Topics in Algorithmic Game Theory and Economics: Background material

Pieter Kleer
Max Planck Institute for Informatics
Saarland Informatics Campus
pkleer@mpi-inf.mpg.de

November 9, 2020
Version 1

Abstract

This write-up contains all the relevant results from combinatorics, optimization and probability theory that will be used during the lectures of the course "Topics in Algorithmic Game Theory and Economics" taught by the author in the Winter Semester 2020/2021 at the Saarland Informatics Campus. It will be updated throughout the semester.

None of the sections represent (by far) a solid introduction to the respective area; we only discuss some definitions and results relevant to the course. If things are still unclear after reading this document, please brows through a standard text book in the respective area (some suggestions are given in the text), or send an e-mail to the address above.[1]

As not all statements are mathematically rigorous and self-contained, this document should not be used as reference beyond the homework sets and final exam.

[1] Also feel free to report any typos.

Contents

1 Linear optimization

A linear optimization problem, or linear program, consists of maximizing (or minimizing) a linear objective function subject to a number of linear constraints, for example

$$\begin{array}{ll} \max & x_1 + 3x_2 \\ \text{subject to} & x_1 + x_2 \leq 5 \\ & 3x_1 + x_2 \leq 2 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{array}$$

The notion of 'linearity' here refers to the fact that the objective function $x_1 + 3x_2$, as well as the constraints $x_1 + x_2 \leq 5, 3x_1 + x_2 \leq 2, x_1 \geq 0$ and $x_2 \geq 0$ depend linearly on the *variables* x_1 and x_2. Constraints that are not linear, such as $x_1^2 + x_1 x_2 \leq 1$, are called non-linear. More generally, a linear optimization problem can be represented in a canonical form:

$$\begin{array}{ll} \max & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array} \tag{1}$$

The matrix $A = (a_{ij})_{i=1,\ldots,m, j=1,\ldots,n} \in \mathbb{R}^{m \times n}$, the vector $b \in \mathbb{R}^m$, and the vector $c \in \mathbb{R}^n$ are known information. The goal is to find a vector $x \in \mathbb{R}^n$ that maximizes the objective function $c^T x = \sum_{i=1}^n c_i x_i$ subject to the constraints $Ax \leq b$ and $x \geq 0$. Furthermore, the expression $Ax \leq b$ is used as short hand notation for the m linear inequalities

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$$

for $i = 1, \ldots, m$. The same applies to the notation $x \geq 0$.[2] The collection of vectors $y \geq 0$ that satisfy $Ay \leq b$ is often referred to as the *feasible region* of the linear program, and the vectors y themselves are called *feasible solutions*.

1.1 Polyhedra

A subset $P \subset \mathbb{R}^n$ is called a *polyhedron* if there exists an $m \times n$-matrix A and vector $b \in \mathbb{R}^m$ such that $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A subset P is *bounded* if there exists a constant $M \geq 0$ such that $P \subseteq [-M, M]^n$. We say that P is a *polytope* if it is a bounded polyhedron. The inequalities $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i$ are usually referred to as *half-spaces* (as each individual inequality divides $\mathbb{R}^n$ in two parts).

One important property of polyhedrons is that they define *convex* sets.

Definition 1.1 (Convexity)**.** *A subset $P \subset \mathbb{R}^n$ is called convex if for every $x, y \in P$ and $0 \leq \lambda \leq 1$ it holds that $\lambda x + (1-\lambda)y \in P$. For $\lambda \in (0,1)$, the vector $\lambda x + (1-\lambda)y$ is called a* convex combination *of x and y.*

Exercise 1.2. *Show that every polyhedron $P \subset \mathbb{R}^n$ is convex.*

[2] In fact, one can incorporate these latter constraints in A and b as well, by rewriting them as $-x \leq 0$ and then extending the matrix A and vector b appropriately.

