Chapter 25

MSMYM4 Combinatorial Optimisation

(25.1) Ef□diently Solvable Problems Using Linear Programming

(25.1.1) Statement Of The Problem And Basic Results

A linear programming problem in standard form may be written as

$$\begin{aligned} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} &\to \min \\ \mathbf{a}_i^T\mathbf{x} &= b_i \quad \text{for } i \in I \\ \mathbf{a}_i^T\mathbf{x} &\geqslant b_i \quad \text{for } i \notin I \\ x_j &\geqslant 0 \quad \text{for } j \in J \\ x_j &\text{ free} \quad \text{for } j \notin J \end{aligned}$$

The matrix A of constraint coefficients may then be expressed as

$$A = \begin{pmatrix} \mathbf{a}_i^T \mid i \in I \\ \mathbf{a}_i^T \mid i \notin I \end{pmatrix} = \begin{pmatrix} \mathbf{A}_j \mid j \in J & \mathbf{A}_j \mid j \notin J \end{pmatrix}$$

where the column vector $\mathbf{a}_i$ is the ith row of A and the column vector $\mathbf{A}_j$ is the jth column of A. The dual problem so this is constructed as follows.

- The new objective function is $\phi(\mathfrak{ß}) = \mathfrak{ß}^T\mathbf{b}$ and is to be maximised.
- For each constraint inequality introduce a dual variable $\pi_i \geqslant 0$.
- For each constraint equation introduce a free dual variable.
- For each constrained primal variable x_j with $j \in J$ introduce a constraint inequality $\mathfrak{ß}^T\mathbf{A}_j \leqslant c_j$.
- For each free primal variable x_j with $j \notin J$ introduce a constraint equation $\mathfrak{ß}^T\mathbf{A}_j = c_j$.

Definition 1 *Let M_P and M_D be the sets of all feasible solutions to the primal and dual problems respectively. Let M_P^{opt} and M_D^{opt} be the sets of optimal solutions.*

Theorem 2 (The Weak Duality Theorem) $\forall \mathbf{x} \in M_P \; \forall \mathfrak{ß} \in M_D \; \mathbf{c}^T\mathbf{x} \geqslant \mathfrak{ß}^T\mathbf{b}$

Corollary 3 *Following from the weak duality theorem*

1. *If $\mathbf{x} \in M_P$ and $\mathfrak{ß} \in M_D$ and $\mathbf{c}^T\mathbf{x} = \mathfrak{ß}^T\mathbf{b}$ then $\mathbf{x}$ and $\mathfrak{ß}$ are optimal.*
2. *If $\min\limits_{\mathbf{x} \in M_P} \mathbf{c}^T\mathbf{x} = -\infty$ then $M_D = \emptyset$.*
3. *If $\max\limits_{\mathfrak{ß} \in M_D} \mathfrak{ß}^T\mathbf{b} = \infty$ then $M_P = \emptyset$.*

Theorem 4 (The Strong Duality Theorem) *$M_P \neq \emptyset \Leftrightarrow M_D \neq \emptyset$ and when this is so $\mathbf{c}^T\mathbf{x} = \boldsymbol{ß}^T\mathbf{b}$.*

These results provide useful information about when a linear programming problem can be solved. A summary is given in Table 25.1.1.

Dual Solution (this column)	Primal Problem		
	$M_P^{\text{opt}} \neq \emptyset$	$\min_{\mathbf{x}\in M_P} \mathbf{c}^T\mathbf{x} = -\infty$	$M_P = \emptyset$
$M_D^{\text{opt}} \neq \emptyset$	$\Leftrightarrow$	impossible	impossible
$\max_{\boldsymbol{ß}\in M_D} \boldsymbol{ß}^T\mathbf{b} = \infty$	impossible	impossible	row $\Rightarrow$ column
$M_D = \emptyset$	impossible	column $\Rightarrow$ row	possible

Table 1: Possibilities for solutions to the primal and dual problems.

The bottom right hand entry in Table 25.1.1 says simply that the combination is possible: finding, say, $M_D = \emptyset$ gives no information about M_P. However, the table is very useful in determining whether a given linear programming problem can be solved. Indeed, given the large amount of work needed to solve a linear programming problem and the relatively small amount of work needed to check these criteria, it is prudent to use them.

For example, suppose it is found that $M_D = \emptyset$—this is plausible since sometimes in the translation to the dual the constraint equations become inconsistent. It must then be the case that either $M_P = \emptyset$, or $f(\mathbf{x})$ is not bounded below. Exhibiting a single feasible solution to the prime will then show that $f(\mathbf{x})$ is not bounded below.

It is often the case that the dual will have less variables than the primal and so be easier to solve. To this end it is useful to be able to solve a dual problem with information only about the primal (remember the dual to the dual is the primal).

Theorem 5 (The Complementary Slackness Theorem) *Suppose $\mathbf{x} \in M_P$ and $\boldsymbol{ß} \in M_D$ then $\mathbf{x}$ and $\boldsymbol{ß}$ are optimal if and only if*

$$\left(c_j - \boldsymbol{ß}^T\mathbf{A}_j\right) x_j = 0 = \pi_i \left(\mathbf{a}_i^T\mathbf{x} - b_i\right)$$

for all i and all j.

Proof. Define for all i and j

$$u_j = \left(c_j - \boldsymbol{ß}^T\mathbf{A}_j\right) x_j$$
$$v_i = \pi_i \left(\mathbf{a}_i^T\mathbf{x} - b_i\right)$$

For a linear programming problem in standard form $\left(c_j - \boldsymbol{ß}^T\mathbf{A}_j\right) \geqslant 0$ and $\left(\mathbf{a}_i^T\mathbf{x} - b_i\right) \geqslant 0$ by definition. Furthermore, for a free primal(dual) variable equality holds in the dual(primal) constraint and so $u_j \geqslant 0$ and $v_i \geqslant 0$. Define $u = \sum_j u_j$ and $v = \sum_i v_i$ then both u and v are non-negative, as is $u + v$. Now,

$$\begin{aligned} u + v &= \sum_j c_j x_j - \sum_j \boldsymbol{ß}^T\mathbf{A}_j x_j + \sum_i \pi_i \mathbf{a}_i^T\mathbf{x} - \sum_i \pi_i b_i \\ &= f(\mathbf{x}) - \sum_j \boldsymbol{ß}^T\mathbf{A}_j x_j + \sum_i \pi_i \mathbf{a}_i^T\mathbf{x} - g(\boldsymbol{ß} \\ &= f(\mathbf{x}) - g(\boldsymbol{ß}) \end{aligned}$$

This is because

$$\sum_j \mathbf{ß}^T \mathbf{A}_j x_j = \mathbf{ß}^T A\mathbf{x} = \sum_i \pi_i \mathbf{a}_i^T \mathbf{x}$$

($\Rightarrow$) Suppose that $\mathbf{ß}$ and $\mathbf{x}$ are optimal, then $f(\mathbf{x}) = g(\mathbf{ß})$ and so $u + v = 0$. But u and v are sums of non-negative terms, hence $u_j = 0 = v_i$ for all i and j.

($\Leftarrow$) Suppose that $u_j = 0 = v_i$ for all i and j then $u + v = 0 = f(\mathbf{x}) - g(\mathbf{ß})$ and hence $f(\mathbf{x}) = g(\mathbf{ß})$ i.e. the solution is optimal. □

The complementary slackness theorem is useful for solving a primal problem when the solution to the dual is known as it allows simultaneous equations to be constructed for the solution to the primal problem.

Information About The Dual In The Simplex Table

The initial simplex table consists of a matrix B and the identity matrix I. Now, $f(\mathbf{x}) = z_0 = \mathbf{c}_{\mathcal{B}}^T B^{-1}\mathbf{b}$ and since $f(\mathbf{ß}) = \mathbf{ß}^T\mathbf{b}$ it must be the case that when $\mathbf{x}$ (and $\mathbf{ß}$) are optimal $\mathbf{ß} = \mathbf{c}_{\mathcal{B}}^T B^{-1}$. Hence $\mathbf{ß}$ can be calculated from the final simplex table as follows:

- Let $\mathbf{e}_k$ be the kth ordered basis vector appearing in column j of the original simplex table. Therefore $z_j = c_{\mathcal{B}}^T B^{-1}\mathbf{e}_k$ giving $z_j = \pi_j$.
- Let $\bar{c}_j$ be the relative cost in column j of the final simplex table, so $\bar{c}_j = c_j - z_j = c_j - \pi_k$.
- Hence in the final simplex table, $\pi_k = c_j - \bar{c}_j$.

(25.1.2) The Primal-Dual Algorithm

The primal-dual algorithm takes a feasible solution to the dual and keeps modifying it until an optimal solution to the primal can be found. Let $\mathbf{ß} \in M_D$ then $\mathbf{ß}^T\mathbf{A}_j \leqslant c_j$ for all j. Now, by complimentary slackness (Theorem 5) when $\mathbf{ß}$ and $\mathbf{x}$ are optimal $\left(c_j - \mathbf{ß}^T\mathbf{A}_j\right) x_j = 0$. Therefore if $c_j - \mathbf{ß}^T\mathbf{A}_j > 0$ set $x_j = 0$. Define

$$J = \left\{ j \mid \mathbf{ß}^T\mathbf{A}_j = c_j \right\}$$

so that for $j \notin J$ it must be the case that $x_j = 0$. The next task is to find values for x_j when $j \in J$. Observe that

$$\mathbf{b} = \sum_{j=1}^{n} \mathbf{A}_j x_j = \sum_{j\in J} \mathbf{A}_j x_j + \sum_{j\notin J} \mathbf{A}_j x_j = \sum_{j\in J} \mathbf{A}_j x_j$$

In order to find the missing values in $\mathbf{x}$ a trivial solution is found by adding in artificial variables. The simplex method is then used to eliminate these in favour of the missing values. Hence define the restricted primal problem

$$\xi = \sum_{i=1}^{m} x_i^a \to \min$$
$$x_i^a + \sum_{j\in J} a_{ij} x_j = b_i$$
$$x_j, x_i^a \geqslant 0$$

To solve this the auxiliary costs $c_j = 0$ for $j \in J$ and $c_i = 1$ for x_i^a are used.

