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

November 8, 2026
Marco Chiarandini

DM545/DM871 – Linear and integer programming

Exercise Sheet 8, Autumn 2026

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$^+$

Show that the maximum matching in arbitrary graphs is a special sort of set packing problem.

Solution:

The set packing problem asks to find a maximum size subset of the columns of a matrix in such a way that no row is covered more than once. The maximum matching asks to find the maximum size subset of the set of edges such that no vertex is covered more than once. Hence, in the matching problem the incidence matrix has the same role as in the matrix of the set packing. It is a special sort of set packing because the incidence matrix has exactly two 1s in each column.

More formally:
The set packing problem is defined as follows. Let $U = \{1, ..., m\}$ be the universal set and let $S_1, ..., S_n$ be a collection of subsets of U. A packing is a subcollection of the sets S_i such that no element of U is contained in more than one of the sets in the subcollection. The set packing problem is to find a packing of maximum cardinality.
It can be formulated as a MILP problem as follows: Let A be a $m \times n$ matrix with entries a_{ij} equal 1 if element i from U is contained in S_j and zero otherwise. Let also $\mathbf{1}$ be a vector of m elements all equal to 1. The set packing problem is to find a vector $\mathbf{x} \in \{0,1\}^n$ such that $A\mathbf{x} \leq \mathbf{1}$ and $\sum_{j=1}^{n} x_j$ is maximized.
Let $G = (V, E)$ be a graph. The maximum matching problem is to find a subset $M \subseteq E$ such that no two edges in M share a vertex and $|M|$ is maximized.
The problem can be formulated as a set packing problem as follows. Let A be the edge-vertex incidence matrix of the graph G and let b be the vector of all ones. Then the maximum matching problem is to find a vector $x \in \{0,1\}^{|E|}$ such that $A\mathbf{x} \leq \mathbf{1}$ and $\sum_{j=1}^{|E|} x_j$ is maximized. The incidence matrix has exactly two 1s in each column, one for each vertex of the edge and it is therefore a special case of the matrix A that appears in the set packing problem.

Exercise 2 Bin Packing

The binpacking problem is to assign a given set of $n \in \mathbb{N}$ items, each of a size $s_i \in \mathbb{N}$, to bins with capacity $\kappa \in \mathbb{N}$, such that the used bins are not overloaded. The goal is to minimize the number of used bins. Model the problem as a MILP problem.
Example: Given 9 items with sizes: 2, 1, 2, 1, 1, 2, 3, 2 and 1, and a bin capacity of $\kappa = 4$. Figure 1 shows a feasible solution which needs 5 bins.

Solution:

Let i be an index for the set of items $I = \{1, \ldots, n\}$. Let x_{ij}, $i \in I, j \in J = \{1, \ldots, B\}$ be binary variables to decide whether i goes in bin j. Let y_j, $j \in J$ be binary variables that indicates whether a bin is used.

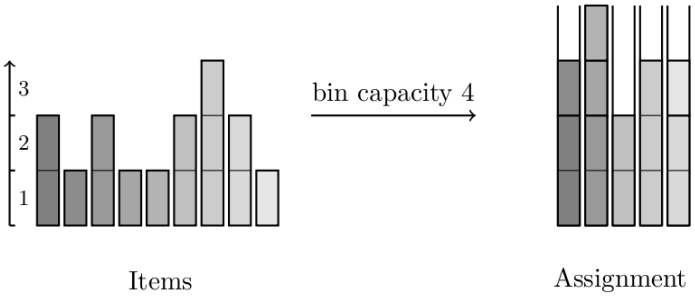


Figure 1: A feasible solution to the bin packing problem.

The bin packing problem can be formulated as follows:

$$\min \quad \sum_{j \in J} y_j \tag{1}$$

$$\sum_{i \in I} s_i x_{ij} \leq \kappa y_j \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad i \in I \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad i \in I, j \in J \tag{4}$$

$$y_j \in \{0,1\} \qquad j \in J \tag{5}$$

The first set of constraints ensures that the capacity of the bins is not exceeded and that the variable y_j is set to one if the bin contains items. The second set of constraints ensures that every item is packed somewhere. The objective function minimizes the number of used bins.