Convexity plays an important role in linear optimization. It guarantees that if a feasible solution x is 'locally' optimal, then it is also globally optimal. Local optimality, very roughly speaking, means that there is a small neighbourhood around x, so that x maximizes the objective function $c^T x$ within this small neighborhood. From an algorithmic point of view, this means that if one can efficiently compute a local optimum, then one can also efficiently compute a solution to (1), i.e., a global optimum.

Exercise 1.3. *Let P be a polytope, $\epsilon > 0$ and $c \in \mathbb{R}^m$. For $x \in P$, define*

$$B_\epsilon(x) = \{y \in P : ||y - x||_2 \leq \epsilon\}.$$

Let c be a given objective vector. Show that if $x = argmax\{c^T y : y \in B_\epsilon\}$, then also $x = argmax\{c^T y : y \in P\}$.

1.2 Optimization over polytopes

A *vertex (or extreme point)* x of a polyhedron P is a vector that cannot be written as a convex combination of two vectors $x', x'' \in P$ where $x', x'' \neq x$. If all vertices of P are integral (i.e., integer-valued), we say that P is *integral*.

Interestingly, an optimal solution to (1) is always attained by at least one vertex of P (assuming P is non-empty). In particular, the following algorithmic result is of great importance.

Theorem 1.4 (Linear Programming)**.** *Consider the program (1) and assume that the feasible region $\{x : Ax \leq b, x \geq 0\}$ is non-empty and bounded. There is an algorithm for computing a vertex $x \in P$ maximizing $c^T x$. It runs in time polynomial in n, m and the representation size of the input A, b and c.*

The first algorithm that proves Theorem 1.4 is the so-called *ellipsoid method*. This algorithm was shown to run in polynomial time by Khachiyan in 1979. One of its drawbacks, despite its theoretical guarantee, is that it does not perform well in practice.

An alternative algorithm for linear programming is the simplex method introduced by Dantzig in 1947. This algorithm works very well in practice, but, unfortunately, it is known to have a worst-case exponential running time.[3]

Exercise 1.5. *For the linear program example given at the beginning of Section 1, draw the feasible region in the (x_1, x_2)-plane and determine all vertices. Which vertex maximize the objective? Next, change the objective function to $3x_1 + x_2$, and identify the vertices maximizing the objective.*

[3] As a side note, within the framework of *smoothed analysis*, the simplex method has been shown to converge quickly to an optimal solution by Spielman and Teng [6]. Smoothed analysis, very roughly speaking, studies the running time of algorithms after adding perturbations to the input data. The result of Spielman and Teng shows that 'bad' instances for the simplex method are very fragile, in the sense that a small perturbation to the input data turns it into an instance for which the simplex method is very efficient.

1.3 Duality

One important aspect of linear programs is the notion of its *dual program*. Given a *(primal) maximization problem* (with at least one feasible solution)

$$\begin{array}{rll} p^* = & \max & c^T x \\ & \text{subject to} & Ax \leq b \\ & & x \geq 0 \end{array}$$

its *dual minimization problem* is given by

$$\begin{array}{rll} d^* = & \min & b^T y \\ & \text{subject to} & A^T y \geq c \\ & & y \geq 0 \end{array} \tag{2}$$

One way to interpret the dual program is that it tries to find the best possible upper bound on the value $p^* = \max\{c^T x : Ax \leq b, x \geq 0\}$. We will illustrate this with an (informal) example.

Consider the primal program, given at the beginning of Section 1,

$$\begin{array}{ll} \max & x_1 + 3x_2 \\ \text{subject to} & x_1 + x_2 \leq 5 \\ & 3x_1 + x_2 \leq 2 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{array}$$

