NOPT048 Linear Programming and Combinatorial Optimization*

Notes by

Marcel K. Goh (Prague, Czech Rep.)

9 June 2019

1. DEFINITIONS AND THEOREMS

1.1. Linear and integer programs

The following is a *linear program*:

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b. \end{array}$$

Here $x, b, c \in \mathbf{R}^n$ and A is an $m \times n$ matrix over $\mathbf{R}$. The function $c^T x$ is called the *objective function* and it is what the linear program is trying to maximise (or minimise). Any vector x that satisfies the inequality $Ax \leq b$ is called a *feasible solution*. A vector x^* is called an *optimum solution* if $c^T x^* \geq c^T x$ for all feasible solutions x. A linear program is called *infeasible* if there are no feasible solutions. Even if a linear program is feasible, it might be *unbounded*. This happens when its objective function can be made arbitrarily "good".

In a certain sense, linear programs are easy to solve. Adding an integrality constraint to a linear program gives an *integer program*:

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b, \\ & x \in \mathbf{Z}^n. \end{array}$$

Integer programs are much harder to solve in the general case. Suppose we want to solve the following integer program:

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b, \\ & x \in \{0, 1\}. \end{array}$$

It is often a good strategy to solve the *linear program relaxation* of the integer program, which gives an approxmiate answer:

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b, \\ & 0 \leq x \leq 1. \end{array}$$

Then we can take the solution of the linear program and round each element to the nearest integer value to get a solution of the original integer program. In some cases, the linear program relaxation gives quite a good answer. For example, if G is a graph with minimum vertex cover of size n, then formulating an integer program and solving its linear program relaxation will spit out a vertex cover S with $|S| \leq 2n$.

1.2. Geometric interpretation

The inequality $Ax \leq b$ defines a *polyhedron*. More precisely, we say that a polyhedron is the intersection of finitely many *halfspaces*, where a halfspace is the set $\{x : a^T x \leq b\}$ for some $a \in \mathbf{R}^n$ and $b \in \mathbf{R}$. If a polyhedron is bounded, we call it a *polytope*. A polyhedron is closed and convex. The finite intersection of closed sets is closed and the intersection of convex sets is convex.

Let $S \subset \mathbf{R}^n$ be a a set of m points. Then the *convex hull* of S is the set

$$\operatorname{conv} S = \{x : x = \sum_{i=1}^{m} \lambda_i s_i, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0\}.$$

* Course given by Prof. Hans Raj Tiwary at Charles University in Prague

The *conic hull* of S is the set

$$\operatorname{cone} S = \{x : x = \sum_{i=1}^{m} \lambda_i s_i, \lambda_i \geq 0\}.$$

The *affine hull* of S is the set

$$\operatorname{aff} S = \{x : x = \sum_{i=1}^{m} \lambda_i s_i, \sum_{i=1}^{m} \lambda_i = 1\}.$$

The convex hull can be obtained from the conic hull by intersecting it with a hyperplane. The affine hull is the smallest-dimensional subspace that contains all the points in S.

Let $P = \{x : Ax \leq b\}$ be a polyhedron. An inequality $\alpha^T x \leq \beta$ is said to be a *valid inequality* if and only if for all $y \in P$, $\alpha^T y \leq \beta$. Then a subset $F \subset P$ is a *face* of P if and only if there exists a valid inequality $\alpha^T x \leq \beta$ such that $F = P \cap \{x : \alpha^T x \leq \beta\}$. The trivial faces of any polyhedron P are $\emptyset$ and P itself (formed by intersecting P with the valid inequalities $0^T x \leq 1$ and $0^T x \leq 0$ respectively). Any face that is not trivial is called *proper*.

The *dimension* of a polyhedron P, denoted $\dim P$, is the largest dimension of a ball inside P. A *vertex* is a face of dimension 0, an *edge* is a face of dimension 1, and a *facet* is a face of dimension $\dim P - 1$. A point $x \in P$ is called an *extreme point* if it is not a convex combination of points of P. Note that a point x is an extreme point if and only if it is a vertex.

Theorem M (*Minkowski-Weyl*). *A set of points P is a polyhedron if and only if there exist finite sets V and Y such that $P = \operatorname{conv} V + \operatorname{cone} Y$ where the sum $A + B$ denotes the set $\{a + b : a \in A, b \in B\}$.*

1.3. Simplex method

A polyhedron is *pointed* if it does not contain a line.