Exercise 3*

Give two MILP formulations for the chromatic number problem: given a graph find a coloring (i.e., a mapping from the natural numbers to the set of vertices) such that every pair of vertices connected by an edge receives different colors and the number of colors used is the least possible.

Solution:

Let $G = (V, E)$ be an undirected graph and $\Gamma = \{1, 2, \ldots, k\}$ a set of k labels represented by the natural numbers. A coloring that uses k colors is a k-labelling of the vertices, that is a mapping $\varphi : V \to \Gamma$. A k-labelling is *proper* if for every pair of vertices connected by an edge the natural numbers that the vertices receive is different, that is, $\varphi(v) \neq \varphi(u)$ for all $uv \in E$. We want to determine the least number of labels k for which the graph G admits a proper k-labelling.
The k-labelling is known as k-coloring and the least number k for which the graph G admits a k-coloring is called the chromatic number of the graph and is denoted by $\chi(G)$.
We give two formulations a compact and an extended one.

Formulation 1 Let $x_{v\ell} \in \{0,1\}$ be the binary variable that is 1 if the vertex v receives color ℓ and zero otherwise and let $y_\ell \in \{0,1\}$ be 1 if the color ℓ is used in the coloring φ and zero otherwise. Let $\Gamma = \{1, 2, \ldots, ub\}$, where ub is a trivial upper bound to the chromatic number, for example, $ub = |V|$.

$$\min \quad \sum_{\ell \in \Gamma} y_\ell \tag{6}$$

$$\text{s.t.} \quad x_{uk} + x_{vk} \leq y_\ell \quad \forall\, uv \in E \tag{7}$$

$$x_{u\ell} \in \{0,1\} \quad \forall\, v \in V, \ell \in \Gamma \tag{8}$$

$$y_\ell \in \{0,1\} \quad \forall\, \ell \in \Gamma \tag{9}$$

The objective function (6) gives the number of colors used from Γ. Constraints (7) ensure that if the color ℓ is used then y_ℓ is 1 and it is used in at most one of the two vertices that are end points of the edge.

It is possible to strengthen the model adding the redundant clique constraints. Let $\mathcal{C}$ be a set of cliques in G (it can be a clique cover or a maximal clique or all cliques of G):

$$\sum_{v \in C} x_{v\ell} \leq y_\ell \quad \forall C \in \mathcal{C}$$

Formulation 2 Let $\mathcal{I}$ be the set of all independent sets of the graph G. Let y_I be a binary variable that is 1 if the independent set $I \in \mathcal{I}$ is selected.

$$\min \quad \sum_{I \in \mathcal{I}} y_I \tag{10}$$

$$\text{s.t.} \quad \sum_{I \in \mathcal{I}} y_I \geq 1 \quad \forall\, v \in V \tag{11}$$

$$y_I \quad \forall\, I \in \mathcal{I} \tag{12}$$

There are exponentially many independent sets in general, hence the model has an exponential number of variables, nevertheless this model provides a tighter linear programming relaxation than the previous. In practice, the model can be solved by delayed column generation and branch and price.

Exercise 4*

Formulate the Minimum Spanning Tree problem as a mathematical programming problem.