- If $\xi = 0$ then disregard the artificial variables to give optimal solution the the original problem, $\mathbf{x}$.

- If $\xi > 0$ then not all of the missing values in $\mathbf{x}$ can be found. Hence consider the dual to the restricted primal.

Let $\overline{\mathfrak{ß}}$ be an optimal solution to the dual restricted primal. $\overline{\mathfrak{ß}}$ is now used to modify $\mathfrak{ß}$ to create a new solution, say $\mathfrak{ß}^* = \mathfrak{ß} + \theta\overline{\mathfrak{ß}}$. Certainly $\mathfrak{ß}^{*T}\mathbf{b} > \mathfrak{ß}^T\mathbf{b}$ as $\mathfrak{ß}$ is not optimal. Hence $\theta > 0$. For feasibility it is required that $\mathfrak{ß}^*\mathbf{A}_j \leqslant c_j$ or rather that

$$\mathfrak{ß}^T\mathbf{A}_j + \theta\overline{\mathfrak{ß}}^T\mathbf{A}_j \leqslant c_j$$

- If $\overline{\mathfrak{ß}}^T\mathbf{A}_j \leqslant 0$ then

  $$\mathfrak{ß}^T\mathbf{A}_j + \theta\overline{\mathfrak{ß}}^T\mathbf{A}_j \leqslant \mathfrak{ß}^T\mathbf{A}_j \leqslant c_j$$

  with the second inequality following from the feasibility of $\mathfrak{ß}$. Hence in this case $\mathfrak{ß}^*$ is feasible for all θ, except when the conditions of Theorem 6 apply.
- If $\overline{\mathfrak{ß}}^T\mathbf{A}_j > 0$ then θ must be chosen so as to maintain $\mathfrak{ß}^{*T}\mathbf{A}_j \leqslant c_j$ for $j \in J$. Hence require

$$\begin{aligned}
\mathfrak{ß}^T\mathbf{A}_j + \theta\overline{\mathfrak{ß}}^T\mathbf{A}_j &\leqslant c_j \\
\theta &\leqslant \frac{c_j - \mathfrak{ß}^T\mathbf{A}_j}{\overline{\mathfrak{ß}}^T\mathbf{A}_j} \quad \forall j \\
&\leqslant \min\left\{ \frac{c_j - \mathfrak{ß}^T\mathbf{A}_j}{\overline{\mathfrak{ß}}^T\mathbf{A}_j} \mid \overline{\mathfrak{ß}}^T\mathbf{A}_j > 0,\ j \notin J \right\}
\end{aligned}$$

Note that if this gives $\theta \leqslant 0$ then there is in fact no solution for θ as $\theta > 0$. The condition $j \notin J$ is required to prevent $\theta = 0$ and indeed it is the x_j with $j \notin J$ for which solutions are sought. If no such θ can be found then $M_P = \emptyset$ whereas otherwise the process is repeated with the new feasible solution to the dual problem $\mathfrak{ß} = \mathfrak{ß}^*$.

Theorem 6 *For the restricted dual problem if $\xi > 0$ and $\overline{\mathfrak{ß}}^T\mathbf{A}_j \leqslant 0$ for all j then $M_P = \emptyset$.*

Proof. By the weak duality theorem it is sufficient to show that $\mathfrak{ß}^T\mathbf{b} \to \infty$.
If $j \in J$ then $\overline{\mathfrak{ß}}\mathbf{A}_j \geqslant 0$.
If $j \notin J$ then by hypothesis $\overline{\mathfrak{ß}}\mathbf{A}_j \geqslant 0$. Hence

$$\mathfrak{ß}^T\mathbf{A}_j + \theta\overline{\mathfrak{ß}}^T\mathbf{A}_j \leqslant \mathfrak{ß}^T\mathbf{A}_j \leqslant c_j$$

because $\mathfrak{ß}$ is dual feasible. Hence for all $\theta > 0$ $\mathfrak{ß} + \theta\overline{\mathfrak{ß}}$ is feasible and

$$\mathfrak{ß}^T\mathbf{b} + \theta\overline{\mathfrak{ß}}^T\mathbf{b} \to \infty \text{ as } \theta \to \infty$$

□

(25.1.3) The Maximum Flow Problem

The maximum flow problem is formulated in terms of a digraph. One vertex is the source from which edges only leave, and another is the sink to which edges only arrive. The weight of each edge is the capacity of that edge. The task is to find the maximum flow through the graph from the source to the sink.

Definition 7 *A digraph N is a network if there exists a vertex s (called the source) where no edges arrive and there exists another vertex t (called the sink) from which no edges leave. Where b is the weight function, $b\colon E \to \mathbb{R}^+$ the network is denoted $N = (V, E, s, t, b)$.*

Definition 8 *The function $f\colon E \to \mathbb{R}^+$ is a flow in a network N if*

1. $0 \leqslant f(v_i, v_j) \leqslant b(v_i, v_j)$ *for all* $(v_i, v_j) \in E$. *These are called the capacity constraints.*
2. *The flows at intermediate vertices are conserved, i.e.*

$$\sum_{\{j|(v_i,v_j)\in E\}} f(v_i, v_j) = \sum_{\{j|(v_j,v_i)\in E\}} f(v_i, v_j)$$

Notation 9 *The following alternatives may be used to aid clarity.*

- *Let* $f_{ij} = f_k = f(v_i, v_j)$ *where* $e_k = (v_i, j_j)$.
- *Let* $\mathbf{f} = \begin{pmatrix} f_1 & f_2 & \dots & f_n \end{pmatrix}^T$.
- *Let* $b_{ij} = b_k = b(v_i, v_j)$ *where* $e_k = (v_i, j_j)$.
- *Let* $\mathbf{b} = \begin{pmatrix} b_1 & b_2 & \dots & b_n \end{pmatrix}^T$.

From this the conservation law may be expressed more simply as

$$\sum_{e_k \text{ enters } v} f_k = \sum_{e_k \text{ leaves } v} f_k \quad \text{for } v \in \{v \in V \mid v \neq s, v \neq t\}$$

Definition 10 *The value of a flow* f, $v(f)$ *is the amount of flow that runs through the network, hence*

$$v(f) = \sum_{e_k \, leaves \, s} f_k$$

Definition 11 *The incidence matrix* A *for a network* (V, E, s, t, b) *is defined by*

$$a_{ij} = \begin{cases} 1 & \text{if } e_j \text{ leaves } v_i \\ -1 & \text{if } e_j \text{ enters } v_i \\ 0 & \text{otherwise} \end{cases}$$

So rows correspond to vertices and columns correspond to edges.

It is immediately obvious from the sum $\sum_j a_{ij} f_j$ that for fixed i

$$\sum_j a_{ij} f_j = \sum_{e_j \text{ leaves } v_i} f_j - \sum_{e_j \text{ enters } v_i} f_j = 0 \quad \text{(by conservation) for } v_i \neq s, v_i \neq t$$

$$\sum_j a_{ij} f_j = \sum_{e_j \text{ leaves } s} f_j = v$$

$$\sum_j a_{ij} f_j = -\sum_{e_j \text{ enters } t} f_j = v'$$

$$A\mathbf{f} = \begin{cases} v & \text{in the position correpsonding to the row for } s \\ v' & \text{in the position correpsonding to the row for } t \\ 0 & \text{otherwise (by conservation)} \end{cases}$$

Intuitively, whatever leaves the source should reach the sink, so that $v' = -v$. Hence $A\mathbf{f} = -v\mathbf{d}$ where $-\mathbf{d}$ has a 1 in the position corresponding to the source and a -1 in the position corresponding to the sink. Hence $A\mathbf{f} + v\mathbf{d} = \mathbf{0}$. Furthermore it is required that $\mathbf{f} \leqslant \mathbf{b}$ and that $\mathbf{f} \geqslant \mathbf{0}$. Hence the maximum flow problem

is in fact the linear programming problem

$$\begin{aligned} v &\to \max \\ A\mathbf{f} + v\mathbf{d} &= 0 \\ \mathbf{f} &\leqslant \mathbf{b} \\ \mathbf{f} &\geqslant \mathbf{0} \end{aligned}$$

Note this problem has $n+1$ variables—the n flows and v. There are n (the number of vertices) constraint equations (because A is $N \times m$ and $\mathbf{f}$ is $m \times 1$) and n (the number of edges) constraint inequalities.

Certainly v is non-negative, and clearly $\mathbf{f} = \mathbf{0}$ is always a feasible solution.

Lemma 12 *A maximal flow exists whenever the capacities of edges leaving the source and entering the sink are finite.*

Proof. Every linear programming problem is either infeasible, $f^{\min} \to -\infty$ or has at least one optimal solution. For the maximal flow problem $\mathbf{f} = \mathbf{0}$ is a feasible solution and by finiteness of capacities $f^{\min} \to -\infty$ is impossible. Hence an optimal solution must exist. □

The Ford-Fulkerson Algorithm

Having formulated the maximum flow problem as a linear programming problem the obvious thing to do would be to apply the dual simplex algorithm. However, instead the dual simplex algorithm is used to prove the correct operation of the Ford-Fulkerson algorithm which is now presented.