Lemma. *Every non-empty pointed polyhedron P has at least one vertex.*

Theorem. *Consider the linear program*

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b. \end{array}$$

If the polyhedron P defined by $Ax \leq b$ is pointed and if an optimum exists, then there exists a vertex of P that attains the maximum.

The result of the above statements is an algorithm for solving linear programs, called the *simplex method*. Here is the general idea:

Algorithm X (*Simplex method*). Find an optimum solution of a linear program.

X1. Find an initial vertex. It is a 0-dimensional face with some 1-dimensional faces incident on it.

X2. At any vertex, check for optimality. If optimal, we are done.

X3. If the vertex is not optimal, increase the value of the objective function by following an edge to a new vertex. Return to step X2. ∎

Let A be an $m \times n$ matrix, $b \in \mathbf{R}^m$, $x \in \mathbf{R}^n$, with $n \geq m$. Let $B \subset [n] = \{1, \ldots, n\}$ of cardinality m. Then A_B denotes the square matrix formed by selecting the columns of A indexed by the elements of B and likewise x_B denotes the vector of length m formed by selecting the elements indexed by the elements of B. We define a vector x to be a *basic feasible solution* of the linear program

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$

if and only if there exists $B \subset [n]$ with $|B| = m$ such that:

i) A_B is non-singular;

ii) $x_j = 0$ for all $j \notin B$;

iii) $x_B = A_B^{-1} b$.

A set $B \subset [n]$ is called a *basis* if and only if A_B is non-singular, and B is a *feasible basis* if the unique solution of $A_B x_B = b$ is non-negative. Note that a basic feasible solution is uniquely determined by the feasible basis B, but a vertex does *not* have to be determined by a unique basis.

A matrix A is *totally unimodular* if and only if every square submatrix of A has determinant 0, 1, or -1.

1.4. Duality

As always, we start with the linear program

$$\begin{array}{ll} \text{maximise} & c^T x \\ \text{subject to} & Ax \leq b, \\ & x \geq 0. \end{array}$$

Inequalities of a linear program can be added together without changing the feasible region. Likewise, one can also multiply any inequality by a non-negative scalar. So we can manipulate inequalities to get an upper bound on the optimum value of the objective function. We want to pick coefficients to get the *least* upper bound possible. Let y denote a vector of coefficients. We want to

$$\begin{array}{ll} \text{minimise} & b^T y \\ \text{subject to} & A^T y \geq c, \\ & y \geq 0. \end{array}$$

The original linear program is called the *primal* and this new one is called the *dual*. Any feasible solution of the dual linear program gives an upper bound on the primal and the optimum of the dual is equal to the optimum of the original program.

Theorem W (*Weak Duality Theorem*). *Let P and D be primal and dual linear programs of the above form and let x and y be respective feasible solutions. Then $c^T x \leq b^T y$.*

Theorem S (*Strong Duality Theorem*). *Let P and D be primal and dual linear programs. Then exactly one of the following holds:*

i) P infeasible, D infeasible

ii) P infeasible, D unbounded

iii) P unbounded, D infeasible

iv) P and D both feasible and bounded and furthermore, if x^* and y^* are primal and dual optimal solutions, then $c^T x^* = b^T y^*$.

Lemma F (*Farkas lemma*). *For an arbitrary matrix A and vector b, exactly one of the following holds:*

i) There exists a vector x such that $Ax = b$ and $x \geq 0$.

ii) There exists a vector y such that $A^T y \geq 0$ and $b^T y < 0$.

Lemma G (*Farkas lemma 2.0*). *For an arbitrary matrix A and vector b, exactly one of the following holds:*

i) There exists a vector x such that $Ax \leq b$ and $x \geq 0$.

ii) There exists a vector y such that $A^T y \geq 0$, $y \geq 0$ and $b^T y < 0$.

