Unit 3

1 Integer Linear Programming

In an Integer Linear Program (ILP) all the functions defining the objective and the constraints are linear and the variables move in $\mathbb{Z}^n$. A general form for an ILP is the following:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m \\ & x \in \mathbb{Z}^n. \end{aligned}$$

For the sake of notational simplicity, we will not formally deal with Mixed-Integer Linear Programs (MILPs), that are linear programs in which some variables are discrete while the others are continuous. However all the theoretical results and the methods presented in this unit can be easily generalized to tackle MILPs.

Solving a MILP is not as easy as solving an LP, but, nonetheless, it can be done by using one of the many numerical methods freely available (e.g. GLPK).

1.1 Continuous relaxations

The following LP

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m \\ & x \in \mathbb{R}^n. \end{aligned}$$

is called **continuous relaxation** of the ILP.

Remarks (all are due to the fact that the feasible set of the continuous relaxation contains the feasible set of the ILP):

- The optimal value of the continuous relaxation cannot be greater than the optimal value of the ILP.
- If a point $\overline{x} \in \mathbb{Z}^n$ is an optimal point of the continuous relaxation, then it is an optimal point of the ILP.
- Let $\overline{x} \in \mathbb{R}^n$ be an optimal point of the continuous relaxation. If a point $\widehat{x} \in \mathbb{Z}^n$ is feasible for the ILP and $\sum_{i=1}^{n} c_i \widehat{x}_i = \sum_{i=1}^{n} c_i \overline{x}_i$, then it is an optimal point of the ILP.

1.2 The branch and bound algorithm

The branch and bound algorithm can compute optimal points of any ILP that has at least one feasible point and that is not unbounded. The algorithm works by dividing the feasible region of the continuous relaxation in pieces, and by iteratively computing optimal points of the continuous relaxations whose feasible regions are restricted to these pieces.

The algorithm starts with a list of pieces that contains one single element representing the entire feasible set: $\mathcal{L} = \{S_0\}$, where $S_0 = \{x \in \mathbb{R}^n\}$. And it needs a point $\widehat{x}$, that is feasible for the ILP, that is the incumbent solution. At any iteration the algorithm performs these tasks:

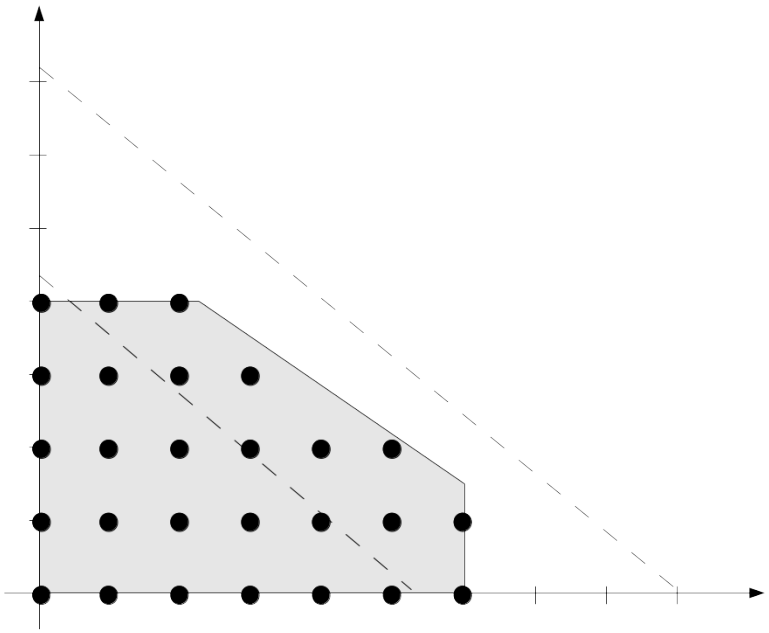
1. take a piece S_i from $\mathcal{L}$
2. if the feasible region of the continuous relaxation restricted to S_i is empty, then throw S_i away [end 1]
3. solve the continuous relaxation whose feasible region is restricted to S_i and obtain $\overline{x}$
4. **(bounding)** if the objective value of $\overline{x}$ is worse than the value of the incumbent solution $\widehat{x}$, then throw S_i away (eliminating many non-optimal candidates) [end 2]
5. **(branching)** if $\overline{x}$ is not feasible for the ILP, then get rid of it by splitting S_i into smaller pieces and put these pieces in $\mathcal{L}$ [end 3]
6. Set the candidate point $\overline{x}$ as the new incumbent solution $\widehat{x} \leftarrow \overline{x}$ and throw S_i away (eliminating many non-optimal candidates) [end 4]

The algorithm stops when $\mathcal{L}$ is empty. The final incumbent solution is a solution of the ILP.

