbf{y} \in \mathbb{B}^{|\mathcal{T}|}, & \\ & \mathbf{z} \in \mathbb{R}_+^{|\mathcal{T}|}. & \end{array}$$

   The set $\mathcal{T}$ is defined as $\{1, 2, 3, 4\}$. The vectors $\mathbf{c}$ and $\mathbf{d}$ contains various costs: $\mathbf{c} = \begin{pmatrix} 15.2 & 14.8 & 15.9 & 14.3 \end{pmatrix}$ and $\mathbf{d} = \begin{pmatrix} 0.2 & 0.2 & 0.1 & 0.6 \end{pmatrix}$. The vector $\mathbf{e}$ contains other constants: $\begin{pmatrix} 5 & 40 & 20 & 50 \end{pmatrix}$

2. Solve it.

Part I

Mixed-integer programming

Chapter 2

Modelling

Using optimisation in practice first needs a suitable representation of the problem, a *mathematical model*. Once this translation of the situation at hand is performed, efficient numerical algorithms can be used to find the optimum decisions to take in a particular situation—how to plan the production in a plant, how to manufacture the drilling part of a printed circuit, what is the best way to combine assets in a portfolio, etc.
From a software engineering point of view, this way of solving problems is quite appealing: complex algorithms are written once by specialised persons (many optimisation solvers exist on the market), and most users just *describe* their problem with some specific formalism that is amenable to those solvers.

An optimisation model first defines an *objective function*, a quantitative criterion to tell whether a given solution is good or bad with respect to the problem at hand (for example, minimise the total cost). To improve this objective function, the solver will be able to change the value of the *variables* (e.g., when to produce what quantity of which item). However, the possible values for these variables are limited by *constraints*, often representing some physical limits (the total budget for a portfolio, the order of the holes to drill on a PCB, etc.).

Especially for discrete optimisation, not all optimisation variables are actual degrees of freedom: some of them do not correspond to actual decisions that can be taken, but derive from some other variables (for example, a variable indicating whether the temperature in an oven has reached a certain threshold). As a consequence, variables may be divided in two groups:

1. The actual *decision* variables, which should only be constrained by the physics of the problem
2. *Dependent* variables, whose value must be fixed as soon as those of the decision variables are

Remark. In this course, due to the nature of the targeted solvers, all objective functions and constraints must be expressed *in a linear way*, but the domain of some variables can be constrained to be integer: the problems must be *mixed-integer linear*. In other words, the canonical form for the objective functions is $\mathbf{c}^\mathbf{T}\,\mathbf{x}$, where $\mathbf{c}$ is a vector of constants and $\mathbf{x}$ the vector of variables; that of constraints is $\mathbf{a}^\mathbf{T}\,\mathbf{x} = \mathbf{b}$, where $\mathbf{a}$ and $\mathbf{b}$ are vectors of constants.

Remark. There is **at most** one objective function (like minimising the costs); in rare cases, there is no objective function, because there is no way to indicate when a solution is better than another.

Remark. For each exercise, you may consider to implement in with a modelling environment (JuMP, Convex.jl, Pyomo, ZIMPL, CVX, etc.).

2.1 Warm-up

2.1.1 Hospital placement (discrete facility location)

A country wants to reconsider its health system and seek for better ways of using the scarcer and scarcer resources available to the State. To this end, the government asks a consulting company where should the hospital be located based on the actual demand, if they could rebuild the complete system from scratch.

The demand is represented as a series of discrete points having some demand (for example, one point for each city, be it large or small). The hospital can only be built at some points (usually corresponding to the large cities). Meeting the demand has some cost that depends on the city where the hospital is located (the largest part coming from ambulances, which must travel a long way before rescuing people; casualties due to the emergency services arriving too late are already considered in these costs); these costs are given in a matrix (one dimension corresponds to the demand points, the other to the potential hospital). Opening a hospital can only be done at a high price.

More precisely, hospitals can be located at three places, with the costs $\begin{pmatrix}500 & 800 & 1800\end{pmatrix}$. The demand is modelled as six points: $\begin{pmatrix}24 & 12 & 8 & 96 & 1 & 48\end{pmatrix}$. The cost matrix of serving the demand from the three possible hospitals is:

$$\begin{pmatrix} 20 & 600 & 80 & 480 & 1200 & 410 \\ 435 & 950 & 890 & 420 & 700 & 20 \\ 450 & 1400 & 1400 & 20 & 800 & 420 \end{pmatrix}.$$

The government goal is to minimise their total costs (i.e. building and operating the hospitals). Model the situation as a mixed-integer linear program.

2.1.1.1 Variants

1. Add a maximum capacity to each hospital (instead of having an infinite capacity: a hospital can only fulfil some maximum demand).

2. Allow to build various sizes of hospitals (small hospital, small cost, small demand, up to large hospital).

Day	0-2	2-4	4-6	6-8	8-10	10-12
Minimum nurses	15	14	14	17	22	20
Day	12-14	14-16	16-18	18-20	20-22	22-24
Minimum nurses	10	20	18	15	20	20

Table 2.1: Nurses requirements for the hospital.

3. Consider a set of hospitals already exists and already meets some demand: their building cost is then zero, except when they are upgraded (a small hospital can be extended to become a medium-sized one). What should you modify to account for this situation?

2.1.2 Nurse scheduling

A working day in a hospital is subdivided in twelve periods of two hours. The staff requirements change from period to period. A nurse works eight hours a day and is entitled to a break after four hours.

1. Determine the minimum number of nurses required to cover all requirements using a mixed-integer linear program.

2. If only 55 nurses are available (which it is not enough for the given data), the management allows for a certain number of nurses to work for a fifth period right after the last one. Determine a schedule the minimum number of nurses working overtime.

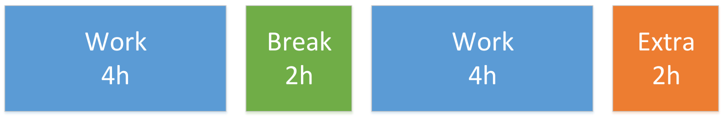

Figure 2.1: A nurse working day.

2.1.3 Team making

40 people participate in a game. For this purpose they must be divided in five groups of eight people. For each pair (i, j), a matrix has a value

- 0 if they don't know each other.
- 1 if they know each other a bit.
- 2 if they know each other well.

The goal is to create groups so that participants know as few other people as possible in each group. Choose a plausible objective function and formulate this problem as a mixed-integer linear program.

2.1.4 Steel mill planning (lot sizing)

A steel mill manufactures I-beams; the director wants to plan the production for the next twelve weeks. The demand for I-beams is exactly known for each week. The cost for using the furnace can be divided into two parts: a fixed cost to get the furnace turned on and a variable cost proportional to the number of beams produced. Storing beams from a time period to another is costly, and is proportional to the number of beams to store.

Week	1	2	3	4	5	6	7	8	9	10	11	12
Demand [10 beams]	7	5	3	5	5	9	1	8	5	6	2	2
Variable cost [1000€ per 10 beams]	2	2	2	2	2	2	2	2	2	2	2	2
Fixed cost [1000€]	16	16	16	16	16	16	16	16	16	16	16	16
Storage cost [1000€ per 10 beams]	1	1	1	1	1	1	1	1	1	1	1	1

Table 2.2: I-beams factory planning requirements.

1. The mill director wants to derive a production plan that minimises the total cost. Formulate the problem as a mixed-integer linear program.
2. Reformulate the planning problem when adding the following constraint: the blast furnace must stay active at least three and at most four consecutive periods once started.

2.1.5 Ground meat preparation

In the food-processing industry, ground-meat preparation is a basic step for many recipes (from the most basic ones, like sausages or hamburger patties, to more complex ones, such as keema or lasagne). Its quality is dictated by the quantity of fat.

To prepare this ground meat, industrial butchers prepare trays with pieces of meat, after the slaughterhouse and the retrieval of the best pieces of meat. These trays have some quantity of meat, of which a certain proportion of fat.

For the rest of the food-processing supply chain, some quantity of ground meat must be produced by assembling those trays (it is not possible to split a tray, e.g. in two and just use half of the total tray). As all trays do not have predictable weight or fat content, the butchers have a tolerance of 5% with the weight (i.e. they may assemble trays whose total weight is at most 5% less or more than what required). Their goal is to be as close as possible to the requested quantity of fat for one recipe at a time by selecting a subset of the available trays (there may be more or less fat than requested).

Formulate this problem as a mixed-integer linear program.

2.1.5.1 Variants

Consider the butchers must prepare multiple ground meat recipes (each of them having its own total weight and fat content target). Their goal is now to minimise the total fat content differences among all prepared recipes. Of course, a tray can only be selected for one recipe.

2.1.6 Unit commitment (exam September 2016)

In the unit-commitment problem, a company needs to schedule its electricity production for a given number of periods T. It has P plants of maximum power $P_i^{\max}$. It must satisfy a demand d_t at each time period t and pays a fixed cost $f_{t,i}$ for using a plant i and a varying cost $p_{t,i}$ depending on the amount of the electricity produced.

Furthermore, we want to introduce start-up costs $g_{t,i}$, that means that we pay each time that a plant i starts when it was down the period before.

We also want to impose that, each time a plant starts, it needs to run for at least four consecutive periods.

Formulate the problem of minimizing the costs of the producer while satisfying the demand as a mixed-integer program.

2.1.7 Knapsack

A gang of thieves is stealing objects in a house. They may take the items there, but they have a limited bag to bring those back to their hideout. Each item has a weight and a value; the bag's capacity only considers the weight of the objects that are put inside. The goal of the thieves is to steal the maximum value from the house.

Model this problem as a mixed integer linear program.

2.2 Boolean algebra

2.2.1 Negation (NOT)

1. How to model a Boolean NOT, i.e. $\neg a$ where a is a binary variable?

2.2.2 Disjunction (OR)

1. How to model a binary Boolean OR, i.e. the relationship $a \vee b$ between two binary variables a and b?

2. How to model a ternary Boolean OR, i.e. the relationship $a \vee b \vee c$-between three binary variables a, b, and c?

3. * How to model compactly a n-ary Boolean OR (i.e. use as few supplementary variables and constraints as possible)?

2.2.3 Conjunction (AND)

1. How to model a binary Boolean AND, i.e. the relationship $a \wedge b$ between two binary variables a and b?
2. How to model a ternary Boolean AND, i.e. the relationship $a \wedge b \wedge c$- between three binary variables a, b, and c?
3. * How to model a n-ary Boolean AND?
4. ** How to make it more compact (i.e. use as few supplementary variables and constraints as possible)?

2.2.4 Implication

How to model the fact than an implication $a \Rightarrow b$ must be true, where a and b are binary variables?

2.2.5 If-then

How to model the following expression, where $x \in [0, 100]$ and $y \in \mathcal{B}$?

$$\text{if } y = 1, \text{ then } x = 100$$

2.3 Threshold

2.3.1 Binary activation

How to model the constraint that the variable $x \in \mathbb{R}^+$ can take a nonzero variable if and only if the variable $y \in \mathcal{B}$ is 1?

2.3.2 Semicontinuous variable

How to model the constraint that the variable $x \in \mathbb{R}^+$ can take a value in the set $\{0\} \cup [m, M]$, where $0 < m < M$?

2.3.3 Threshold detection

How to model the following expression, where $x \in \mathcal{B}$ and $y \in [0, 100]$?

$$x = \begin{cases} 1 & \text{if } y \geq 50 \\ 0 & \text{otherwise} \end{cases}$$

How does your solution behave for $y = 50$?

2.3.4 Piecewiselinear function

A model needs to use a nonlinear function in a constraint:

$$f(x) = x^3 + \log_2 x.$$

A piecewise-linear approximation with sixteen points, equally distributed between 2 and 9.5, would be sufficient in this application. In other words, the previous equation would be replaced by:

$$f(x) = \begin{cases} 15.894\,x - 22.788 & \text{if } 2 \le x \le 2.5 \\ 23.276\,x - 41.243 & \text{if } 2.5 \le x \le 3 \\ & \vdots \\ 256.906\,x - 1579.98 & \text{if } 9 \le x \le 9.5 \end{cases}.$$

1. Propose a MILP formulation for the constraint that forces y to take an approximation of $f(x)$, with $x \in [2, 9.5]$ and $y \in \mathbb{R}^+$. Use the set of points in Table 2.3 to this end.
   Hint: for sixteen points to interpolate, use sixteen continuous variables and sixteen binary variables.

