Department of Mathematics and Computer Science
University of Southern Denmark, Odense

March 22, 2024
Marco Chiarandini

DM545/DM871 – Linear and integer programming

Sheet 8, Spring 2024

Solution:

Included.

Exercises with the symbol $^+$are to be done at home before the class. Exercises with the symbol *will be tackled in class and should be at least read at home. The remaining exercises are left for self training after the exercise class.

Exercise 1$^+$ [DT97]

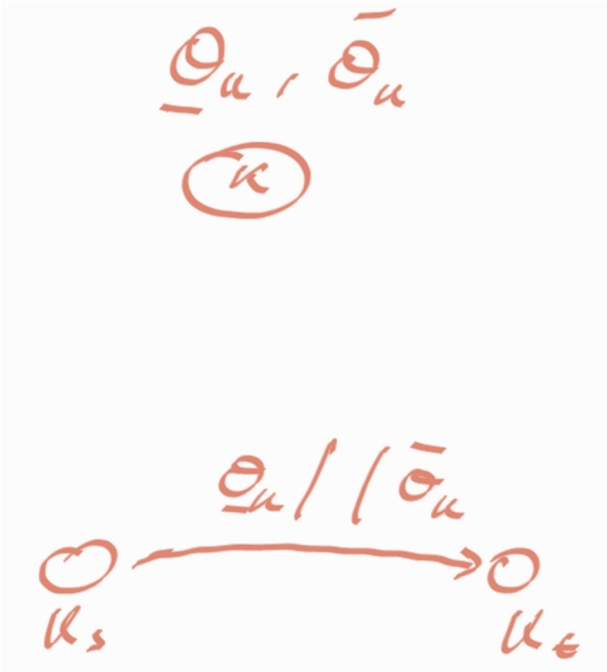
Suppose that in a minimum cost flow problem restrictions are placed on the total flow leaving a node k, i.e.

$$\underline{\theta}_k \leq \sum_{(k,j)\in E} x_{kj} \leq \overline{\theta}_k$$

Show how to modify these restrictions to convert the problem into a standard cost flow problem.

Solution:

Bounds on nodes are not possible in our model of min cost flow. However, we can transform the network by splitting the vertex in two and introducing an arc between the new vertices with the given bounds.

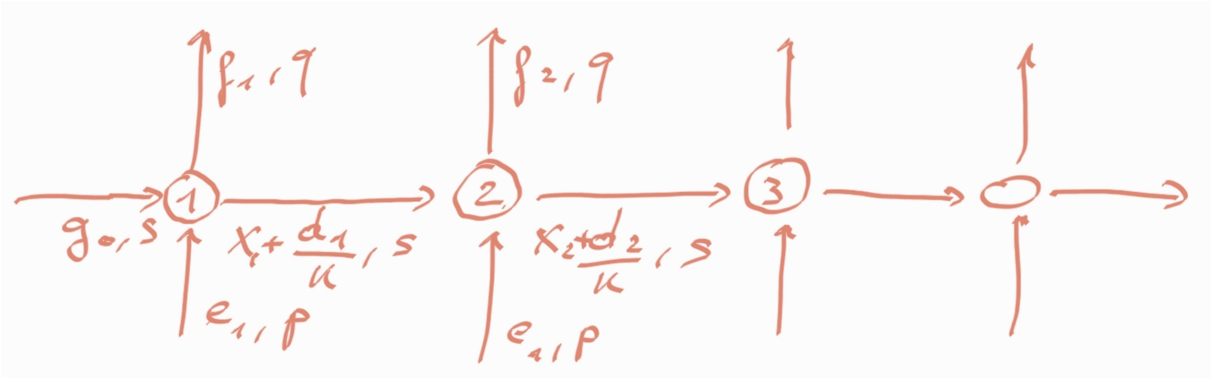


Exercise 2$^+$

The production plan of a factory for the next year is to produce d_t units of product per month t, $t = 1, \ldots, 12$. Each worker can produce k units of product in a month. The monthly salary is equal to s. Employing and firing personnel has costs: precisely, employing one person costs p while firing one costs q. Assuming that initially there are g_0 workers, determine the number of workers that must be present during every month such that the demand is always satisfied and the overall costs of salary, employment, and firing are minimized.

Solution:

It is possible to model the employment and firing of workers as a flow in a network.



Exercise 3^+ Directed Chinese Postman Problem

Suppose a postman has to deliver mail along all the streets in a small town. Assume furthermore that on one-way streets the mail boxes are all on one side of the street, whereas for two-way streets, there are mail boxes on both sides of the street. For obvious reasons the postman wishes to minimize the distance he has to travel in order to deliver all the mail and return home to his starting point. Show how you can solve this problem using minimum cost flows. A similar model can be formulated for the Snow Plow problem or the Salt Spreding problem.

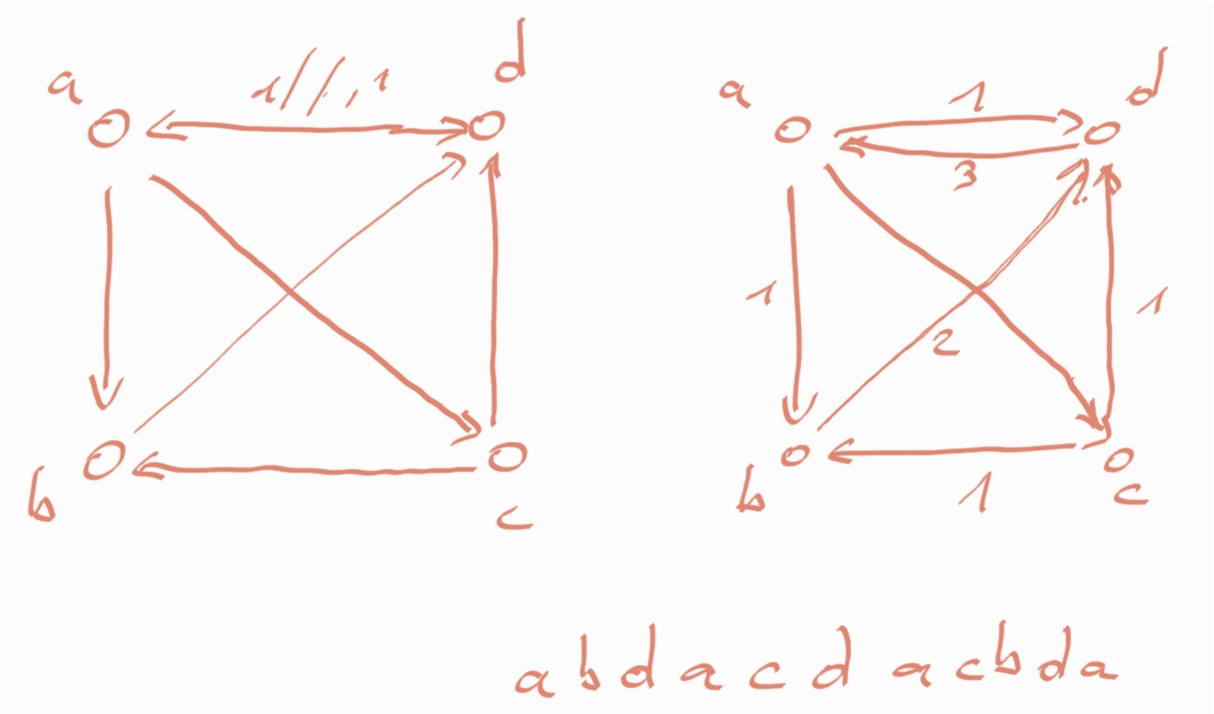
Solution:

The solution to this problem can be found on page 174 of J. Bang-Jensen and G. Gutin. Digraphs: Theory, Algorithms and Applications, Springer London, 2009 http://dx.doi.org/10.1007/978-1-84800-998-1_4. See the extract enclosed in the next pages.
In short, the solution to the Chinese postman problem is a min cost flow that traverses each arc at least once in a network where arcs are streets and nodes are street instersections.
We must assume that the graph is strongly connected otherwise there is no solution (any closed walk is strongly connected.)
A digraph where there exists a walk that visits arcs exactly once is called Eulerian.

$$N = (V, A, \mathbf{l}, \mathbf{u} = \infty, \mathbf{c})$$

$$x_{ij} = \#\text{ of times arc is used}$$

The cost os a min cost circulation in N equals the cost of a Chinese postman walk in D.



The case $|X| = |Y|$ in Hall's theorem, the so-called marriage theorem, was proved much earlier (in 1917) by Frobenius.

Corollary 4.11.4 (The Marriage theorem) *[363] A bipartite graph $B = (X, Y; E)$ has a matching covering X if and only if $|X| = |Y|$ and (4.25) holds.* □

4.11.2 The Directed Chinese Postman Problem

Suppose a postman has to deliver mail along all the streets in a small[14] town. Assume furthermore that on one-way streets the mail boxes are all on one side of the street, whereas for two-way streets, there are mail boxes on both sides of the street. For obvious reasons the postman wishes to minimize the distance he has to travel in order to deliver all the mail and return home to his starting point. We show below how to solve this problem in polynomial time using minimum cost flows.

We can model the problem by a directed graph $D = (V, A)$ and a cost function $c : A \to \mathbb{R}_+$ where V contains a vertex for each intersection of streets in the town and the arcs model the streets. A 2-cycle corresponds to a two-way street and an arc which is not in a 2-cycle corresponds to a one-way street in the obvious way. The cost of an arc corresponds to the length of the corresponding street. Now it is easy to see that an optimal route for the postman corresponds to a minimum cost closed walk in D which traverses each arc at least once.

We have seen in Theorem 1.7.2 that if a digraph is eulerian, then it contains a closed trail which covers all arcs precisely once. Thus if D is eulerian, the optimal walk is simply a eulerian trail in D (using each arc exactly once). Below we show how to solve the general case by reducing the problem to a minimum cost circulation problem. First observe that there is no solution to the problem if D is not strongly connected, since any closed walk is strongly connected as a digraph. Hence we assume below that the digraph in question is strong, a realistic assumption when we think of the postman problem.

Let $D = (V, A)$ be a strong digraph and let c be a cost function on A. The cost $c(W)$ of a walk W is $\sum_{ij \in A} c_{ij} w_{ij}$ where w_{ij} denotes the number of times the arc ij occurs on W. Define $\mathcal{N}$ as the network $\mathcal{N} = (V, A, l \equiv 1, u \equiv \infty, c)$, that is, all arcs have lower bounds one, capacity infinity and cost equal to the cost on each arc.

Theorem 4.11.5 *The cost of a minimum cost circulation in $\mathcal{N}$ equals the minimum cost of a Chinese postman walk in D.*

Proof: Suppose W is a closed walk in D which uses each arc $ij \in A$ exactly $w_{ij} \geq 1$ times. Then it is easy to see that we can obtain a feasible circulation of cost $c(W)$ in $\mathcal{N}$ just by sending w_{ij} units of flow along each arc $ij \in A$.

[14] This assumption is to make sure that the postman can carry all the mail in his backpack, say. Without this assumption the problem becomes much harder.

Conversely, suppose x is an integer feasible circulation in $\mathcal{N}$. Form a directed multigraph $D' = (V, A')$ by letting A' contain x_{ij} copies of the arc ij for each $ij \in A$. It follows from the fact that x is an integer circulation that D' is an eulerian directed multigraph (see Figure 4.18). Hence, by Theorem 1.7.2, D' has an eulerian tour T. The tour T corresponds to a closed walk W in D which uses each arc at least once and clearly we have $c(W) = c^T x$. □

Figure 4.18 An instance of the directed Chinese postman problem. Part (a) shows a digraph with cost 1 (not shown) on every arc. Part (b) shows the values of a minimum cost circulation in the corresponding network. This circulation corresponds to the postman tour *abdacdacbda*.

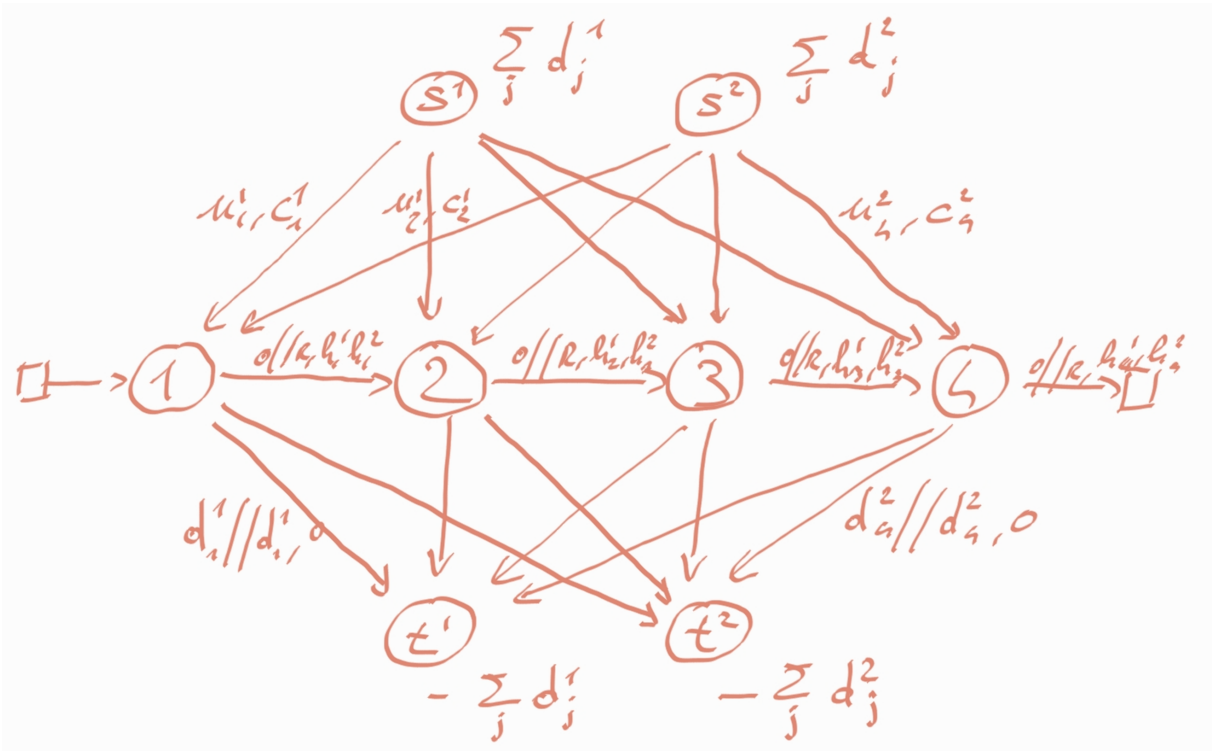
The corresponding problem can be considered for a connected edge-weighted undirected graph $G = (V, E)$. Here the goal is to find a tour which traverses all edges in E at least once and minimizes the total weight of the tour. It is not hard to see that this problem is equivalent to finding a minimum cost subset of E so that by duplicating these edges we obtain a supergraph G^* of G in which all vertices have even degree. This is equivalent to G^* having an orientation as a strongly connected eulerian directed multigraph. Edmonds and Johnson showed how to solve this problem via matching techniques [287] . Finally there is also the MIXED CHINESE POSTMAN PROBLEM (MCPP) in which the input is a mixed graph $M = (V, A, E)$ with weights on the arcs and edges. Again the goal is to find minimum cost subsets $E' \subseteq E$ and $A' \subseteq A$ so that duplicating these edges and arcs in M results in a mixed supergraph M^* of M which can be oriented as a strongly connected eulerian directed multigraph. A necessary and sufficient condition for a mixed graph to have an orientation as an eulerian directed multigraph is given in Corollary 11.7.4. The MCCP is NP-hard [741]. From the point of view of practical relevance the MCPP is probably the most important of the three variants as it directly models situations such as garbage collection, snow removal, street sweeping, etc. For an exact algorithm for the MCPP see, e.g., Nobert and Picard [730].