In terms of A, b and c, we have

$$A = \begin{pmatrix} 1 & 1 \\ 3 & 1 \end{pmatrix}$$

together with $b = (5, 2)$ and $c = (1, 3)$. Let $y = (y_1, y_2) \in \mathbb{R}^2$ and let x be any feasible point of the primal program. Suppose we multiply the inequality $x_1 + x_2 \leq 5$ with a factor y_1, and the inequality $3x_1 + x_2 \leq 2$ with a factor y_2. Adding up these inequalities, we get

$$y_1(x_1 + x_2) + y_2(3x_1 + x_2) \leq 5y_1 + 2y_2 = b^T y.$$

if we assume that both $y_1, y_2 \geq 0$. (This assumption gives rise to the constraint $y \geq 0$ in the dual program.) Note that the new inequality can be rewritten, in a somewhat cumbersome way, as $(A^T y)^T x \leq b^T y$. We would like to have that $b^T y$ is an upper bound on the value p^*. A sufficient condition for this is that $c \leq A^T y$ as it then follows that

$$c^T x \leq (A^T y)^T x \leq b^T y, \tag{3}$$

for any feasible x (that in particular satisfies $x \geq 0$).

To summarize, this shows that if $y \geq 0$ and $A^T y \geq c$, the value $b^T y$ is in fact an upper bound on p^*, as this reasoning holds for any feasible solution x of the primal, and so in particular for a maximizer. The inequality in (3) is known as *weak duality*.

One question that remains is if the dual program does in fact compute the smallest possible upper bound on the value p^*. This is true, and known as *strong duality*. That is, it holds that

$$p^* = \max\{c^T x : Ax \leq b, x \geq 0\} = \min\{b^T y : A^T y \geq c, y \geq 0\} = d^*. \tag{4}$$

In fact, any pair of optimal solutions (x, y) to the primal and dual program, respectively, satisfies the so-called *complementary slackness* conditions:

$$y_i(A_i x - b_i) = 0 \quad \text{and} \quad x_j((A^T)_j y - c_j) = 0, \tag{5}$$

for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Here, we use the notation A_i for the i-th row of A, and $(A^T)_j$ for the j-th row of A^T. This means, e.g., that if a dual variable $y_i > 0$, then its corresponding constraint $A_i x \leq b_i$ in the primal program must be *tight*, i.e., it holds with equality (rather than strict inequality).

Exercise 1.6. *Consider the primal program*

$$\begin{array}{ll} \max & x_1 + 2x_2 + x_3 + 3x_4 \\ \textit{subject to} & x_1 + x_2 + 3x_3 + x_4 \leq 10 \\ & 2x_1 + x_2 + 2x_3 + 2x_4 \leq 5 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{array}$$

Write down the dual program, and use the approach of Exercise 1.5 to determine an optimal solution to the dual program. Then, use the complementary slackness conditions in (5) to determine an optimal solution of the primal program.

1.4 Integer linear programming

An *integer linear program* consist of finding an integer-valued solution that maximizes a given linear objective function. Its canonical form is

$$\begin{array}{ll} \max & c^T x \\ \text{subject to} & Ax \leq b \\ & x \in \mathbb{Z}_+^n \end{array} \tag{6}$$

where $\mathbb{Z}_+ = \{0, 1, 2, 3, \ldots\}$ is the set of non-negative integers. Whereas linear programming can be solved in polynomial time (Theorem 1.4), integer linear programs are hard to solve in general. One of the (intuitive) reasons for this is that, in general, not all vertices of the polytope $\{x : Ax \leq b, x \geq 0\}$ are integral, and, hence, Theorem 1.4 cannot be used. Nevertheless, it turns out that many polytopes, in particular those whose integral points describe certain combinatorial objects, are in fact integral.

Example 1.7. *Let $G = (A \cup B, E)$ be a bipartite graph with node set $A \cup B$, where $|A| = |B| = n$, and $E \subseteq \{\{a, b\} : a \in A, b \in B\}$ with $|E| = m$. A perfect matching of G is a collection of edges $M \subseteq E$ such that every node in $A \cup B$ is adjacent to precisely one edge in M. That is, for every $c \in A \cup B$, there is precisely one edge $e \in M$ such that $e \cap \{c\} \neq \emptyset$.*