Let us consider the following ILP

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -25x_1 - 30x_2 \\ \text{subject to} \quad & 7x_1 + 10x_2 \leq 56 \\ & 0 \leq x_1 \leq 6 \\ & 0 \leq x_2 \leq 4 \\ & x \in \mathbb{Z}^2. \end{aligned}$$

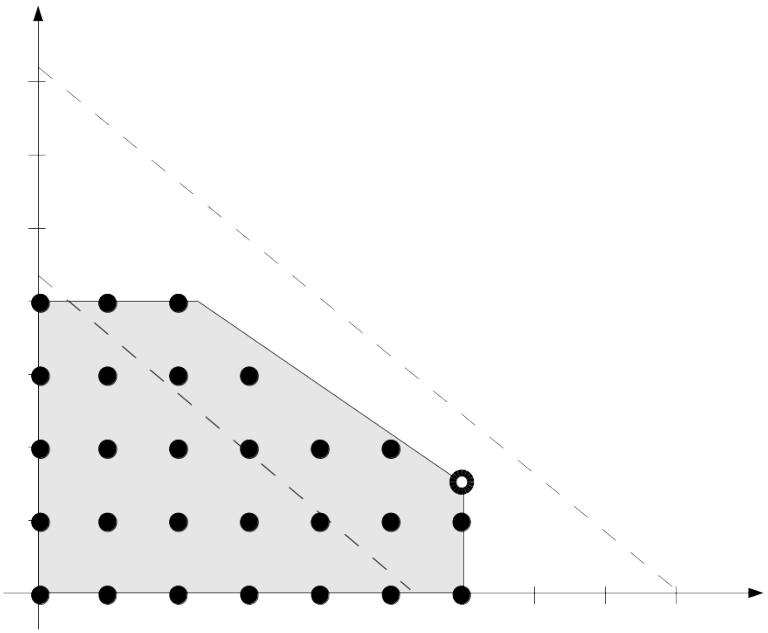
Whose graphical representation is the following



The optimal point is $x = (5 \;\; 2)^T$. [Check it numerically with GLPK]

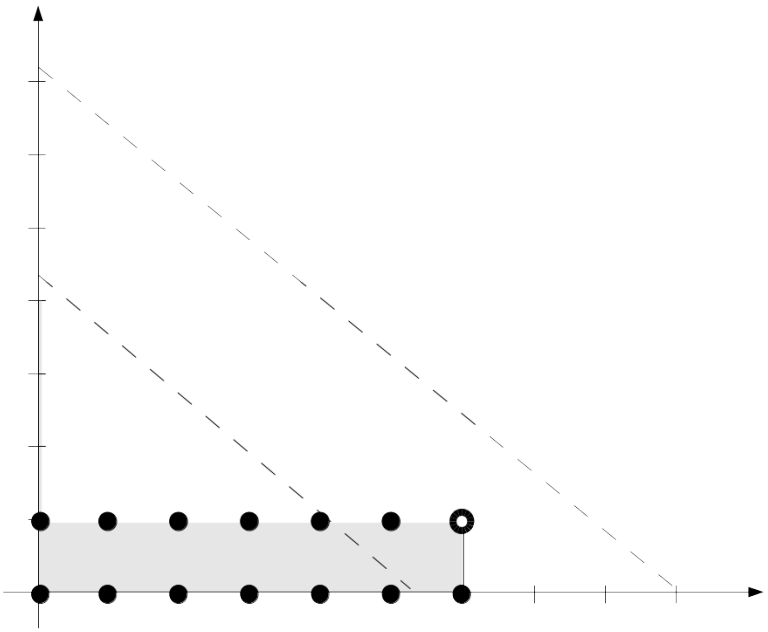
Let us simulate the iterations of the branch and bound algorithm. We initialize the list $\mathcal{L} = \{\{\mathbb{R}^2\}\}$ and the incumbent solution $\widehat{x} = (0 \;\; 0)^T$ whose objective value is 0.

- We consider $S_0 = \{\mathbb{R}^2\}$.



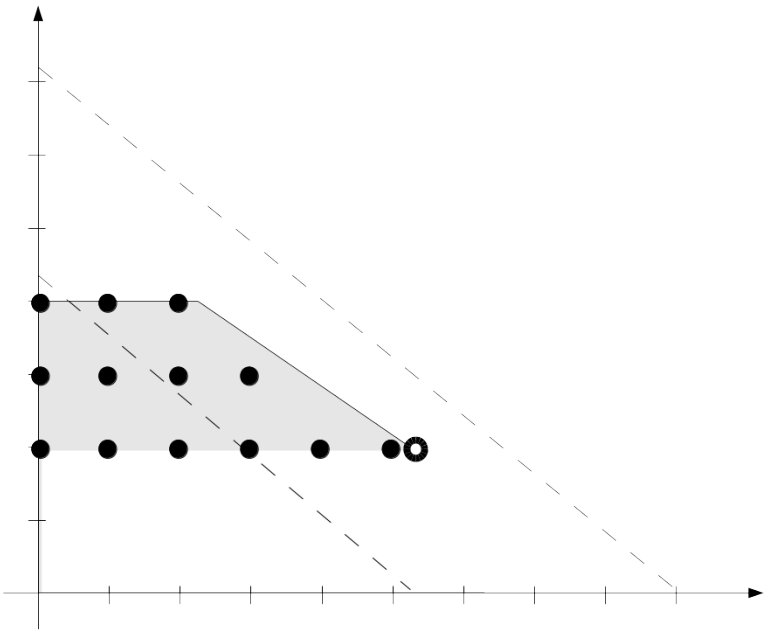
The optimal point of the continuous relaxation restricted to S_0 is $\overline{x} = (6 \;\; 1.4)^T$ whose objective value is -192. We cannot bound. We must branch: $\mathcal{L} = \{\{x_2 \leq 1\}, \{x_2 \geq 2\}\}$.