2. ** Propose an equivalent MILP formulation that only uses a fraction of the binary variables. It is possible to find one that uses $\lceil \log_2 N \rceil$ binary variables for N points to interpolate.

x	$f(x)$
2	9
2.5	16.947
3	28.585
3.5	44.682
4	66
4.5	93.295
5	127.322
5.5	168.834
6	218.585
6.5	277.325
7	345.807
7.5	424.782
8	515
8.5	617.212
9	732.170
9.5	860.623

Table 2.3: Points for a linear approximation of the function $f(x) = x^3 + \log_2 x$ between 2 and 10.

2.4 More modelling

Remark. The exercises in this section are harder than those of Section 2.1, and often require techniques from Sections 2.2 and 2.3.

2.4.1 Volleyball teams

Make teams for a volleyball tournament. Each team has exactly six members.

Then, consider the gender of the players: impose that each time has at least two men and two women.

For the next step, add the notion of level for each player; they are noted on a scale of one to four. Each team must have at least one person for three different levels.

Finally, define the objective: teams must be as similar to each other as possible. More precisely, the total "variance" of the level and presence of each gender must be as low as possible (the objective must remain linear).

2.4.2 * Greatest common divisor

Write a mixed-integer linear optimisation program that computes the greatest common divisor (GCD) of a and b, two non-zero natural numbers.

2.4.3 Attic problem

1. Your aunt passes away, you and your sister are her only heirs. How to solve inheritance problems? Minimise the unfairness when splitting the heritage (consisting of a list of objects to divide amongst the two of you), computed with various criteria: monetary value, volume, sentimental value. Each of you shall get approximately the same amount for each of these criteria out of the heritage of your late aunt.

2. * However, your notary is in a hurry, and would want the actual distribution rather soon. Why is your model so slow to solve with many objects? Find an explanation that your notary could understand.
   Hint: what is the value for the first LP relaxation?
   To solve this question, you will need the material of Chapter 3.

3. Later on, you hear about a mysterious hidden child. With the help of your genealogist friend, you can find her. Decide again how to split the heritage among the three of you, even though your notary is far from happy with this situation.

4. To help him solve For the next step, add the notion of level for each player; they are noted on a scale of one to four. Each team must have at least one person for three different levels.

Finally, define the objective: teams must be as similar to each other as possible. More precisely, the total "variance" of the level and presence of each gender must be as low as possible (the objective must remain linear).

1. such problems quickly, your notary would like an automated tool to solve such problems quickly. Impressed with your pedagogical skills and explanations, he proposes you a consulting job to implement an enhanced

version of your model that solves these deficiencies. What amount should you bill him (100€ per hour, including all taxes)?

2.4.4 Truck transportation (exam January 2014)

A company has a limited number of trucks to ship objects from their storehouse in Germany to the one in Belgium. Trucks have all the same cross section but differ in length. The objects have different lengths, but their width and height are standardized to the cross section of trucks. Model the following 3 **independent** problems a mixed-integer linear programs.

1. The company wants to minimize the number of trucks needed to ship the objects.
2. Each truck can carry a limited weight and the company wants to minimize the shipping cost. For each truck the shipping cost is composed of a fixed cost to pay if the truck takes the road and a variable cost proportional to the weight of the truck.
3. Truck drivers are divided in teams that stay together on the road. The company has a list of dependency between objects. If an object a is dependent of an object b, it means that a must be shipped by the same team that ships object b. The company now wants to minimize the maximum empty space among the trucks assigned to take the road.

2.4.5 Tennis championship (exam August 2013)

A tennis club wants to organise a small championship between its 20 members. As time is limited, only 6 sessions of 10 simultaneous matches can be organised. Each player will have to play against 6 different opponents.

1. If each potential pair of opponents (i, j) is associated to a value $c_{i,j}$, formulate the problem of determining the programme of the matches that maximises the sum of all the $c_{i,j}$ for the organised matches as a mixed integer linear program.
2. Each player is associated with a strength f_i. We can define S_j as the sum of all the f_i of the opponents that player i meets during their 6 matches. Formulate as a mixed integer linear program the problem of determining a matches programme such that, among all players, the difference between the largest S_j and the smallest S_j is minimum.

2.4.6 Exam schedule (exam January 2013)

You wish to write a tool that automatically determines an exam schedule. To this end, you have a list S of students and a list C of courses. Each student i follows a subset C_i of courses. Each course j is followed by a subset S_j of students. Finally, all the exams must take place in a two-week period, from Monday to Friday. You can suppose that there are enough rooms.

Type	Strength	Stamina	Price
1	1	10	3
2	3	7	3
3	7	7	6
4	8	7	7
5	10	7	9

Table 2.4: Data for exercise 2.4.7.

1. Formulate the discrete problem of finding a valid schedule such that no student ever has two exams on the same day or during two consecutive days as a mixed-integer linear program.

2. Formulate the discrete optimisation problem of finding a valid schedule such that no student ever has two exams on the same day and that maximises the minimum number of days between two exams for each student as a mixed-integer linear program.

2.4.7 Schmurf game (exam August 2014)

A player wants to write an optimisation model to set up his best team in the video game *Schmurf*. In the game *Schmurf*, a player has a budget of 1000 *coins* to set up a team of as many *warriors* as they want. Each warrior has a given *strength* and a given *stamina*. There are five different types of warriors whose characteristics are given in the table below.

Formulate as a linear (mixed-)integer optimisation program the following problems:

1. Buying a team with the highest strength.

2. Buying a team with a total stamina that is as close as possible to twice the total strength (considering that it is maybe not possible to have a stamina exactly equal to twice the total strength) and a strength of at least 200.

3. A team will play against opponents lying is seven different rooms. The team must be split in seven smaller teams with a winning probability in each room given by a linear function of the number of warriors of each type. Explain how to create a split in seven teams so as to maximise the expected number of winning rooms of the whole team.

2.4.8 Storing files on flash drives (exam January 2015)

We want to store computer files on flash drives. For technical reasons, each file cannot be split and stored in pieces on separate drives. We assume that we have n_2 files of 2 GB, n_3 files of 3 GB, n_5 files of 5 GB and n_7 files of size 7 GB.

1. Model the problem of storing the $n_2 + n_3 + n_5 + n_7$ files on a minimum number of flash drives of size 16 GB each.
2. Assume flash drives of 32 GB cost 9 euros, flash drives of 16 GB cost 5 euros and flash drives of 8 GB cost 3 euros. Model the problem of minimising the cost of storing the $n_2 + n_3 + n_5 + n_7$ files.

2.4.9 Scout staffs

A chief scout needs to make the new staffs for the next year to come. He must choose which people goes in which staff. They are three different staffs: *wolf clubs*, *boy scouts*, and *rover scouts*. Obviously, everyone cannot go in each staff and everyone has referred in which staff they may go. Each staff must have a reasonable size and everyone should be assigned to a staff. The goal of the chief scout is to maximize the harmony in each staff. To this end, he has an *affinity matrix* giving a value corresponding to the degree of friendship between two persons.

We generalize the problem for N staffs and M persons. Each staff k must be leaded by minimum l_k and maximum u_k persons. For every person i and every staff k, we have a Boolean $s_{i,k}$ equal to 1 if the i can go in staff k, 0 otherwise. The degree of friendship between i and j is given by $a_{i,j}$ (the affinity matrix).

Model this problem as a mixed-integer linear program.

2.5 Graph

The following notations are commonly used when dealing with graphs:

- $G = (E, V)$ is a graph whose edges are in set E and vertices in V. An edge is a pair of vertices: $E \subseteq V^2$.
- $\delta(i)$ is the set of edges that go in or out of vertex i.
$$\delta(i) = \left\{(i,j) \;\middle|\; j \in V, (i,j) \in E\right\}.$$
The same notation is often used when $\mathcal{I}$ is a set of vertices, and represents the set of edges that start within the set $\mathcal{I}$ and end outside $\mathcal{I}$.
$$\delta(\mathcal{I}) = \left\{(i,j) \;\middle|\; i \in \mathcal{I}, (i,j) \in E, j \notin \mathcal{I}\right\}.$$
In the particular case where $S = V$,
$$\delta(S) = \emptyset.$$
- $E(S)$ is the set of edges between the nodes of set $S \subseteq V$.
$$E(S) = \left\{(i,j) \;\middle|\; (i,j) \in E, i \in S, j \notin E\right\}.$$
In the particular case where $S = V$,
$$E = E(V)\,.$$

2.5.1 Travelling salesman problem

Given a list of cities and the distances between each pair, the undirected travelling salesman problem finds the shortest path that visits each city exactly once while returning to the original city.

Hint. Two main formulations are possible: one eliminates subtours and is called subtour elimination, the other imposes connexity and is named cut-set.

2.5.2 Prize collection with fixed path length

Let $G = (E, V)$ be an undirected graph, with E being its set of edges and V its set of vertices. Each vertex has a prize p_i. Formulate as a mixed-integer linear program the problem of finding the path of length exactly L that maximises the total collected prizes, without cycle.

Chapter 3

Branch-and-bound algorithm

3.1 Algorithm behaviour

3.1.1 Geometric solving

Solve the following problems geometrically using the branch-and-bound algorithm (i.e. by drawing the feasible area, finding the extreme points by hand—as the simplex algorithm would).

1.

$$\begin{array}{rc} \max & 9\,x + 5\,y \\ \text{s.t.} & 4\,x + 9\,y \leq 35 \\ & x \leq 6 \\ & x - 3\,y \geq 1 \\ & 3\,x + 2\,y \leq 19 \\ & (x, y) \in \mathbb{Z}^2_+. \end{array}$$

2.

$$\begin{array}{rc} \max & 2\,x + 3\,y \\ \text{s.t.} & -\frac{2}{3}\,x + y \leq \frac{5}{2} \\ & \frac{1}{3}\,x + y \leq \frac{9}{2} \\ & 2\,x + y \leq 14 \\ & (x, y) \in \mathbb{Z}^2_+. \end{array}$$

Solution. For the first problem, the solution is $x = 6$ and $y = 0$. For the second problem, the solution is $x = 6$ and $y = 2$.

3.1.2 Tree construction

1. Give the tightest lower and upper bounds on the optimal value for the objective based on the tree of Figure 3.1. Compute the gap.
2. With the given partial tree, give the nodes which should be pruned (and why), and which could be explored further.

3. Explore your solver's MIP log and link it to the branch-and-bound tree.

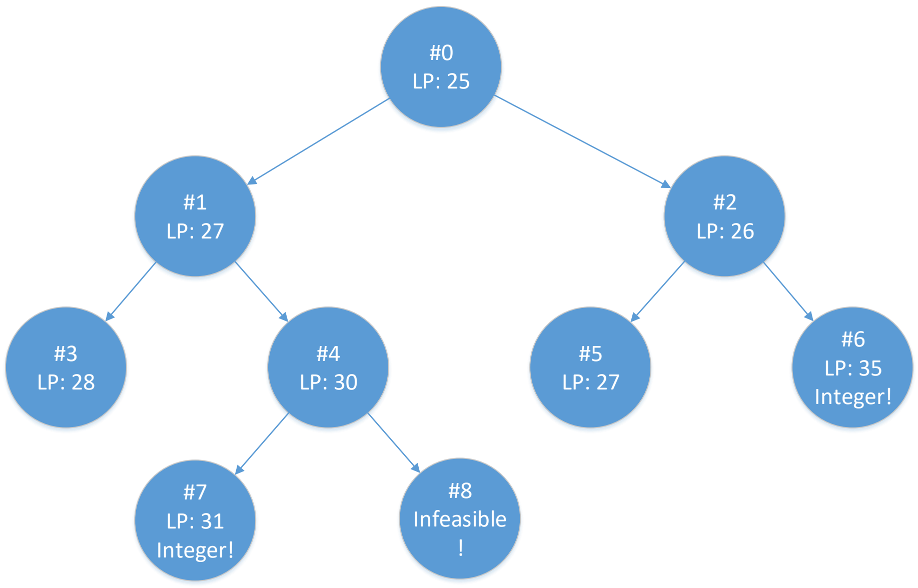

Figure 3.1: A partial branch-and-bound tree.