Definition 13 *For a network (V, E, s, t, b) and flow f an augmenting path p is a path from s to t in the undirected graph (V, E') where $E' = \{\{i, j\} \mid (i, j) \in E\}$ with the following properties.*

- *If p traverses $\{i, j\} \in E'$ and $(i, j) \in E$ (a 'forwards edge') then $f_{ij} < b_{ij}$. So forwards edges are unsaturated.*
- *If p traverses $\{j, i\} \in E'$ and $i, j) \in E$ (a 'backwards edge') then $f_{ij} > 0$. So backwards edges are non-empty.*

The objective now is to increase flow to saturation on forward edges and decrease flow on backward edges to zero. Hence along such a path the maximum change in flow that can be made is

$$\delta = \min_{(i,j) \text{ on } p} \{f_{ij} \mid (i, j) \text{ is a backward edge}\} \cup \{b_{ij} - f_{ij} \mid (i, j) \text{ is a forward edge}\}$$

The task now is to find such a path.

Algorithm 14 (Labelling Algorithm) *The vertices in a network are labelled as follows.*

1. *Let f be a feasible flow and $L := \{s\}$. Label s by $(\emptyset, \infty)$.*
2. *Select a vertex $x \in L$ which has label $(l_1(x), l_2(x)$ say, and set $L := L \setminus \{x\}$.*
3. *Scan the network from x as follows.*
   (a) *If $(x, y) \in E$ and $f_{xy} < b_{xy}$ then label y by*

$$\big(x, \min\{l_2(x), b_{xy} - f_{xy}\}\big)$$

   (b) *If $(y, x) \in E$ and $f_{yx} > 0$ then label y by*

$$\big(-x, \min\{l_2(x), f_{yx}\}\big)$$

When y is labelled set $L := L \cup \{y\}$.

4. *Repeat from 2 and terminate when $L = \emptyset$.*

Observe that by construction $\delta = l_2(t)$. However, if t is not reached the flow must be optimal—this is called 'nonbreakthrough'.

Algorithm 15 (Ford-Fulkerson) *A maximal flow may be found as follows.*

1. *Let f be a feasible flow.*
2. *With the flow f, remove any existing labels and run the labelling algorithm.*
3. *If t is labelled then construct an augmentation path and increase the flow along it giving flow f'. Set $f = f'$ and continue from 2.*
4. *If t is not labelled terminate.*

Two questions must now be answered in order to show that the Ford-Fulkerson algorithm works. Firstly, it must be determined whether the algorithm terminates. Secondly, if the algorithm does terminate, does it give an optimal flow?

Maximal Flow

To prove optimality at termination requires rather a lot of work. The strategy is to find feasible solutions to the dual of the maximum flow problem and use it to establish—through a rather convoluted process—an upper bound on $v(f)$. The final step is to show the maximal flow to be equal to the minimum value of this bound. (This sounds contradictory, but a little thought may show otherwise.)

First of all construct the dual to the maximum flow problem

$$\begin{aligned} v &\to \max \\ A\mathbf{f} + v\mathbf{d} &= 0 \\ \mathbf{f} &\leqslant \mathbf{b} \\ \mathbf{f} &\geqslant \mathbf{0} \end{aligned}$$

Recall $|V| = n$ and $|E| = m$. To the first n conservation constraint equations assign the new free variable $\pi(x)$ where x is the vertex corresponding to the particular equation. To the m capacity constraint inequalities assign a new non-negative variable $\gamma(x, y)$ where $(x, y) \in E$ is the edge corresponding to the particular inequality. The coefficients of the various constraints may be represented in the form

$$\begin{array}{cc} A & \mathbf{d} \\ I & \mathbf{0} \end{array}$$

The left hand column corresponds to the elements of $\mathbf{f}$. Any column in A has a 1 where $x \in V$ and -1 where $y \in V$ say. The bottom part then has 1 when $(x, y) \in E$. This gives rise to the constraints

$$\pi(x) - \pi(y) + \gamma(x, y) \geqslant 0 \quad \forall (x, y) \in E$$

The right hand side is 0 because the elements of $\mathbf{v}$ have cost coefficient 0 in the objective function, $v \to \max$. The right of the constraint coefficient array given above corresponds to v. Hence by the definition of $\mathbf{d}$ and

the form of the objective function the single equation $\pi(t) - \pi(s) = 1$ is obtained. Hence the dual to the maximum flow problem is

$$\begin{aligned} \sum_{(x,y)\in E} b(x,y)\gamma(x,y) &\to \min \\ \pi(x) - \pi(y) + \gamma(x,y) &\geqslant 0 \quad \forall (x,y) \in E \\ \pi(t) - \pi(s) &= 1 \\ \gamma(x,y) &\geqslant 0 \quad \forall (x,y) \in E \\ \pi(x) \text{ free } &\forall x \in V \end{aligned}$$

Definition 16 *An s-t cut in a network N is a pair $(W, \overline{W})$ such that $V = W \dot\cup \overline{W}$, $s \in W$ and $t \in \overline{W}$. The capacity of the cut is*

$$c(W, \overline{W}) = \sum_{\substack{x\in W \\ y\in\overline{W} \\ (x,y)\in E}} b(x,y)$$

Lemma 17 *Every cut determines a feasible solution to the dual of the maximum flow problem as follows*

$$\gamma(x,y) = \begin{cases} 1 & \text{if } (x,y) \in E \text{ and } x \in W, y \in \overline{W} \\ 0 & \text{otherwise} \end{cases}$$

$$\pi(x) = \begin{cases} 0 & \text{if } x \in W \\ 1 & \text{if } x \in \overline{W} \end{cases}$$

Furthermore, the value of the objective function is equal to the value of the cut.

Proof. Let $(x,y) \in E$ then by the definition of this solution

$$\pi(x) - \pi(y) + \gamma(x,y) = \begin{cases} 0 & \text{if } x \in W \text{ and } y \in W \\ 0 & \text{if } x \in W \text{ and } y \in \overline{W} \\ 1 & \text{if } x \in \overline{W} \text{ and } y \in W \\ 0 & \text{if } x \in \overline{W} \text{ and } y \in \overline{W} \end{cases}$$

$$\pi(t) - \pi(s) = 1$$

So the solution is feasible. The value of the objective function is given by

$$\sum_{(x,y)\in E} b(x,y)\gamma(x,y) = \sum_{\substack{x\in W \\ y\in\overline{W} \\ (x,y)\in E}} b(x,y) = c(W, \overline{W})$$

□

Lemma 18 *For any cut $(W, \overline{W})$ and any flow f*

$$v(f) = \sum_{\substack{x\in W \\ y\in\overline{W}}} f(x,y) - \sum_{\substack{x\in\overline{W} \\ y\in W}} f(x,y)$$

Proof. Let $(W, \overline{W})$ be a cut and f be a flow. Since flow is conserved at vertices

$$\sum_{\{y|(x,y)\in E\}} f(x,y) - \sum_{\{y|(x,y)\in E} f(y,x) = 0 \quad \forall x \notin \{s,t\}$$

$$\text{and} \sum_{\{y|(s,y)\in E\}} f(s,y) = v$$

$$\begin{aligned}
\text{let } \overline{v} &= \sum_{\substack{x\in W\\ y\in V\\ (x,y)\in E}} f(x,y) - \sum_{\substack{x\in W\\ y\in V\\ (y,x)\in E}} f(y,x) \\
&= \sum_{\substack{x\in W\\ y\in W\\ (x,y)\in E}} f(x,y) + \sum_{\substack{x\in W\\ y\in \overline{W}\\ (x,y)\in E}} f(x,y) - \sum_{\substack{x\in W\\ y\in W\\ (y,x)\in E}} f(y,x) - \sum_{\substack{x\in W\\ y\in \overline{W}\\ (y,x)\in E}} f(y,x) \\
&= \sum_{\substack{x\in W\\ y\in W\\ (x,y)\in E}} f(x,y) - \sum_{\substack{x\in \overline{W}\\ y\in W\\ (x,y)\in E}} f(x,y)
\end{aligned}$$

□

Theorem 19 *Let N be a network, let f be a flow in N and let $(W, \overline{W})$ be a cut. Then*

1. *for every flow f and every cut $(W, \overline{W})$, $v(f) \leqslant c(W, \overline{W})$ and the value of the maximal flow is equal to the value of the minimal cut.*
2. *if the Ford-Fulkerson algorithm terminates then it does so at maximal flow.*

Proof. 1. Let f be a flow and $(W, \overline{W})$ be a cut. By Lemma 17 there exists a feasible solution to the dual of the maximum flow problem with

$$c(W, \overline{W}) = \sum_{(x,y)\in E} b(x,y)\gamma(x,y)$$

Since this value is to be minimised duality theory gives $v(f) \leqslant c(W, \overline{W})$.

Suppose now that f^* is an maximal flow and run the Ford-Fulkerson algorithm starting with this flow. Since f^* is optimal the algorithm must terminate in the first iteration without finding an augmenting path. Define

$$W = \{v \in V \mid v \text{ is labelled when the Ford-Fulkeson algorithm terminates}\}$$
$$\overline{W} = W^C$$

which, trivially, is a cut. Now by Lemma 18

$$\begin{aligned}
v(f^*) &= \sum_{\substack{x\in W\\ y\in W\\ (x,y)\in E}} f^*(x,y) - \sum_{\substack{x\in \overline{W}\\ y\in W\\ (x,y)\in E}} f^*(x,y) \\
&= \sum_{\substack{x\in W\\ y\in W\\ (x,y)\in E}} b(x,y) \\
&= c(W, \overline{W})
\end{aligned}$$

2. Let f^* be a flow found by the Ford-Fulkerson algorithm, then a cut can be defined in the same way as above. But then by the same calculation $v(f^*) = c(W, barW)$ and so has the same flow as the maximal flow. □

Termination

Although maximal flow is found at termination there is no guarantee that the Ford-Fulkerson algorithm terminates at all.