- We consider $S_1 = \{x_2 \leq 1\}$.



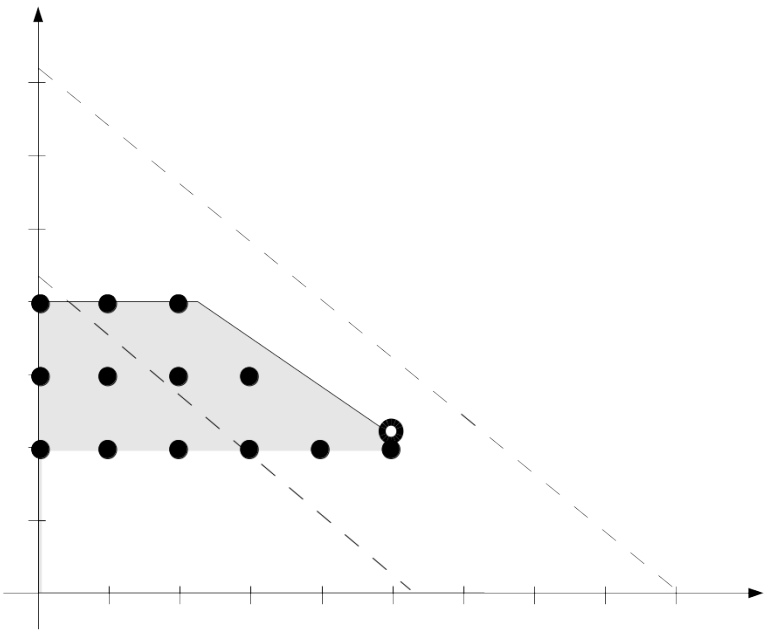
The optimal point of the continuous relaxation restricted to S_1 is $\overline{x} = (6 \;\; 1)^T$ whose objective value is -180. We cannot bound. We must not branch. We update the incumbent solution $\widehat{x} = (6 \;\; 1)^T$. The list is the following $\mathcal{L} = \{\{x_2 \geq 2\}\}$.

- We consider $S_2 = \{x_2 \geq 2\}$.



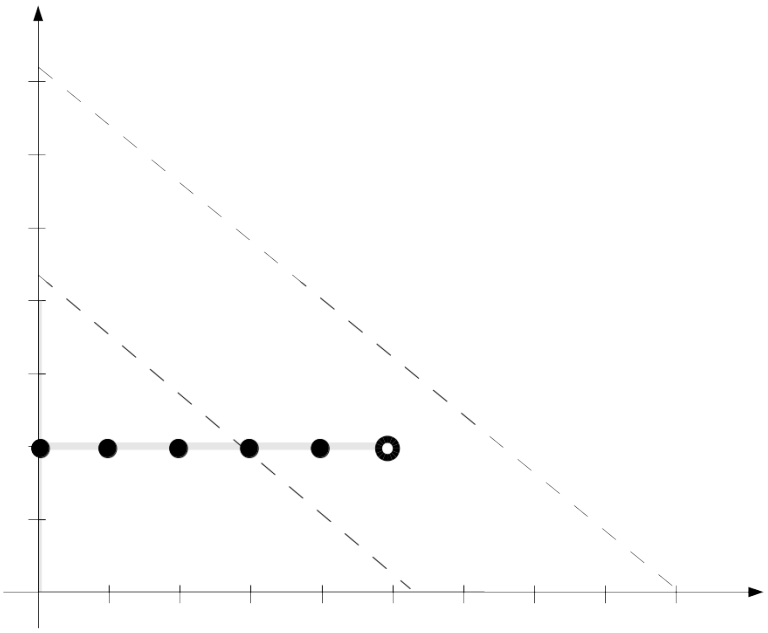
The optimal point of the continuous relaxation restricted to S_2 is $\overline{x} = \left(\frac{36}{7} \quad 2\right)^T$ whose objective value is -188.57. We cannot bound. We must branch: $\mathcal{L} = \{\{x_2 \geq 2, x_1 \leq 5\}, \{x_2 \geq 2, x_1 \geq 6\}\}$.

- We consider $S_3 = \{x_2 \geq 2, x_1 \leq 5\}$.