Solution. The upper bound is 31, the lower bound is 27, with a gap of 12.9%. You should explore further nodes #3 and #5.

3.1.3 * Worst case

Give an example where the branch-and-bound algorithm performs a large number of iterations (when compared to the number of variables) to find a first integer solution.

Solution. A worst case can be found when having two integer variables and a linear equality: the number of integer solutions along this line is very small, but branching does not really help reaching a solution quickly.

3.1.4 Tree interpretation (exam August 2014)

In the branch-and-bound tree of Figure 3.2, in each node, a number located at the top of the node indicates an upper bound, whereas a number located at the bottom indicates a lower bound.

1. Is it a minimisation or a maximisation problem?
2. Compute the tightest lower and upper bound for the problem.

3. What are the nodes that must be explored further?

4. Assume that we are interested in the two best solutions. What are the nodes that should be explored further?

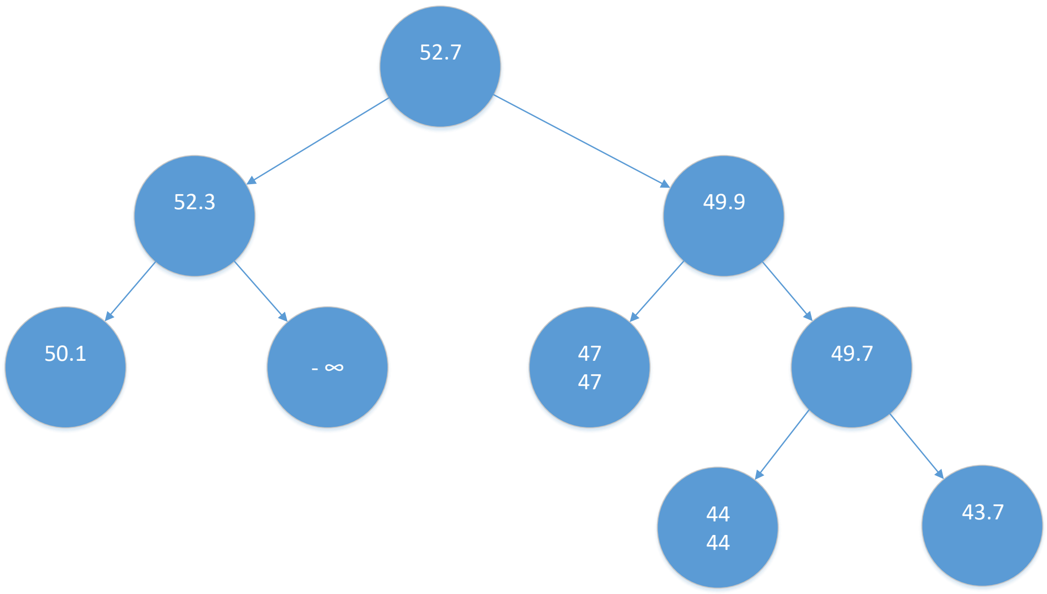

Figure 3.2: A partial branch-and-bound tree.

3.1.5 Tree interpretation (exam August 2014)

Consider the branch-and-bound tree which is depicted in Figure 3.3. For each node, the above number (if present) represents an upper bound on the optimal value of the given subproblem, while the below number represents a lower bound on the optimal value of the given subproblem.

1. Determine the sense of optimization (maximisation or minimisation).

2. Give the tightest possible lower and upper bound for the full problem.

3. Determine which nodes need to be explored further.

4. Assume that we are interested in the best two solutions (instead of just one), determine which leaves of the tree can potentially contain the second best solution.

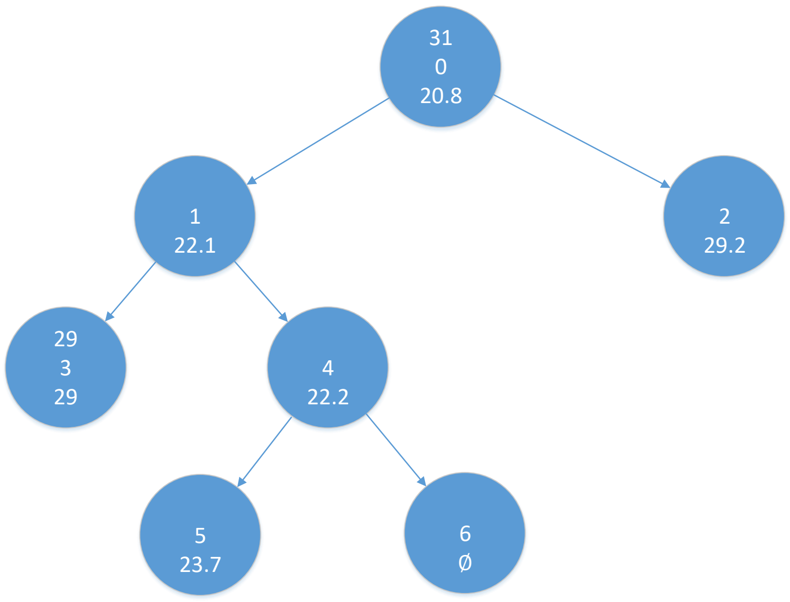

Figure 3.3: A partial branch-and-bound tree.

3.2 Implementation choices

3.2.1 Pruning (exam January 2014)

In the branch-and-bound algorithm, give the different opportunities to prune the search and explain briefly.

Solution. By bound, by optimality, by infeasibility.

3.2.2 Exploration strategies

Explain two simple node selection strategies that can be used to explore the branch-and-bound tree. In particular, detail their effect on the bounds, and on the number of nodes to find an integer solution.

Solution. Depth-first (improves primal bound) or breadth-first search (improves dual bound).

3.2.3 ** Implementation

1. Develop a MILP model for the knapsack problem (as in Exercise 2.1.7).
2. Implement an algorithm that solves to optimality instances of this problem by pure enumeration of all solutions.

3. Implement a branch-and-bound that solves to optimality instances of this problem.
4. Implement a pseudopolynomial dynamic algorithm that solves to optimality instances of this problem.
5. Compare those three approaches, in terms of worst case and average case time complexity.

Chapter 4

Formulation comparison

Due to the way the branch-and-bound algorithm works, adapting the formulation can make the solving process faster: if fewer non-integer-feasible points are present, then fewer branching are required to find the optimum solution, as the LP relaxation more often yields integer-feasible solutions. The ideal case is when all extreme points of the feasible polyhedron are integer (this is always the case for totally unimodular matrices).

A formulation P_1 is *stronger* than P_2 if the first one is included in the second one, i.e. $P_1 \subsetneq P_2$ (see Figure 4.1). Both formulations must be correct for the problem to solve (i.e. have the same set of integer-feasible points): the comparison is based on the LP relaxation of the integrality constraints. The best formulation for a problem is the convex hull (all its extreme points are integer solutions).

- To prove that $P_1 \subseteq P_2$, any point of P_1 must be a point of P_2; to this end, the main technique is to work with the set of constraints of P_1 and to show that it implies the set of constraints of P_2. However, this is not enough to prove that P_1 is a stronger formulation: both might be equal. (See Figure 4.2.)
  Simply checking that one point of P_1 is in P_2 is not enough, as this would conclude in cases where no real conclusion is possible (like in Figure 4.1).

- To prove that $P_1 \subsetneq P_2$ (having proved that $P_1 \subseteq P_2$), it is sufficient to find a point that is within P_2 but not within P_1. As the two formulations have the same set of integer-feasible points (they represent the same problem), such a point cannot be found as an integer solution, but rather a fractional one. (See Figure 4.3.)

- Proving that $P_1 = P_2$ is equivalent to proving that $P_1 \subseteq P_2$ and $P_2 \subseteq P_1$.

Remark. A *stronger* formulation is not always *better* in all possible meanings of the word "better": this formulation might require many more constraints and/or variables than the first one, which may make it harder to implement and to solve

(unless specific care is taken, which often means lazy constraints, when they are supported by the solver). Usually, an exponential number of constraints is required to describe the convex hull.
For many problems, it is possible to deal with an exponential number of constraints using either lazy constraints (the most useful constraint is selected by a separation procedure within the total set of possible constraints) or cutting planes (to bring the formulation closer to the convex hull, as in Chapter 5). Most commercial solvers allow these procedures.

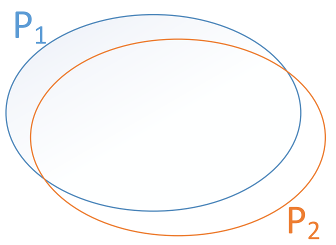

Figure 4.1: The LP-feasible sets of two formulations.

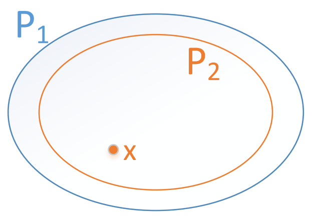

Figure 4.2: Prove that $P_1 \subseteq P_2$: any point of P_1 must be a point of P_2.

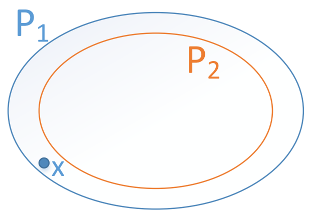

Figure 4.3: Prove that $P_1 \subsetneq P_2$: considering that $P_1 \subseteq P_2$, there is a point x in P_1 that is not in P_2.

4.1 General problems

4.1.1 Discrete facility location[1]

The uncapacitated facility location problem deals with the optimal opening of facilities (their position cannot be changed) to meet some demand (always one unit) while minimising the total cost (opening facilities, producing the goods, delivering them to the customers). There is no bound on the amount of goods a facility can produce. A client's demand is always met by only one facility.

1. Propose two formulations for the uncapacitated facility location.
   Hint: one has an aggregated constraint, not the other one.

2. Is a formulation stronger than the other, or are they equal?

3. Implement both formulations and compare the solving times when increasing the size of the problem. Is one better than the other? Does it correspond to your expectations?

Solution. First formulation:

$$\begin{array}{lcr} \min & \sum_{j\in\mathcal{F}} o_j\, x_j + \sum_{i\in\mathcal{D}i} \sum_{j\in\mathcal{F}} s_{ij}\, y_{ij} & \\ \text{s.t.} & \sum_{j\in\mathcal{F}} y_{ij} = 1, & \forall i \in \mathcal{C}, \\ & y_{ij} \le x_j, & i \in \mathcal{C}, \forall j \in \mathcal{F}. \end{array}$$

Second formulation:

$$\begin{array}{lcr} \min & \sum_{j\in\mathcal{F}} o_j\, x_j + \sum_{i\in\mathcal{D}i} \sum_{j\in\mathcal{F}} s_{ij}\, y_{ij} & \\ \text{s.t.} & \sum_{j\in\mathcal{F}} y_{ij} = 1, & \forall i \in \mathcal{C}, \\ & \sum_{i\in\mathcal{C}} y_{ij} \le |\mathcal{C}|\, x_j, & \forall j \in \mathcal{F}. \end{array}$$

The first formulation is strictly stronger than the second one.

4.1.2 Exams schedule (exam January 2016)

We want to create a schedule for the exam session. In order to do so, we formulate the problem of finding a time slot for each exam in such a way that no student has more than one exam on the same day. We define binary variables as

$$\begin{aligned} x_{it} &= 1 \text{ if exam } i \text{ takes place on day } t \\ &= 0 \text{ otherwise.} \end{aligned}$$

We denote by L the set of courses, by T the set of days and by S the set of all students. For a given student $k \in S$, we denote by $C(k)$ the index set of all courses followed by student k. To encode the constraints, we suggest two options:

[1]Also see exercise 2.1.1.

1.
$$\begin{aligned}\sum_{t\in T} x_{it} &= 1 &&\text{for all } i \in L, t \in T\\ \sum_{i\in C(k)} x_{it} &\le 1 &&\text{for all } k \in S, t \in T.\end{aligned}$$

2.
$$\begin{aligned}\sum_{t\in T} x_{it} &= 1 &&\text{for all } i \in L, t \in T\\ x_{it} + x_{jt} &\le 1 &&\text{for all } \ (i,j) \in C(k) \times C(k)\,, k \in S, t \in T, i \neq j.\end{aligned}$$

Determine between (a) and (b) which one is the best formulation and prove that it is the case.