Theorem 20 *The Ford-Fulkerson algorithm terminates whenever the edge capacities are rational.*

Proof. First of all suppose the capacities are natural numbers. The Ford-Fulkerson algorithm begins with a zero flow, which is an integer flow. All flows found by the algorithm will be integer because

$$\delta = \min_{(i,j) \text{ on } p} \{f_{ij} \mid (i,j) \text{ is a backward edge}\} \cup \{b_{ij} - f_{ij} \mid (i,j) \text{ is a forward edge}\}$$

is an integer and $\delta \geqslant 1$. But therefore the flow increases by at least 1 at every iteration, so if f^* is a maximum flow (one always exists) then the algorithm will terminate in at most $v(f^*)$ iterations.

Suppose now that $b\colon E \to \mathbb{Q}^+$ so that $b_j = \frac{p_j}{q_j}$ for $p_j \in \mathbb{N}$ and $q_j \in \mathbb{N}$, say. Take $D = \operatorname{lcm}\{q_j\}$ then Db_j is integer for all j. Consider the maximum flow problem with the capacity function Db, which are all integers. Hence the algorithm terminates in at most $Dv(f^*)$ iterations where f^* is an optimal flow in the new maximum flow problem. □

If $b\colon E \to \mathbb{R}$ then there is no guarantee that the algorithm will terminate.

(25.1.4) The Transportation Problem

The transportation problem is to match producers $A_1, A_2, \ldots, A_m$ with capacities $a_1, a_2, \ldots, a_m$ with consumers $B_2, B_2, \ldots, B_n$ with capacities $b_1, b_2, \ldots, b_n$ in such a way as to minimise the total cost of transport, where the cost of transportation from producer A_i to consumer B_j is c_{ij}.

It is easy to solve such problems using the Hungarian Method described in Chapter **??**. However, the problem may be formulated as a linear programming problem and so as with the maximum flow problem a solution using the primal-dual algorithm is sought. The formulation as a linear programming problem is

$$\begin{aligned} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} &\to \min \\ \sum_{j=1}^{m} x_{ij} &= a_i \\ \sum_{i=1}^{n} x_{ij} &= b_j \\ x_{ij} &\geqslant 0 \quad \forall i,j \end{aligned}$$

From Chapter **??** recall that

- If the transportation problem has a feasible solution then it has an optimal solution.
- The transportation problem has a feasible solution if and only if $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. When this condition holds the problem is said to be balanced.
- If $\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j$ a 'dummy' producer A_{m+1} with the required capacity is added with costs of transportation zero.

The first step in the application of the primal-dual algorithm is to find the dual. This appears to verge on the impossible until the following matrix representation of the constraints is constructed.

$$
\begin{array}{ccccccccc}
x_{11}+ & x_{12}+ & \cdots & +x_{1n} & & & & & = a_1 \\
 & x_{21}+ & x_{22}+ & \cdots & +x_{2n} & & & & = a_2 \\
 & & & & & \ddots & & & \vdots \\
 & & & & & & x_{m1}+ & x_{m2}+ \cdots +x_{mn} & = a_m \\
x_{11}+ & & x_{21}+ & \ldots & & \cdots & +x_{m1} & & = b_1 \\
 & x_{21}+ & & x_{22}+ & \ldots & & \cdots + & x_{m2} & = b_2 \\
 & & \ddots & & \ddots & & & \ddots & \vdots \\
 & & x_{1n}+ & & x_{2n}+ & \ldots & & \cdots +x_{mn} & = b_n
\end{array}
$$

For them m constraint equations for the producers introduce the new free variables α_i. For the n constraint equations for the consumers introduce the new free variables β_j. The dual is then

$$
\begin{aligned}
\sum_{i=1}^{m} \alpha_i a_i + \sum_{j=1}^{n} \beta_j b_j &\to \max \\
\alpha_i + \beta_j &\leqslant c_{ij} \quad \forall i, j
\end{aligned}
$$

Clearly a feasible solution is $\alpha_i = 0\ \forall i$ and $\beta_j = \min\{c_{ij}\ \forall i\}$ for all j. Now to find the restricted primal let $K = \{(i,j) \mid \alpha_i + \beta_j = c_{ij}\}$ to give

$$
\begin{aligned}
\xi = \sum_{k=1}^{m+n} x_k^a &\to \min \\
x_i^a + \sum_{\{j|(i,j)\in K} x_{ij} &= a_i \text{ for } 1 \leqslant i \leqslant m \\
x_{m_j}^a + \sum_{\{i|(i,j)\in K} x_{ij} &= b_j \text{ for } 1 \leqslant j \leqslant n \\
x_{ij} &\geqslant 0 \text{ for all } (i,j) \in K \\
x_k^a &\geqslant 0 \text{ for all } 1 \leqslant i \leqslant m+n
\end{aligned}
$$

At this point a useful relationship can be deduced. Summing the objective function and all the constraint equations gives

$$
\xi = \sum_{i=1}^{m} a_i + \sum_{j=1}^{m} b_j - 2 \sum_{\{(i,j)\in K\}} x_{ij} \tag{21}
$$

Using this the artificial variables can be eliminated from the restricted primal. Note that since the first two summations are simply constants they can be removed from the objective function. The factor of -2 can

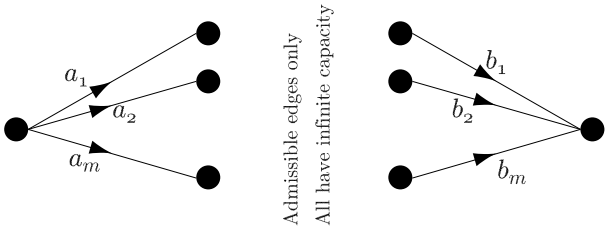

Figure 1: Maximum flow graph for the transportation problem.

also be eliminated, remembering to change minimisation to maximisation. This gives

$$\begin{aligned}
\sum_{\{(i,j)\in K\}} x_{ij} &\to \max \\
\sum_{\{j|(i,j)\in K\}} x_{ij} &\leqslant a_i \text{ for } i \leqslant i \leqslant m \\
\sum_{\{i|(i,j)\in K\}} x_{ij} &\leqslant b_j \text{ for } i \leqslant j \leqslant n \\
x_{ij} &\geqslant 0 \text{ for } (i,j) \in K
\end{aligned}$$

The form of this is simply a maximum flow problem for a graph as shown in Figure 25.1.4, to which the Ford-Fulkerson algorithm can be applied.

Theorem 22 *Every feasible solution to the restricted primal determines a unique flow f in the network N. Conversely, every flow in N determines a feasible solution to the restricted primal, and in both cases*

$$v(f) = \sum_{\{(i,j)\in K} x_{ij}$$

Proof. Let X where $(X)_{ij} = x_{ij}$ be a feasible solution to the restricted primal, then construct the following flow $f(i,j) = x_{ij}$ with flows from/to the source and sink determined by the equations.

$$f(s,i) = \sum_{\{j|(i,j)\in K\}} x_{ij} \quad \text{and} \quad f(j,t) = \sum_{\{i|(i,j)\in K\}} x_{ij}$$

These equations guarantee conservation. Non-negativity follows from the non-negativity of the solution to the transportation problem. For the source and sink the capacity constraints are met by construction. For the other edges the capacities are infinite, so trivially these are not violates.

Conversely let f be a flow in a network N and set $x_{ij} = f(i,j)$ for all $(i,j) \in K$. This is then a feasible solution to the restricted primal as the capacity constraints ensure the constraint inequalities are satisfied.

In both cases

$$v(f) = \sum_{i\in I} f(s,i) = \sum_{i\in I} \sum_{\{j|(i,j)\in K} f(i,j) = \sum_{(i,j)\in K} x_{ij}$$

□

From equation (21) a solution $\mathbf{x}$ is optimal (so $\xi = 0$) if and only if

$$\sum_{(i,j)\in K} x_{ij} = \sum_{i=1}^{m} a_i$$

noting that the problem is balanced so that $\sum a_i = \sum b_j$. Hence when inequality is found an optimal solution to the restricted primal must be found so that the solution can be updated.

Definition 23 *When the Ford-Fulkerson Labelling Algorithm terminates (at 'nonbreakthrough') let*

$$I^* = \{i \in I \mid i \text{ is labelled}\}$$
$$J^* = \{j \in J \mid j \text{ is labelled}\}$$

Lemma 24 *At nonbreakthrough $i \in I^* \Rightarrow j \in J^*$ for every $(i,j) \in J$.*

Proof. When the optimal flow is reached $f(i,j)$ is finite but the capacity of the edge (i,j) is infinity, therefore whenever one of the producer vertices is labelled any consumer to which it is joined must also be labelled.□

Theorem 25 *At nonbreakthrough an optimal solution to the dual of the restricted primal is given by*

$$\overline{\alpha}_i = 1 \quad \text{for } i \in I^*$$
$$\overline{\alpha}_i = -1 \text{ for } i \notin I^*$$
$$\overline{\beta}_j = -1 \text{ for } j \in J^*$$
$$\overline{\beta}_j = 1 \quad \text{for } j \notin J^*$$

Proof. The proof is done in two stages. First of all $\left(\overline{\mathbf{ff}} \quad \overline{\mathbf{fi}}\right)$ is shown to be feasible, then secondly optimality is shown by showing that the same objective function value is obtained in the restricted primal. However, it is first necessary to find the dual to the restricted primal, which is

$$\sum_{i=1}^{m} \alpha_i a_i + \sum_{j=1}^{n} \beta_j b_j \to \max$$
$$\alpha_i + \beta_j \leqslant 0$$
$$\alpha_i \leqslant 1 \text{ for } i \in I$$
$$\beta_j \leqslant 1 \text{ for } j \in J$$

Clearly the proposed solution satisfies the last two constraint inequalities. Now, for $(i,j) \in K$

- If $i \notin I^*$ then $\overline{\alpha}_i = -1$ so the first constraint inequality is satisfied whichever value $\overline{\beta}_j$ takes.
- If $i \in I^*$ then $\overline{\alpha}_i = 1$ and by Lemma 24 $j \in J^*$ so that $\overline{\beta}_j = -1$ and the first constraint inequality is satisfied.

Hence this solution is feasible.

Calculating the value of the objective function for this solution gives

$$g\left(\overline{\mathbf{ff}}, \overline{\mathbf{fi}}\right) = \sum_{i \in I^*} a_i - \sum_{i \notin I^*} a_i + \sum_{j \in J^*} b_j - \sum_{j \notin J^*} b_j \tag{26}$$

Consider now the objective function of the restricted primal

$$\xi = \sum_{i \in I} a_i + \sum_{j \in J} b_j - 2v(f) \tag{27}$$

Let f be the maximal flow as found by the Ford-Fulkerson algorithm. Therefore

$$v(f) = \sum_{i \in I} f(s,i) \quad \text{by definition}$$
$$= \sum_{i \in I^*} f(s,i) + \sum_{i \notin I^*} f(s,i)$$

Now, if $i \notin I^*$ then vertex i was not labelled, meaning that the edge (s,i) is saturated. Hence

$$
\begin{aligned}
&= \sum_{i\in I^*} f(s,i) + \sum_{i\notin I^*} a_i \\
&= \sum_{i\notin I^*} a_i + \sum_{i\in I^*} \sum_{\substack{j\in J \\ (i,j\in K)}} f(i,j) \quad \text{by conservation} \\
&= \sum_{i\notin I^*} a_i + \sum_{i\in I^*} \left(\sum_{\substack{j\in J^* \\ (i,j)\in K}} f(i,j) + \sum_{\substack{j\notin J^* \\ (i,j)\in K}} f(i,j) \right) \\
&= \sum_{i\notin I^*} a_i + \sum_{i\in I^*} \sum_{\substack{j\in J^* \\ (i,j)\in K}} f(i,j) \quad \text{by Lemma 24} \\
&= \sum_{i\notin I^*} a_i + \sum_{\substack{j\in J^* \\ (i,j)\in K}} \sum_{i\in I^*} f(i,j) \\
&= \sum_{i\notin I^*} a_i + \left(\sum_{\substack{i\in I^* \\ (i,j)\in K}} f(i,j) + \sum_{\substack{i\notin I^* \\ (i,j)\in K}} f(i,j) \right) \quad \text{because the second summation is zero} \\
&= \sum_{i\notin I^*} a_i + \sum_{j\in J^*} \sum_{i\in I} f(i,j) \\
&= \sum_{i\notin I^*} a_i + \sum_{j\in J^*} f(j,t) \quad \text{by conservation} \\
&= \sum_{i\notin I^*} a_i + \sum_{j\in J^*} b_j
\end{aligned}
$$

This last step is made for the following reason: As the Ford-Fulkerson algorithm has terminated t was not labelled. Therefore all edges (j,t) must be saturated. Substituting into equation (27) and comparing to equation (26) gives

$$\xi = \sum_{i\in I} a_i + \sum_{j\in J} b_j - 2\left(\sum_{i\notin I^*} a_i + \sum_{j\in J^*} b_j \right) = g\left(\overline{\mathbf{ff}}, \overline{\mathbf{fi}}\right)$$

Hence by the strong duality theorem the suggested solution is optimal. □

Having found an optimal solution to the dual of the restricted primal the next step is to find θ and modify the solution to the primal.

- $\overline{\alpha}_i + \overline{\beta}_j \leqslant 0$ for all $(i,j) \notin K$ means that the primal is infeasible, but this is impossible for the transportation problem.

- There exists $(i,j) \in K$ such that $\overline{\alpha}_i + \overline{\beta}_j > 0$ so that

$$\theta = \min\left\{ \frac{c_{ij} - \alpha_i - \beta_j}{\overline{\alpha}_i + \overline{\beta}_j} \mid \overline{\alpha}_i + \overline{\beta}_j > 0 \right\} = \min\left\{ \frac{c_{ij} - \alpha_i - \beta_j}{\overline{\alpha}_i + \overline{\beta}_j} \mid i \in I^*, j \notin J^* \right\} \tag{28}$$

	Demands			
Capacities	45	20	30	30
35	8	6	10	9
50	9	12	13	7
40	14	9	16	5

Table 2: Transportation costs for Example 33.

The updated solution, **ß**, is now given by

$$\alpha_i := \alpha_i + \theta\overline{\alpha}_i = \begin{cases} \alpha_i + \theta & \text{if } i \in I^* \\ \alpha_i - \theta & \text{if } i \notin I^* \end{cases} \tag{29}$$

$$\beta_j := \beta_j + \theta\overline{\beta}_j = \begin{cases} \beta_i - \theta & \text{if } j \in J^* \\ \beta_j + \theta & \text{if } j \notin J^* \end{cases} \tag{30}$$

Theorem 31 *For all $(i,j) \in K$ if $x_{ij} > 0$ then $\alpha_i^* + \beta_j^* = c_{ij}$.*

This theorem means that edges which carry flow at nonbreakthrough never become inadmissible in subsequent iterations. The whole of the above process may be formulated as an algorithm.

Algorithm 32 (Alphabeta) *For a transportation problem with inputs a_i, b_j and c_{ij}:*

1. *Put $\alpha_i = 0$ for $i \leqslant i \leqslant m$ and $\beta_j = \min\{c_{ij} \mid 1 \leqslant i \leqslant m\}$ for all $1 \leqslant j \leqslant n$.*
2. *Let $K = \{(i,j) \in I \times J \mid \alpha_i + \beta_j = c_{ij}\}$.*
3. *Use the Ford-Fulkerson algorithm to solve the maximum flow problem where the admissible edges are those in K.*
4. *If $\sum_{(i,j)\in K} x_{ij} = \sum_{i=1}^{m} a_i$ then $\mathbf{x}$ is an optimal solution.*
5. *Otherwise let I^* and J^* be the sets of vertices labelled at nonbreakthrough. Find θ as given by equation (28), update (**ff**, **fi**) as given by equations (29) and (30) and goto 2.*

This apparently convoluted process is in fact rather easy to follow in practise, as is shown by the following example.

Example 33 *Solve the transportation problem given in Table 25.1.4.*

Proof. Solution Initially $\mathbf{ff} = \mathbf{0}$ and β_j is the minimum of column j, so $\mathbf{fi}^T = \begin{pmatrix} 8 & 6 & 10 & 5 \end{pmatrix}$. Clearly $\alpha_i + \beta_j = c_{ij}$ in precisely those cells which determined the values of **fi** so that

$$K = \{(1,1),(1,2),(1,3),(3,4)\}$$

Hence for the first iteration the maximum flow problem is as shown in Figure **??**. Rather than run the Ford-Fulkerson algorithm beginning from the zero flow the exhibited flow is trivial to construct. Starting from the zero flow is largely to allow machines to perform the algorithm. Running the Ford-Fulkerson algorithm on this network, the algorithm terminates and $I^* = \{2,3\}$ and $J^* = \{4\}$. Hence

$$\theta = \min\left\{\frac{c_{ij} - \alpha_i - \beta_j}{2} \mid i \in I^*, j \notin J^*\right\} = -\min\left\{\frac{1}{2}, 3, 3, \frac{3}{2}, \frac{3}{2}, 3\right\} = \frac{1}{2}$$

The solution is now updated, and it is most convenient to display this in tabular form as shown in Table 25.1.4, marked as $\mathbf{ff}'$ and $\mathbf{fi}'$.

	45	20	30	30	$\mathbf{ff}'$	$\mathbf{ff}''$
35	8	6	10	9	$\frac{-1}{2}$	$\frac{-3}{2}$
50	9	12	13	7	$\frac{1}{2}$	$\frac{3}{2}$
40	14	9	16	5	$\frac{1}{2}$	$\frac{3}{2}$
$\mathbf{fi}'$	$\frac{17}{2}$	$\frac{13}{2}$	$\frac{21}{2}$	$\frac{9}{2}$		
$\mathbf{fi}''$	$\frac{15}{2}$	$\frac{15}{2}$	$\frac{21}{2}$	$\frac{7}{2}$		

Table 3: Transportation table & intermediate dual solutions for Example 33.

Now beginning the next iteration

$$K = \{(1,1),(1,2),(1,3),(2,1),(3,4)\}$$

Again the Ford-Fulkerson algorithm is run, as shown in the central diagram of Figure **??**. This gives $I^* = \{2,3\}$ and $J^* = \{1,4\}$. Hence

$$\theta = \min\left\{\frac{12-\frac{13}{2}-\frac{1}{2}}{2}, \frac{9-\frac{13}{2}-\frac{1}{2}}{2}, \frac{16-\frac{21}{2}-\frac{1}{2}}{2}\right\} = 1$$

This gives the new solution shown in Table 25.1.4 as $\mathbf{ff}''$ and $\mathbf{fi}''$. For the third iteration

$$K = \{(1,2),(1,3),(2,1),(2,3),(3,2),(3,4)\}$$

This gives rise to the right hand diagram in Figure **??** from which it is clear that $\sum_{(i,j)\in K} x_{ij} = \sum_{i=1}^{m} a_i$ and so an optimal solution is found. □

(25.2) Other Ef□diently Solvable Problems

(25.2.1) The Shortest Path Problem

Definition 34 *Let $\Gamma = (V, E, d)$ be a weighted digraph where $d\colon E \to \mathbb{R}$. The direct distances matrix of Γ is then given by*

$$a_{ij} = \begin{cases} d(v_i, v_j) & \text{if} (v_i, v_j) \in E \\ \infty & \text{if } (v_i, v_j) \notin E \text{ and } i \neq j \\ 0 & \text{if } (v_i, v_j) \notin E \text{ and } i = j \end{cases}$$

Definition 35 *Let $\Gamma = (V, E, d)$ be a weighted digraph and let $p = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ be a path in Γ. Define the weight of p, $d(p)$ as*

$$d(p) = \sum_{j=1}^{k-1} d(v_{i_j}, d_{i_{j+1}})$$

The shortest path between two vertices has the obvious definition. It is worth noting problems that may occur when negative cycles exist in a graph. Using a negative cycle it is possible to construct a path of arbitrarily low weight. As following theorems will show, the situation is thus

- The shortest path problem does not, in general, have a solution.
- The shortest elementary path problem always has a solution.
- When there are no negative cycles the shortest path problem and the shortest elementary path problem have the same solution.

Theorem 36 *In every digraph whenever a uv path exists, a shortest uv path exists.*

Proof. In an elementary path no repetition of vertices occurs, so as V is finite there are only finitely many uv paths. There must, therefore, be at least 1 shortest such path. □

Theorem 37 *In a weighted digraph with no negative cycles if a uv path exists then there is a shortest uv path, and at least one shortest uv path is elementary.*

Proof. Suppose p is a shortest uv path that is not elementary and let p' be obtained from p by removing cycles on p. Certainly p' is elementary and since edges have been removed $d(p') \leqslant d(p)$. □

Definition 38 *The shortest distances matrix A^* for a weighted digraph is defined as*

$$a^*_{ij} = \begin{cases} \text{the length of the shortest } v_iv_j \text{ path, if it exists.} \\ \infty \text{ otherwise} \end{cases}$$

Definition 39 (Shortest Distances Problem) *Given a direct distances matrix A for a weighted digraph find the shortest distances matrix A^*, or show that there exists negative cycles in the graph.*

Dijkstra's algorithm, as given in Chapter **??** provides a way to find the shortest distance between one particular vertex and all other vertices, but requires non-negative weights. By increasing the efficiency from $O(m^2)$ to $O(m^3)$ the shortest distance between all nodes can be found, and the weights can be in $\mathbb{R}$.

Define a sequence of $m \times m$ matrices $\Delta^{(p)}$ such that $\Delta^{(p)}_{ij}$ is the shortest v_iv_j path that does not pass through the vertices $v_p, v_{p+1}, \ldots, v_m$ except for v_i and v_j so that $\Delta^{(1)} = A$ and $\Delta^{(m+1)} = A^*$.

To calculate this sequence of matrices a recurrence relation is required. Let q be the shortest v_iv_j path not containing $v_{p+1}, v_{p+2}, \ldots, v_m$ (except v_i and v_j), which may be assumed to be elementary by Theorem 37. Then either

- If q passes through v_p then it is a composition of two paths: a v_iv_p path with vertices from $\{v_1, v_2, \ldots, v_{p-1}\}$ and a v_pv_j path with vertices from the same set. Hence
$$\Delta^{(p+1)}_{ij} = \Delta^{(p)}_{ip} + \Delta^{(p)}_{pj}$$
- If q does not pass through v_p then $\Delta^{(p+1)}_{ij} = \Delta^{(p)}_{ij}$.

The required recurrence is, therefore

$$\Delta^{(p+1)}_{ij} = \min\left(\Delta^{(p)}_{ij}, \Delta^{(p)}_{ip} + \Delta^{(p)}_{pj}\right) \tag{40}$$

At iteration p each element must be compared with the sum of its projection into row p and its projection into column p.

Now, a negative cycle is identified precisely when $\Delta^{(p)}_{ii} < 0$. It may therefore be assumed that if the algorithm is still running then $\Delta^{(p)}_{ii} \geqslant 0$ so that in row p the comparisons must give

$$\Delta^{(p+1)}_{pi} = \min\left(\Delta^{(p)}_{pj}, \Delta^{(p)}_{pp} + \Delta^{(p)}_{pj}\right) = \Delta^{(p)}_{pj}$$

Similarly for column p, so row and column p are copied from $\Delta^{(p)}$ to $\Delta^{(p+1)}$.

A method for finding the matrix A^* may now be constructed (this is done in Algorithm 41) but A^* only gives the weight of the shortest distance: what is the shortest path? For this purpose define the auxiliary

matrix E such that e_{ij} is the greatest index of the intermediate nodes on the shortest v_iv_j path. E may be found by initialising it to the null matrix then setting $e_{ij} = p$ if $\Delta_{ip}^{(p)} + \Delta_{pj}^{(p)} < \Delta_{ij}^{(p)}$. E is used to construct the shortest path iterating on the relationship that the shortest path between v_i and v_j contains only vertices numbered less than e_{ij}.

Algorithm 41 (Floyd) *Given a direct distances matrix A:*

1. *It at any point $\Delta_{ii} < 0$ then stop: a negative cycle has been found.*
2. *For all i and all j ($1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant m$) do $\Delta_{ij} := a_{ij}$ and $e_{ij} := 0$.*
3. *For*
   (a) $1 \leqslant p \leqslant m$
   (b) $1 \leqslant i \leqslant m$ and $i \neq p$
   (c) $1 \leqslant j \leqslant m$ and $n \neq p$

   If $\Delta_{ip} + \Delta_{pj} < \Delta_{ij}$ then $\Delta_{ij} := \Delta_{ip} + \Delta_{pj}$ and $e_{ij} := p$.

Theorem 42 *Floyd's algorithm solves the shortest path problem in $O(m^3)$ operations.*

Proof. In step 2 there are $O(m^2)$ assignments. In step 3 there are 4 operations, and parts (a), (b), and (c) there are performed $m(m-1)(m-1)$ times, giving a total of $O(m^3)$. □

Example 43 *Find the shortest v_1v_4 path in the weighted digraph with direct distances matrix*

$$A = \begin{pmatrix} 0 & 5 & 2 & 7 \\ 6 & 0 & 9 & 2 \\ 3 & 1 & 0 & 4 \\ -3 & 1 & -1 & 0 \end{pmatrix}$$

Proof. Solution Using Floyd's algorithm

$$A \to \begin{pmatrix} 0 & 5 & 2 & 7 \\ 6 & 0 & 8 & 2 \\ 3 & 1 & 0 & 4 \\ -3 & 1 & -1 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 5 & 2 & 7 \\ 6 & 0 & 8 & 2 \\ 3 & 1 & 0 & 3 \\ -3 & 1 & -1 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 3 & 2 & 3 \\ 6 & 0 & 8 & 2 \\ 3 & 1 & 0 & 3 \\ -3 & 0 & -1 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 3 & 2 & 5 \\ -1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 3 \\ -3 & 0 & -1 & 0 \end{pmatrix}$$

while

$$E \to \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 3 & 0 & 3 & 0 \\ 4 & 0 & 4 & 0 \\ 4 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

From A^* the shortest v_1v_4 path has weight 5, and from E the path is of the form $v_1, \ldots, v_3, \ldots, v_4$. But there are no intermediate vertices on the shortest v_1v_3 path (from $e_{13} = 0$) and so since $e_{34} = 2$ this path must actually be of the form $v_1, v_3, \ldots, v_2, \ldots, v_4$. But $e_{32} = 0 = e_{24}$ so the shortest v_1v_4 path is v_1, v_3, v_2, v_4. □

(25.2.2) The Greedy Algorithm

While the greedy algorithm is an obvious way to try so solve a problem, it is frequently the case that it does not work. What problems, then, does the greedy algorithm solve?

The Minimal Spanning Tree Problem

In Chapters **??** and **??** it has been shown that the greedy algorithm solves the minimal spanning tree problem.

Definition 44 (Minimal Spanning Tree Problem) *Given a weighted connected graph $\Gamma = (V, E)$ with weight function $w\colon E \to \mathbb{R}$ find a spanning tree $T = (V, E^*)$ of Γ such that $e(E^*) = \sum_{e \in E^*} w(e)$ is minimal.*

Definition 45 (Maximum Weight Spanning Forest Problem) *Given a weighted graph $\Gamma = (V, E)$ with weight function $w\colon E \to \mathbb{R}$ find a spanning forest* $T = (V, E^*)$ of Γ such that $e(E^*) = \sum_{e \in E^*} w(e)$ is maximal.*

The minimal spanning tree problem can be solved as a maximal weight forest problem by means of a few modifications to the problem. Let $\Gamma = (V, E)$ be a graph with weight function $w\colon E \to \mathbb{R}$ and let $T = (V, E^*)$ be a minimal spanning tree. Observe that

- If the new weight function w' is defined as $w'(e) = w(e) + k$ then for any spanning tree (V, E_1), $w'(E_1) = (|E_1| - 1)w(E_1)$. Therefore any minimal spanning tree remains a minimal spanning tree.
- If the new weight function w' is defined as $e'(e) = -w(e)$ then for any spanning tree (V, E_1), $w'(E_1) = -w(E_1)$. Therefore a minimal spanning tree with respect to w is maximal with respect to w'.
- Let
$$W = \max_{e \in E} w(e)$$
then if T is a minimal spanning tree with respect to W then it is a maximal spanning tree with respect to the new weight function w' defined by $w'(e) = W - w(e)$

(25.2.3) Independence Systems & Matroids

Independence Systems

Definition 46 *The pair $S = (E, \mathcal{F})$ is an independence system if*

- *E is a finite set called the background set.*
- *$\mathcal{F}$ is a family of subsets of E that is closed under inclusion, i.e. if $A \in \mathcal{F}$ and $A' \subseteq A$ then $A' \in \mathcal{F}$.*

Definition 47 *Let $\Gamma = (V, E)$ be a graph, then $M \subseteq E$ is a matching in Γ if $\forall e, f \in M\ e \cap f = \emptyset$.*

So in a matching no two edges have a common vertex. This definition is used to form a particular class of independence system: Particular classes of independence system are of interest.

Class 1: Let $\Gamma = (V, E)$ be an acyclic graph and let $\mathcal{A} = \{A \subseteq E \mid (V, A) \text{ is acyclic}\}$ then clearly $(E, \mathcal{A})$ is an independence system. Note that $|A| \leqslant |V| - p$ where Γ has p connected components.

Class 2: Let $\Gamma = (V, E)$ be a graph and let $\mathcal{M} = \{M \subseteq E \mid M \text{ is a matching}\}$ then $(E, \mathcal{M})$ is an independence system since any subset of a matching is a matching. Since in a matching there can be at most $\left\lfloor \frac{|V|}{n} \right\rfloor$ it must be the case that $|M| \leqslant \left\lfloor \frac{|V|}{n} \right\rfloor$.

Class 3: Let E be the set of all cells in an $m \times m$ matrix A. Let $I \in \mathcal{I}$ if and only if I is a set of independent cells of A, the clearly $(E, \mathcal{I}$ is an independence system. Clearly $|I| \leqslant m$.

*A forest is a collection of trees which form the connected components of a graph. A spanning forest is therefore the trees which span each connected component.

Class 4: Let A be a real $m \times n$ matrix and let C be the set of column indices of A. Define

$$\mathcal{L} = \{X \subseteq C \mid \text{columns with indices from } X \text{ are linearly independent}\}$$

then $(C, \mathcal{L})$ is an independence system. Clearly X is less than or equal to the rank of A.

Definition 48 (The Combinatorial Optimisation Problem Of An Independence System) *Let $S = (E, \mathcal{F})$ be an independence system and $w\colon E \to \mathbb{R}^+$ then the combinatorial optimisation problem associates with S is to find an independent set of the greatest total weight, i.e. to find $X^* \in \mathcal{F}$ such that*

$$w(X^*) = \max_{X \in \mathcal{F}} w(X)$$

Algorithm 49 (Greedy) *Given an independence system $S = (E, \mathcal{F})$, and weight function $w\colon E \to \mathbb{R}^+$ find $X^* \in \mathcal{F}$ such that $w(X^*)$ is maximal.*

1. $X^* := \emptyset$.
2. *Find $e \in E$ such that $w(e) = \max_{z \in E} w(z)$.*
3. $E := E \setminus \{e\}$.
4. *If $X^* \cup \{e\} \in \mathcal{F}$ then $X^* := X^* \cup \{e\}$.*
5. *If $E = \emptyset$ then stop, else goto 2.*

Theorem 50 *Let $S = (E, \mathcal{F})$ be an independence system, then the following are equivalent.*

1. *The Greedy algorithm correctly solves every combinatorial optimisation problem associated with S for any weight function.*
2. *If $F, F' \in \mathcal{F}$ and $|F'| = |F| + 1$ then there is an element $e \in F' \setminus F$ such that $F \cup \{e\} \in \mathcal{F}$. This is called the exchange proposition.*
3. *If $A \subset E$ and F and F' are maximal independent subsets of A then $|F| = |F'|$. This is called the rank proposition.*

Proof. 1⇒2 Equivalently, it is shown that $\neg 2 \Rightarrow \neg 1$. Suppose that $F, F' \in \mathcal{F}$ with $|F| = p$, $|F'| = p + 1$ and for every $e \in F' \setminus F$ $F \cup \{e\} \notin \mathcal{F}$. Define $w\colon E \to \mathbb{R}$ as follows.

$$w(e) = \begin{cases} p + 2 & \text{for } e \in F \\ p + 1 & \text{for } e \in F' \setminus F \\ 0 & \text{for } e \notin F \cup F' \end{cases}$$

Running the Greedy algorithm will then do the following

1. Accept all elements of F.
2. Reject all elements of $F' \setminus F$ because by hypothesis $\nexists e \in F' \setminus F$ such that $F \cup \{e\} \in \mathcal{F}$.
3. Possibly accept some elements of $E \setminus (F \cup F')$.

The weight of the found set, X^* is then

$$w(X^*) = p(p + 2) + 0 = p^2 = 2p$$

But

$$w(F') = (p + 1)(p + 1) = p^2 + 2p + 1 > w(X^*)$$

Since F' is an independent set this is a contradiction, so the result is shown.

2⇒3 Suppose that F and F' are maximally independent subsets of A and that $|F| < |F'|$. Take any $F'' \subseteq F'$ such that $|F''| = |F| + 1$. Since $F' \in \mathcal{F}$, $F'' \in \mathcal{F}$ and by hypothesis $\exists e \in F'' \setminus F$ such that $F \cup \{e\} \in \mathcal{F}$. Certainly $F \cup \{e\} \subseteq A$ and $F \neq F \cup \{e\}$, but this contradicts that F was maximally independent.

3⇒1 Let $w : e \to \mathbb{R}$ be an arbitrary weight function and suppose that the Greedy algorithm finds the solution $F = \{e_1, e_2, \ldots, e_r\}$. Let $F' = \{e'_1, e'_2, \ldots, e'_s\}$ be an optimal solution—one exists by the finiteness of E.

By construction F is maximal independent, and although F' may not be maximal it is independent and only edges of zero weight can be added. Without loss of generality assume, therefore, that F' is extended in this way to be a maximal independent set. Hence by hypothesis $|F| = |F'|$, so $r = s$. Certainly $w(F') \geqslant w(F)$ because F' is optimal and so has maximal weight. Without loss of generality assume that

$$\begin{aligned} w(e_1) \leqslant w(e_2) \leqslant \cdots \leqslant w(e_r) \\ \text{and } w(e'_1) \leqslant w(e'_2) \leqslant \cdots \leqslant w(e'_r) \end{aligned} \tag{51}$$

Now, since e_1 is chosen by the Greedy algorithm is must have maximal weight, and therefore $w(e_1) \geqslant w(e'_1)$. Suppose that $w(e_i) \geqslant w(e'_i)$ for $1 \leqslant i \leqslant k-1$.

- Suppose that

$$w(e_k) < w(e'_k) \tag{52}$$

and let

$$A = \{e \in E \mid w(e) \geqslant w(e'_k)\}$$

By the induction hypothesis and equations (51) $|A| \geqslant k$. Let $F'' = \{e_1, e_2, \ldots, e_{k-1}\}$ then by the induction hypothesis $F'' \subseteq A$. Furthermore F'' is independent because $F'' \subset F$ and $F \in \mathcal{F}$.
Now, $F' \cap A \subseteq F'$ and so is independent.

  * Suppose that F'' is not a maximally independent subset of A then $\exists e \notin F''$ such that $F'' \cup \{e\} \in \mathcal{F}$ and is also a subset of A. But therefore $e \in A$ and so by the definition of A, $w(e) \geqslant w(e'_k) > w(e_k)$ by equation (52).

    Let $\tilde{F}$ be the subset of F'' at the stage when e was processed. $\tilde{F} \cup \{e\} \subseteq F'' \cup \{e\} \in \mathcal{F}$ and so by independence $\tilde{F} \cup \{e\} \in \mathcal{F}$. Since $w(e) > w(e_k)$ and e is feasible e would have been chosen by the Greedy algorithm in preference to e_k. But e_k was chosen, so this is a contradiction meaning that F'' must be maximally independent.

  Now, F'' is maximally independent and $|F''| = k-1$. But $F' \cap A$ is an independent set and $|F' \cap A| \geqslant k$ which contradicts the hypothesis. Therefore the assumption of equation (52) must be false, and hence $w(e_k) \geqslant w(e'_k)$.

Since $w(e_k) \geqslant w(e'_k)$ the induction holds and so equations (51) show that $w(F) \geqslant w(F')$ and hence the solution found by the Greedy algorithm is optimal. □

Definition 53 *Let $S = (E, \mathcal{F})$ be an independence system.*

1. *An independence system satisfying one (and hence all) of the conditions of Theorem 50 is called a matroid.*
2. *For $A \subseteq E$ the rank of A, $r(A)$, is the size of the maximally independent subsets of A.*
3. *A maximally independent subset of a set $A \subseteq E$ is called a basis for A.*

Each of the 4 classes of independence system described earlier can now be examined for being a matroid.

1. Let $H \subset E$ be maximally independent, then it corresponds to a maximal acyclic subgraph. But this is precisely ("if and only if") a spanning forest which always has $|V| - p$ elements where the forest has p connected components. Hence rank proposition holds, so this class of independence system is a matroid. This kind of matroid is denoted $\mathrm{Gr}(\Gamma)$.
2. Consider the graphs shown in Figure **??**. From the graph on the left take the subgraph on the right, which has two matchings as shown. Both matchings are maximally independent in this subgraph but contain differently many elements. Hence the rank proposition does not hold and this is not a matroid.
3. Consider a 4×4 matrix with ij entry a_{ij}. Taking the subset of entries $\{a_{11}, a_{12}, a_{21}\}$ there are two maximal independent subsets $\{a_{11}\}$ and $\{a_{12}, a_{21}\}$ that are of unequal size. Hence the rank proposition does not hold, so this is not a matroid.
4. From linear algebra, every maximally independent set of columns has the same size, namely the rank of the matrix. So by the rank proposition this is a matroid.

Definition 54 *Let A be an $m \times n$ matrix over a field K and let A' be the set of column indices of A. Then where*

$$\mathcal{L}_A = \{X \subseteq A' \mid \textit{columns with indices from } X \textit{ are linearly independent}\}$$

the independence system $(A', \mathcal{L}_A)$ is a matroid, denoted $M(A)$.