Solution:

A tree is a connected graph with no cycles, and a spanning tree is a subgraph in G that is a tree and includes all vertices of G i.e. V. The minimum spanning tree (MST) problem calls for finding a spanning tree whose total cost is minimum. The total cost is measured as the sum of the costs of all the edges in the spanning tree.
Let $G = (V, E)$ be an undirected graph with n vertices and m edges. Let w_e be the weight of edge $e \in E$. Let x_e be a binary variable that is 1 if edge e is in the spanning tree and zero otherwise. We can try to formulate the MST problem as follows:

$$\min \sum w_e x_e \tag{13}$$

$$\sum_{e \in E} x_e = n - 1 \tag{14}$$

$$\sum_{v : v \in e, e \in E} x_e \geq 1 \qquad \forall v \in V \tag{15}$$

$$x_e \in \{0, 1\} \tag{16}$$

Constraint (14) enforces that the solution contains exactly $n - 1$ edges, and constraints (15) ensure that every vertex is touched by the tree, ie, that the tree is spanning V. However, these constraints alone do not ensure that the solution is connected. We can still obtain a disconnected graph with $n-1$ edges where all vertices are incident to at least one edge but there is a cycle plus some isolated edges. We need to add additional constraints to ensure that the solution is connected and acyclic. There are several ways to do this, for example, we can use the following theorem:

Theorem 1 (Spanning Trees) *For an undirected graph $G = (V, E)$ and subgraph $G' \subseteq G$, the following are equivalent:*
(a) $G' = (V, E')$ is a tree (acyclic and connected);
(b) $G' = (V, E')$ is acyclic and has $n - 1$ arcs; and
(c) $G' = (V, E')$ is connected and has $n - 1$ arcs.

Using (b), we can add the subtour elimination constraints to the model as follows:

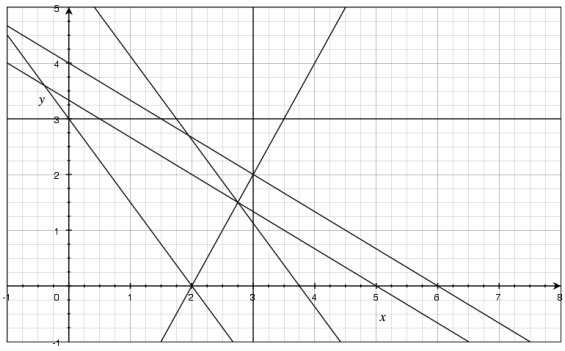


Figure 2: The polyhedron of exercise 5.

$$\min \sum w_e x_e \tag{17}$$
$$\sum_{e \in E} x_e = n - 1 \tag{18}$$
$$\sum_{e \in \gamma_S} x_e \leq |S| - 1 \qquad \forall \emptyset \subset S \subset V \tag{19}$$
$$x_e \in \{0, 1\} \tag{20}$$

where $\gamma_S = \{ij \in E \mid i \in S, j \in S\}$.
To pursue (c), we need the following definition: A *cut* $C = \{X, X'\}$ is a partition of the set of vertices into two subsets. For $X \subset V$, we denote the set of arcs crossing the cut $\delta_{X,X'} = \{(u, v) \in E : u \in X, v \in V \setminus X\}$.
Hence, (c) states that for a spanning tree of n vertices we need $n - 1$ edges and for each possible cut C at least one of the edges crossing the cut must be in the solution. The cut set formulation of MST is:

$$\min \sum w_e x_e \tag{21}$$
$$\sum_{e \in E} x_e = n - 1 \tag{22}$$
$$\sum_{e \in \delta_X} x_e \geq 1 \qquad \forall \emptyset \subset X \subset V \tag{23}$$
$$x_e \in \{0, 1\} \tag{24}$$

More on MST in sec. 3.5 of [Wo] (where it is called maximum weight Tree) and on page 23 of [CL].

Exercise 5* Consider the polyhedron $P \subseteq R^2$ described by the following inequalities and depicted in Figure 2:

$$\begin{array}{rcrcr} 2x & - & y & \leq & 4 \\ 2x & + & 3y & \leq & 12 \\ & & y & \leq & 3 \\ 3x & + & 2y & \geq & 6 \\ x & & & \leq & 3 \\ 3/2x & + & y & \leq & 45/8 \\ 2x & + & 3y & \leq & 10 \end{array}$$