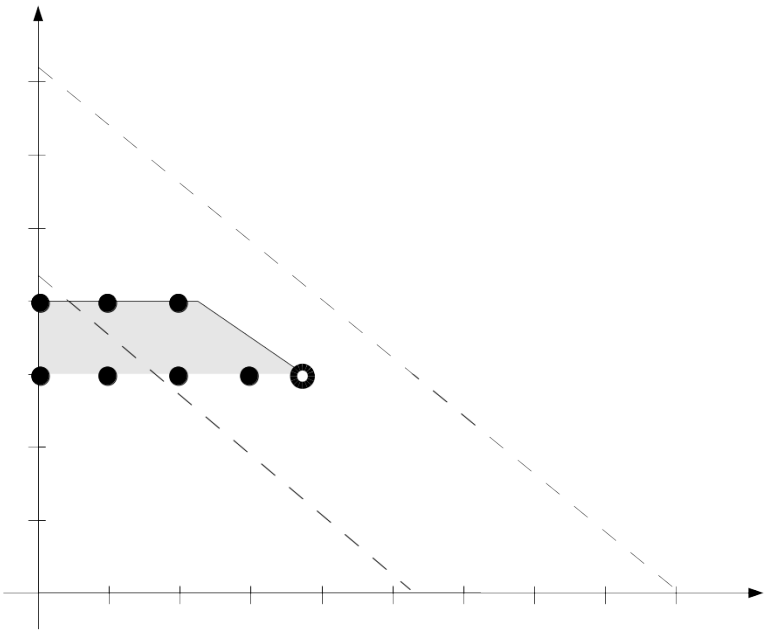
The optimal point of the continuous relaxation restricted to S_3 is $\overline{x} = (5 \;\; 2.1)^T$ whose objective value is -188. We cannot bound. We must branch: $\mathcal{L} = \{\{x_2 \geq 2, x_1 \geq 6\}, \{x_2 \geq 2, x_1 \leq 5, x_2 \leq 2\}, \{x_2 \geq 2, x_1 \leq 5, x_2 \geq 3\}\}$.

- We consider $S_4 = \{x_2 \geq 2, x_1 \geq 6\}$. We can throw it away because the feasible region of the continuous relaxation restricted to S_4 is empty. The list is the following $\mathcal{L} = \{\{x_2 \geq 2, x_1 \leq 5, x_2 \leq 2\}, \{x_2 \geq 2, x_1 \leq 5, x_2 \geq 3\}\}$.
- We consider $S_5 = \{x_2 \geq 2, x_1 \leq 5, x_2 \leq 2\}$.



The optimal point of the continuous relaxation restricted to S_5 is $\overline{x} = (5 \;\; 2)^T$ whose objective value is -185. We cannot bound. We must not branch. We update the incumbent solution $\widehat{x} = (5 \;\; 2)^T$. The list is the following $\mathcal{L} = \{\{x_2 \geq 2, x_1 \leq 5, x_2 \geq 3\}\}$.

- We consider $S_6 = \{x_2 \geq 2, x_1 \leq 5, x_2 \geq 3\}$.



The optimal point of the continuous relaxation restricted to S_6 is $\overline{x} = \left(\frac{26}{7} \;\; 3\right)^T$ whose objective value is -182.86. We can bound. The list is empty.

- $\widehat{x} = (5 \;\; 2)^T$ is an optimal point of the ILP.

Remarks:

- There are many ways to perform the branching phase. Here we simply split the region in 2 by adding the following alternative constraints: $x_i \leq \lfloor \overline{x}_i \rfloor$ and $x_i \geq \lceil \overline{x}_i \rceil$, where $\overline{x}_i$ is any non integer element.
- There are many different ways to select an element of $\mathcal{L}$ at any iteration. Here we use the First-In-First-Out strategy (FIFO), but also the LIFO or the random ones can be used as well.

1.3 Linear formulations

Let us denote as $P \triangleq \{x \in \mathbb{R}^n : \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \ldots, m\}$ the polyhedron defining the ILP.

Definition 1.1 *A polyhedron $\overline{P}$ is a* **linear formulation** *of the ILP if $\overline{P} \cap \mathbb{Z}^n = P \cap \mathbb{Z}^n$.*

Definition 1.2 *Let $\overline{P}$ and $\widehat{P}$ be two linear formulations of the ILP. We say that $\overline{P}$ is better than $\widehat{P}$ if $\overline{P} \subseteq \widehat{P}$.*

Remarks:

- Let $\overline{P}$ be a generic linear formulation of the ILP. The set of the optimal points of

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & x \in \overline{P} \cap \mathbb{Z}^n \end{aligned}$$

  coincides with that of the ILP.

- Let $\overline{P}$ and $\widehat{P}$ be two linear formulations of the ILP with $\overline{P}$ better than $\widehat{P}$. Let $\overline{x}$ be an optimal point of

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & x \in \overline{P}, \end{aligned}$$

  and $\widehat{x}$ be an optimal point of

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & x \in \widehat{P}. \end{aligned}$$

  It holds that $\sum_{i=1}^{n} c_i \overline{x}_i \geq \sum_{i=1}^{n} c_i \widehat{x}_i$.

Theorem 1.3 *Assume that any a_{ij} and any b_j are rational numbers. An* **optimal formulation** *P^* exists such that it is the convex hull of $P \cap \mathbb{Z}^n$.*

Remarks:

- The optimal formulation is the best linear formulation.

- A linear formulation of the ILP is optimal if and only if all its vertices are integer vectors.