We can model the (perfect) matchings of G as the integral points of a suitable polytope. To this end, we define a variable x_{ab} for every edge $\{a, b\} \in A \cup B$ with the interpretation that for a given matching M, we have $x_{ab} = 1$ if $\{a, b\} \in M$, and $x_{ab} = 0$ otherwise. The polytope of interest is the polytope P given by the constraints

$$\sum_{b \in B : \{a,b\} \in E} y_{ab} = 1 \quad \forall a \in A,$$

$$\sum_{a \in A : \{a,b\} \in E} y_{ab} = 1 \quad \forall b \in B,$$

$$0 \leq y_{ab} \leq 1 \quad \forall \{a, b\} \in E.$$

We note that this system can equivalently be written as one of the form $Cy \leq d, y \geq 0$ for suitable C and d (i.e., in the canonical form introduced before). The first two constraints simply say that every node in A and B should be adjacent to precisely one edge of a perfect matching.

It is not hard to see that there is a one-to-one correspondence between the perfect matchings of G and the integral points in P. In fact, the integral points of P are precisely its vertices.

Exercise 1.8. *Show that every integral vector $x \in \{0, 1\}^m$ that is contained in the perfect matching polytope P is a vertex of P.*

Hint: Show that for every fractional feasible solution $x \in P$, there is a cycle of edges in G that have a fractional value in x. Then show that one can always perturb the values of x on this cycle in two directions, and conclude that therefore x can be written as a convex combination of two other feasible points from P (and hence is not a vertex).

1.5 Further reading

For a general treaty on linear programming, see, e.g., the book of Bertsimas and Tsitsiklis [1] or the books of Schrijver [4, 5]. There are also many lecture notes by experts available on the web.

2 Probability theory

A (discrete) probability distribution over a finite set Ω is a function $f : \Omega \to [0, 1]$ that maps every element $i \in \Omega$ to a non-negative real number $0 \leq p_i \leq 1$ such that $\sum_{i\in\Omega} p_i = 1$. A (discrete) random variable X with probability distribution f is, informally speaking, a random object that realizes state $x \in \Omega$ with probability $p_i = \mathbb{P}(X = i)$. The probability that an event $A \subseteq \Omega$ happens is denoted by $\mathbb{P}(A) = \sum_{i\in A} \mathbb{P}(X = i)$.

Example 2.1. *Consider a (fair) die represented by* $\Omega = \{1, 2, 3, 4, 5, 6\}$ *and* $p_i = \mathbb{P}(X = i) = 1/6$ *for* $i = 1, \ldots, 6$. *An event* A *could be the event that a throw turns up even, or that the sum of the eyes is greater or equal than* 7.

A simple, but often useful, inequality for bounding the probability that at least one of two events A and B happens is the *union bound*

$$\mathbb{P}(A \cup B) \leq \mathbb{P}(A) + \mathbb{P}(B).^{4}$$

More general, for events $A_1, \ldots, A_n$, we have

$$\mathbb{P}\left(\cup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} \mathbb{P}(A_i).$$

The expectation of a random variable X over a finite set $\Omega \subseteq \mathbb{R}$ is given by

$$\mathbb{E}(X) = \sum_{i\in\Omega} i \cdot \mathbb{P}(X = i).$$

A useful property of the expectation operator is that it is linear. That is, for random variables X and Y and $\alpha, \beta \in \mathbb{R}$, it holds that

$$\mathbb{E}(\alpha X + \beta Y) = \alpha \cdot \mathbb{E}(X) + \beta \cdot \mathbb{E}(Y).$$

Exercise 2.2. *Let* $\Omega = \mathbb{Z}_+$. *Show that* $\mathbb{E}[X] = \sum_{i\in\Omega} \mathbb{P}(X \geq i)$.

The variance of a random variable X is given by

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right].$$

Exercise 2.3. *Show, by writing out the definition, that* $Var(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

2.1 Concentration inequalities

Concentration inequalities can be used to bound the probability that (non-negative) random variables take on large values. The simplest one is Markov's inequality.