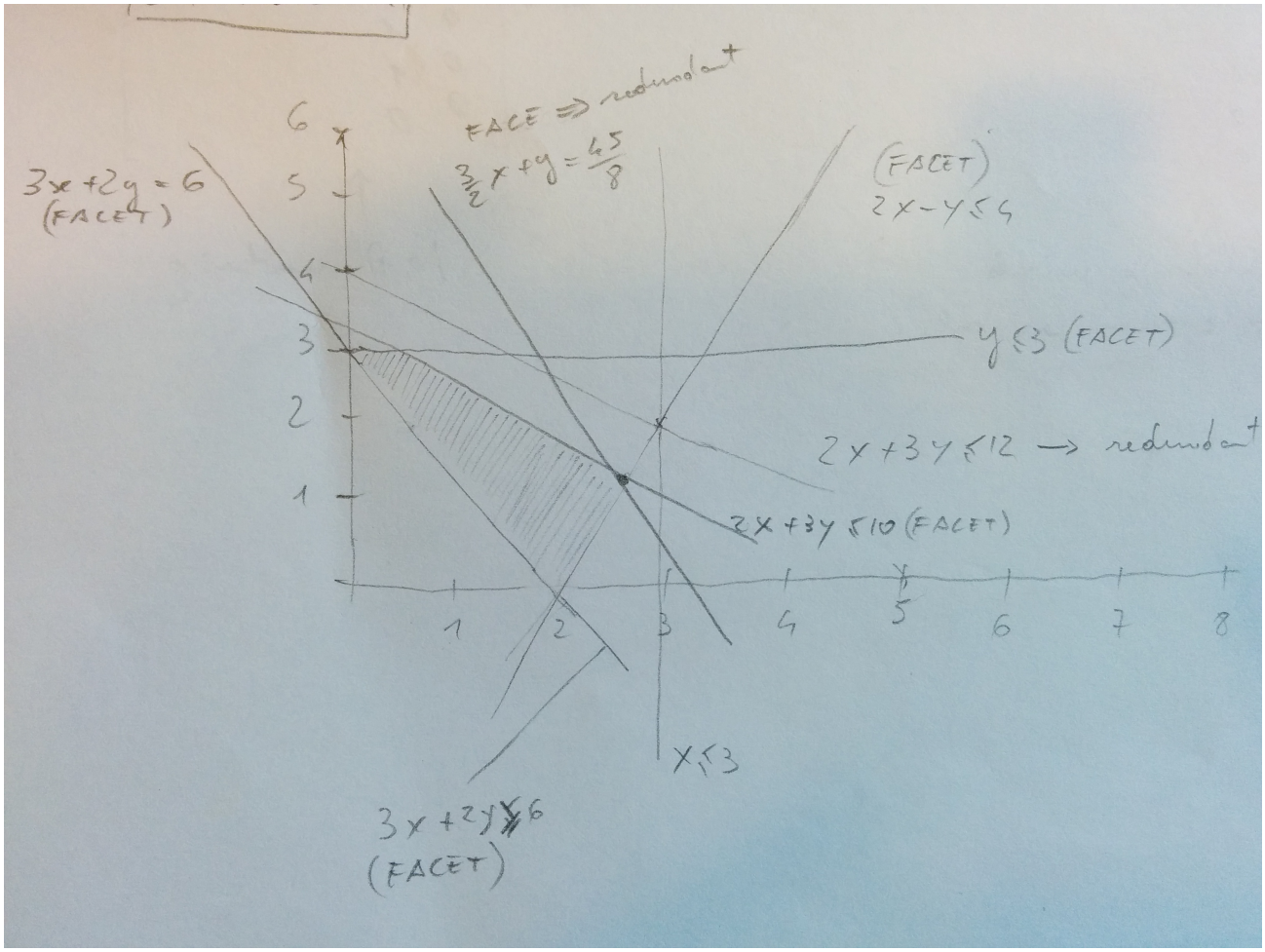
Which inequality is a face? Which is a facet? Which is redundant?