- Let P^* be the optimal formulation of the ILP. Any optimal vertex of

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & x \in P^* \end{aligned}$$

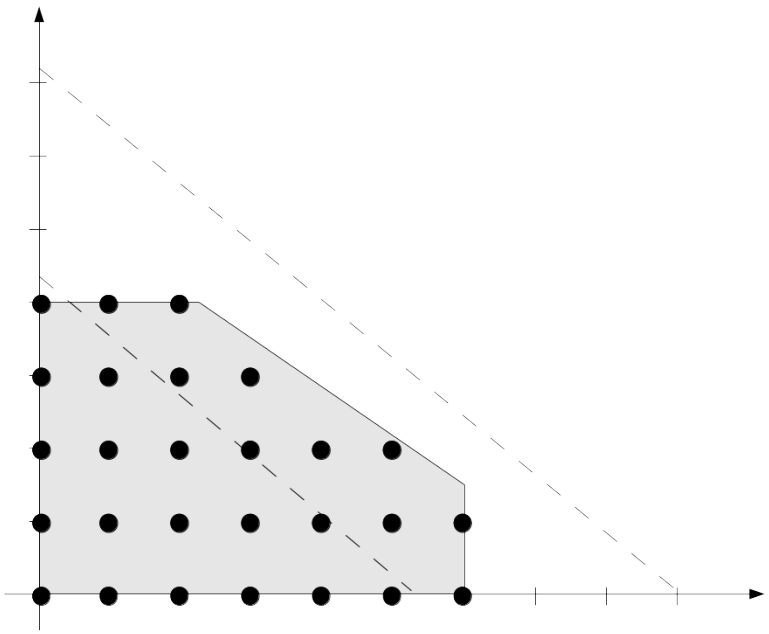
  is an optimal point of the ILP. Moreover its optimal value coincides with the optimal value of the ILP.

1.4 Adding cuts to get the optimal formulation

Given the ILP

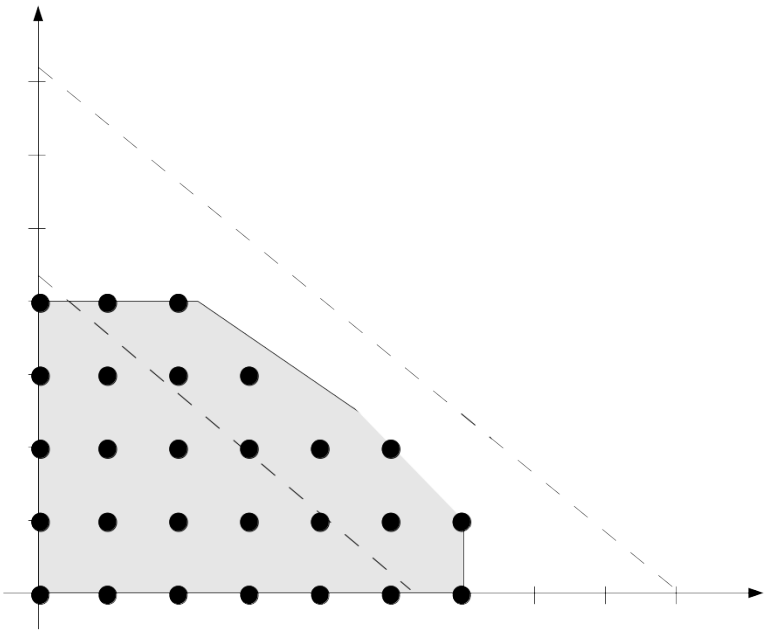
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -25x_1 - 30x_2 \\ \text{subject to} \quad & 7x_1 + 10x_2 \leq 56 \\ & 0 \leq x_1 \leq 6 \\ & 0 \leq x_2 \leq 4 \\ & x \in \mathbb{Z}^2, \end{aligned}$$

whose graphical representation is the following

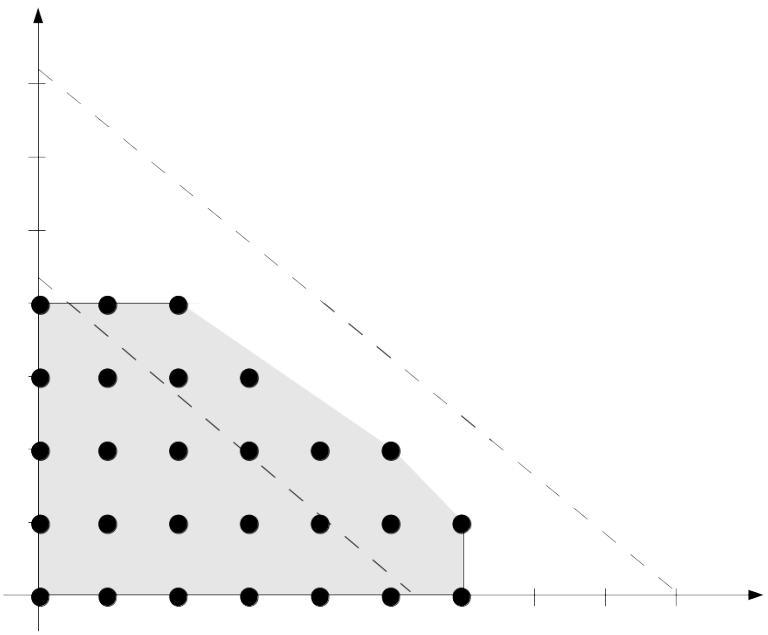


we can add to the formulation some cuts (that is some further constraints) in order to make the polyhedron the optimal formulation of the ILP. To do this, we simply need to add those constraints that delete all the non integer vertices without deleting any feasible point:

- To delete the vertex $(6 \;\; 1.4)^T$, we add the cut $x_1 + x_2 \leq 7$.