Lemma 2.4 (Markov's inequality)**.** *Let* $X \geq 0$ *be a non-negative random variable. For any* $t > 0$ *it holds that*

$$\mathbb{P}[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

[4] Its proof follows from the simple fact that $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) \leq \mathbb{P}(A) + \mathbb{P}(B)$.

Exercise 2.5. *Prove Markov's inequality.*

We continue with Chebyshev's inequality, which can be proven by applying Markov's inequality on the variable $Y = (X - \mathbb{E}[X])^2$.

Lemma 2.6 (Chebyshev's inequality)**.** *Let X be a random variable. For any $t > 0$, we have*

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq t] \leq \frac{Var[X]}{t}.$$

The last, and most involved concentration inequality, is Chernoff's bound. For this we need the notion of independence of random variables. Random variables $X_1, \ldots, X_n$ defined on a finite set Ω, respectively, are said to be *independent* if

$$\mathbb{P}[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n] = \prod_{i=1}^{n} \mathbb{P}[X_i = x_i].$$

Exercise 2.7. *Show that if $X_1, \ldots, X_n$ are independent, then*

$$\mathbb{E}[X_1 \cdot X_2 \cdot \cdots \cdot X_n] = \prod_{i=1}^{n} \mathbb{E}[X_i].$$

Lemma 2.8 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables with $\mathbb{P}[X_i = 1] = p_i$ and $\mathbb{P}[X_i = 0] = 1 - p_i$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X] = \sum_i p_i$. Then for any $t \in (0, 1)$, it holds that*

$$\mathbb{P}[|X - \mu| \geq t \cdot \mu] \leq 2 \exp\left(-\frac{\mu \cdot t^2}{3}\right).$$

There are many different versions available in the literature similar to the Chernoff bound above (in particular some stronger than the one stated here).

2.2 Further reading

For an introduction into probability theory, see, e.g., the book of Blitzstein and Hwang [2].

3 Matroids

Matroids are combinatorial objects that generalize, e.g., the idea of linearly independent vectors in $\mathbb{R}^n$ (or any other vector space for that matter). There are many natural combinatorial objects that can be seen as a matroid. For a finite set A and element e, we use the shorthand notation $A + e$ for $A \cup \{e\}$. Furthermore, if $f \in A$, we write $A - f$ for $A \setminus \{f\}$.

A *matroid* $\mathcal{M} = (E, \mathcal{I})$ is given by a finite ground set of elements $E = \{e_1, \ldots, e_m\}$ and a non-empty collection of subsets $\mathcal{I} \subseteq 2^E = \{X : X \subseteq E\}$ that satisfy the following conditions.

i) (Downward-closed property) If $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$.

ii) (Augmentation property) If $X, Y \in \mathcal{I}$ and $|X| > |Y|$ then there exists an $e \in X \setminus Y$ such that $Y + e \in \mathcal{I}$.

The first property states that the *set system* $\mathcal{I}$ is closed under taking subsets, and the second property states that if we have two sets of which one has strictly larger size, then we can add an element from the larger set to the smaller set, and again get a set in $\mathcal{I}$. That is, we can augment the smaller set with an element from the larger set.

Any non-empty collection $\mathcal{I}$ of subsets of E that satisfies the downward-closed property (but not necessarily the augmentation property) is usually referred to as an *independent set system*, and the sets in $\mathcal{I}$ are called *independent sets*.

Example 3.1 (Linear matroid)**.** *Consider a non-empty collection of vectors $E = \{e_1, \ldots, e_m\}$ with $e_i \in \mathbb{R}^n$ for $i = 1, \ldots, m$. Define a subset $X \subseteq E$ independent, i.e., $X \in \mathcal{I}$, if and only if the vectors in X are linearly independent. That is, if*

$$\sum_{i \in X} \gamma_i \cdot e_i = 0$$

for real coefficients γ_i, then they must all be zero.

Exercise 3.2. *Show that the linear matroid in fact satisfies downward-closedness and the augmentation property.*

Example 3.3 (Graphic matroid)**.** *Let $G = (V, E)$ be an undirected graph. A subset $X \subseteq E$ of edges is said to be independent, i.e., $X \in \mathcal{I}$, if and only if the subgraph formed by the edges of X is acyclic. That is, it does not contain a cycle. Such subgraphs are usually referred to as forests in the graph theory literature.*

It is easy to see that the downward-closed property holds: If any subset of edges of X would contain a cycle, then X itself would also contain a cycle, which is a contradiction with X being independent. For the augmentation property, one can look at the connected components of Y. Since X has the property that $|X| > |Y|$ there must be at least one edge $e \in X \setminus Y$ that connects two connected components of Y (check this yourself). Adding this edge to X does not create a cycle.

A non-example of a matroid, which we give here in order to avoid confusion of the terminology, are independent (node) sets of a given undirected graph $G = (V, E)$. A set of

nodes $X \subseteq V$ is said to be independent (in the graph theory literature) if no two nodes in X are adjacent in G, i.e., there is no edge in G between any two nodes in X. (So here the nodes V would be the ground set of elements.)

Although the downward-closedness property is satisfied, it can be shown that the augmentation property does not hold: Simply consider the graph $G = (V, E)$ with $V = \{a, b, c\}$ and $E = \{\{a, b\}, \{b, c\}\}$. Then $\{b\}$ forms an independent set, as well as a and c together (as the edge $\{a, c\}$ is not present), but we cannot augment $\{b\}$ with either a or c.

3.1 Bases

One special collection of independent sets of a given matroid $\mathcal{M} = (E, \mathcal{I})$ are the set of *bases* $\mathcal{B}$. These are the maximal independent sets of $\mathcal{I}$. We say that a set $X \in \mathcal{I}$ is *maximal* if there does not exist a $Y \in \mathcal{I}$ with $X \subseteq Y$ and $Y \neq X$. Said differently, X cannot be augmented with another element and still be independent. We next describe some properties of bases.

Proposition 3.4. *All bases in $\mathcal{B}$ have the same cardinality (or size).*

Exercise 3.5. *Prove Proposition 3.4.*

Example 3.6 (Bases of graphic matroid)**.** *Assuming that the graph $G = (V, E)$ is connected, the bases of the graphic matroid are the* spanning trees *of G. A spanning tree is a subgraph consisting of one connected component, the whole node set V, that does not contain any cycle.*

Alternatively, a spanning tree can be seen as a subgraph of G in which there is a unique path between any two nodes in V, or as a subgraph with a path between any two nodes and no cycles. (See also Exercise 3.12 for two example of spanning trees.)

Proposition 3.7 (Exchange property)**.** *Let $B, B' \in \mathcal{B}$. Then for every $e \in B \setminus B'$, there exists an $e' \in B' \setminus B$ such that $B - e + e' \in \mathcal{B}$.*

Exercise 3.8. *Prove Proposition 3.7.*

There actually exists a stronger form of the exchange property, which is a bit harder to prove.

Proposition 3.9 (Strong exchange property)**.** *Let $B, B' \in \mathcal{B}$. Then for every $e \in B \setminus B'$, there exists an $e' \in B' \setminus B$ such that $B - e + e', B' + e - e' \in \mathcal{B}$.*

Exercise 3.10. *Prove Proposition 3.9 for the graphic matroid in Example 3.3.*

For a given independent set $X \in \mathcal{I}$, we define the directed graph $D(X) = (E, A(X))$. Note that its nodes are the elements of the ground set E. We have

$$A(X) = \{(y, z) : y \in X, z \in E \setminus X, X - y + z \in \mathcal{I}\}.$$

That is, we have a directed arc from y to z in D if and only if we can exchange y with z in X and still obtain an independent set. A useful property of the digraph D is stated below.