Solution:

We note first that the inequalities $\mathbf{ax} \leq b$ and $\lambda\mathbf{ax} \leq \lambda b$ are identical for any $\lambda > 0$.
A constraint $\mathbf{ax} \leq b$ is *valid* for $P \subseteq \mathbb{R}^n_+$ if $P \subseteq \{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{ax} \leq b\}$.

A valid constraint for $P \subseteq \mathbb{R}^n_+$, $\mathbf{a}_1\mathbf{x} \leq b_1$, *dominates* another valid constraint for P, $\mathbf{a}_2\mathbf{x} \leq b_2$, if $\{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{a}_1\mathbf{x} \leq b_1\} \subseteq \{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{a}_2\mathbf{x} \leq b_2\}$.
A constraint $\mathbf{a}_0\mathbf{x} \leq b_0$ is *redundant* in the description of $P \subseteq \mathbb{R}^n_+$ if there are $k \geq 1$ valid constraints for P such that $\{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{a}_i\mathbf{x} \leq b_i \text{ for } i = 1, \ldots, k\} \subseteq \{\mathbf{x} \in \mathbb{R}^n_+ \mid \mathbf{a}_0\mathbf{x} \leq b_0\}$.
A facet is a face of maximal dimension: in R^2 it is an edge of dimension 1.

$2x$	$-$	y	$\leq$	4	Facet
$2x$	$+$	$3y$	$\leq$	12	Redundant
		y	$\leq$	3	Facet
$3x$	$+$	$2y$	$\geq$	6	Facet
x			$\leq$	3	Redundant
$3/2x$	$+$	y	$\leq$	45/8	Face, Redundant
$2x$	$+$	$3y$	$\leq$	10	Facet



Exercise 6*

Model:

- the vertex covering problem: given a graph find the set of vertices such that each edge of the graph is incident to at least one vertex of the set.
- the dominating set problem: given a graph find a set of the vertices such that every vertex not in the set is adjacent to at least one member of the set.

Are the solutions to the two problems always the same, i.e., are the two problems equivalent?
The two problems have several applications, for example: in wireless networking, dominating sets are used to find efficient routes within ad-hoc mobile networks; they also encode the art gallery problem or museum problem, that is, guarding an art gallery with the minimum number of guards who together can observe the whole gallery.

Solution:

No.
If by any graph you also consider non-connected graphs, then you have a trivial counterexample in $G = (V, E)$ with $V \neq \emptyset$ and $E = \emptyset$, i.e. at least one vertex and no edges. The minimal vertex cover is $C = \emptyset$, since there are no edges to cover, but the only dominating set is $D = V$, and clearly $D \nsubseteq C$.
Even if you restrict it to connected graphs, the answer is still no. Consider the cycle graph $G = (V, E)$ of six vertexes $V = a, b, c, d, e, f$ and $E = (a, b), (b, c), (c, d), (d, e), (e, f), (a, f)$
The vertex cover sets of minimal size have three elements, obtained by skipping one vertex in two: a, c, e and b, d, f. The dominating sets of minimal size are obtained with opposing vertexes, a, d, b, e, and c, f. None of the dominating sets is a subset of any vertex cover set.

Exercise 7*

Given a collection of sets, a set which intersects all sets in the collection in at least one element is called a *hitting set*, and a simple NP-hard problem is to find a hitting set of smallest size or minimum hitting set. Formulate this problem in MILP terms.

Solution:

Set covering is equivalent to the hitting set problem. That is seen by observing that an instance of set covering can be viewed as an arbitrary bipartite graph, with sets represented by vertices on the left, the universe represented by vertices on the right, and edges representing the inclusion of elements in sets. The task is then to find a minimum cardinality subset of left-vertices which covers all of the right-vertices. In the Hitting set problem, the objective is to cover the left-vertices using a minimum subset of the right vertices. Converting from one problem to the other is therefore achieved by interchanging the two sets of vertices.
The model for the problem is:

$$\begin{aligned} \min \sum_{v} x_v & \\ \sum_{v \in V : u \in W_v} x_v \geq 1 & \qquad \forall u \in V \\ x_v \in \{0, 1\} & \qquad \forall v \in V \end{aligned}$$

The LP relaxation will give us a lower bound when the solution is optimal. Non-optimal solutions to the LP relaxation are not necessarily lower bounds.