- To delete the vertices $\left(\frac{16}{7} \;\; 4\right)^T$ and $\left(\frac{14}{3} \;\; \frac{7}{3}\right)^T$, we add the cut $2x_1 + 3x_2 \leq 16$.



- All the vertices are integers, therefore we reach the optimal formulation. The point $(5 \;\; 2)^T$ is a solution of both the continuous relaxation and the ILP.

2 Models with integer variables

Including integer variables in a model can be useful for the following different reasons.

Indivisible quantities. If in the model some variables represents some indivisible quantities, then these variables must be considered in $\mathbb{Z}$ rather than in $\mathbb{R}$. For example both in the product mix and in the blending problems some quantities can be available only in stocks, in this case the models are MILPs.

Alternative choices. Integer variables can be used to represent alternative choices. In this case we have n binary variables $x \in \{0,1\}^n$ that model n possible different choices ($x \in \{0,1\}^n$ is equivalent to $x \in [0,1]^n$ and $x \in \mathbb{Z}^n$). For any choice i: $n_i = 0$ if the alternative i is not selected, while $n_i = 1$ if the alternative i is selected. Some possible constraints including these variables:

- The amount of alternatives selected cannot be less than $s \geq 0$:
$$\sum_{i=1}^{n} x_i \geq s.$$
- The amount of alternatives selected cannot be more than $p \geq 0$:
$$\sum_{i=1}^{n} x_i \leq p.$$
- The jth alternative can be selected only if all the alternatives $i \in I \subseteq \{1,\dots,n\} \setminus \{j\}$ are selected:
$$|I| x_j \leq \sum_{i \in I} x_i.$$
For example, alternative 1 can be selected only if alternatives 2, 3, and 5 are selected: $3x_1 \leq x_2 + x_3 + x_5$.

Indicator variables. An indicator variable $\delta \in \{0,1\}$ is a binary variable that is used to control the sign of a continuous variable $x \geq 0$. Let $M > 0$ be a huge number, let $\varepsilon > 0$ be a tiny number. We get the following results:

$$x \leq M\delta \quad \text{gives the relations} \quad \begin{cases} x > 0 & \Longrightarrow & \delta = 1 \\ \delta = 0 & \Longrightarrow & x = 0, \end{cases}$$

$$x \geq \varepsilon\delta \quad \text{gives the relations} \quad \begin{cases} x = 0 & \Longrightarrow & \delta = 0 \\ \delta = 1 & \Longrightarrow & x \geq \varepsilon, \end{cases}$$

$$x \leq M(1-\delta) \quad \text{gives the relations} \quad \begin{cases} x > 0 & \Longrightarrow & \delta = 0 \\ \delta = 1 & \Longrightarrow & x = 0, \end{cases}$$

$$x \geq \varepsilon(1-\delta) \quad \text{gives the relations} \quad \begin{cases} x = 0 & \Longrightarrow & \delta = 1 \\ \delta = 0 & \Longrightarrow & x \geq \varepsilon. \end{cases}$$

An indicator variables can be used, for example, to model these situations:

- At least one of the variables $x_1 \in [0, 1000]$ and $x_2 \in [0, 1000]$ must be equal to 0. In this case we add the constraints $\delta \in \{0, 1\}$, $x_1 \leq 1000\delta$ and $x_2 \leq 1000(1 - \delta)$. In fact we get the relations: $x_1 > 0 \;\Rightarrow\; \delta = 1 \;\Rightarrow\; x_2 = 0$ and $x_2 > 0 \;\Rightarrow\; \delta = 0 \;\Rightarrow\; x_1 = 0$.
- At least one of the variables $x_1 \in [0, 1000]$ and $x_2 \in [0, 1000]$ must be positive (that is, not less than ε). In this case we add the constraints $\delta \in \{0, 1\}$, $x_1 \geq \varepsilon\delta$ and $x_2 \geq \varepsilon(1 - \delta)$. In fact we get the relations: $x_1 = 0 \;\Rightarrow\; \delta = 0 \;\Rightarrow\; x_2 \geq \varepsilon$ and $x_2 = 0 \;\Rightarrow\; \delta = 1 \;\Rightarrow\; x_1 \geq \varepsilon$.
- At most $p \geq 0$ of the variables $x \in [0, 1000]^n$ can be positive. In this case we add the constraints $\delta \in \{0, 1\}^n$, $x_i \leq 1000\delta_i$, $i = 1, \ldots, n$, and $\sum_{i=1}^{n} \delta_i \leq p$.
- At least $s \geq 0$ of the variables $x \in [0, 1000]^n$ must be positive (that is, not less than ε). In this case we add the constraints $\delta \in \{0, 1\}^n$, $x_i \geq \varepsilon\delta_i$, $i = 1, \ldots, n$, and $\sum_{i=1}^{n} \delta_i \geq s$.