Exercise 4* Warehousing of Seasonal Products

A company manufactures multiple products. The products are seasonal with demand varying weekly, monthly, or quarterly. To use its work-force and capital equipment efficiently, the company wishes to "smooth" production, storing pre-season production to supplement peak-season production. The company has a warehouse with fixed capacity R that it uses to store all the products it produces. Its decision problem is to identify the production levels of all the products for every week, month, or quarter of the year that will permit it to satisfy the demands incurring the minimum possible production and storage costs.
We can represent this warehousing problem as a relevant generalization of the min cost network flow problem encountered in the course.
For simplicity, consider a situation in which the company makes two products and then it needs to schedule its production for each of the next four quarters of the year. Let d_j^1 and d_j^2 denote the demand for products 1 and 2 in quarter j. Suppose that the production capacity for the jth quarter is u_j^1 and u_j^2, and that the per unit cost of production for this quarter is c_j^1 and c_j^2. Let h_j^1 and h_j^2 denote the storage (holding) costs per unit of the two products from quarter j to quarter $j+1$.
Represent graphically the network in the two products four periods case and write the Linear Programming formulation of the problem. Which network flows problem models this application? If all input data are integer, will the solution be integer?

Solution:

We can model this problem as a multicommodity flow in a network. The network has 4 nodes, one for each quarter, two target nodes for each quarter (one per product), two nodes for each plant and for each quarter (one per product) and two global sources for the two products.



See page 655 of [AMO].

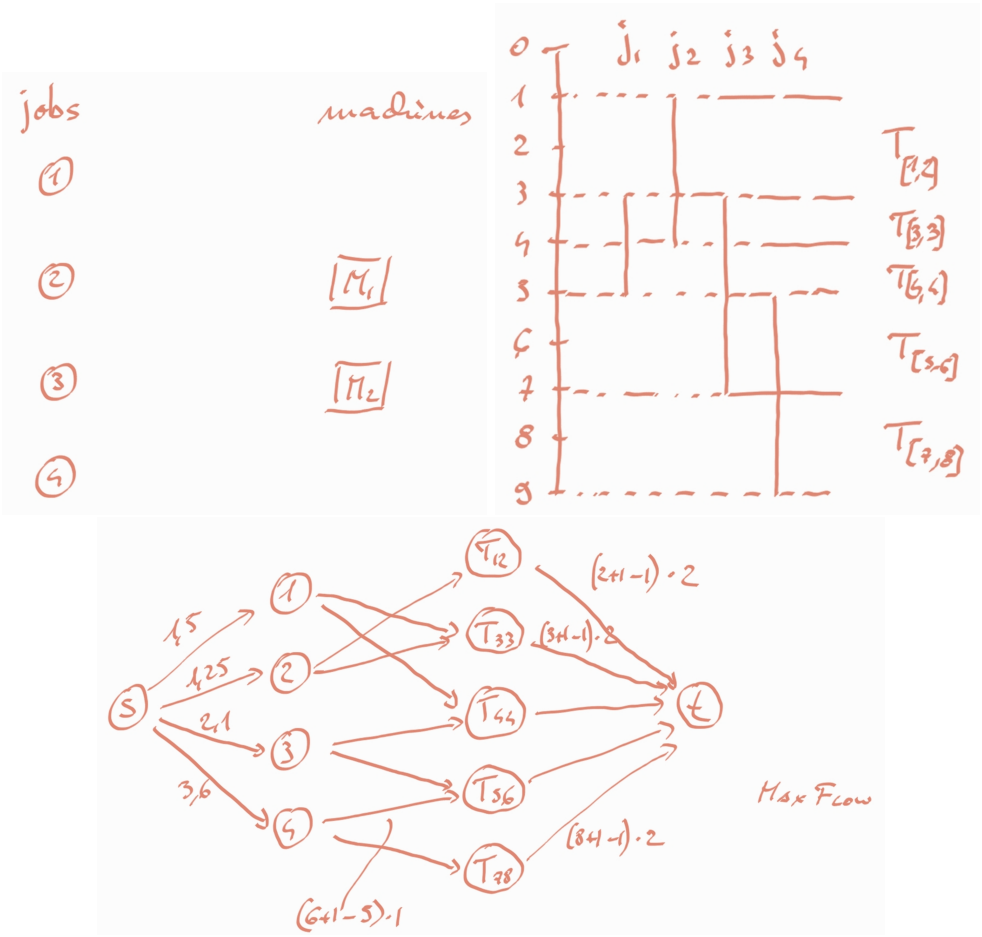
Exercise 5* Scheduling on Uniform Parallel Machines

We consider scheduling a set J of jobs on M uniform parallel machines. Each job $j \in J$ has a processing requirement p_j (denoting the number of machine days required to complete the job), a release data r_j (representing the beginning of the day when job j becomes available for processing), and a deadline $d_j \geq r_j + p_j$ (representing the beginning of the day by which the job must be completed). We assume that a machine can work on only one job at a time and that each job can be processed by at most one machine at a time. However we allow preemptions (ie, we can interrupt a job and process it on different machines on different days). The scheduling problem is to determine a feasible schedule that completes all jobs before their due dates or to show that no such schedule exists.
Formulate the feasible scheduling problem as a maximum flow problem.
It may help to consider an example of the problem, for example, the one given in Table 1.