Definition 55 *Independence systems $S + (E, \mathcal{F})$ and $S' = (E', \mathcal{F}')$ are isomorphic, written $S \approx S'$, if there exists a bijection $\phi \colon E \to E'$ such that $F \in \mathcal{F} \Leftrightarrow \phi(F) \in \mathcal{F}'$.*

Definition 56 *A matroid S is called a matric matroid if there exists a matrix A over an appropriate field for which $S \approx M(A)$. Similarly S is said to be a graphic matroid if there exists a graph Γ such that $S \approx \mathrm{Gr}(\Gamma)$.*

(25.3) Hard Problems

(25.3.1) Classifying Problems As 'Hard'

When assessing the computational complexity of algorithms the complexity is expressed as a function of the size of the input to the algorithm. If the function is polynomial then the problem is efficiently solvable. Exponential and factorial expressions increase so quickly that increasing the length of input causes a remarkable increase in the operations needed.

Efficient therefore means polynomial and hard means not polynomial. For a particular problem it is possible that no polynomial algorithm is known, but this does not mean that no polynomial algorithm exists. An example of this is the problem of finding a Hamiltonian cycle in a graph. In fact it cannot even be shown that no polynomial algorithm exists. There follows definitions of problems that currently have no known polynomial algorithm.

Definition 57 (Integer Linear Programming) *Given an integer matrix A and integer vectors $\mathbf{b}$ and $\mathbf{c}$ find an integer vector $\mathbf{x}$ which solves the optimisation problem*

$$\begin{aligned} \mathbf{c}^T\mathbf{x} &\to \max \\ A\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geqslant 0 \end{aligned}$$

This problem reduces to normal linear programming when the integer requirement is omitted. The integer problem remains hard even when A has only one row.

Definition 58 (Integer Knapsack) *Given integers $a_1, a_2, \ldots, a_n$ and K find integers $x_1, x_2, \ldots, x_n$ such that $\sum_{i=1}^{n} a_i x_i = K$.*

The problem can be simplified further by considering only $x_i \in \{0, 1\}$, so the problem is to find $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i = K$. This is still a hard problem which is called 0-1 Knapsack.

Definition 59 (Satisfiability Of Boolean Functions) *A boolean formula $f(x_1, x_2, \ldots, x_n)$ is satisfiable if there exists $x_1^*, x_2^*, \ldots, x_n^* \in \{0, 1\}$ such that $f(x_1^*, x_2^*, \ldots, x_n^*) = 1$.*

Note that is is required that f be in conjunctive normal form, so that $f = c_1 \wedge c_2 \wedge \cdots \wedge c_k$ where c_j is a function of $x_1, x_2, \ldots, x_n$ that involves only 'or' and negation and at most one of each of x_i and $\neg x_i$ is used.

(25.3.2) Class $\mathcal{NP}$

Computational Complexity

The computational complexity of an algorithm may be expressed as a function of either the number of parameters in the input, L_1, or the number of symbols in the input, L_2. Needless to say $L_2 \geqslant L_1$, and moreover where $\mathcal{A}$ is an algorithm and

$$cc(\mathcal{A}) = f(L_1) = g(L_2)$$

then for all x, $g(x) \leqslant f(x)$.

- If f is bounded by a polynomial function then $\mathcal{A}$ is called strongly polynomial.
- If g is bounded by a polynomial function then $\mathcal{A}$ is called polynomial.

In order to simplify proceedings consider only problems with integer entries.

Recognition Problems

Recognition problems have output that is either "yes" or "no". The satisfiability of a boolean formula is clearly such a formula, as is determining whether a graph contains a Hamilton cycle. However the maximum clique problem is not a recognition problem.

Most problems have a recognition version, i.e. a version that is a recognition problem. This is generally done by asking "does a solution exist?" so for example the maximum clique problem transforms to the following

- Given a graph Γ and $k \in \mathbb{N}$ does there exist a subgraph Γ^* of Γ with r vertices where $r \geqslant k$.

In order to simplify proceedings consider only recognitions.

Definition 60 *Let $\mathcal{R}$ be the class of recognition problems with integer entries. Furthermore define for a problem $A \in \mathcal{R}$*

- $\text{inst}(A)$ *to be the class of all instances of A.*
- $\text{tinst}(A)$ *to be the class of all true instances of A i.e. those instances where the solution is "true".*
- $\text{alg}(A)$ *to be the set of all algorithms that correctly solve A.*
- Σ *to be the alphabet for the problem A, so that all problems, instances and algorithms are expressed as strings of symbols from Σ.*

- Σ^* *to be the set of all strings of symbols of* Σ.
- $|x|$ *to be the length of the string* $x \in \Sigma^*$.
- \$ *to be a distinguished element of* Σ.

If $\mathcal{A} \in \text{alg}(A)$ *then let* $\text{op}(\mathcal{A}, x)$ *be the number of operations necessary to perform* $\mathcal{A}$ *when applied to* x.

Definition 61 *For* $A \in \mathcal{R}$ *and* $\mathcal{A} \in \text{op}(A)$ *define*

- $cc(\mathcal{A}) = \sup\{\text{op}(\mathcal{A}) \mid x \in \text{inst(A)}, |x| = L\} = f(L)$.
- $cc(A) = \inf_{\mathcal{A} \in \text{alg}(A)} cc(\mathcal{A})$.

An algorithm $\mathcal{A}$ *is called polynomial of there exists a polynomial* p *such that* $f(z) = O(p(z))$, *so define the class of polynomially solvable problems*

$$\mathcal{P} = \{A \in \mathcal{R} \mid \exists \mathcal{A} \in \text{alg}(A) \text{ with } \mathcal{A} \text{ polynomial}\}$$

Checking Solutions

Even if a problem is hard it may be very easy to verify a solution. To determine whether $2^{67} - 1$ is prime is very hard indeed, yet verifying the solution

$$2^{67} - 1 = 193,707,121 \times 761,838,257,287$$

is quite easy, and can certainly be done polynomially. To simplify the proceedings consider only problems to which a solution can be checked by a polynomial algorithm.

The problems satisfying the three restrictions are called the problems of class $\mathcal{NP}$.

Definition 62 *A problem is of class* $\mathcal{NP}$ *if it obeys the following criteria.*

$$\mathcal{NP} = \left\{ A \in \mathcal{R} \mid \exists p \quad \exists \mathcal{C} \quad \forall x \in \Sigma^* \quad x \in \text{tinst}(A) \Leftrightarrow \exists c(x) \in \Sigma^*, |c| \leqslant p(|x|) \text{ with } x\$c(x) \xrightarrow{\mathcal{C}} \text{“true”} \right\}$$

In this rather intricate definition $c(x)$ is called a certificate and $\mathcal{C}$ is the (polynomial) algorithm that checks that $c(x)$ solves A. $\mathcal{C}$ must be supplies with both an input and an output that it may verify the validity of, so $c(x)$ is the output which depends on the input x. The expression “$x\$c(x)$” is the concatenation of these strings, separated by the distinguished character: the input taken by $\mathcal{C}$.

Theorem 63 *The problem to satisfy a boolean formula in conjunctive normal form is of class* $\mathcal{NP}$.

Proof. Let $C(x_1, x_2, \ldots, x_n)$ be a boolean formula in conjunctive normal form that is satisfiable, observe that it may have up to $2n$ literals. If there are k such clauses in a formula then the length of the input is $L = 2kn$. Now

$$C \in \text{sat}() \Leftrightarrow \exists x_1^*, x_2^*, \ldots, x_n^* \in \{0,1\} \text{ such that } C(x_1^*, x_2^*, \ldots, x_n^*) = 1$$

The certificate is therefore $x_1^*, x_2^*, \ldots, x_n^*$ which is of length $O(n) \leqslant L$. The verification algorithm simply substitutes these values in and evaluates the resulting expression. Each clause will require at most $2n - 1$ additions, n negations, $k - 1$ clauses will then have to be multiplied, and finally a check for equality to 1 will be made. This gives

$$cc(\mathcal{C}) = k(2n - 1) + nk + k - 1 + 1 = 3nk = \frac{3}{2}L$$

Hence $p(z) = \frac{3}{2}z$ and the definition is satisfied. □

Theorem 64 *The problem to detect the presence of a Hamilton cycle in a graph is of class $\mathcal{NP}$.*

Proof. The input is a graph $\Gamma = (V, E)$ with $|V| = m$ say. This will be encoded by the adjacency matrix, so that $L = m^2$. Now, $\Gamma \in$ tinst() if and only if a Hamilton cycle exists in Γ.

Take as the certificate, therefore, a sequence of indices for the vertices, $(i_1, i_2, \ldots, i_m)$ such that the sequence of corresponding vertices forms a Hamilton cycle. This certificate is of length $m < L = m^2$. The algorithm $\mathcal{C}$ must check that the certificate is an m-tuple of distinct numbers from 1 to m, and that $(v_{i_1}, v_{i_2}, \ldots, v_{i_m})$ is a Hamilton cycle by checking that the appropriate edges exist. This may be done as follows

1. (a) Check $i_1 \neq i_2$, $i_1 \neq i_3$ etc. which takes $m - 1$ comparisons.
   (b) Check $i_2 \neq i_3$, $i_2 \neq i_4$ etc. which takes $m - 2$ comparisons.
   (c) Continue, giving a total of $\sum_{i=1}^{m-1} i = \frac{1}{2}m(m-1)$ comparisons.
2. For $1 \leqslant r \leqslant m - 1$ check that $\{v_{i_r}, v_{i_{r+1}})\} \in E$.
3. Check that $\{v_m, v_1\} \in E$.

This gives

$$cc(\mathcal{C}) = O(m^2) + O(m) = O(m^2) = O(L)$$

Hence $p(z) = z$ and the definition is satisfied. □

Similarly the maximum clique problem, the travelling salesman problem, and the 0-1 knapsack problem are all of class $\mathcal{NP}$.

Theorem 65 $\mathcal{P} \subseteq \mathcal{NP}$.

It is not clear as to whether certain problems are of class $\mathcal{NP}$, for example whether a number is prime.

Conjecture 66 $\mathcal{P} \neq \mathcal{NP}$