Disjunctive constraints. Constraints that must be partially satisfied, in the sense that a maximum amount of them can be violated, are called disjunctive constraints. Let $g_j(x) \leq b_j$ with $j \in J$ be k disjunctive constraints of which at least $\overline{k}$ must be satisfied. It is possible to model them by using k binary variables $\delta \in \{0, 1\}^k$:

$$g_j(x) \leq b_j + M(1 - \delta_j), \quad j = 1, \ldots, k, \qquad \sum_{j=1}^{k} \delta_j \geq \overline{k},$$

where $M > 0$ is a huge number.

3 Matching

Here the decision problem consists in how to match objects coming from two different groups. Any object can be matched at most with one object, and the two objects must come from different groups. Any possible match coupling an object i from group I with object j from group J has an utility u_{ij}. An optimal matching maximizes the total utility.

The model is an ILP:

$$\begin{aligned}
\underset{x}{\text{maximize}} \quad & \sum_{i \in I} \sum_{j \in J} u_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j \in J} x_{ij} \leq 1, \quad i \in I \\
& \sum_{i \in I} x_{ij} \leq 1, \quad j \in J \\
& x \in \{0, 1\}^{|I|} \times \{0, 1\}^{|J|}.
\end{aligned}$$

Notice that the variables x represent alternative choices: $x_{ij} = 1$ if and only if the match between i and j is active.

The model is defined in `matching1.mod`:

```
set I;
set J;

param u{I,J};

var x{I,J} binary;

maximize utility: sum{i in I, j in J} u[i,j]*x[i,j];

s.t. first_group {i in I}:
  sum{j in J} x[i,j] <= 1;

s.t. second_group {j in J}:
  sum{i in I} x[i,j] <= 1;

end;
```

Possible data is defined in `matching1.dat`:

```
set I := I1 I2 I3 I4;
set J := J1 J2 J3 J4 J5;

param u: J1 J2 J3 J4 J5 :=
I1 1 2 3 4 5
I2 3 2 1 2 2
I3 3 1 2 1 4
I4 2 2 4 1 1;
```

The output file:

```
Problem:    matching1
Rows:       10
Columns:    20 (20 integer, 20 binary)
Non-zeros:  60
Status:     INTEGER OPTIMAL
Objective:  utility = 15 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 utility                    15
     2 first_group[I1]
```

```
                                    1                           1
     3 first_group[I2]
                                    1                           1
     4 first_group[I3]
                                    1                           1
     5 first_group[I4]
                                    1                           1
     6 second_group[J1]
                                    1                           1
     7 second_group[J2]
                                    0                           1
     8 second_group[J3]
                                    1                           1
     9 second_group[J4]
                                    1                           1
    10 second_group[J5]
                                    1                           1

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 x[I1,J1]     *              0             0             1
     2 x[I1,J2]     *              0             0             1
     3 x[I1,J3]     *              0             0             1
     4 x[I1,J4]     *              1             0             1
     5 x[I1,J5]     *              0             0             1
     6 x[I2,J1]     *              1             0             1
     7 x[I2,J2]     *              0             0             1
     8 x[I2,J3]     *              0             0             1
     9 x[I2,J4]     *              0             0             1
    10 x[I2,J5]     *              0             0             1
    11 x[I3,J1]     *              0             0             1
    12 x[I3,J2]     *              0             0             1
    13 x[I3,J3]     *              0             0             1
    14 x[I3,J4]     *              0             0             1
    15 x[I3,J5]     *              1             0             1
    16 x[I4,J1]     *              0             0             1
    17 x[I4,J2]     *              0             0             1
    18 x[I4,J3]     *              1             0             1
    19 x[I4,J4]     *              0             0             1
    20 x[I4,J5]     *              0             0             1

Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
```

```
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```

4 Product mix with fixed costs

Consider again the product mix problem with concurrent resources. Assume that the quantities are integers. Moreover, assume that if the quantity q_i of product i is positive, then there is an additional fixed cost f_i to pay. The decision problem can be modeled as an ILP:

$$\begin{aligned}
\underset{q,\delta}{\text{maximize}} \quad & \sum_{i=1}^{n}(p_i q_i - f_i \delta_i) \\
\text{subject to} \quad & \sum_{i=1}^{n} a_{ij} q_i \leq b_j, \quad j = 1, \ldots, m \\
& l_i \leq q_i \leq u_i, \quad i = 1, \ldots, n \\
& q_i \leq u_i \delta_i, \quad i = 1, \ldots, n \\
& q \in \mathbb{Z}^n, \quad \delta \in \{0,1\}^n.
\end{aligned}$$

Notice that we added the indicator variables δ to model the fixed costs: if $q_i > 0$ then $\delta_i = 1$. Consider the same instance of before, with $f = (1000 \;\; 1100 \;\; 2500)^T$. We can compute an optimal point by employing GLPK. The model is defined in `product_mix5.mod`:

```
set P;
set R;

param p{P};
param a{P,R};
param b{R};
param l{P};
param u{P};
param f{P};

var q{i in P} >= l[i], <= u[i], integer;
var delta{P} binary;

maximize utility:
  sum{i in P} (p[i]*q[i] - f[i]*delta[i]);

subject to alternative_resources {j in R}:
```

```
  sum{i in P} a[i,j]*q[i] <= b[j];

subject to activation {i in P}:
  q[i] <= u[i]*delta[i];

end;
```

And the data is specified in `product_mix5.dat`:

```
set P := P1 P2 P3;
set R := R1 R2 R3;

param: p l u f :=
P1 1000 0 100 1000
P2 1500 0 100 1100
P3 2200 0 100 2500;

param a: R1 R2 R3 :=
P1 20 31 16
P2 30 42 81
P3 62 51 10;

param: b :=
R1 480
R2 480
R3 300;
```

The output file:

```
Problem:    product_mix5
Rows:       7
Columns:    6 (6 integer, 3 binary)
Non-zeros:  21
Status:     INTEGER OPTIMAL
Objective:  utility = 14700 (MAXimum)

   No.   Row name        Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 utility                 14700
     2 alternative_resources[R1]
                                 472                         480
     3 alternative_resources[R2]
                                 461                         480
     4 alternative_resources[R3]
                                 140                         300
     5 activation[P1]
                                 -95                          -0
```

```
     6 activation[P2]
                                    0                          -0
     7 activation[P3]
                                  -94                          -0

   No. Column name       Activity     Lower bound   Upper bound
------ ------------    ------------- ------------- -------------
     1 q[P1]        *              5             0           100
     2 q[P2]        *              0             0           100
     3 q[P3]        *              6             0           100
     4 delta[P1]    *              1             0             1
     5 delta[P2]    *              0             0             1
     6 delta[P3]    *              1             0             1

Integer feasibility conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
        High quality

End of output
```

5 Scheduling

There is a pool of n jobs that can be scheduled in a program. Each job i has an utility u_i and a time to completion t_i. Given two jobs i and j, they cannot be scheduled in parallel, and, if i is completed before j, then there is an utility penalty equal to p_{ij}. The program must end not after time T.

Maximizing the utility of this decision problem can be modeled as a MILP:

$$
\begin{aligned}
\underset{x,\delta,\gamma}{\text{maximize}} \quad & \sum_{i=1}^{n} u_i \delta_i - \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \gamma_{ij} \\
\text{subject to} \quad & t_i \delta_i \leq x_i, \quad i = 1, \ldots, n \\
& x_i \leq M \delta_i, \quad i = 1, \ldots, n \\
& x_i + t_j \delta_j \leq x_j + M \gamma_{ij}, \quad i = 1, \ldots, n, \; j = 1, \ldots, n, \; i \neq j \\
& \gamma_{ij} + \gamma_{ji} \leq 1, \quad i = 1, \ldots, n, \; j = 1, \ldots, n \\
& 0 \leq x_i \leq T, \quad i = 1, \ldots, n
\end{aligned}
$$

$$x \in \mathbb{R}^n, \quad \delta \in \{0,1\}^n, \quad \gamma \in \{0,1\}^n \times \{0,1\}^n.$$

Notice that

- any variable x_i indicates the time at which the ith job is completed (if $x_i = 0$ then the ith job is not in the program)
- δ are indicator variables: if $x_i > 0$ (that is the ith job is in the program), then $\delta_i = 1$
- γ are used to model the disjunctive precedence constraints: $\gamma_{ij} = 1$ if and only if the ith job is completed before the jth job.

The model is defined in `scheduling1.mod`:

```
set J;

param u{J};
param p{J,J};
param t{J};
param T;

var x{J} >= 0, <= T;
var delta{J} binary;
var gamma{J,J} binary;

maximize utility:
  sum{i in J} u[i]*delta[i] - sum{i in J, j in J} p[i,j]*gamma[i,j];

s.t. completion_time {i in J}:
  t[i]*delta[i] <= x[i];

s.t. activation {i in J}:
  x[i] <= 2*T*delta[i];

s.t. precedence1 {i in J, j in J : i != j}:
  x[i] + t[j]*delta[j] <= x[j] + 2*T*gamma[i,j];

s.t. precedence2 {i in J, j in J}:
  gamma[i,j] + gamma[j,i] <= 1;

end;
```

The data is specified in `scheduling1.dat`:

```
set J := J1 J2 J3 J4 J5 J6 J7 J8;

param: u t :=
J1 10 11
```

```
J2 20 19
J3 10 12
J4 20 21
J5 30 30
J6 10 16
J7 10 17
J8 20 26;

param p: J1  J2  J3  J4  J5  J6  J7  J8 :=
J1        0   2   2   1   1   1   2   1
J2      0.1   0   1   2   3   2   3   3
J3      0.1 0.1   0   1   2   3   2   1
J4      0.1 0.1 0.1   0   2   2   4   5
J5      0.1   3 0.1 0.1   0   6   7   6
J6      0.1 0.1 0.1   5 0.1   0   3   3
J7      0.1 0.1   8 0.1 0.1 0.1   0   4
J8      0.1  10 0.1 0.1 0.1 0.1 0.1   0;

param T := 100;
```