Proposition 3.11. *For any $B, B' \in \mathcal{B}$, the directed graph $D(B)$ contains a perfect matching*[5] *on the nodes in $B \Delta B'$, i.e., the nodes (elements in E) that form the symmetric difference of B and B'.*

Exercise 3.12. *Consider the graph G in Figure 1 with the two indicated spanning trees B and B'. Remember that spanning trees are the bases of the graphic matroid, as explained in Example 3.6.*

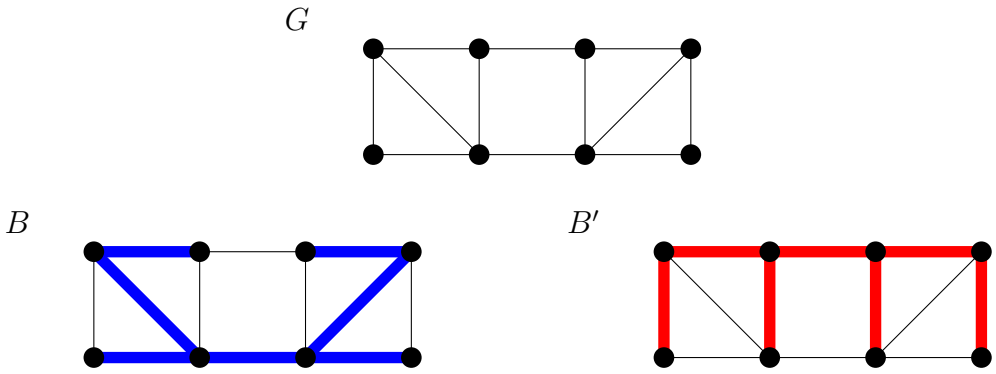

Figure 1: The graph and spanning trees of Exercise 3.12

Construct (or draw) the graph $D(B)$ and indicate a perfect matching on $B \Delta B'$.

3.2 Greedy algorithm

An important problem for an arbitrary (downward closed) independent set system $(E, \mathcal{I})$[6] with $E = \{e_1, \ldots, e_m\}$ is the following (linear) optimization problem for a given objective vector $c \in \mathbb{R}^m$ with $c_1 \geq c_2 \geq \cdots \geq c_m$.[7]

$$\begin{array}{ll} \max & \sum_{i \in X} c_i \\ \text{subject to} & X \in \mathcal{I} \end{array} \tag{7}$$

One straightforward way of trying to solve this problem is the *greedy algorithm* depicted in Algorithm 1. It tries to find an optimal solution by 'greedily' selecting elements, i.e., it first tries to select the element with the highest value, then the second highest value, and so on. Perhaps somewhat surprisingly, it turns out that the greedy algorithm returns an optimal solution if the independent set system is a matroid. In fact, it turns out that matroids characterize the class of independent set systems for which the greedy algorithm returns an optimal solution.

Theorem 3.13. *The greedy algorithm returns an optimal solution to (7) for any objective vector $c \in \mathbb{R}^m$ if and only if $(E, \mathcal{I})$ is a matroid.*

Exercise 3.14. *Prove that the greedy algorithm returns an optimal solution when $(E, \mathcal{I})$ is a matroid.*

[5] A perfect matching P on a subset $A \subseteq E$ in D is a collection of directed arcs P such that every node in A is the head or tail of precisely one arc in P.

[6] That does not necessarily satisfy the augmentation property.

[7] This assumption is without loss of generality.

ALGORITHM 1: Greedy algorithm

Input : Objective function $c = (c_1, \ldots, c_m)$ with $c_1 \geq c_2 \geq \cdots \geq c_m$ and (downward closed) independent set system $(E, \mathcal{I})$.

Output: Independent set $X \in \mathcal{I}$.

```
Set X = ∅.
for i = 1, ..., m do
    if X + e_i ∈ I then
        Set X ← X + e_i
    end
end
```

3.3 Further reading

For an extensive treaty of matroids, see, e.g., the book of Oxley [3] or Schrijver [5].

References

[1] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[2] Joseph K Blitzstein and Jessica Hwang. *Introduction to probability*. Crc Press, 2019.

[3] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

[4] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[5] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[6] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.