Exercise 8

Four friends are going on holiday together and need to plan out a schedule of activities for the next two weeks. Each day has three time slots, in which an activity can be placed, representing morning, midday and evening. The goal is to schedule as many different activities as possible. Each activity can either be relaxing, active or cultural (denoted by r, a and u). Associated with each activity is a cost. The total cost for the two weeks must not exceed 14.000kr. All slots should be assigned to exactly one activity. Because they do not want to go overtired, each day can at most have one active activity and

it cannot be in the morning. They would also like to reserve the first day for only cultural and relaxing activities. The three last days may not exceed 700kr for each day, because they want to save money for souvenirs. They do not want to do the same cultural or active activities twice, but would accept doing the same relaxing activities three times. Further, each person has a list of activities they would like to add, so atleast one activity from each friend's list must be scheduled.

Solution:

[This problem was designed and solve by a previous student of this course.]
The model: Let n be the total number of activities and let t be the total number of timeslots during the two weeks. Then let N be the set of integers in the range 1 to n and T be the set of integers in the range 1 to t. Let D be the set of integers from one to 14, representing the days. Let T_d be the subset of T, made of the indexes of the timeslots in day d, eg, $T_1 = \{1, 2, 3\}$. Let p be the number of friends and let F be the $n \times p$ parameter matrix representing the preferences of the four friends for the activities. The objective will then be to maximize the number of different activities scheduled or the variety.
Variables Let the binary variables $x = \{x_{11}, x_{12}, ..x_{nt}\}$ represent the scheduling of an activity in some timeslot (ie. x_{ij} is 1 if activity i is scheduled in the timeslot j). Let the binary variables $\alpha = \{\alpha_1, \alpha_2, ..\alpha_n\}$ be the auxiliary variables representing whether or not an activity has been chosen for the schedule.

$$\text{maximize} \sum_{i \in N} \alpha_i$$

In order to link α_j to x_i, we enforce α_i to be equal to one if we assign activity i to some timeslot. Because we are maximizing over α, α_i will be set to one if $\sum_{jinT} x_{ij} \geq 0$. Dealing with auxiliary variables is an important part of MILP, since they give a different view to the problem and help to specify some constraints.

$$\sum_{j \in T} x_{ij} \geq \alpha_i \quad \forall i \in N$$

To restrict the cost of the schedule, we sum the cost of all choosen activities and force this to be less than 14000kr.

$$\sum_{i \in N} \sum_{j \in T} c_i x_{ij} \leq 14000$$

Next, we need to ensure that each timeslot is filled with exactly one activity. As with the linking constraints of the auxiliary variables, constraints such as these may be easy to forget when trying to write a mathematical model. But failing to do so would in this case lead to double-bookings.

$$\sum_{i \in N} x_{ij} = 1 \quad \forall j \in T$$

To express the constraint of no active activities in the morning, we constrain any active activity in all morning timeslots. To ensure that only active activities are considered we use the parameter a that will be one only when the activity is in fact an active activity.

$$\sum_{i \in N} a_i x_{ij} = 0 \quad \forall j \in \min\{T_d\}, d \in D$$

In order to restrict the number of active activities scheduled during a day, we sum through all activities and the timeslots in the day, creating a constraint for each day.

$$\sum_{i \in N} \sum_{j \in T_d} a_i x_{ij} \leq 1 \quad \forall d \in D$$

The next constraint removes the possibility of assigning active activities during the first day, which is equivalent to only allowing cultural or relaxing activities.

$$\sum_{j \in T_1} a_i x_{ij} = 0 \quad \forall i \in N$$

The cost constraint on the last three days is represented by three constraints, one for each day, where we restrict the total cost of the activities scheduled during this time to be less or equal than 700kr.