Job (j)	1	2	3	4
Processing time (p_j)	1.5	1.25	2.1	3.6
Release time (r_j)	3	1	3	5
Due date (d_j)	5	4	7	9

Table 1:

Solution:



See explanation in the solution of the next exercise.

Exercise 6* Tanker Scheduling Problem

A steamship company has contracted to deliver perishable goods between several different origin-destination pairs. Since the cargo is perishable the customers have specified precise dates (ie, delivery dates) when the shipments must reach their destinations. (The cargoes may not arrive early or late).

The steamship company wants to determine the minimum number of ships needed to meet the delivery dates of the shiploads.
Formulate this problem as a network flow problem modeling the example in Table 2 with four shipments. Each shipment is a full shipload with the characteristics shown in Table 2. For example, as specified by the first row in this figure, the company must deliver one shipload available at port A and destined for port C on day 3.

shipment	origin	destination	delivery date
1	Port A	Port C	3
2	Port A	Port C	8
3	Port B	Port D	3
4	Port B	Port C	6

	C	D
A	3	2
B	2	3

	A	B
C	2	1
D	1	2

Table 2: Data for the tanker scheduling problem: Left shipment characteristics; Center, shipment transit times; Right return times.

Solution:

Solutions for this and the previous exercise are from [AMO] pages 170-176, enclosed in the next pages.

For Exercise 6 the [AMO] book says that we solve a minimum value problem using a maximum flow algorithm. The minimum flow problem is very similar to the maximum flow problem: instead of upper bounds we have lower bounds to flow on arcs and instead of maximizing we are asked to minimize the flow circulating in the network. More precisely, let $N = (V, A, \mathbf{l}, \mathbf{u})$ be a network with source s, sink t and non-negative lower bounds on the arcs. A minimum feasible (s,t)-flow in N is a feasible (s,t)-flow whose value is minimum possible among all feasible (s,t)-flows.
In mathematical programming terms a formulation of this problem is very similar to a max flow problem:

$$
\begin{aligned}
z = \max\ & x_{ts} \\
& \sum_{j:ji\in A} x_{ij} - \sum_{j:ij\in A} x_{ji} = 0 && \forall i \in V \\
& x_{ij} \le u_{ij} && \forall ij \in A \\
& x_{ij} \ge 0 && \forall ij \in A
\end{aligned}
\qquad\qquad
\begin{aligned}
z = \min\ & x_{ts} \\
& \sum_{j:ji\in A} x_{ij} - \sum_{j:ij\in A} x_{ji} = 0 && \forall i \in V \\
& (x_{ij} \le u_{ij}) && \forall ij \in A \\
& x_{ij} \ge \ell_{ij} && \forall ij \in A
\end{aligned}
$$

What follows is a digression outside of the scope of this course.

Algorithmically, the minimum value problem is solved by solving two the maximum flow problems.

1. Suppose x is a feasible (s,t)-flow. Assume $|x| > 0$ else there is a trivial solution to the problem with $|x| = 0$. We can find x by applying one of the transformations seen in class to remove lower bounds. The transformation modifies the upper bounds and introduces balancing terms at some nodes. Then, we can find a feasible flow, not necessarily the one of minimum value, by solving a max flow problem in this transformed network (with the additional element of having to satisfy the balance restrictions at the nodes).

2. Let y be the maximum (t,s)-flow in $N(x)$ (hence, a flow backward). Let $(T, \overline{T})$ be the min (t,s)-cut and $r(T, \overline{T}) = |y|$ be the capacity of the cut.

$$
\begin{aligned}
|y| = r(T,\overline{T}) &= \\
&= \sum_{ij\in(T,\overline{T})} (u_{ij} - x_{ij}) + \sum_{pq\in(\overline{T},T)} (x_{pq} - \ell_{pq}) \\
&= u(T,\overline{T}) - \ell(\overline{T},T) + x(T,\overline{T}) - x(\overline{T},T) = \\
&= u(T,\overline{T}) - \ell(\overline{T},T) + |x|
\end{aligned}
$$

From where:

$$|x| - |y| = \ell(\overline{T},T) - u(T,\overline{T})$$

$|x| - |y|$ is the flow value obtained by the composition of the two flows x and y. On the other hand, noting that $s \in \overline{T}$ and $t \in T$, the quantity $\ell(\overline{T}, T) - u(T, \overline{T})$ identifies the demand of the network, that is, how much flow has at least to got from s to t. Then, the equality between left and right hand side, indicates that the composite flow of x and y (in rough terms, x subtracted y), is the minimum feasible flow satisfying the demand of the network.

For more details you may consult J. Bang-Jensen and G. Gutin. Digraphs: Theory, Algorithms and Applications, Springer London, 2009 (http://dx.doi.org/10.1007/978-1-84800-998-1_4) from page 185.

The network also contains a source node s and a sink node t. It contains an arc (s, C_i) for each node C_i denoting a club, an arc (C_i, R_j) whenever the resident R_j is a member of the club C_i, and an arc (R_j, P_k) if the resident R_j belongs to the political party P_k. Finally, we add an arc (P_k, t) for each $k = 1, \ldots, 3$ of capacity u_k; all other arcs have unit capacity.

We next find a maximum flow in this network. If the maximum flow value equals q, the town has a balanced council; otherwise, it does not. The proof of this assertion is easy to establish by showing that (1) any flow of value q in the network corresponds to a balanced council, and that (2) any balanced council implies a flow of value q in the network.

This type of model has applications in several resource assignment settings. For example, suppose that the residents are skilled craftsmen, the club C_i is the set of craftsmen with a particular skill, and the political party P_k corresponds to a particular seniority class. In this instance, a balanced town council corresponds to an assignment of craftsmen to a union governing board so that every skill class has representation on the board and no seniority class has a dominant representation.

Application 6.3 Matrix Rounding Problem

This application is concerned with consistent rounding of the elements, row sums, and column sums of a matrix. We are given a $p \times q$ matrix of *real* numbers $D = \{d_{ij}\}$, with row sums α_i and column sums β_j. We can round any real number a to the next smaller integer $\lfloor a \rfloor$ or to the next larger integer $\lceil a \rceil$, and the decision to round up or down is entirely up to us. The matrix rounding problem requires that we round the matrix elements, and the row and column sums of the matrix so that the sum of the rounded elements in each row equals the rounded row sum and the sum of the rounded elements in each column equals the rounded column sum. We refer to such a rounding as a *consistent rounding*.

We shall show how we can discover such a rounding scheme, if it exists, by solving a feasible flow problem for a network with nonnegative lower bounds on arc flows. (As shown in Section 6.7, we can solve this problem by solving two maximum flow problems with zero lower bounds on arc flows.) We illustrate our method using the matrix rounding problem shown in Figure 6.2. Figure 6.3 shows the maximum flow network for this problem. This network contains a node i corresponding to each row i and a node j' corresponding to each column j. Observe that this network

				Row sum
	3.1	6.8	7.3	17.2
	9.6	2.4	0.7	12.7
	3.6	1.2	6.5	11.3
Column sum	16.3	10.4	14.5	

Figure 6.2 Matrix rounding problem.

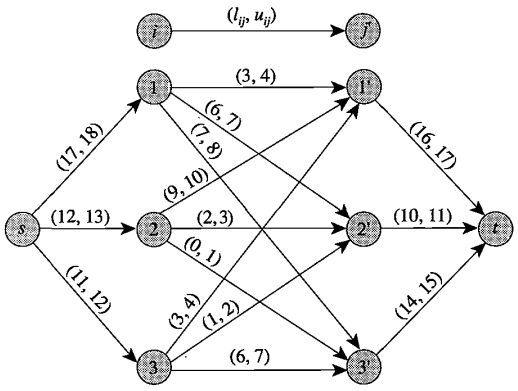


Figure 6.3 Network for the matrix rounding problem.

contains an arc (i, j') for each matrix element d_{ij}, an arc (s, i) for each row sum, and an arc (j', t) for each column sum. The lower and the upper bounds of each arc (i, j') are $\lfloor d_{ij} \rfloor$ and $\lceil d_{ij} \rceil$, respectively. It is easy to establish a one-to-one correspondence between the consistent roundings of the matrix and feasible flows in the corresponding network. Consequently, we can find a consistent rounding by solving a maximum flow problem on the corresponding network.

This matrix rounding problem arises in several application contexts. For example, the U.S. Census Bureau uses census information to construct millions of tables for a wide variety of purposes. By law, the bureau has an obligation to protect the source of its information and not disclose statistics that could be attributed to any particular person. We might disguise the information in a table as follows. We round off each entry in the table, including the row and column sums, either up or down to a multiple of a constant k (for some suitable value of k), so that the entries in the table continue to add to the (rounded) row and column sums, and the overall sum of the entries in the new table adds to a rounded version of the overall sums in the original table. This Census Bureau problem is the same as the matrix rounding problem discussed earlier except that we need to round each element to a multiple of $k \geq 1$ instead of rounding it to a multiple of 1. We solve this problem by defining the associated network as before, but now defining the lower and upper bounds for any arc with an associated real number α as the greatest multiple of k less than or equal to α and the smallest multiple of k greater than or equal to α.

Application 6.4 Scheduling on Uniform Parallel Machines

In this application we consider the problem of scheduling of a set J of jobs on M uniform parallel machines. Each job $j \in J$ has a processing requirement p_j (denoting the number of machine days required to complete the job), a release date r_j (representing the beginning of the day when job j becomes available for processing), and a due date $d_j \geq r_j + p_j$ (representing the beginning of the day by which the job must be completed). We assume that a machine can work on only one job at a time and that each job can be processed by at most one machine at a time. However, we

allow *preemptions* (i.e., we can interrupt a job and process it on different machines on different days). The scheduling problem is to determine a feasible schedule that completes all jobs before their due dates or to show that no such schedule exists.

Scheduling problems like this arise in batch processing systems involving batches with a large number of units. The feasible scheduling problem, described in the preceding paragraph, is a fundamental problem in this situation and can be used as a subroutine for more general scheduling problems, such as the maximum lateness problem, the (weighted) minimum completion time problem, and the (weighted) maximum utilization problem.

Let us formulate the feasible scheduling problem as a maximum flow problem. We illustrate the formulation using the scheduling problem described in Figure 6.4 with M = 3 machines. First, we rank all the release and due dates, r_j and d_j for all j, in ascending order and determine $P \leq 2 \mid J \mid - 1$ mutually disjoint intervals of dates between consecutive milestones. Let $T_{k,l}$ denote the interval that starts at the beginning of date k and ends at the beginning of date $l + 1$. For our example, this order of release and due dates is 1, 3, 4, 5, 7, 9. We have five intervals, represented by $T_{1,2}$, $T_{3,3}$, $T_{4,4}$, $T_{5,6}$, and $T_{7,8}$. Notice that within each interval $T_{k,l}$, the set of available jobs (i.e., those released but not yet due) does not change: we can process all jobs j with $r_j \leq k$ and $d_j \geq l + 1$ in the interval.

Job (j)	1	2	3	4
Processing time (p_j)	1.5	1.25	2.1	3.6
Release time (r_j)	3	1	3	5
Due date (d_j)	5	4	7	9

Figure 6.4 Scheduling problem.

We formulate the scheduling problem as a maximum flow problem on a bipartite network G as follows. We introduce a source node s, a sink node t, a node corresponding to each job j, and a node corresponding to each interval $T_{k,l}$, as shown in Figure 6.5. We connect the source node to every job node j with an arc with capacity p_j, indicating that we need to assign p_j days of machine time to job j. We connect each interval node $T_{k,l}$ to the sink node t by an arc with capacity $(l - k + 1)M$, representing the total number of machine days available on the days from k to l. Finally, we connect a job node j to every interval node $T_{k,l}$ if $r_j \leq k$ and $d_j \geq l + 1$ by an arc with capacity $(l - k + 1)$ which represents the maximum number of machines days we can allot to job j on the days from k to l. We next solve a maximum flow problem on this network: The scheduling problem has a feasible schedule if and only if the maximum flow value equals $\sum_{j \in J} p_j$ [alternatively, the flow on every arc (s, j) is p_j]. The validity of this formulation is easy to establish by showing a one-to-one correspondence between feasible schedules and flows of value $\sum_{j \in J} p_j$ from the source to the sink.

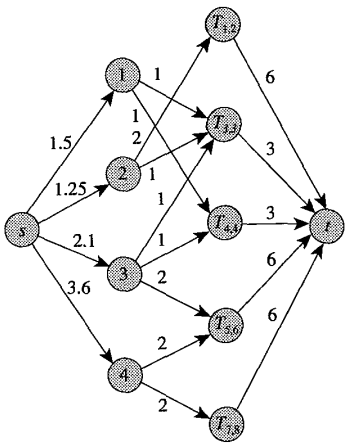


Figure 6.5 Network for scheduling uniform parallel machines.

Application 6.5 Distributed Computing on a Two-Processor Computer

This application concerns assigning different modules (subroutines) of a program to two processors in a way that minimizes the collective costs of interprocessor communication and computation. We consider a computer system with two processors; they need not be identical. We wish to execute a large program on this computer system. Each program contains several modules that interact with each other during the program's execution. The cost of executing each module on the two processes is known in advance and might vary from one processor to the other because of differences in the processors' memory, control, speed, and arithmetic capabilities. Let α_i and β_i denote the cost of computation of module i on processors 1 and 2, respectively. Assigning different modules to different processors incurs relatively high overhead costs due to interprocessor communication. Let c_{ij} denote the interprocessor communication cost if modules i and j are assigned to different processors; we do not incur this cost if we assign modules i and j to the same processor. The cost structure might suggest that we allocate two jobs to different processors—we need to balance this cost against the communication costs that we incur by allocating the jobs to different processors. Therefore, we wish to allocate modules of the program on the two processors so that we minimize the total cost of processing and interprocessor communication.

We formulate this problem as a minimum cut problem on an undirected network as follows. We define a source node s representing processor 1, a sink node t representing processor 2, and a node for every module of the program. For every node i, other than the source and sink nodes, we include an arc (s, i) of capacity β_i and an arc (i, t) of capacity α_i. Finally, if module i interacts with module j during program execution, we include the arc (i, j) with a capacity equal to c_{ij}. Figures 6.6 and 6.7 give an example of this construction. Figure 6.6 gives the data for this problem, and Figure 6.7 gives the corresponding network.

We now observe a one-to-one correspondence between s–t cuts in the network

i	1	2	3	4
α_i	6	5	10	4
β_i	4	10	3	8

(a)

$\{c_{ij}\} =$

	1	2	3	4
1	0	5	0	0
2	5	0	6	2
3	0	6	0	1
4	0	2	1	0

(b)

Figure 6.6 Data for the distributed computing model.

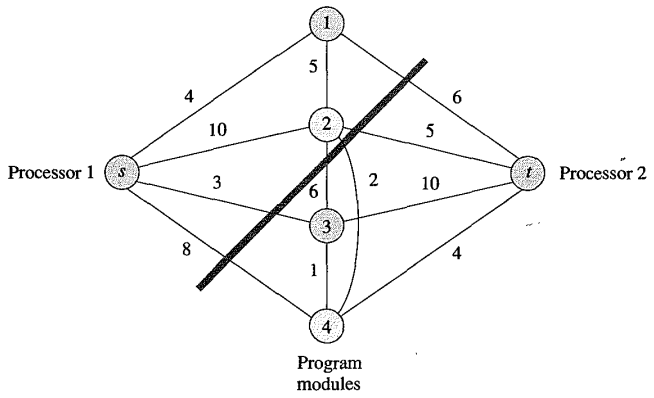


Figure 6.7 Network for the distributed computing model.

and assignments of modules to the two processors; moreover, the capacity of a cut equals the cost of the corresponding assignment. To establish this result, let A_1 and A_2 be an assignment of modules to processors 1 and 2, respectively. The cost of this assignment is $\sum_{i \in A_1} \alpha_i + \sum_{i \in A_2} \beta_i + \sum_{(i,j) \in A_1 x A_2} c_{ij}$. The s–t cut corresponding to this assignment is $(\{s\} \cup A_1, \{t\} \cup A_2)$. The approach we used to construct the network implies that this cut contains an arc (i, t) for every $i \in A_1$ of capacity α_i, an arc (s, i) for every $i \in A_2$ of capacity β_i, and all arcs (i, j) with $i \in A_1$ and $j \in A_2$ with capacity c_{ij}. The cost of the assignment A_1 and A_2 equals the capacity of the cut $(\{s\} \cup A_1, \{t\} \cup A_2)$. (We suggest that readers verify this conclusion using

the example given in Figure 6.7 with $A_1 = \{1, 2\}$ and $A_2 = \{3, 4\}$.) Consequently, the minimum s–t cut in the network gives the minimum cost assignment of the modules to the two processors.

Application 6.6 Tanker Scheduling Problem

A steamship company has contracted to deliver perishable goods between several different origin–destination pairs. Since the cargo is perishable, the customers have specified precise dates (i.e., delivery dates) when the shipments must reach their destinations. (The cargoes may not arrive early or late.) The steamship company wants to determine the minimum number of ships needed to meet the delivery dates of the shiploads.

To illustrate a modeling approach for this problem, we consider an example with four shipments; each shipment is a full shipload with the characteristics shown in Figure 6.8(a). For example, as specified by the first row in this figure, the company must deliver one shipload available at port A and destined for port C on day 3. Figure 6.8(b) and (c) show the transit times for the shipments (including allowances for loading and unloading the ships) and the return times (without a cargo) between the ports.

Shipment	Origin	Destination	Delivery date
1	Port A	Port C	3
2	Port A	Port C	8
3	Port B	Port D	3
4	Port B	Port C	6

(a)

	C	D
A	3	2
B	2	3

(b)

	A	B
C	2	1
D	1	2

(c)

Figure 6.8 Data for the tanker scheduling problem: (a) shipment characteristics; (b) shipment transit times; (c) return times.

We solve this problem by constructing a network shown in Figure 6.9(a). This network contains a node for each shipment and an arc from node i to node j if it is possible to deliver shipment j after completing shipment i; that is, the start time of shipment j is no earlier than the delivery time of shipment i plus the travel time from the destination of shipment i to the origin of shipment j. A directed path in this network corresponds to a feasible sequence of shipment pickups and deliveries. The tanker scheduling problem requires that we identify the minimum number of directed paths that will contain each node in the network on exactly one path.

We can transform this problem to the framework of the maximum flow problem as follows. We split each node i into two nodes i' and i'' and add the arc (i', i''). We set the lower bound on each arc (i', i''), called the *shipment arc*, equal to 1 so that at least one unit of flow passes through this arc. We also add a source node s and connect it to the origin of each shipment (to represent putting a ship into service),

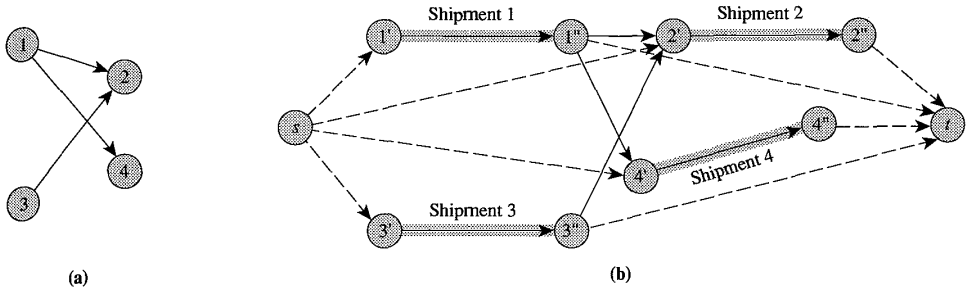


Figure 6.9 Network formulation of the tanker scheduling problem: (a) network of feasible sequences of two consecutive shipments; (b) maximum flow model.

and we add a sink node t and connect each destination node to it (to represent taking a ship out of service). We set the capacity of each arc in the network to value 1. Figure 6.9(b) shows the resulting network for our example. In this network, each directed path from the source s to the sink t corresponds to a feasible schedule for a single ship. As a result, a feasible flow of value v in this network decomposes into schedules of v ships and our problem reduces to identifying a feasible flow of minimum value. We note that the zero flow is not feasible because shipment arcs have unit lower bounds. We can solve this problem, which is known as the *minimum value problem*, using any maximum flow algorithm (see Exercise 6.18).

6.3 FLOWS AND CUTS

In this section we discuss some elementary properties of flows and cuts. We use these properties to prove the max-flow min-cut theorem to establish the correctness of the generic augmenting path algorithm. We first review some of our previous notation and introduce a few new ideas.

Residual network. The concept of *residual network* plays a central role in the development of all the maximum flow algorithms we consider. Earlier in Section 2.4 we defined residual networks and discussed several of its properties. Given a flow x, the residual capacity r_{ij} of any arc $(i, j) \in A$ is the maximum additional flow that can be sent from node i to node j using the arcs (i, j) and (j, i). [Recall our assumption from Section 6.1 that whenever the network contains arc (i, j), it also contains arc (j, i).] The residual capacity r_{ij} has two components: (1) $u_{ij} - x_{ij}$, the unused capacity of arc (i, j), and (2) the current flow x_{ji} on arc (j, i), which we can cancel to increase the flow from node i to node j. Consequently, $r_{ij} = u_{ij} - x_{ij} + x_{ji}$. We refer to the network $G(x)$ consisting of the arcs with positive residual capacities as the *residual network* (with respect to the flow x). Figure 6.10 gives an example of a residual network.

s–t cut. We now review notation about cuts. Recall from Section 2.2 that a cut is a partition of the node set N into two subsets S and $\overline{S} = N - S$; we represent this cut using the notation $[S, \overline{S}]$. Alternatively, we can define a cut as the set of

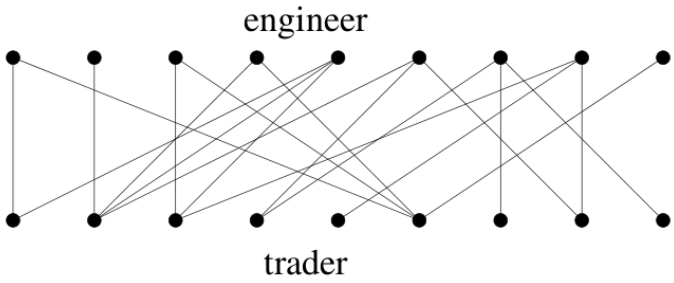


Figure 1:

Exercise 7 [Goe11]

A managing director has to launch the marketing of a new product. Several candidate products are at his disposal and he has to choose the best one. Hence, he let each of these products be analysed by a team made of an engineer and a trader who write a review together. The teams are made along the graph in Figure 1; each edge corresponds to a product and its endvertices to the engineer and trader examining it.

a) How many people at least does the managing director gather in order to have the report on all the products? (The report can be given by either the engineer or the trader.)

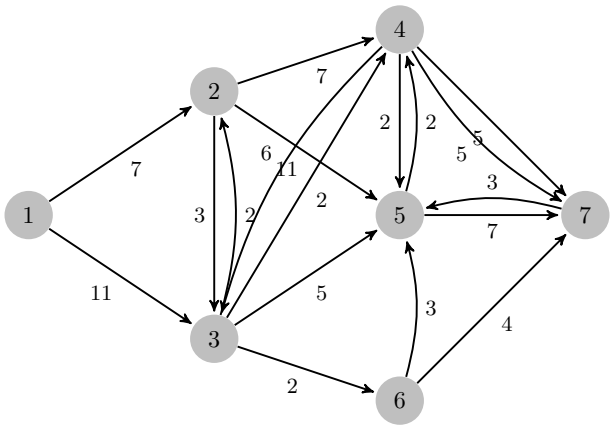
b) Assuming now that the report must be done jointly by an engineering and a trader, and that each engineer and trader can be occupied with only one candidate product, give a polynomial time algorithm to identify which products will for sure not have the possibility to obtain a report.

Solution:

a) This is an application of the vertex cover problem and its strong duality with maximum matching.
b) This can be done by finding all maximum matching of the graph. The edges that are never in a matching are those that will be never reviewed. We could solve $|E|$ linear programs formulations of the max matching problem for bipartite graphs in each of which a different edge is enforced to be in the solution. Other, more efficient methods based on direct algorithms exist.

Exercise 8

Given the Network in Figure 2, determine the max flow from 1 to 7 and indicate the min cut.



footnotesize

```
\documentclass{standalone}
\usepackage{tikz}

%%% TIKZ STUFF %%%
\usetikzlibrary{arrows}
\tikzstyle{vertex}=[circle,fill=black!25,minimum size=20pt,inner sep=0pt]
\tikzstyle{selected vertex} = [vertex, fill=red!24]
\tikzstyle{edge} = [draw,thick,-]
\tikzstyle{arc} = [draw,thick,->,shorten >=1pt,>=stealth']
\tikzstyle{arcl} = [draw,thick,->,shorten >=1pt,>=stealth',bend left=25]
\tikzstyle{arcr} = [draw,thick,->,shorten >=1pt,>=stealth']
\tikzstyle{weight} = [font=\small]
\tikzstyle{selected edge} = [draw,line width=5pt,-,red!50]
\tikzstyle{ignored edge} = [draw,line width=5pt,-,black!20]
%%% TIKZ STUFF %%%


\begin{document}

\begin{tikzpicture}[scale=0.9, auto,swap]
    % First we draw the vertices
  \foreach \pos/\name in {{(0,3)/1}, {(3,1)/3}, {(3,5)/2},
                          {(6,0)/6}, {(6,3)/5}, {(6,6)/4}, {(9,3)/7}}
       \node[vertex] (\name) at \pos {$\name$};
    % Connect vertices with edges and draw weights
    \foreach \source/ \dest /\weight in {
      1/2/7, 1/3/11, 2/3/3, 3/6/2,
      2/4/7, 2/5/11, 3/4/2, 3/5/5,
      4/7/5, 4/5/2, 5/7/7, 6/7/4}
    \path[arcr] (\source) -- node[weight] {$\weight$} (\dest);
    \foreach \source/ \dest /\weight in {
      3/2/2, 4/3/6, 4/7/5, 5/4/2, 7/5/3, 6/5/3}
        \path[arcl,bend right] (\source) edge [bend right=15] node[weight] {$\weight$} (\dest);
\end{tikzpicture}
\end{document}
```

Figure 2: Find the maximum flow from 1 to 7. Numbers on arcs are capacity values. [In preparation for the exam, below the graph you find the excerpt of latex code to produce the picture. You can use it to experiment whether its use is fast enough for an exam session.]

Solution:

See https://github.com/DM871/dm871.github.io/blob/main/notebooks/net_flow.ipynb.

Exercise 9

Consider the following IP problem:

$$\begin{aligned} \max \quad & 4x_1 + 7x_2 \\ s.t. \quad & x_1 + 3x_2 \leq 12 \\ & 4x_1 + 6x_2 \leq 27 \\ & 4x_1 + 2x_2 \leq 20 \\ & x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z} \end{aligned} \tag{1}$$

Subtask a

Give a heuristic primal bound and describe how you determined it.

Solution:

$x = [0, 0]$ is feasible because it satisfies all constraints and has value $z = 0$. This is a lower bound to the optimal solution.

Subtask b

Write the LP relaxation (2lp) of (2) to obtain a dual bound. Explain the relation between the optimal solution of (2lp) and the optimal solution of (2).

Solution:

We relax x_1 and x_2. The problem (2lp) becomes:

$$\begin{aligned} \max \quad & z_{LP} = 4x_1 + 7x_2 \\ s.t. \quad & x_1 + 3x_2 \leq 12 \\ & 4x_1 + 6x_2 \leq 27 \\ & 4x_1 + 2x_2 \leq 20 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2}$$

(2lp) gives an upper bound to the problem (2).

Subtask c

Write the first simplex tableau of (2lp) and indicate which variables constitute a basic solution. Call s_1, s_2, s_3 the slack variables.

Subtask d

Explain which variable leaves the basis and which variable enters the basis in the first iteration of the simplex algorithm with largest coefficient pivot rule. Show that the answer would be the same if, instead, the largest increase pivot rule was used.

Subtask e

After a number of iterations the tableau is the following:

x1	x2	s1	s2	s3	-z	b
0	1	2/3	-1/6	0	0	7/2
1	0	-1	1/2	0	0	3/2
0	0	8/3	-5/3	1	0	7
0	0	-2/3	-5/6	0	1	-61/2

Argue that an optimal solution for (2lp) has been found and give for it the value of x_1 and x_2 together with its objective function value. Report the optimality gap for (2) at this stage.

Subtask e

Show how you can reconstruct the tableau at the previous point by just knowing that x_2, x_1 and s_3 are in basis and that:

$$A_B^{-1} = \begin{bmatrix} 2/3 & -1/6 & 0 \\ -1 & 1/2 & 0 \\ 8/3 & -5/3 & 1 \end{bmatrix}.$$

Solution:

```
import numpy as np
from fractions import Fraction as f
A=np.array([[ 1, 3,0],[4, 6,0],[4,2,1]])
# print np.linalg.inv(A)
A_1= np.array([[f(2,3),f(-1,6),0],[f(-1),f(1,2),0],[f(8,3),f(-5,3),1]])
print np.dot(A[:,[1,0,2]],A_1)
```

Subtask f

From the second row of the last tableau derive a Gomory cut and write it in the space of the original variables.
Argue shortly that the cut is a valid inequality for (2) and that it will make the current optimal solution of (2lp) infeasible.

Subtask g

Introduce the cut in the tableau and explain how the solution algorithm will continue. Indicate the new pivot and explain how you found it. (You do not need to carry out the simplex iteration.)

Subtask h

After the introduction of the cut the tableau of the optimal solution to the new LP problem is the following.

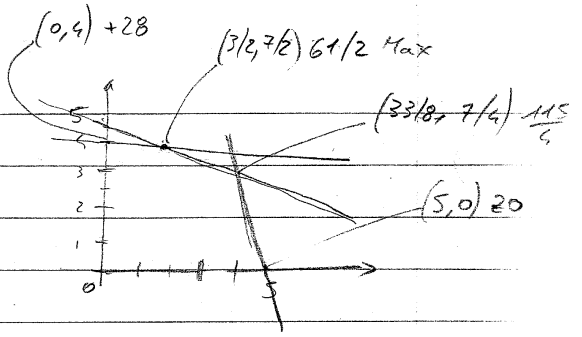
x1	x2	s1	s2	s3	s4	-z	b
0	1	2/3	0	0	-1/3	0	11/3
0	0	0	1	0	-2	0	1
0	0	8/3	0	1	-10/3	0	26/3
1	0	-1	0	0	1	0	1
0	0	-2/3	0	0	-5/3	1	-89/3

Explain how the solution process would continue from this stage by branch and bound. Define the next branching and indicate what can be done in each open node.

Solution:

TASK 1

$$\max z = 4x_1 + 7x_2$$
$$x_1 + 3x_2 \le 12$$
$$4x_1 + 6x_2 \le 27$$
$$4x_1 + 2x_2 \le 20$$



(a) (0,0) is feasible $\Rightarrow z = 0 = LB$

(c) (d)

x_1	x_2	s_1	s_2	s_3	$-z$	b	b_i/a_{i1}	b_i/a_{i2}	$c_j \cdot b_i/a_{i2}$
1	3	1	0	0	0	12	12	4	$7 \cdot 4 = 28$
4	6	0	1	0	0	27	$27/4 = 8$	$\frac{27}{6} \simeq 4$	
4	2	0	0	1	0	20	5	10	$5 \cdot 4 = 20$
4	7	0	0	0	1	0			

Largest coefficient $\rightsquigarrow 7$

Largest increase $\rightsquigarrow 7$

1/3	1	1/3	0	0	0	4
2	0	−2	1	0	0	3
10/3	0	−2/3	0	1	0	12
−5/3	0	7/3	0	0	1	28

(e)

0	1	2/3	−1/6	0	0	7/2
1	0	−1	1/2	0	0	3/2
0	0	8/3	−5/2	1	0	7
0	0	−2/3	−5/6	0	1	−61/2

$$GAP = \frac{61/2 - 0}{61/2} = 1$$

(g)

I row: $\frac{2}{3}s_1 + \frac{5}{6}s_2 \ge \frac{1}{2}$ $\qquad \frac{1}{2}s_1 \ge \frac{1}{2}$

$$\frac{2}{3}(12 - x_1 - 3x_2) + \frac{5}{6}(27 - 4x_1 - 6x_2) \ge \frac{1}{2}$$
$$8 - \frac{2}{3}x_1 - 2x_2 + \frac{135}{6} - \frac{10}{3}x_1 - 5x_2 \ge \frac{1}{2}$$
$$-4x_1 - 7x_2 \ge \frac{1}{2} - 8 - \frac{135}{6}$$

II row: $\frac{1}{2}s_2 \ge \frac{1}{2}$ $\qquad$ (i)-form

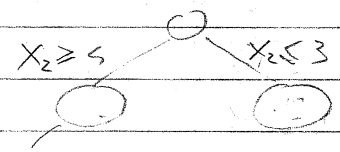
$$\frac{1}{2}(27 - 4x_1 - 6x_2) \ge 1$$
$$-4x_1 - 6x_2 \ge -26$$
$$-2x_1 - 3x_2 \ge -13$$
$$2x_1 + 3x_2 \le 12$$ $\qquad$ (ii)-form

(g) Introducing the i-form we have a basis but infeasible.
Introducing the ii-form we do elementary row operations to arrive to a canonical form with infeasible basis.
We use dual simplex:

0	0	0	$\boxed{-1/2}$	0	0	1	$-1/2$
0	1	$2/3$	$-1/6$	0	0	0	$7/2$
1	0	-1	$1/2$	0	0	0	$3/2$
0	0	$8/3$	$-5/3$	1	0	0	7
0	0	$-2/3$	$-5/6$	0	1	0	$-61/2$

1) pivot < 0
2) row with b term negative
3) col that $\min \left|\frac{c_j}{a_{ij}}\right|$ $\Rightarrow$ $-\frac{1}{2}$ is pivot

(h) Using branch and bound on we have



are could use heuristics again at each node

the LP requires a dual-simplex step
sol will be (0,4) of val 28

sol will be (2,3) of val 29

References

[DT97] George B. Dantzig and Mukund N. Thapa. *Linear Programming*. Springer, 1997.

[Goe11] Michel X. Goemans. Lecture notes on bipartite matching. http://www-math.mit.edu/~goemans/18433S11/matching-notes.pdf, 2011.