1.5. Applications

1.5.1. Matching and vertex cover in bipartite graphs

Given a graph $G = (V, E)$, a *matching* is a set $M \subset E$ such that for all $v \in V$ there is at most one edge $e \in M$ with $v \in e$. The linear program that finds the maximal matching in a bipartite graph is

$$\begin{array}{lll} \text{maximise} & \displaystyle\sum_{e \in E} x_e & \\ \text{subject to} & \displaystyle\sum_{v \in e} x_e \leq 1 & \text{for all } v \in V, \\ & x_e \geq 0 & \text{for all } e \in E. \end{array}$$

The dual of this linear program is the following:

$$\begin{array}{ll} \text{minimise} & \displaystyle\sum_{v\in V} y_v \\ \text{subject to} & y_u + y_v \geq 1 \quad \text{for all } \{u,v\} \in E, \\ & y_v \geq 0 \qquad \text{for all } v \in V. \end{array}$$

Given a graph $G = (V, E)$, a *vertex cover* is a set $S \subset V$ such that for all $\{u, v\} \in E$, $\{u, v\} \cap S \geq 1$. The dual linear program above finds the minimum vertex cover in a bipartite graph. The duality of these two programs results in Kőnig's theorem.

Theorem K (*Kőnig's theorem*). *Suppose $G = (V, E)$ is a bipartite graph. Let $M \subset E$ denote the maximum matching and let $S \subset V$ denote the minimum vertex cover. Then $|M| = |S|$.* ∎

1.5.2. Matching in weighted graphs

Let $G = (V, E)$ be a graph (not necessarily bipartite) with a weight function $w : E \to \mathbf{R}^+$. Suppose we want to find the maximum weight matching in G. For any subset $E' \subset E$, the *characteristic vector* of E' is the vector $\chi^{E'}$ given by

$$\chi_e^{E'} = \begin{cases} 1 & \text{if } e \in E' \\ 0 & \text{if } e \notin E' \end{cases}$$

for all $e \in E$. Then the *matching polytope* of the graph G is the set

$$M(G) = \operatorname{conv}\{x \in \{0,1\}^{|E|} : x \text{ is the characteristic vector of some matching in } G\}.$$

To get the maximum weight matching, we need only solve the linear program

$$\begin{array}{ll} \text{maximise} & w^T x \\ \text{subject to} & x \in M(G). \end{array}$$

What does the polytope $M(G)$ look like in terms of inequalities? It turns out that $M(G)$ is the set of all vectors x that satisfy

1. $\displaystyle\sum_{v\in e} x_e \leq 1$ for all $v \in V$ and
2. $\displaystyle\sum_{e\in E[U]} x_e \leq \frac{|U|-1}{2}$ for every set $U \subset V$ of odd cardinality.

In a graph $G = (V, E)$, a matching $M \subset E$ is *perfect* if every vertex in V is incident to some edge in M. The perfect matching polytope is the set

$$PM(G) = \operatorname{conv}\{x \in \{0,1\}^{|E|} : x \text{ is the characteristic vector of some perfect matching in } G\}.$$

For a set $U \subset V$ of vertices, let $\delta(U)$ denote the number of edges that have one endpoint in U and one endpoint outside U. Then we find that $PM(G)$ is the set of all vectors x satisfying

1. $\displaystyle\sum_{v\in e} x_e = 1$ for all $v \in V$,
2. $\displaystyle\sum_{e\in\delta(U)} x_e \geq 1$ for every set $U \subset V$ of odd cardinality, and
3. $x_e \geq 0$ for every $e \in E$.

Note that the number of inequalities in these polytopes may be very large. For this reason, a polynomial-time algorithm may not exist.

1.5.3. The travelling salesman problem

Given a graph $G = (V, E)$, a *closed tour* is a walk in G with no repeated edges and no repeated vertices except for the first and last vertex, which are the same. In a graph with edge costs $(c_e : e \in E)$, *travelling salesman problem* asks us to find the minimum-weight closed tour that contains each vertex exactly once. To solve the travelling salesman problem, we start with the following linear program:

$$\begin{array}{ll} \text{minimise} & \displaystyle\sum_{e \in E} c_e x_e \\ \text{subject to} & \displaystyle\sum_{v \in e} = 2 \text{ for all } v \in V, \\ & 0 \leq x_e \leq 1 \text{ for all } e \in E. \end{array}$$

What we really want to do is add the inequalities $\sum_{e \in \delta(S)} x_e \geq 2$ for every $\emptyset \neq S \neq V$. This is called *subtour elimination* because it forbids any tours smaller than the entire closed tour we're trying to build. Adding this inequality, we get the linear program relaxation of the integer program for the travelling salesman problem. The problem is that eliminating subtours adds an exponential number of ineqalities to the program. Instead of adding all of these inequalities, we approach the problem as follows:

Algorithm T (*Subtour elimination*). This algorithm solves the travelling salesman problem by adding violated subtour constraints one at a time.