$$\sum_{i \in N} \sum_{j \in T_d} c_i x_{ij} \leq 700 \quad \forall d \in D, d \geq \max\{D\} - 3$$

In order to restrict the number of times a particular activity is scheduled, we partition the set of activities in three sets one per each type, ie, $N = N_u \cup N_r \cup N_d$. Then for each type we create a constraint summing the activities scheduled over all timeslots.

$$\sum_{i \in T} x_{ij} \leq 1 \quad \forall j \in N_u$$

$$\sum_{i \in T} x_{ij} \leq 3 \quad \forall j \in N_r$$

$$\sum_{i \in T} x_{ij} \leq 1 \quad \forall j \in N_a$$

Finally, we ensure that all friends get at least one favourite activity scheduled, by creating a constraint for each person. Here we use the auxiliary variable again, because we only care about whether or not some activity on some person's favourites list has been scheduled at some time.

$$\sum_{j \in N} F_{ij} \alpha_j \geq 1 \quad \forall i \in \{1, ..p\}$$

The model in ZIMPL language is reported in Figure 3
The format of the data file is such that the first column is the name of the activity, the second is the cost, the third marks if it is an active activity, fourth for cultural and fifth for relaxing. Finally the four last columns represent the four friends' preferences. A dat file:

Exercise 9

The Classical airline crew scheduling problem is described as follows. A flight segment is a nonstop flight between two airports with particular departure/arrival times and airplane requirements. Crew schedules are determined for each fleet (airplanes of a particular type) separately. First a subset of *rotations (or rosters)* are identified. Rotations are collections of flight segments that satisfy aviation regulations, company regulations and labor union rules and that are meant to be served by a single crew. They have typically duration of 2-3 days. Then, crews are assigned to the selected rotations, often by a bidding system.
The cost of selecting a particular rotation includes flight time of the crew in that rotation, lodging and per day expenses, cost of transporting crews to their next departure city in their next flight segment (they are not serving on these flight segments), etc. Let c_j denote the cost of selecting rotation $j \in \{1, \ldots, n\}$. Suppose there are m flight segments, then a rotation j is given by an incidence vector of flight segments in that rotation, i.e., for rotation j, $\mathbf{a}_j$ is an m-dimensional vector with $a_{ij} = 1$ if flight segment i is on rotation j and 0 otherwise. Formulate the problem of selecting the rotations that minimize the total cost such that all flight segments are served, and each flight segment is served by exactly one rotation.

Solution:

This is an example of application of the set partitioning model.

```
param n := read path as "1n" use 1 comment "#";
set T := {1 .. 42};
set D := {<i> in T with ((i-1)mod 3) ==0};
set N := {read path as "<1s>" skip 1 use n comment "#"};
set P := {1 .. 4};
param c[N] := read path as "2n" skip 1 use n comment "#";
param a[N] := read path as "3n" skip 1 use n comment "#";
param u[N] := read path as "4n" skip 1 use n comment "#";
param r[N] := read path as "5n" skip 1 use n comment "#";
param f[N*P] := read path as "n+" skip (2+n) comment "#";

var s[N] binary;
var a_s[N*T] binary;

maximize variety: sum <j> in N : s[j];

subto choose: forall <j> in N do sum <i> in T : a_s[j,i] >= s[j];

subto totalcost: sum <j> in N do sum <i> in T : c[j]*a_s[j,i] <= 14000;

subto fillallonce: forall <i> in T do sum <j> in N : a_s[j,i] == 1;

subto noactivemorning: forall <i> in D do sum <j> in N: a[j]*a_s[j,i] == 0;

subto atmostoneactive: forall <l> in D do sum <i> in {1,2}: sum <j> in N : a[j]*a_s[j,l+i]
     <= 1;

subto dayonecultrelax: forall <j> in N do sum <i> in {1,2,3}: a[j]*a_s[j,i] == 0;

subto savelastdays: forall <d> in D with d >= max(D)-6 do sum <j> in N : sum <i> in {d,d
    +1,d+2}: c[j] * a_s[j,i] <= 700;

subto timeschosencult: forall <j> in N do sum <i> in T : a_s[j,i]*u[j] <= 1;

subto timeschosenrelax: forall <j> in N do sum <i> in T : a_s[j,i]*r[j] <= 3;

subto timeschosenactive: forall <j> in N do sum <i> in T : a_s[j,i]*a[j] <= 1;


subto favorites1: forall <i> in P do sum <j> in N : f[j,i]*s[j] >= 1;
```