Solution. The first formulation is strictly stronger than the second one.

4.1.3 Courses schedule (exam September 2016)

We want to create a schedule for the courses. In order to do so, we formulate the problem of finding a time slot for each course in such a way that no students and no professors have two courses at the same time slot. We define binary variables as

$$\begin{aligned}x_{it} &= 1 \text{ if course } i \text{ takes place on time slot } t\\ &= 0 \text{ otherwise.}\end{aligned}$$

We denote by L the set of courses, by T the set of time slots, by S the set of all students, and by P the set of professors. For a given student $k \in S$, we denote by $C(k)$ the index set of all courses followed by student k. Similarly for a given professor $j \in$, we denote by $D(j)$ the index set of all courses taught by professor j.To encode the constraints, we suggest two options:

1.
$$\begin{aligned}\sum_{t\in T} x_{it} &= 1 &&\text{for all } i \in L, t \in T,\\ \sum_{i\in C(k)} x_{it} &\le 1 &&\text{for all } k \in S, t \in T,\\ \sum_{i\in D(j)} x_{it} &\le 1 &&\text{for all } j \in P, t \in T.\end{aligned}$$

2.
$$\begin{aligned}\sum_{t\in T} x_{it} &= 1 &&\text{for all } i \in L, t \in T,\\ x_{it} + x_{\hat{\imath}t} &\le 1 &&\text{for all } \ (i,\hat{\imath}) \in C(k) \times C(k)\,, k \in S, t \in T, i \neq \hat{\imath},\\ x_{it} + x_{\hat{\imath}t} &\le 1 &&\text{for all } \ (i,\hat{\imath}) \in D(j) \times D(j)\,, j \in P, t \in T, i \neq \hat{\imath}.\end{aligned}$$

Determine between (a) and (b) which one is the best formulation and prove that it is the case.

Solution. The first formulation is strictly stronger than the second one.

4.1.4 Lot sizing (exam January 2017)

We consider the lot-sizing problem, i.e. we want to produce items by batch in order to satisfy a demand. There are T time periods, and, at each time period t, there is a demand d_t. For each batch produced, we pay a fixed price f_t (irrespective of the batch size) and a variable price p_t (proportional to the batch size). We neglect here the stock prices. We consider two formulations of such a problem.

1. We define continuous variables $x_t \forall t \in [1, T]$ indicating the amount produced at each time period t, binary variables $y_t \forall t \in [1, T]$ indicating whether there was any production at period t, and variables $s_t \forall t \in [1, T]$ indicating the stock going from period $t-1$ to t.

$$P_1 = \left\{ (x, y, s) \in \mathbb{R}_+^T \times \mathcal{B}^T \times \mathbb{R}_+^T \,\middle|\, \begin{array}{ll} x_t + s_{t-1} = d_t + s_t & \forall t \in [1, T-1], \\ x_T + s_{T-1} = d_T, & \\ s_0 = 0, & \\ x_t \leq M_t\, y_t & \forall t \in [1, T] \end{array} \right\},$$

   where $M_t = \sum_{i=t}^{T} d_i$.

2. We define $\mathcal{O}(T^2)$ additional continuous variables $w_{st} \forall s \in [1, T]\ \forall t \in [s, T]$ indicating the fraction of the demand of period t that is served from a production in period s. The formulation reads as follows.

$$P_2 = \left\{ (x, y, w) \in \mathbb{R}_+^T \times \mathcal{B}^T \times \mathbb{R}_+^{T(T+1)/2} \,\middle|\, \begin{array}{ll} x_t = \sum_{i=t}^{T} d_i\, w_{ti} & \forall t \in [1, T], \\ \sum_{s=1}^{t} w_{st} = 1 & \forall t \in [1, T], \\ w_{st} \leq y_s & \forall t \in [1, T] \end{array} \right\}.$$

Then:

1. Prove that the second formulation is at least as strong as the first one. **Hint**: to be able to compare both formulations, you need to consider (project onto) the (x, y) space.
2. Give some intuition on why the second formulation is *strictly* stronger than the first one.

4.2 Graph

The following notations are often used for graphs in optimisation:

- $G = (E, V)$ is a graph whose edges are in set E and vertices in V. An edge is a pair of vertices: $E \subseteq V^2$.
- $\delta(i)$ is the set of edges that go in or out of vertex i.

  $$\delta(i) = \left\{ (i, j) \,\middle|\, j \in V, (i, j) \in E \right\}.$$

  The same notation is often used when $\mathcal{I}$ is a set of vertices, and represents the set of edges that start within the set $\mathcal{I}$ and end outside $\mathcal{I}$.

  $$\delta(\mathcal{I}) = \left\{ (i, j) \,\middle|\, i \in \mathcal{I}, (i, j) \in E, j \notin \mathcal{I} \right\}.$$

- $E(S)$ is the set of edges between the nodes of set $S \subseteq V$.

  $$E(S) = \left\{ (i, j) \,\middle|\, (i, j) \in E, i \in S, j \notin E \right\}.$$

4.2.1 * Travelling salesman problem (TSP)[2]

Given a list of cities and the distances between each pair, the undirected travelling salesman problem finds the shortest path that visits each city exactly once while returning to the original city. It has applications outside operational research; for example, to manufacture microchips, in order to minimise delays, every component must be placed as closely as possible to the other ones; the components' position can also be optimised.

1. Propose two formulations for the undirected travelling salesman.
   Hint: one eliminates subtours and is called subtour elimination (P_{sub}), the other imposes connexity and is named cut-set (P_{cut}).
2. Prove that the two formulations are equivalent or that one is stronger than the other. To this end, follow these steps.
   (a) For a set of vertices $S \subsetneq V$, what kind of relationship is there between the set of outgoing edges $\delta(S)$, the set of edges within S $E(S)$, and the whole set of edges $E = E(V)$?
   (b) What can you say about $P_{\text{cut}} \subseteq P_{\text{sub}}$?
   (c) What can you say about $P_{\text{sub}} \subseteq P_{\text{cut}}$?
3. Implement both formulations and compare the solving times when increasing the size of the problem. Is one better than the other? Does it correspond to your expectations?

Solution. First formulation, P_{sub}:

$$\begin{array}{lll}
\min & \sum_e c_e \, x_e & \\
\text{s.t.} & \sum_{e \in \delta(i)} x_e = 2, & \forall i \in V, \\
& \sum_{e \in E} x_e = |V|, & \\
& \sum_{e \in E(S)} x_e \leq |S| - 1, & \forall S \subsetneq V : \, 2 \leq |S| \leq |V| - 1.
\end{array}$$

[2] Also see exercise 2.5.1.

Second formulation, P_{cut}:

$$\begin{array}{lll} \min & \sum_e c_e\, x_e & \\ \text{s.t.} & \sum_{e\in\delta(i)} x_e = 2, & \forall i \in V, \\ & \sum_{e\in E} x_e = |V|\,, & \\ & \sum_{e\in\delta(S)} x_e \geq 2, & \forall S \subsetneq V :\ 2 \leq |S| \leq |V| - 1. \end{array}$$

The following relationship holds:

$$\underbrace{\delta(S)}_{\substack{\text{edges going}\\ \text{out of } S}} = \underbrace{E(V)}_{\text{all edges}} \setminus \underbrace{E(S)}_{\text{edges in } S} \setminus \underbrace{E(V\setminus S)}_{\text{edges not in } S}$$

Chapter 5

Cuts and valid inequalities

Cuts and valid inequalities help make a formulation stronger: more corners of the feasible region will be integer-feasible solutions (at least for some variables), without needing branching. The goal is to reach the convex hull of the integer-feasible points.

In theory, it would be possible to solve exactly mixed-integer programs with a cutting plane generator and the simplex algorithm; however, in practice, this was shown to have very poor performance. Nevertheless, cutting planes help the branch-and-bound algorithm be more efficient, and all mixed-integer solvers use them nowadays.

Many commercial solvers also allow the users to write their own cutting planes, than can depend on the problem structure (for example, Section 5.2 derives cutting planes for a specific kind of constraint). This is done through callback functions that are called at branch-and-bound nodes: they are allowed to add constraints *that remove fractional solutions.*

An orthogonal but related notion is *lazy constraints*: as opposed to cutting planes, they do not strengthen the formulation, they are required for it to be correct. However, in many applications, the complete formulation is too large (for example, the subtour elimination and cut-set constraints for the TSP, as in Exercise 2.5.1): it is often desirable to ignore some constraints of the formulation, and to add them back only when they are violated. In this case, solvers often implement a similar mechanism called *lazy constraints* to add those missing constraints. As opposed to user cuts, these are allowed to remove *integer-feasible solutions.*

5.1 Integer and mixed-integer cuts

Remark. There are multiple correct answers for each question. Even if you use the exact same techniques as indicated in the solutions, you might get different cuts, but still correct.

5.1.1 Valid inequalities

Find a valid inequality for the following sets:

1. $X = \{x \in \mathcal{B}, y \in \mathcal{B} \,|\, 12\,x - 6\,y \leq 7\}$.
2. $X = \{x \in \mathbb{R}^+, y \in \mathcal{B} \,|\, x \leq 20\,y, \quad x \leq 7\}$.
3. $X = \{x \in \mathbb{R}^+, y \in \mathbb{N} \,|\, x \leq 6\,y, \quad x \leq 16\}$.
4. $X = \big\{x \in \mathbb{R}^+, (y_1, y_2, y_3) \in \mathbb{N}^3 \,|\, -x - \frac{10}{3}y_1 + y_2 + \frac{11}{4}y_3 \leq \frac{21}{2}\big\}$.
5. $X = \big\{(x, y, z) \in \mathcal{B}^3, s \in \mathbb{R}^+ \,|\, 2\,x + 3\,y + 9\,z - s = 32\big\}$.

Solution. Rounding for the first constraint:

$$2\,x - y \leq 1.$$

Disjunction on the binary/integer variable for the next two:

$$x \leq 7\,y \qquad \text{and} \qquad x \leq 4 + 4\,y.$$

MIR for the last two:

$$y + 3\,z - s \leq 10 \qquad \text{and} \qquad \frac{1}{3}\,y + \frac{5}{3}\,z + \frac{1}{3}\,s \leq 6.$$

5.1.2 Cutting planes

Find a valid inequality for X which cuts the point x^*:

$$X = \big\{(v, w, x, y, z) \in \mathbb{Z}^5 \,|\, 9\,v + 12\,w + 8\,x + 17\,y + 13\,z \geq 50\big\},$$

$$x^* = \left(0, \frac{25}{6}, 0, 0, 0\right).$$

Solution. $3\,v + 3\,w + 2\,x + 5\,y + 4\,z \geq 13$.

5.2 Knapsack covers

5.2.1 Knapsack covers

Find valid covers for the following X which cuts the points x^*:

1. $X = \big\{x \in \mathcal{B}^5 \,|\, 9\,v + 8\,w + 6\,x + 6\,y + 5\,z \leq 14\big\}$, $x^* = \big(0, \frac{5}{8}, \frac{3}{4}, \frac{3}{4}, 0\big)$.
2. $X = \big\{x \in \mathcal{B}^5 \,|\, 9\,v + 8\,w + 6\,x + 6\,y + 5\,z \leq 14\big\}$, $x^* = \big(\frac{1}{2}, \frac{1}{8}, \frac{3}{4}, \frac{3}{4}, 0\big)$.
3. $X = \big\{x \in \mathcal{B}^5 \,|\, 7\,v + 6\,w + 6\,x + 4\,y + 3\,z \leq 14\big\}$, $x^* = \big(\frac{1}{7}, 1, \frac{1}{2}, \frac{1}{4}, 1\big)$.
4. $X = \big\{x \in \mathcal{B}^5 \,|\, 12\,v - 9\,w + 8\,x + 6\,y - 3\,z \leq 2\big\}$, $x^* = \big(0, 0, \frac{1}{2}, \frac{1}{6}, 1\big)$.

Solution. In order: $w + x + y \leq 2$, $v + x \leq 1$, $v + w + z \leq 2$, $-w + x \leq 0$ (with a change of variables: $w' = 1 - w$ and $z' = 1 - z$).

5.2.2 Cover lifting

For the set $X = \big\{(u, v, w, x, y, z) \in \mathcal{B}^6 \mid 12\,u + 9\,v + 7\,w + 5\,x + 5\,y + 3\,z \leq 14\big\}$ and the cover $w + y + z \leq 2$:

1. Determine whether the cover is a facet of $X \cap \big\{(u, v, w, x, y, z) \in \mathcal{B}^6 \mid u = v = x = 0\big\}$.
2. Lift the inequality for X.

Solution. The cover is a facet of the given set, as it has three affinely independent points.

After one iteration of lifting (on u), the cover becomes $2\,u + w + y + z \leq 2$. After lifting v and x: $2\,u + v + w + y + z \leq 2$.

5.2.3 Solution to a knapsack problem only with covers and lifting

Solve the following knapsack instance (knapsack capacity: twenty kilograms) using a computer and only a LP solver (without support for integer variables).

Item	1	2	3	4	5	6	7
Profit (€)	4	5	6	6	6	10	15
Weight (kg)	5	5	6	7	8	12	16

1. Write the corresponding integer linear program. Find a feasible solution: this is a lower bound on the objective function. You should get 17.
2. Write its linear relaxation and solve it to optimality. The objective value is a first upper bound; you should get 19.4375.
3. Find as many violated cover inequalities as possible by the following process:
   (a) Solve the linear relaxation,
   (b) Find (at least) one cover inequality that is violated by the solution to this LP

   After adding four constraints, you could get the upper bound 18.6363 and the optimal solution

$$(0, 0.9090, 0.9090, 0.9090, 0, 0.1818, 0.0909)\,.$$

4. Extend some cover inequalities by inspection (try to add a new variable into the mix without changing the right-hand side).
   For example, with the inequality $x_2 + x_3 + x_6 \leq 2$, you can generate $x_2 + x_3 + x_6 + x_7 \leq 2$, as only two objects (at most) amongst them can be taken at once: if you take x_2 and x_3, then it is no more possible to take x_6 or x_7.

After adding three constraints (the previous one and two slight variations thereof), you could get the upper bound 18.6 and the optimal solution

$$(0.4, 1, 1, 1, 0, 0, 0)\,.$$

5. Show that an extended inequality by the previous procedure is less strong than one generated by lifting. To do this, start from $x_1+x_2+x_3+x_4 \leq 3$, generate one valid inequality by extension, and another one by lifting, compare them by optimising the corresponding relaxations (is one bound tighter than the other?).
   After adding the lifted cover, you could get the upper bound 17.6666 and the optimal solution

$$(0, 0.3333, 1, 1, 0.6666, 0, 0)\,.$$

6. Add new cover inequalities and lift them until the problem is solved; you should get an optimal value of 17 with the optimal solution

$$(0, 1, 1, 1, 0, 0, 0)\,.$$

Remark. Multiple paths are possible, getting different intermediate results. Some may be shorter than others; the intermediate results are shown for the shortest one.

5.3 Exam exercises

5.3.1 Exam January 2013

For each of the following sets, give a valid inequality for the discrete set that cuts the point $\overline{x}$.

1. $X_1 = \left\{x \in \{0,1\}^4 \;\middle|\; 5\,x_1 + 7\,x_2 + 8\,x_3 + 9\,x_4 \leq 15\right\}$ and $\overline{x}^{(1)} = \left(1, 1, \frac{3}{8}, 0\right)$.
2. $X_2 = \left\{x \in \mathbb{Z}_+^4 \;\middle|\; x_1 - \frac{2}{3}\,x_2 + \frac{7}{6}\,x_3 + \frac{4}{3}\,x_4 = \frac{2}{3}\right\}$ and $\overline{x}^{(2)} = \left(\frac{2}{3}, 0, 0, 0\right)$.
3. $X_3 = \left\{x \in \mathbb{Z}_+^4 \;\middle|\; x_1 - \frac{1}{3}\,x_2 + \frac{5}{3}\,x_3 + \frac{1}{3}\,x_4 = \frac{1}{3}\right\}$ and $\overline{x}^{(3)} = \left(\frac{1}{2}, \frac{1}{2}, 0, 0\right)$.

5.3.2 Exam August 2013

We consider the set $X = \left\{x \in \{0,1\}^4 \;\middle|\; 3\,x_1 + 5\,x_2 + 6\,x_3 + 11\,x_4 \leq 13\right\}$.

1. Show that the inequality $x_1 + x_2 + x_3 \leq 2$ is valid for X.
2. Show that $x_1 + x_2 + x_3 \leq 2$ is not a facet of $\operatorname{conv}(X)$.
3. Lift the variable x_4 into the inequality.
4. Show that the new inequality in a facet of $\operatorname{conv}(X)$.

5.3.3 Exam January 2014

Find a valid inequality for X which cuts the point x^* for the following cases.

1. $X = \left\{ x \in \mathbb{Z}^4 \,\middle|\, 5\,x_1 + 10\,x_2 - 9\,x_3 + 20\,x_4 \leq 42 \right\}$ and $x^* = (9, 1, 1, 0)$.
2. $X = \left\{ x \in \{0,1\}^7 \,\middle|\, 5\,x_1 + 5\,x_2 + 6\,x_3 + 7\,x_4 + 8\,x_5 + 12\,x_6 + 16\,x_7 \leq 20 \right\}$ and $x^* = \left(0, \frac{1}{3}, 1, 1, \frac{2}{3}, 0, 0\right)$.
3. $X = \left\{ x \in \mathbb{Z}^3 \,\middle|\, 3\,x_1 + 4\,x_2 + 9\,x_3 \leq 33 \right\}$ and $x^* = (-4, 6, 2)$.

5.3.4 Exam August 2014

For each discrete set and each point, find a valid inequality that cuts off the given point.

1. $X_1 = \left\{ x \in \{0,1\}^4 \,\middle|\, 7\,x_1 + 9\,x_2 + 10\,x_3 + 12\,x_4 \leq 25 \right\}$ and $x = \left(0, 1, 1, \frac{1}{2}\right)$.
2. $X_2 = \left\{ (x, y) \in \mathbb{R}^+ \times \{0,1\} \,\middle|\, x \leq 20\,y, \quad x \leq 5 \right\}$ and $x = \left(5, \frac{1}{4}\right)$.
3. $X_3 = \left\{ x \in \mathbb{Z}^4_+ \,\middle|\, 6\,x_1 + 7\,x_2 - 3\,x_3 - x_4 \leq 1 \right\}$ and $x = \left(\frac{1}{6}, 0, 0, 0\right)$.

5.3.5 Exams January and August 2015

For each of the following sets, determine a valid inequality for the discrete set separating the given point $\overline{x}$.

1. $X_1 = \left\{ x \in \{0,1\}^4 \,\middle|\, 5\,x_1 + 7\,x_2 + 8\,x_3 + 13\,x_4 \leq 17 \right\}$ and $\overline{x}^{(1)} = \left(1, 1, \frac{1}{2}, 0\right)$. If possible, provide a facet-defining inequality of $\text{conv}(X_1)$.
2. $X_2 = \left\{ x \in \mathbb{Z} \times \mathbb{R}^+ \,\middle|\, x_1 + x_2 \geq \frac{5}{4} \right\}$ and $\overline{x}^{(2)} = \left(\frac{5}{4}, 0\right)$.
3. $X_3 = \left\{ x \in \mathbb{Z}^4_+ \,\middle|\, 3\,x_1 - \frac{1}{2}\,x_2 + 5\,x_3 + \frac{3}{2}\,x_4 \leq \frac{1}{2} \right\}$ and $\overline{x}^{(3)} = \left(\frac{1}{6}, 0, 0, 0\right)$.

5.3.6 Exam September 2016

For each of the following sets, determine a valid inequality for the discrete set separating the given point $\overline{x}$.

1. $X_1 = \left\{ x \in \{0,1\}^4 \,\middle|\, 9\,x_1 + 5\,x_2 + 7\,x_3 + 13\,x_4 \leq 17 \right\}$ and $\overline{x}^{(1)} = \left(1, 1, \frac{3}{7}, 0\right)$. If possible, provide a facet-defining inequality of $\text{conv}(X_1)$.
2. $X_2 = \left\{ x \in \mathbb{Z} \times \mathbb{R}^+ \,\middle|\, x_1 + x_2 \geq \frac{7}{5} \right\}$ and $\overline{x}^{(2)} = \left(\frac{7}{5}, 0\right)$.
3. $X_3 = \left\{ x \in \mathbb{Z}^4_+ \,\middle|\, 3\,x_1 - \frac{1}{3}\,x_2 + \frac{6}{5}\,x_3 - \frac{3}{2}\,x_4 \leq \frac{1}{2} \right\}$ and $\overline{x}^{(3)} = \left(\frac{1}{2}, 0, 0, 0\right)$.

5.4 Lazy constraints

5.4.1 Travelling salesman problem

The generalised subtour elimination constraint can be used to formulate the prize-collecting travelling salesman problem: with respect to the travelling salesman problem, travelling through an edge e has a cost c_e, and visiting a city j allows them to make a profit of f_j; not all cities have to be visited, but the salesman still must follow a cycle, starting at the first city.

1. Give (at least) one MILP model for the travelling salesman problem (see Exercise 2.5.1).

2. Which constraints would be interesting to be added as lazy constraints during the optimisation?

3. Implement the TSP with lazy constraints.

Solution. The subtour elimination formulation:

$$\begin{array}{lll} \min & \sum_e c_e\, x_e & \\ \text{s.t.} & \sum_{e \in \delta(i)} x_e = 2, & \forall i \in V, \\ & \sum_{e \in E(S)} x_e \leq |S| - 1, & \forall S \subsetneq V :\ 2 \leq |S| \leq |V| - 1. \end{array}$$

Consider adding lazily the subtour elimination, based on a subtour you find in the graph (any tour that does not include the origin is a subtour).

5.4.2 Prize-collecting travelling salesman problem

The generalised subtour elimination constraint can be used to formulate the prize-collecting travelling salesman problem: with respect to the travelling salesman problem, travelling through an edge e has a cost c_e, and visiting a city j allows them to make a profit of f_j; not all cities have to be visited, but the salesman still must follow a cycle, starting at the first city.

1. Give a MILP model for the prize-collecting travelling salesman problem using the generalised subtour elimination constraint (GSEC).

2. What could be practical problems of implementing this formulation?

3. Derive a separation procedure that could be used for a lazy-constraint-based implementation of the problem. The generated constraint must maximise the violation.

4. Solve the following instance of the prize-collecting travelling salesman

problem with lazy constraint generation.

$$c_e = \begin{pmatrix} 0 & 4 & 3 & 3 & 5 & 2 & 5 \\ 4 & 0 & 5 & 3 & 3 & 4 & 7 \\ 3 & 5 & 0 & 4 & 6 & 0 & 4 \\ 3 & 3 & 4 & 0 & 4 & 4 & 6 \\ 5 & 3 & 6 & 4 & 0 & 5 & 8 \\ 2 & 4 & 0 & 4 & 5 & 0 & 3 \\ 5 & 7 & 4 & 6 & 8 & 3 & 0 \end{pmatrix},$$

$$f_j = \begin{pmatrix} 2 & 4 & 1 & 3 & 7 & 1 & 7 \end{pmatrix}.$$

Solution. A formulation:

$$\begin{array}{lll} \min & \max \sum_{j \in N} f_j \, y_j - \sum_{e \in E} c_e \, x_e & \\ \text{s.t.} & \sum_{e \in \delta(j)} x_e = 2 \, y_j, & \forall j \in V, \\ & y_{\text{origin}} = 1, & \\ & \sum_{e \in E(S)} x_e \le \sum_{j \in S \setminus \{k\}} y_j, & \forall k \in S, \quad \forall S \subseteq V \setminus \{\text{origin}\}. \end{array}$$

Consider adding lazily the subtour elimination, based on a subtour you find in the graph (any tour that does not include the origin is a subtour).

Chapter 6

Lagrangian duality

When a problem is too complex to solve, decomposition techniques can be used. One of them is *Lagrangian duality.* It allows removing problematic constraints, moving them into the objective function (their violation is thus penalised). these constraints are chosen such that, once they are removed, the problem becomes much easier to solve (for example, very efficient algorithms are known, for example based on flows or other polynomially solvable problems, as in Part III).

6.0.1 Symmetric travellingsalesman problem

Use Lagrangian relaxation to solve the STSP instance with distances:

$$\begin{pmatrix} - & 8 & 2 & 14 & 26 & 13 \\ - & - & 7 & 4 & 16 & 8 \\ - & - & - & 23 & 14 & 9 \\ - & - & - & - & 12 & 6 \\ - & - & - & - & - & 5 \\ - & - & - & - & - & - \end{pmatrix}.$$

Remark. The *symmetric* travelling salesman problem is a regular TSP (like Exercise 2.5.1) where distances are supposed to be symmetric, i.e. $d(A, B) = d(B, A)$.

6.0.2 Generalised assignment

Use Lagrangian relaxation to solve the following *generalised assignment* instance:

$$\begin{array}{lll} \max & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} & \\ \text{s.t.} & \sum_{j=1}^{n} x_{i,j} \leq 1 & \forall i \in [1, m] \\ & \sum_{i=1}^{m} a_{i,j} x_{i,j} \leq b_j & \forall j \in [1, n] \\ & x \in \mathcal{B}^{mn}, & \end{array}$$

where the costs are given by the matrices:

$$\mathbf{A} = \begin{pmatrix} 5 & 7 & 2 \\ 14 & 8 & 7 \\ 10 & 6 & 12 \\ 8 & 4 & 15 \\ 6 & 12 & 5 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 15 \\ 15 \\ 15 \end{pmatrix}, \qquad \mathbf{C} = \begin{pmatrix} 6 & 10 & 1 \\ 12 & 12 & 5 \\ 15 & 4 & 3 \\ 10 & 3 & 9 \\ 8 & 9 & 5 \end{pmatrix}.$$

6.0.3 * Generalised assignment (exam January 2014)

With the following *generalised assignment* instance:

$$\begin{array}{lcl} \max & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} & \\ \text{s.t.} & \sum_{j=1}^{n} x_{i,j} \leq 1 & \forall i \in [1,m]\,, \\ & \sum_{i=1}^{m} a_{i,j} x_{i,j} \leq b_j & \forall j \in [1,n]\,, \\ & x \in \mathcal{B}^{mn}, & \end{array}$$

where the costs are given by the matrices:

$$\mathbf{A} = \begin{pmatrix} 5 & 6 & 2 \\ 3 & 6 & 6 \\ 8 & 5 & 9 \\ 4 & 8 & 10 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix}, \qquad \mathbf{C} = \begin{pmatrix} 6 & 10 & 3 \\ 12 & 12 & 4 \\ 15 & 8 & 7 \\ 10 & 7 & 10 \end{pmatrix}.$$

1. Discuss the strength of **all** possible Lagrangian relaxations, and the ease or difficulty of solving the Lagrangian subproblems, and the Lagrangian dual.

2. Use Lagrangian relaxation to solve the given generalized assignment problem instance.

As a reminder, the formula for the subgradient algorithm is:

$$u_{k+1} = u_k - \epsilon_k \frac{z_k(u_k) - \underline{z}}{||d - Dx(u_k)||^2} \left(d - Dx(u_k)\right).$$

6.0.4 * Assignment problem and relaxation strength

Consider the assignment problem with a budget constraint:

$$\begin{array}{lcl} \max & \sum_{i \in M} \sum_{j \in N} c_{i,j} x_{i,j} & \\ \text{s.t.} & \sum_{j \in N}^{n} x_{i,j} = 1 & \forall i \in M, \\ & \sum_{i \in M}^{n} x_{i,j} = 1 & \forall j \in N, \\ & \sum_{i=1}^{m} \sum_{jinN} a_{i,j} x_{i,j} \leq b, & \\ & x \in \mathcal{B}^{mn}. & \end{array}$$

Discuss the strength of different possible Lagrangian relaxations, and the ease or difficulty of solving the Lagrangian subproblems, and the Lagrangian dual.

6.0.5 * Uncapacitated facility location

Consider the following uncapacitated facility location instance:

$$m = 6, \qquad n = 5,$$

$$c_{ij} = \begin{pmatrix} 6 & 2 & 1 & 3 & 5 \\ 4 & 10 & 2 & 6 & 1 \\ 3 & 2 & 4 & 1 & 3 \\ 2 & 0 & 4 & 1 & 4 \\ 1 & 8 & 6 & 2 & 5 \\ 3 & 2 & 4 & 8 & 1 \end{pmatrix} \qquad \text{(delivery costs)}\,,$$

$$f = \begin{pmatrix} 4 & 8 & 11 & 7 & 5 \end{pmatrix} \qquad \text{(fixed costs)}\,.$$

Solve the Lagrangian subproblem IP$(\mathbf{u})$ with the dual vector $\mathbf{u} = \begin{pmatrix} 5 & 6 & 3 & 1 & 2 & 1 \end{pmatrix}$ in order to obtain an optimal dual solution $(\mathbf{x}(\mathbf{u})\,, \mathbf{y}(\mathbf{u}))$ and a lower bound $\underline{z(\mathbf{u})}$.
Then, modify the solution $(\mathbf{x}(\mathbf{u})\,, \mathbf{y}(\mathbf{u}))$ so get a good feasible *primal* solution.

Part II

Combinatorial optimisation

Chapter 7

Dynamic programming

7.1 Knapsack

7.1.1 Basic knapsack

Solve the binary knapsack problem for five objects: the values are $(3, 4, 6)$ and the weights $(2, 3, 5)$, with a total capacity of 8.

1. Propose a MILP formulation.

2. Solve it by dynamic programming.

Solution. A formulation:

$$\begin{array}{cc} \max & 3\,x + 4\,y + 6\,z \\ \text{s.t.} & 2\,x + 3\,y + 5\,z \leq 8, \\ & x \in \mathcal{B}, y \in \mathcal{B}, z \in \mathcal{B}. \end{array}$$

The optimal solution is to take the second and third objects.

7.1.2 Exam knapsack (January 2013)

Solve the following knapsack.

$$\begin{array}{cc} \max & 4\,w + 5\,x + 3\,y + z \\ \text{s.t.} & 5\,w + 6\,x + 4\,y + 2\,z \leq 14, \\ & w \in \mathcal{B}, x \in \mathcal{B}, \\ & y \in \mathbb{Z}^+, z \in \mathbb{Z}^+. \end{array}$$

7.1.3 Exam knapsack (August 2015)

Solve the following knapsack.

$$\begin{array}{cc} \max & 5\,w + 4\,x + 7\,y + 13\,z \\ \text{s.t.} & 4\,w + 4\,x + 5\,y + 9\,z \leq 13, \\ & w \in \mathbb{Z}^+, x \in \mathbb{Z}^+, \\ & y \in \mathbb{Z}^+, z \in \mathbb{Z}^+. \end{array}$$

7.1.4 Exam knapsack (September 2016)

Solve the following knapsack.

$$\begin{array}{rc} \max & 7\,w + 9\,x + 12\,y + 3\,z \\ \text{s.t.} & 5\,w + 6\,x + 8\,y + 3\,z \leq 15, \\ & w \in \mathcal{B}, x \in \mathcal{B}, y \in \mathcal{B}, \\ & z \in \mathbb{Z}^+. \end{array}$$

7.1.5 Knapsack duality

Solve the binary knapsack problem for five objects: the values are $(3, 4, 6, 5, 8)$ and the weights $(412, 507, 714, 671, 920)$, with a total capacity of 1794.

1. Why would instance this cause problems to a dynamic programming algorithm?

2. With the following definitions ($X \subseteq \mathbb{R}^n$)

$$\begin{array}{rclc} f(\lambda) & = & \max & \mathbf{c}^\mathbf{T}\,\mathbf{x} \\ & & \text{s.t.} & \mathbf{a}^\mathbf{T}\,\mathbf{x} \leq \lambda, \\ & & & x \in X \end{array} \qquad \begin{array}{rclc} h(t) & = & \min & \mathbf{a}^\mathbf{T}\,\mathbf{x} \\ & & \text{s.t.} & \mathbf{c}^\mathbf{T}\,\mathbf{x} \geq t \\ & & & x \in X \end{array}$$

   prove the two following statements:

   (a) $f(\lambda) \geq t$ if and only if $h(t) \leq \lambda$.

   (b) $f(\lambda) = \max \left\{ t \,\middle|\, h(t) \leq \lambda \right\}$.

3. Dualise this instance using the previous statements and solve this dual by dynamic programming. ($f(\lambda)$ is a knapsack, $h(t)$ is its dual, and can still be solved as a sequence of smaller and related knapsacks when properly rewritten.)

Solution. The complexity of the dynamic algorithm is $\mathcal{O}(|\mathcal{O}|\ C)$.

First, the proof revolves around the optimal $\mathbf{x}$ and the corresponding λ and t. Then, prove the second statement by injecting the first one in the maximum.

To rewrite the dual as a knapsack, use a change of variables: $\mathbf{x}' = 1 - \mathbf{x}$. This leads to the definition of

$$\begin{array}{rclcc} j(s) & = & 3224 & - & \max & 412\,v' + 507\,w' + 714\,x' + 671\,y' + 920\,z' \\ & & & & \text{s.t.} & 3\,v' + 4\,w' + 6\,x' + 5\,y' + 8\,z' \leq s \\ & & & & & \mathbf{x} \in \mathcal{B}^5. \end{array}$$

The optimal solution to this dual knapsack is linked to the first one by:

$$f(\lambda = 1794) = 26 - \min \left\{ s \,\middle|\, j(s) \geq 1430 \right\}.$$

The optimum is to take the third and fifth objects.

7.1.6 Exam dual knapsack (January 2015)

Apply the dual knapsack technique to solve the following knapsack.

$$\begin{array}{ll} \max & 3\,w + 4\,x + 6\,y + 8\,z \\ \text{s.t.} & 412\,w + 507\,x + 714\,y + 881\,z \leq 1394, \\ & w \in \mathcal{B}, x \in \mathcal{B}, y \in \mathcal{B}, z \in \mathcal{B}. \end{array}$$

7.2 Lot sizing

7.2.1 Basic lot sizing

A steel mill manufactures beams; the director wants to plan the production for the next four weeks. The demand for beams is exactly known for each week. The cost for using the furnace can be divided into two parts: a fixed cost to get the furnace turned on and a variable cost proportional to the number of beams produced. Storing beams from a time period to another is costly, and is proportional to the number of beams to store.

1. Propose a MILP formulation.
2. Propose an algorithm to solve it by dynamic programming and apply it on this instance.

Week	t	1	2	3	4
Demand [10 spools]	d_t	8	5	13	4
Variable cost [1000€ per 10 spools]	p_t	1	1	1	2
Fixed cost [1000€]	f_t	20	10	45	15
Storage cost [1000€ per 10 spools]	h_t	1	1	1	1

Table 7.1: Spool factory planning requirements.

Solution. A formulation is:

$$\begin{array}{lll} \min & \sum_{t=1}^{4} (f_t\, y_t + p_t\, x_t + h_t\, s_t) & \\ \text{s.t.} & s_0 = 0, & \\ & s_t = s_{t-1} + x_t - d_t, & \forall t \in [2,4]\,, \\ & x_t \leq \left(\sum_{\tau=1}^{4} d_\tau\right) y_t, & \forall t \in [1,4]\,, \\ & x_t \in \mathbb{Z}^+, y_t \in \mathcal{B}, s_t \in \mathbb{Z}^+, & \forall t \in [1,4]\,. \end{array}$$

When the storage costs are not considered (the previous problem can be rewritten to include them in the production costs: $c_t = p_t + \sum_{i=t}^{4} h_i$), a dynamic algorithm is:

$$H(0) = 0, \qquad H(1) = f_1 + c_1\, d_1,$$

$$H(k) = \min_{1 \leq t \leq k} \left\{ H(t-1) + f_t + c_t \sum_{i=t}^{k} d_i \right\}.$$

Periods	t	1	2	3	4
Demand	d_t	5	2	7	3
Fixed costs	f_t	20	20	25	15
Varying costs	p_t	3	3	4	2
Inventory costs	h_t	1	1	1	1

Table 7.2: Data for exercise 7.2.2.

A variation that directly takes into account the storage costs:

$$H(k) = \min_{1 \le t \le k} \left\{ H(t-1) + f_t + p_t \sum_{i=t}^{k} d_i + \sum_{i=t}^{k-1} \left[h_i \left(\sum_{j=i+1}^{k} d_j \right) \right] \right\}.$$

7.2.2 Lot sizing (exam January 2016)

In the lot-sizing problem, a company wants to plan its schedule of production by batches for a given number of periods. It must satisfy a demand at each time period and pays a fixed cost for producing a batch, a varying cost depending on the amount produced and a varying inventory cost.

Solve the following lot-sizing problem using dynamic programming with the data in the table below.

7.3 Time-constrained shortest path

7.3.1 Basic time-constrained shortest path

A traveller wants to go from one place to another. However, they must go through customs, whose office has strict opening hours: with the considered time window, only the time at which they start working is important. As such, they want to determine their minimum travel time.

This travel must then be made in a graph $G = (V, E)$, which is annotated with some data: c_{ij} the time to cross edge $e = (i, j) \in E$; r_i, the earliest time to travel through node $i \in V$ (if the traveller gets at i earlier than r_i, they will have to wait until r_i).

1. Propose a MILP formulation.

2. Propose an algorithm to solve it by dynamic programming.

Solution. A formulation is:

$$\begin{array}{lll}
\min & t_n & \\
\text{s.t.} & \sum_{i \in V} x_{i,k} = \sum_{j \in V} x_{j,k}, & \forall k \in V \setminus \{1, n\}, \\
& \sum_{i \in V} x_{1,i} = 1, & \\
& \sum_{i \in V} x_{i,n} = 1, & \\
& & \\
& t_1 = 0, & \\
& t_i \geq 0, & \forall i \in V, \\
& t_j \geq t_i + c_{i,j}\, x_{i,j}, & \forall (i,j) \in E, \\
& t_i \geq r_i, & \forall i \in V, \\
& & \\
& x_e \in \mathcal{B}, & e \in E.
\end{array}$$

A dynamic algorithm for this problem is a variation of Bellman-Ford algorithm for shortest paths:

$$t_1(j) = \begin{cases} \max\{r_j, c_{1,j}\} & \forall i: \ (i,j) \in E \\ +\infty & \text{otherwise} \end{cases}$$

$$t_k(j) = \max\left\{ r_j, \quad \min\left\{ t_{k-1}(j), \qquad \min_{i \in \delta^-(j)} \{t_{k-1}(i) + c_{i,j}\} \right\}\right\}.$$

Chapter 8

Constraint programming

Usually, computer code is *imperative*: it describes a series of operations to follow. On the other hand, *declarative* programming only stands relations between the variables (constraints); afterwards, a solver comes and find a solution that satisfies all those constraints. Integer programming (covered in Part I) is one kind of declarative programming which is particularly well-suited to finding optimal solutions; on the other hand, *constraint* programming excels at finding feasible solutions.

The main difference between mixed-integer programming and constraint programming is the technology behind the solvers: usually, modelling is easier with constraint programming, as the constraints are more expressive; on the other hand, solving mixed-integer programs is easier that constraint programs, as all constraints have the same structure. As a consequence, the algorithms behind each paradigm are very different (even though they can merge: SCIP can solve nonlinear mixed-integer programs with constraint programming techniques).

Constraint programming is very expressive: linear constraints (as those of mixed-integer linear programming) are allowed within constraint programs; nonlinear constraints are too (for example, products of variables). Logic constraints can be added as is, without rewriting them as linear constraints (mixing AND, OR, NOT, etc., but not first-order logic, with $\forall$ and $\exists$). More specific constraints are defined, such as `alldifferent`: its arguments must take distinct values (for example, `alldifferent(1, 2, 3)` holds, but not `alldifferent(2, 2)`). *Reification* is also possible, it allows imposing constraints such as “one or two constraints of this set of three constraints must hold”.

Whereas mixed-integer programming relies on a single family of algorithms for any kind of problem, constraint programming requires some code for handling each kind of constraint, i.e. *propagators*, which actually implement the constraint. In other words, each problem may be formulated as only one constraint—the tedious work being to write the propagator that is specific for this problem. Hence, constraint programming formulation is about *reusing* existing constraints, which have seen a lot of research about the best propagators (the Global Constraint Catalog lists many kinds of constraints and propagators).

8.0.1 Sudoku

Sudoku is a kind of puzzle game. It involves an $N \times N$ grid, some of the spaces being filled. The goal is to fill all spaces with numbers between 1 and N^2 while meeting a series of constraints: "the same single integer may not appear twice in the same row, column or in any of the $N\ \sqrt{N} \times \sqrt{N}$ subregions of the $N \times N$ playing board" [Wikipedia].

1. Explain how a human could solve any sudoku puzzle using the same principles as constraint programming. Apply it on the given grid.
2. Write a mixed-integer programming model to solve sudoku puzzles.
3. * The solver performs many operations on the model before actually solving it (this is called *presolving*), and for the sudoku it might very well solve the problem (without needing any branch-and-bound or cutting plane). How does these operations relate to the way a human would solve the problem (as in 1)?
4. Write a constraint programming model to solve sudoku puzzles. Use only the constraint `alldifferent`.

1			
		2	
	3		4

Table 8.1: Example 2×2 sudoku grid.

Solution. A human tries to apply the constraints to find spaces where only one value is possible. More advanced techniques imply considering all values for each space, and eliminating those that do not respect the constraints.

A MILP formulation is:

$$\begin{array}{lrl}
\text{find} & \text{a solution} & \\
\text{s.t.} & \sum_{k=1}^{N^2} x_{i,j,k} = 1, & \forall i \in \left[1, N^2\right], \forall j \in \left[1, N^2\right], \\
 & \sum_{j=1}^{N^2} x_{i,j,k} = 1, & \forall i \in \left[1, N^2\right], \forall k \in \left[1, N^2\right], \\
 & \sum_{i=1}^{N^2} x_{i,j,k} = 1, & \forall j \in \left[1, N^2\right], \forall k \in \left[1, N^2\right], \\
 & \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} x_{i+l,j+m,k} = 1, & \forall i \in [1, N], \forall j \in [1, N], \forall k \in \left[1, N^2\right], \\
 & x_{i,j,k} = 1, & \forall i \in \left[1, N^2\right], \forall j \in \left[1, N^2\right], \forall k \in \left[1, N^2\right], v_{i,j,k} = 1.
\end{array}$$

Presolving can, based on known values, eliminate a series of variables: if a figure is known for one constraint, all other variables in the constraint can be set to zero. Once the variables are eliminated, the remaining constraints are simpler. In theory, branch-and-bound is only required when multiple solutions exist.

A CP formulation is:

$$\begin{array}{llr}
\text{find} & \text{a solution} & \\
\text{s.t.} & \text{alldifferent}(x_{i1}, x_{i2} \cdots x_{iN^2}) \,, & \forall i \in \left[1, N^2\right], \\
& \text{alldifferent}\big(x_{1j}, x_{2j} \cdots x_{N^2 j}\big) \,, & \forall j \in \left[1, N^2\right], \\
& \text{alldifferent}\begin{pmatrix} x_{i,j}, & x_{i+1,j} & \cdots & x_{i+N,j} \\ x_{i,j+1}, & x_{i+1,j+1} & \cdots & x_{i+N,j+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i,j+N}, & x_{i+1,j+N} & \cdots & x_{i+N,j+N} \end{pmatrix}, & \forall i \in [1, N] \,, \forall j \in [1, N] \,.
\end{array}$$

8.0.2 Magic square

"A magic square is an arrangement of distinct integers in an $N \times N$ grid, where the numbers in each row, and in each column, and the numbers in the main and secondary diagonals, all add up to the same number" [Wikipedia]. This number is given by the following formula, based on the size of the grid N:

$$\frac{N\left(N^2+1\right)}{2}.$$

Thus, for a 3×3 grid, the three digits in each row, column, and diagonal must sum up to 15; for a 4×4 grid, they must sum up to 34; etc.

Write a constraint programming model to solve magic squares. Use only the constraint `alldifferent` and linear equalities.

Solution.

$$\begin{array}{llr}
\text{find} & \text{a solution} & \\
\text{s.t.} & 2\,V = N\left(N^2+1\right), & \\
& \sum_{j=1}^{N} x_{ij} = V, & \forall i \in [1, N] \,, \\
& \sum_{i=1}^{N} x_{ij} = V, & \forall j \in [1, N] \,, \\
& \sum_{i=1}^{N} x_{ij} = V, & \forall j \in [1, N] \,, \\
& \text{alldifferent}(x_{ij} \quad \forall i \in [1, N] \,, \forall j \in [1, N]) \,. &
\end{array}$$

8.0.3 N queens (inspired by exam January 2014)

The N-queens problem is about placing N queens in an $N \times N$ grid such that no two queens can threaten each other. The rules for queens are those of chess: a queen can attack another one if they are on the same row, column, or diagonal.

1. Write a constraint programming model to solve the n-queens problem. Use only the constraint `alldifferent`.
2. Write a mixed-integer linear program to solve the n-queens problem. Compare it to the constraint programming example. How many constraints does your model use? What is the domain of the variables?
3. Solve the problem for $N = 4$ using reasoning (domain filtering based on the constraints) and enumeration. How many solutions exist?

Solution.

$$\begin{array}{rl} \text{find} & \text{a solution} \\ \text{s.t.} & \text{alldifferent}(x_i \quad \forall i \in [1,N])\,, \\ & \text{alldifferent}(x_i - i \quad \forall i \in [1,N])\,, \\ & \text{alldifferent}(x_i + i \quad \forall i \in [1,N])\,. \end{array}$$

8.0.4 * Stable mate matching (exam January 2016)

In the stable room-mate matching problem, we are given $2n$ people that we need to match by pairs in order to occupy n rooms. Each person has a decreasing order of people that he would like to share the room with. The goal of the problem is to find a matching of the people in groups of two in n rooms. We would like to model the stability of the matching, i.e. there cannot exist a pair of persons i and j that are not matched together but such that i is higher in the order of preference of j than the current room-mate of j and j is higher in the order of preference of i than the current room-mate of i (both at the same time).

1. Model the stable room-mate problem using the constraint programming paradigm. You can use the following constraints: linear constraints, propositional logic, `alldifferent`.
2. Model the matching (without the stability requirement) as an integer programming programming problem.
3. Model the additional stability requirement in the integer programming problem modelled in (2).

Example. Assume there are four people whose order of preference is given as follows:

Order(1) : $2, 3, 4$. Order(2) : $1, 4, 3$. Order(3) : $1, 4, 2$. Order(4) : $1, 2, 3$.

If we match the people as $(1, 2)$ and $(3, 4)$, the matching is stable. If we match the people as $(1, 3)$ and $(2, 4)$, the matching is not stable as both 1 and 2 would prefer to leave their roommate to go together in a room.

Part III

Polynomially-solvable problems

Chapter 9

Flow problems

Flows are a kind of polynomially-solvable problems. They can be modelled as mixed-integer linear programs, but more specific algorithms can solve them faster. The first step, however, is to formulate a problem as a maximum flow, as was previously done for mixed-integer programs (Chapter 2) or constraint programming (Chapter 8).

Maximum flows are graph problems. Within a network of nodes, one being the source s and another the destination t, each edge having a defined capacity, the goal is to push as much flow into the network as possible. This flow must start from s and end at t, but also respect the capacities of all the edges in the network. Edges are directed.

9.1 Modelling

9.1.1 Village fair

A village fair gathers the families of the whole village and the close surroundings. The organisers want to have a dinner to end the fair, but they need to sit people to the tables. In order to increase social interactions, at most three members of each family can be present at any given table.

Model this situation as a maximum flow problem, with p families (with a_i members for family i) and q tables (with a capacity b_j for table j).

Solution. Use one node per family (the family i sending a_i units of flow) and one per table (the table j consuming at most b_j units of flow).

9.1.2 Nurses scheduling

Nurses make shifts in a hospital to fulfil the needs for each and every department and still have some rest. Each shift lasts for eight hours (meaning that there are three shifts per day). Each department has minimum requirements in terms of nurses per day (to be hired for the three shifts); due to the nature of their workload, minimum requirements are also imposed on each shift (summing over the departments). Furthermore, the nurses should be allocated to the departments following some bounds.

Model this situation as a minimum flow problem.

Nurses	Emergency	Neonatalogy	Orthopaedic surgery	
Shift 1	Between 6 and 8	Between 11 and 12	Between 7 and 12	At least 26
Shift 2	Between 4 and 6	Between 11 and 12	Between 7 and 12	At least 24
Shift 3	Between 2 and 4	Between 10 and 12	Between 5 and 7	At least 19
	At least 13	At least 32	At least 22	

Table 9.1: Nurses requirements.

Solution. Use one node per shift and one per department.

9.1.3 Capacitated lot sizing

A steel mill manufactures I-beams; the director wants to plan the production for the next twelve weeks. The demand for I-beams is exactly known for each week. The plant is dimensioned for a limited production each week (100 beams). Due to limited space, at most 300 beams can be stored within the mill. The director only wants a feasible production planning (i.e. the number of beams to produce each week), without considering production costs.

1. Formulate this problem as a feasible flow problem.
2. Formulate this problem as a linear mixed-integer optimization problem.

Week	1	2	3	4	5	6	7	8	9	10	11	12
Demand [10 beams]	7	5	3	5	5	9	1	8	5	6	2	2

Table 9.2: I-beams factory planning requirements.

Solution. Use one node per time period.

9.1.4 * Character design

For a new game, a game designer wants to assign a series of C characters to both at least three traits (among T ones) and to exactly one political party (among P ones). For balancing, each trait and each party can have at most $C/2$ characters. The designer longs for a tool that can produce such assignments so they can iterate through them and find other ideas.

1. Formulate this problem as (1) a *feasible* flow problem, and (2) as a linear mixed-integer optimization problem.

2. While working on the lore, the designer desires that some given characters are not assigned to some traits. How do you modify your two models to accommodate this change?

3. Later on, they wish that some characters can have one or two political parties. What is the impact on your two models?

Solution. Use one node per trait and one per party, plus two nodes per character. Then, remove some edges. Finally, modify the capacities of edges towards the parties.

9.2 Maximum flows

9.2.1 Maximum flow with the shortest augmenting path algorithm

Solve the following maximum flow with the shortest augmenting path algorithm.

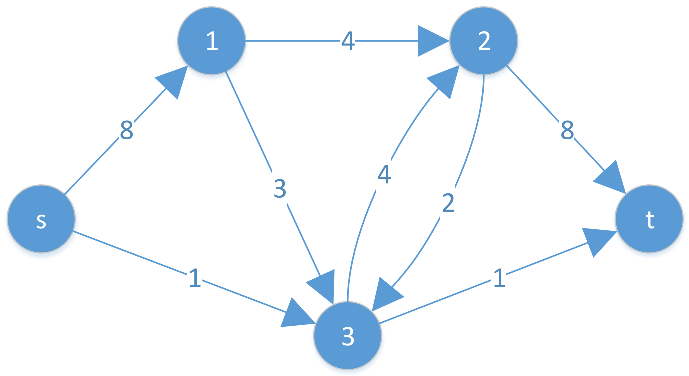

Figure 9.1: Network for exercise 9.2.1.

Solution. The maximum flow is 8.

9.3 Feasible flows

9.3.1 Transformation into a maximum flow

Solve the following feasibility flow with bound constraints as a maximum flow with edge demands.

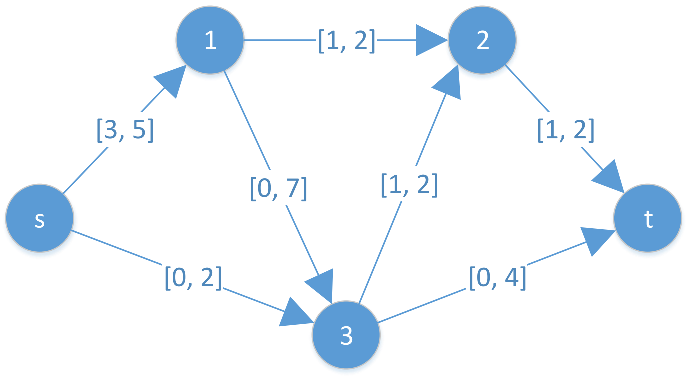

Figure 9.2: Network for exercise 9.3.1.

9.3.2 Transformation into a maximum flow (exam January 2017)

Consider the following directed graph. On each arc, we are given a lower and an upper bound on the ow that should traverse the arc.

1. Formulate the problem of finding a feasible flow in this graph as a maximum flow problem.

2. Apply a maximum flow algorithm in order to find a feasible flow.

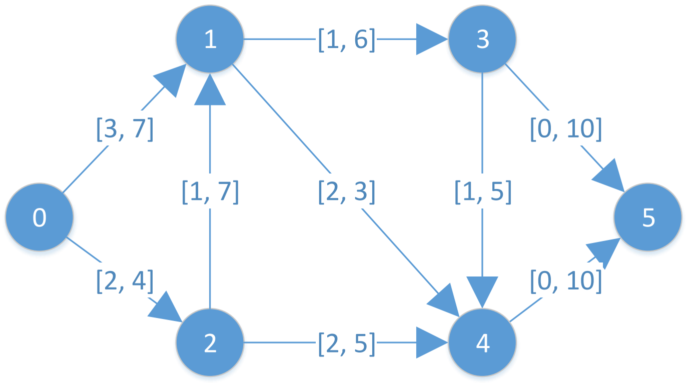

Figure 9.3: Network for exercise 9.3.2.

9.4 Minimum cut

9.4.1 Minimum cut with maximum flows (exams January and August 2015)

In the following graph, apply a maximum flow algorithm of your choice to find a minimum (s, t)-cut.

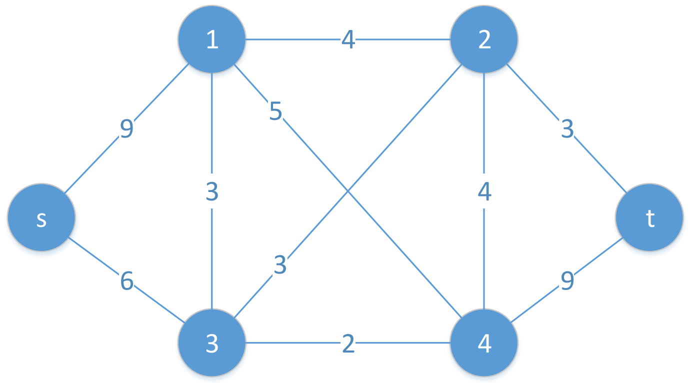

Figure 9.4: Network for exercise 9.4.1.

Solution. A minimum (s, t)-cut is $S = \{s\}$ and $T = \{1, 2, 3, 4, t\}$.

9.4.2 Minimum cut with maximum flows (exam August 2013)

1. In the following graph, determine a maximum flow from s to t using one of the algorithms studied in this course. The values indicated near the edges are their capacities.

2. Prove that your solution is optimum by indicating a minimum (s, t)-cut.

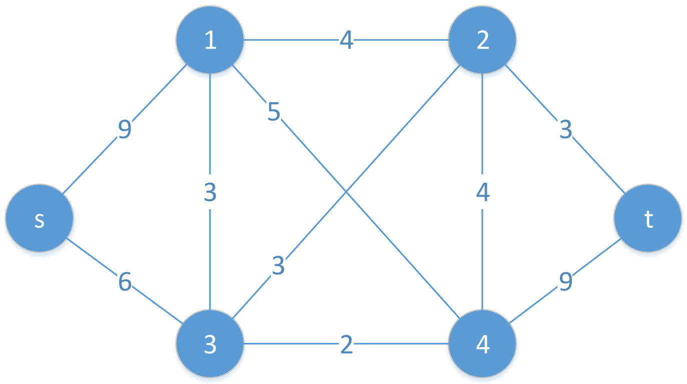

Figure 9.5: Network for exercise 9.4.2.

Chapter 10

Matching and assignment

10.1 Matching

10.1.1 Augmenting paths

1. Find two augmenting paths for the matching M shown below.

2. Derive the matching M' from M using the augmentation procedure and the two augmenting paths.

3. Is the new matching $M^{'}$ optimal? Prove it using the notion of augmenting paths.

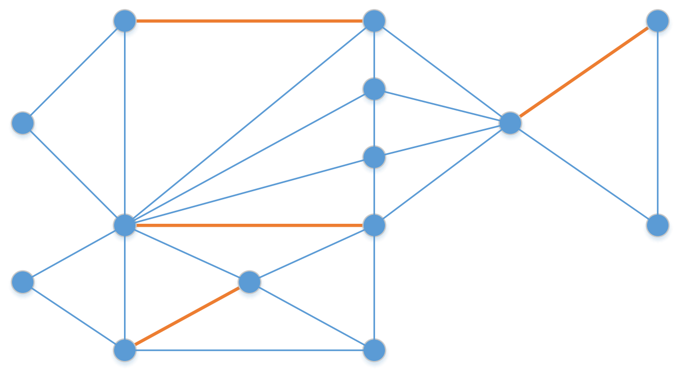

Figure 10.1: Graph for exercise 10.1.2.

10.1.2 Maximum cardinality matching for a bipartite graph

1. Find a matching of *maximum* cardinality in the following bipartite graph.

2. How can you prove the matching has maximum cardinality?

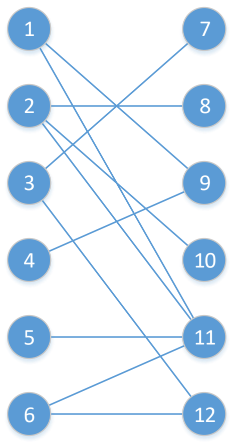

Figure 10.2: Graph for exercise 10.1.2.

Solution. A maximum matching is: 1–9, 2–8, 3–7, 5–11, 6–12. It is optimal as there are no remaining augmenting paths.

10.1.3 Maximum cardinality matching for a bipartite graph (exam January 2013)

1. Find a matching of maximum cardinality in the following bipartite graph.

2. Prove the matching is maximum.

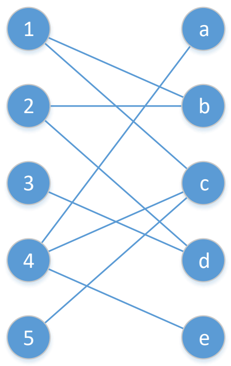

Figure 10.3: Graph for exercise 10.1.2.

10.2 Assignment

10.2.1 Assignment in complete graph

Find a maximal weight assignment in the complete graph where the edges have the following weights:

$$\mathbf{C} = \begin{pmatrix} 6 & 2 & 3 & 4 & 1 \\ 9 & 2 & 7 & 6 & 0 \\ 8 & 2 & 1 & 4 & 9 \\ 2 & 1 & 3 & 4 & 4 \\ 1 & 6 & 2 & 9 & 1 \end{pmatrix}.$$

Solution.

$$1 - 1', \qquad 2 - 3', \qquad 3 - 5', \qquad 4 - 4', \qquad 5 - 4'.$$

10.2.2 Inequality assignment (January 2016)

We want to solve the following (inequality) assignment problem:

$$\begin{array}{lll} \max & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} & \\ \text{s.t.} & \sum_{i=1}^{n} x_{ij} \leq 1, & \forall j \in [1, n]\,, \\ & \sum_{j=1}^{n} x_{ij} \leq 1, & \forall i \in [1, n]\,, \\ & x_{ij} \in \mathcal{B}, & \forall j \in [1, n]\,, \forall i \in [1, n]\,, \end{array}$$

where the costs c_{ij} are given by the following matrix:

$$\mathbf{C} = \begin{pmatrix} 4 & -2 & -2 & -1 \\ 7 & 2 & 1 & -3 \\ 2 & 3 & 0 & 1 \\ 5 & 3 & 3 & -1 \end{pmatrix}.$$

1. Solve the problem with all inequality constraints replaced by equality constraints, i.e. the following problem:

$$\begin{array}{lll} \max & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij} & \\ \text{s.t.} & \sum_{i=1}^{n} x_{ij} = 1, & \forall j \in [1,n]\,, \\ & \sum_{j=1}^{n} x_{ij} = 1, & \forall i \in [1,n]\,, \\ & x_{ij} \in \mathcal{B}, & \forall j \in [1,n]\,, \forall i \in [1,n]\,. \end{array}$$

2. Adapt the solution found to find another solution to the main problem that *increases* the objective function.