T1. We start with the linear program without the subtour constraints.

T2. Find the optimal solution to the linear program.

T3. If we find that some subtour constraints are violated by the optimal solution, we add these inequalities to our program and return to step T2. If no subtour constraints are violated, we have a solution. ∎

Step 3 is our main concern. How do we figure out if there are any violated subtour constraints? Suppose we are at a certain step of the algorithm and the linear program has just given us an optimal solution x^*. Let $G(x^*)$ denote the graph with vertices V and edges $\{e : x_e^* > 0\}$. Without loss of generality, assume $G(x^*)$. Set capacities $u_e = x_e^*$ for every edge e. Now for any $S \subset V$, the weight of the cut $(S, V \setminus S)$ equals the value of $\sum_{e \in \delta(S)} x_e$. So we can solve the min-cut problem over $G(x^*)$ and if we get a value less than 2, we know that some subtour constraint is violated. Of course, we need some heuristics to decide which constraints to add, so there is still no guarantee of polynomial-time termination.

1.6. Ellipsoid algorithm

So far, we do not know if linear programs can be solved in polynomial time. It turns out that optimisation is polynomially equivalent to separating over a polytope. Given $x \in \mathbf{R}^n$ and a polyhedron P, we separate as follows: Output "yes" if $x \in P$; otherwise, output "no" and a pair (α, β) such that $\alpha^T y \leq \beta$ is a valid inequality for P and $\alpha^T x > \beta$. The *ellipsoid algorithm* uses an oracle for separation to answer the question of feasibility: Given a polytope P, find an $x \in P$ or conclude that $P = \emptyset$.

Before we get into the actual method, let's convince ourselves that checking for feasibility is equivalent to optimising. That optimisation implies feasibility is obvious. So let's suppose we have a method to check if a linear program is feasible. Can we use that to optimise the linear program

$$\begin{array}{ll} \text{minimise} & c^T x \\ \text{subject to} & Ax \leq b, \\ & x \geq 0? \end{array}$$

Well, from duality we know that if an optimum exists, it will also be the optimum value of the dual linear program. So we can construct a polyhedron that contains only this value and then check if it is feasible. So we check if the polyhedron

$$\{\begin{pmatrix} x \\ y \end{pmatrix} : c^T x = b^T y, Ax \leq b, A^T y \geq c, x \geq 0, y \geq 0\}$$

is feasible and that will give us the optimum value of x, if it exists.

On to ellipsoids. An *ellipsoid* $E(A, a)$ is the set $\{x \in \mathbf{R}^n : (x-a)^T A^{-1}(x-a) \leq 1\}$ where A is a *positive definite* matrix. This means that $(x-a)^T A^{-1}(x-a)$ is positive and moreover, $A = M \cdot M$ for some square matrix M. If we let B denote the unit ball in $\mathbf{R}^n$, then $E(A, a) = \{Mx + a : x \in B\}$.

For simplicity, assume that P is bounded and assume that P is either empty or full-dimensional. Assume also that we have an oracle that outputs a separating hyperplane. Then the *ellipsoid algorithm* determines if $P = \{x : Ax \leq b\}$ is feasible.

Algorithm E (*Ellipsoid algorithm*). This algorithm presupposes that we have a separation oracle. Given a bounded polytope P and assuming, for the sake of simplicity, that P is either empty or full-dimensional, the algorithm determines either outputs a point c in P or concludes that P is empty.

E1. Start with an ellipsoid large enough that it definitely contains a point in the polytope, if one exists.

E2. Let c be the centre of the ellipsoid. Invoke the separation oracle. If $c \in P$, output c and halt.

E3. If we have reached the maximum amount of iterations, output $P = \emptyset$.

E4. Otherwise, the oracle gives us a separating hyperplane that cuts the ellipsoid. Determine which half of the ellipsoid P lies, then draw a new, smaller ellipsoid that contains this half. Return to step E2. ∎

The maximum number of iterations is calculated using a function polynomial in the size of the linear program, so this algorithm runs in polynomial time.

1.7. Matroids and greedy algorithms

Given a graph $G = (V, E)$, a *spanning tree* is a set $T \subset E$ such that every vertex is incident to some edge in T and $G = (V, T)$ is a contains no cycles. Kruskal's algorithm obtains the minimum weight spanning tree by maintaining a forest F. If F spans every vertex, then it is a spanning tree. If not, add the minimum weight edge e such that $F \cup \{e\}$ does not contain a cycle and repeat.

Let $G = (V, E)$ be a graph and let F denote the set of all forests in G. Then (E, F) is called the *graphic matroid* of G. In fact, there are lots of different types of matroids. A tuple $M = (E, I)$ where $I \subset \mathcal{P}(E)$ is a *matroid* if

i) $\emptyset \in I$,

ii) If $X \subset Y$ and $Y \in I$, then $X \in I$,

iii) If there exist $X, Y \in I$ such that $|Y| > |X|$, then there exists some $y \in Y \setminus X$ such that $X \cup \{y\} \in I$.

A system that satisfies i) and ii) is called an *independence system*. Associated with any independence system $M = (E, I)$ is its *rank function* $r : \mathcal{P}(E) \to \mathbf{R}$ defined by

$$r(X) = \max\{|Y| : Y \subset X, Y \in I\}.$$

We call $r(E)$ the *rank* of M. A set $S \in I$ such that $|S| = r(E)$ is called a *base* of M. If $M = (E, I)$ is a matroid, then the properties of its rank function r are:

1. $0 \leq r(X) \leq |X|$ for all $x \subset E$.
2. If $X \subset Y$ then $r(X) \leq r(Y)$ for all $X, Y \subset E$.
3. $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$ for all $X, Y \subset E$.

Property 3 above is called *submodularity*. If r is the rank function of a general independence system, it need only satisfy properties 1 and 2, as well as the weaker property $r(X \cup Y) \leq r(X) + r(Y)$.

Matroids can be directly characterised by the rank function.

Theorem R (*Rank characterisation of matroids*). *Let E be a finite set and suppose that $r : \mathcal{P}(E) \to \mathbf{R}$ satisfies the three peoperties of the rank function. Then $M = (E, I)$, with*

$$I = \{Y \subset E : |Y| = r(Y)\},$$

is a matroid with r as its rank function.

To further understand the link between spanning trees and matroids, note that Kruskal's algorithm is an instance of the *weighted greedy algorithm*. Let $M = (E, I)$ be a matroid with a weight function c defined on edges. The algorithm proceeds as follows:

Algorithm G (*Greedy algorithm*). Given a matroid $M = (E, I)$ with a weight function c defined on edges, this algorithm finds the set $S \in I$ with maximum weight.

G1. Set $S_0 \leftarrow \emptyset$, and E is the element set of M. Set $i \leftarrow 0$.

G2. Select $e \in E$ with maximum $c(e)$. Set $E \leftarrow E \setminus \{e\}$.

G3. If $S_i \cup \{e\} \in I$, set $S_{i+1} \leftarrow S_i \cup \{e\}$, $i \leftarrow i + 1$.

G4. If $E = \emptyset$, output $S = S_i$. Otherwise, return to step 2. ▮

The following two theorems establishes a certain equivalence between matroids and problems that can be solved with the greedy algorithm.

Theorem. *Let $M = (E, I)$ be a matroid with rank function r. The weighted greedy algorithm finds the maximum weighted independent set S_k of cardinality k for each $1 \leq k \leq r(E)$.* ▮

Theorem G (*Greedy equivalence*). *Let $M = (E, I)$ be an independence system. Then if the weighted greedy algorithm works for all (non-negative) weights, then M is a matroid.* ▮

It is worth it to note that spanning trees and matroids have corresponding polytopes. For a graph $G = (V, E)$, the spanning tree polytope $ST(G)$ is the set of all vectors $x \in \mathbf{R}^{|E|}$ that satisfy

1. $x_e \geq 0$,
2. $\sum_{e \in E} x_e = |V| - 1$, and
3. $\sum_{e \in E[S]} x_e \leq |S| - 1$, for every proper subset S of V with $|S| \geq 2$.

For a matroid $M = (E, I)$ with rank function r, the matroid polytope P_M is the set of all vectors $x \in \mathbf{R}^{|E|}$ satisfying $\sum_{e \in T} x_e \leq r(T)$ for all $T \subset E$, where $x_e \geq 0$.

2. PROBLEMS AND PROOFS

Problem 1. *Describe how to find an initial basic feasible solution for an arbitrary LP in standard (equational) form.*

Solution. Suppose the constraints are $Ax = b, x \geq 0$. Examine the matrix A. We want to create an identity submatrix. If one already exists, we use that as an initial basis. If not, for any row that doesn't have a unique variable with coefficient 1, introduce a new variable. So we introduce new variables $y_1, \ldots, y_k$. Then all these new variables form an initial basis of an auxiliary linear program. We minimise the sum over all y_i. Optimising over this minimisation linear program gives us a basis for our original linear program. ▮

Problem 2. *Write an integer program formulation for the matching/perfect matching problem.*

Solution. The integer program corresponding to the matching problem in a graph $G = (V, E)$ with a cost vector c corresponding to edges.

$$\begin{aligned} \text{maximise} \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{v \in e} x_e \leq 1 \quad \text{for all } v \in V, \\ & x_e \in \{0, 1\}. \end{aligned}$$

For a perfect matching, change the inequality for each vertex to equality. ▮

Problem 3. *Prove that the linear program relaxation of the standard integer program for matching/perfect matching is integral for bipartite graphs.*

Proof. Let $LP(G, c)$ denote the linear program relaxation of the perfect matching integer program over a graph $G = (V, E)$ with weights c. If $LP(G, c)$ is infeasible, then G has no perfect matching. For a vector x, let $I(x)$ denote the set $\{i : x_i \in \{0, 1\}\}$. If $LP(G, c)$ is feasible, there exists at least one optimal solution. Let

x^* an optimal solution that maximises the number of integer elements, i.e. $|I(x^*)| \geq |I(x)|$ for all optimal solutions x. We want to show that $|I(x^*)| = |E|$.

Suppose, towards a contradiction, that $|I(x^*)| < |E|$. Then there exists an i such that $0 < x_i < 1$. It must correspond to some edge $e_1 = \{a_1, b_1\}$ in the graph G. Since the sum at each vertex is 1, there must be another edge e_2 leaving b_1 with $x^*_{e_2}$ non-integral. Continue adding edges in this way. Because the graph has finitely many edges, we must eventually hit a vertex we've seen before and we have a cycle $C \subset E$ in which $0 < x^*_e < 1$ for all edges. Since the graph G is bipartite, the length t of the cycle C is even. To simplify notation let's rename the edges of C to $e_1, e_2, \ldots, e_t$. Now for a small real number ϵ define a vector x' by

$$x'_e = \begin{cases} x^*_e - \epsilon & \text{for } e \in \{e_1, e_3, \ldots, e_{t-1}\} \\ x^*_e + \epsilon & \text{for } e \in \{e_2, e_4, \ldots, e_t\} \\ x^*_e & \text{otherwise} \end{cases}$$

It's easy to see that this new x' still maintains the sum of 1 at each vertex. Now we just have to determine how to pick $\epsilon \in [0, 1]$. Let ϵ be the smallest number that causes one of the x^*_e to hit 0 or 1. Then x' has one more element that is integral compared to x^*, contradicting maximality of $I(x^*)|$. So $I(x^*)| = |E|$. ∎

Problem 4. *The following is the integer program for vertex cover in a graph $G = (V, E)$:*

$$\begin{array}{lll} \text{minimise} & \displaystyle\sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \text{for all } \{u, v\} \in E, \\ & x_v \in \{0, 1\} & \text{for all } v \in V. \end{array}$$

Prove that changing $x_v \in \{0, 1\}$ to $0 \leq x_v \leq 1$ and then solving the resulting linear program results in a 2-approximation.

Proof. Suppose that x is the optimum feasible solution of the linear program relaxation. Define $S = \{v \in V : x^*_v \geq \frac{1}{2}\}$. S is a vertex cover. We want to show that $|S| \leq 2 \cdot |S^*|$. Let S^* be the actual optimal vertex cover and x^* its characteristic vector. Then

$$|S| = \sum_{v \in S} 1 \leq 2 \cdot \sum_{v \in V} x_v \leq 2 \cdot \sum_{v \in V} x^*_v = 2 \cdot |S^*|,$$

which is what we wanted. ∎

Problem 5. *Prove the max-flow min-cut theorem via duality.*

Proof. Given a directed graph $G = (V, E)$ with capacities c for each edge, source vertex s and sink vertex t. Let P be the set of all paths from s to t. Then we can formulate the maximum flow problem as the following linear program:

$$\begin{array}{lll} \text{maximise} & \displaystyle\sum_{p \in P} x_p & \\ \text{subject to} & \displaystyle\sum_{e \in p} x_p \leq c(e) & \text{for all } e \in E, \\ & x_p \geq 0 & \text{for all } p \in P. \end{array}$$

Now we construct the dual. We have one dual variable y_e for every edge $e \in E$.

$$\begin{array}{lll} \text{minimise} & \displaystyle\sum_{e \in E} c(e) y_e & \\ \text{subject to} & \displaystyle\sum_{e \in p} y_e \geq 1 & \text{for all } p \in P, \\ & y_e \geq 0 & \text{for all } e \in E. \end{array}$$

This is the linear program relaxation of the minimum cut integer program. By the Strong Duality Theorem, the optimal value of the objective functions of these two linear programs must be equal, so the maximum flow equals the value of the minimum cut. ∎

Problem 6. *Prove Kőnig's theorem via duality.*

Proof. Let $G = (V, E)$ be a bipartite graph with m edges and n vertices. Consider the integer program

$$\begin{aligned} \text{maximise} \quad & \sum_{j=1}^{m} x_j \\ \text{subject to} \quad & Ax \leq 1, \\ & x \geq 0, \\ & x \in \mathbf{Z}^m. \end{aligned}$$

where A is the incidence matrix of G. Because each row can sum to at most 1 and the incidence matrix of a graph contains only 0s and 1s, we discover that $x_j \in \{0, 1\}$ for all j and that the edges e_j with $x_j^* = 1$ in an optimal solution x^* form a maximum matching in G.

Now consider the integer program

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{n} y_i \\ \text{subject to} \quad & A^T y \geq 1, \\ & y \geq 0, \\ & y \in \mathbf{Z}^n. \end{aligned}$$

where again, A is the incidence matrix of G. In this integer program, every row of A^T corresponding to edge e_j must be at least 1. So in any optimal solution y^* we have $y_i^* \in \{0, 1\}$, since any larger value could be decreased to 1. Then the vertices v_i with $y_i^* = 1$ form a minimum vertex cover of G.

In both integer programs we may drop the integrality constraints, without affecting the optima, because A is unimodular and by Problem 8, b being integral implies that any optimum vector is as well. The resulting linear programs are dual to one another, so their optimal values are equal. Hence the size of a maximum matching equals the size of a minimum vertex cover in G. ∎

Problem 7. *Let $G = (V, E)$ be a graph and with n vertices $v_1, \ldots, v_n$ and m edges $e_1, \ldots, e_m$. The incidence matrix of G is an $n \times m$ matrix A such that $A_{ij} = 1$ if $v_i \in e_j$ and 0 otherwise, for $1 \leq i \leq m$ and $1 \leq j \leq n$. Prove that if G is bipartite, its incidence matrix is totally unimodular.*

Proof. Let $G = (X \cup Y, E)$ be a bipartite graph ($X \cap Y = \emptyset$) and let A denote its incidence matrix. We want to show that every $k \times k$ submatrix Q of A has determinant 0, 1, or -1. We proceed by induction on k. The base case $k = 1$ is easy, because all the entries of an incidence matrix are 0 or 1.

Now consider $k > 1$ and a $k \times k$ submatrix Q. Since the columns of Q correspond to edges, each column of Q has at most two nonzero entries (which are 1). If there is a column with only zero entries, we get $\det Q = 0$, and if there is a column with only one nonzero entry, we can expand the determinant on this column and get that up to sign, $\det Q$ equals the determinant of a $(k-1) \times (k-1)$ submatrix.

The last case is if every column of Q has exactly two 1s. Here we claim that $\det Q = 0$. To see this, observe that the sum of all rows of Q corresponding to vertices in X is the row vector $(1, \ldots, 1)$, since for each column of Q, exactly one of its two 1s comes from a vertex in X. For the same reason, we get $(1, \ldots, 1)$ by summing up the rows for vertices in Y, and it follows that the rows of Q are linearly dependent. ∎

Problem 8. *Prove that a polyhedron described by a totally unimodular matrix and integral right-hand side is integral.*

Proof. Suppose that $P = \{x \in R^n : Ax \leq b\}$ for a totally unimodular matrix A and vector b with integer elements. Let x be a vertex of P. Then there exists some $B \subset [n]$ such that A_B is non-singular, $x_i = 0$ for all $i \notin B$, and $x_B = A_B^{-1} b$. By Cramer's rule, all the entries of A_B^{-1} can be written as rational numbers with common denominator of $\det(A_B)$. But since A is a non-singular unimodular matrix, $\det A_B^{-1} \in \{-1, 1\}$, so every element of A_B^{-1} is integral. So x_B has integral elements and x has integral non-zero elements, which is what we wanted. ∎

Problem 9. *Prove that an independence system is a matroid if and only if the weighted greedy algorithm produces a correct result for all non-negative objective functions.*

Proof. First we prove the "if" direction. Let $M = (E, I)$ be an independence system. We proceed by contraposition. We assume M is not a matroid and show that there exists some non-negative objective function for which the greedy algorithm fails. So suppose that there exist $X, Y \in I$ such that $|Y| > |X|$ and for all $e \in Y \setminus X$, $X \cup \{e\} \notin I$. Let c be a weight function given by

$$c = \begin{cases} 1+\epsilon & \text{if } e \in X \\ 1 & \text{if } e \in Y \setminus X \\ 0 & \text{otherwise} \end{cases}$$

with ϵ to be determined. The Greedy Algorithm should choose a maximum-weight independent set with cardinality $|Y|$. But instead, the algorithm chooses all of X, then it completes X to an independent set X' of cardinality $|Y|$ by choosing 0-weight elements, for a total weight of $|X|(1+\epsilon)$. Now if we set $\epsilon < \frac{1}{|E|}$, then $c(X') < |X| + 1 \leq |Y| \leq c(Y)$. So the Greedy Algorithm chose the wrong set.

Now we prove the "only if" direction. Let $M = (E, I)$ denote a matroid this time and we want to show that if c is a positive objective function, the Greedy Algorithm finds maximum-weight independent sets of cardinality k for every k satisfying $1 \leq k \leq r(E)$.

Suppose, towards a contradiction, that the algorithm fails for a given k. Concretely, suppose the algorithm choose $S_k = \{s_1, s_2, \ldots, s_k\}$ with $c(s_1) \geq c(s_2) \geq \ldots \geq c(s_k)$ and there exists some set $T_k = \{t_1, t_2, \ldots, t_k\}$ with $c(t_1) \geq c(t_2) \geq \ldots \geq c(t_k)$ and $c(T_k) > c(S_k)$. This means there exists some p with $1 \leq p \leq k$ such that $c(t_p) > c(s_p)$. Look at the smallest such p. Consider $X = \{s_1, s_2, \ldots, s_{p-1}\}$ and $Y = \{t_1, t_2, \ldots, t_p\}$. Note that $|Y| > |X|$, so by property 3 of matroids, there exists a $t_j \in Y \setminus X$ such that $X \cup \{t_j\} \in I$. By observation, $c(t_j) \geq c(t_p) > c(s_p)$, so the Greedy Algorithm should have chosen t_j, not s_p, a contradiction. ∎

Problem 10. *Prove that the rank function of a matroid is submodular.*

Proof. Let $M = (E, I)$ be a matroid and let r denote its rank function. We want to show that for all $X, Y \subset E$, $r(X \cup Y) + r(X \cap Y) \leq r(X) + r(Y)$. Let J be a maximal independent subset of $X \cap Y$. Extend J to J_X, a maximal independent subset of X, and likewise to J_Y, a maximal independent subset of Y. We have $r(X \cap Y) = |J| = |J_X \cap J_Y|$. If we can show that $r(X \cup Y) \leq |J_X \cup J_Y|$, then submodularity follows, because $|J_X \cup J_Y| + |J_X \cap J_Y| = |J_X| + |J_Y|$. Extend J_X to a maximal independent subset K of $X \cup Y$.

Suppose, towards a contradiction, that $|K| > |J_X \cup J_Y|$. Because $J_X \setminus J$ is contained in both K and $J_X \cup J_Y$, we have $|K \setminus (J_X \setminus J)| > |J_Y|$. Now, by the choice of J_X, we have that $K \setminus (J_X \setminus J)$ is an independent subset of Y. This contradicts the choice of J_Y. ∎