Figure 3: The mathematical model implemented in ZIMPL

```
"skydiving",1028,1,0,0,1,1,0,0
"scubadiving",2600,1,0,0,0,1,0,0
"rock climbing",692, 1,0,0,0,0,0,1
"hiking",400, 1,0,0,1,0,1,0
"biking",200, 1,0,0,1,0,1,1
"enter marathon",5376,1,0,0,0,0,1,0
"bungeejumping",2000,1,0,0,0,0,0,0
"parkour",412, 1,0,0,0,0,1,1
"capoeira",200, 1,0,0,0,0,0,1
"skiing",2540,1,0,0,0,0,0,1
"snowboarding",609, 1,0,0,1,0,1,1
"horseback riding",2141,1,0,0,0,0,1,0
"rafting",1045, 1,0,0,0,1,1,0
"golf",1068, 1,0,0,0,0,1,0
"fishing",4800,1,0,0,0,1,0,0
"kayaking",2684, 1,0,0,1,1,0,0
"canoeing",779, 1,0,0,0,0,0,0
"ziplining",400, 1,0,0,0,1,1,1
"wind surfing",552, 1,0,0,0,0,0,1
"visit tower",340, 0,1,0,0,0,0,1
"visit great statue",88, 0,1,0,1,1,0,0
"explore great forest",1912,0,1,0,0,0,0,0
"see ancient wonder",316, 0,1,0,0,0,0,0
"visit war museum",333, 0,1,0,0,0,0,1
"visit art museum",528, 0,1,0,1,0,0,1
"visit modern art museum",656, 0,1,0,1,0,1,0
"explore outskirts of city",175, 0,1,0,0,0,0,0
"explore citycenter",88, 0,1,0,0,0,0,1
"visit marketplace",300, 0,1,0,0,1,0,0
"go on guided tour",1643,0,1,0,1,0,0,0
"visit famous church",301, 0,1,0,1,1,1,0
"visit zoo",742, 0,1,0,0,0,0,0
"visit animal park",1218, 0,1,0,0,0,0,0
"see great waterfall",292, 0,1,0,0,1,0,0
"travel by famous train route",2675, 0,1,0,1,0,0,1
"visit famous author's birthplace",155, 0,1,0,1,0,0,0
"attend music festival",2876, 0,1,0,0,0,0,0
"see great mountains",1140,0,1,0,0,0,0,0
"visit great lake",120, 0,1,0,1,0,1,1
"visit spa",3221, 0,0,1,0,0,0,0
"shopping",1600,0 0,1,0,0,1,0
"beach",112, 0,0,1,0,0,0,0
"go to amusement park",1432, 0,0,1,0,0,0,1
"relax at hotel",0, 0,0,1,0,0,0,0
"eat at buffet",753, 0,0,1,1,1,0,1
"eat at restaurant",535, 0,0,1,1,0,0,0
"get massage",275, 0,0,1,1,0,0,0
"go see a movie",202, 0,0,1,0,1,1,0
"go to a concert",2000,0 0,1,1,0,0,0
"go on picnic",113, 0,0,1,0,0,1,1
"buy souvenirs",800, 0,0,1,0,0,0,0
"grill",225, 0,0,1,0,1,0,1
"go sailing",2207, 0,0,1,0,0,1,0
"hot springs",122, 0,0,1,1,1,0,1
"visit cafe",210, 0,0,1,0,1,1,0
"visit bar",369, 0,0,1,0,0,1,1
```

Figure 4: An example of a data file: