DM545

Linear and Integer Programming

Lecture 12
Preprocessing
More IP Modelling
Interior Point Methods

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Preprocessing rules

Consider $S = \{x : a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b, l_j \leq x_j \leq u_j, j = 0..n\}$

- Bounds on variables.
  If $a_0 > 0$ then:

$$x_0 \leq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0$$

  and if $a_0 < 0$ then

$$x_0 \geq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0$$

- Redundancy. The constraint $\sum_{j=0}^{n} a_j x_j \leq b$ is redundant if

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b$$

- Infeasibility: $S = \emptyset$ if

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j > b$$

- Variable fixing. For a max problem in the form

$$\max\{c^T x : Ax \leq b, l \leq x \leq u\}$$

if $\forall i = 1..m a_{ij} \geq 0, c_j < 0$ then fix $x_j = l_j$
if $\forall i = 1..m a_{ij} < 0, c_j > 0$ then fix $x_j = u_j$

- Integer variables:

$$\lceil l_j \rceil \leq x_j \leq \lfloor u_j \rfloor$$

- Binary variables. Probing: add a constraint, eg, $x_2 = 0$ and check what happens

Example

$$
\begin{array}{rl}
\max & 2x_1 + x_2 - x_3 \\
I & 5x_1 - 2x_2 + 8x_3 \leq 15 \\
II & 8x_1 + 3x_2 - x_3 \geq 9 \\
III & x_1 + x_2 + x_3 \leq 6 \\
 & 0 \leq x_1 \leq 3 \\
 & 0 \leq x_2 \leq 1 \\
 & x_3 \geq 1
\end{array}
$$

$$
\begin{array}{ll}
I : 5x_1 \leq 15 + 2x_2 - 8x_3 \leq 15 + 2 \cdot \overbrace{1}^{u_2} - 8 \cdot \overbrace{1}^{l_3} = 9 & \rightsquigarrow x_1 \leq 9/5 \\
\quad 8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 \cdot 1 - 5 \cdot 0 = 17 & \rightsquigarrow x_3 \leq 17/8 \\
\quad 2x_2 \geq 5x_1 + 8x_3 - 15 \geq 5 \cdot 0 + 8 \cdot 1 = -7 & \rightsquigarrow x_2 \geq -7/2, x_2 \geq 0 \\
II : 8x_1 \geq 9 - 3x_2 + x_3 \geq 9 - 3 + = 7 & \rightsquigarrow x_1 \geq 7/8 \\
I : 8x_3 \geq 15 + 2x_2 - 5x_1 \leq 15 + 2 - 5 \cdot 7/8 = 101/8 & \rightsquigarrow x_3 \leq 101/64
\end{array}
$$

$x_1 + x_2 + x_3 \leq 9/5 + 1 + 101/64 < 6$ Hence III is redundant

Example

$$\begin{array}{rl} \max & 2x_1 + x_2 - x_3 \\ I & 5x_1 - 2x_2 + 8x_3 \le 15 \\ II & 8x_1 + 3x_2 - x_3 \ge 9 \\ & 7/8 \le x_1 \le 9/5 \\ & 0 \le x_2 \le 1 \\ & 1 \le x_3 \le 101/64 \end{array}$$

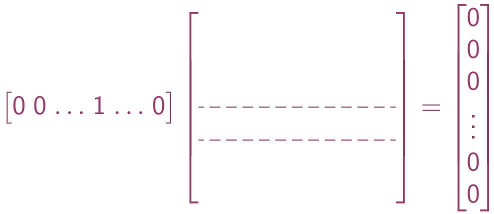
Increasing x_2 makes constraints satisfied $\rightsquigarrow$ $x_2 = 1$
Decreasing x_3 makes constraints satisfied $\rightsquigarrow$ $x_3 = 1$

We are left with:

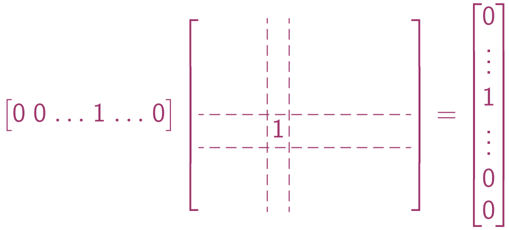
$$\max\{2x_1 : 7/8 \le x_1 \le 9/5\}$$

Preprocessing for Set Covering/Partitioning

1. if $e_i^T A = 0$ then the ith row can never be satisfied



2. if $e_i^T A = e_k$ then $x_k = 1$ in every feasible solution



In SPP can remove all rows t with $a_{tk} = 1$ and set $x_j = 0$ (and then remove cols) for all cols that cover t

3. if $e_t^T A \geq e_p^T A$ then we can remove row t, row p dominates row t (by covering p we cover t)

$$\left[\begin{array}{c|c|c}
 & & \\
\hline
1 & 1 & 1 \\
\hline
 & & \\
 & & \\
\hline
1 & & 1 \\
\hline
 & &
\end{array}\right]$$

In SPP we can remove all cols j:
$a_{tj} = 1, a_{pj} = 0$

4. if $\sum_{j \in S} Ae_j = Ae_k$ and $\sum_{j \in S} c_j \leq c_k$ then we can cover the rows by Ae_k more cheaply with S and set $x_k = 0$
(Note, we cannot remove S if $\sum_{j \in S} c_j \geq c_k$)

$$\left[\begin{array}{c|ccc|c|c|c}
 & & 1 & & & 1 & \\
 & 1 & & & & 1 & \\
 & & & 1 & & 1 & \\
 & 0 & 0 & 0 & & 0 & \\
 & 1 & & & & 1 & \\
 & 0 & 0 & 0 & & 0 &
\end{array}\right]$$

Try on exercise 11 of Sheet 5

Outline

Modeling with IP, BIP, MIP

Iterate:

1. def. variables
2. use variables to express objective function
3. use variables to express constraints

a. problems with discrete input/output (knapsack, factory planning)

b. problems with logical conditions

c. combinatorial problems (sequencing, allocation, transport, assignment, partitioning)

d. network problems

Variables

discrete quantities	$\in \mathbb{Z}^n$
decision variables	$\in \mathbb{B}^n$
indicator/auxiliary variables (for logical conditions)	$\in \mathbb{B}^n$
special ordered sets	$\in \mathbb{B}^n$
incidence vector of S	$\in \mathbb{B}^n$

In the next slides:

x	binary
y	integer
z	continuous

Logical Conditions

Linking constraints $x = 0$ if $z = 0$, $x = 1$ if $z > 0$

$$z > 0 \Longrightarrow x = 1 \Longrightarrow z - Mx \leq 0$$
$$x = 1 \Longrightarrow z > m \Longrightarrow z - mx \leq 0$$

Logical conditions and $0 - 1$ variables :

$$\begin{aligned}
X_1 \vee X_2 &\iff x_1 + x_2 \geq 1 \\
X_1 \wedge X_2 &\iff x_1 = 1, x_2 = 1 \\
\neg X_1 &\iff x_1 = 0 (1 - x_1 = 1) \\
X_1 \rightarrow X_2 &\iff x_1 - x_2 \leq 0 \\
X_1 \leftrightarrow X_2 &\iff x_1 - x_2 = 0
\end{aligned}$$

Examples

- $(X_A \vee X_B) \rightarrow (X_C \vee X_D \vee X_E)$

$$\begin{aligned} &x_A + x_B \geq 1 && x_C + x_D + x_E \geq 1 \\ &x_A + x_B \geq 1 \Longrightarrow x = 1 && x = 1 \Longrightarrow x_C + x_D + x_E \geq 1 \\ &x_A + x_B - 2x \leq 0 && x_C + x_D + x_E \geq x \end{aligned}$$

- Disjunctive constraints (encountered earlier)

- Constraint: $x_1 x_2 = 0$

  1) replace $x_1 x_2$ by x_3
  2) $x_3 = 1 \iff x_1 = 1, x_2 = 1$

$$\begin{aligned} -x_1 \phantom{{}- x_2} + x_3 &\leq 0 \\ - x_2 + x_3 &\leq 0 \\ x_1 + x_2 - x_3 &\leq 1 \end{aligned}$$

- Special ordered sets of type 1/2 (for continuous or integer vars):
  SOS1: set of vars within which exactly one must be non-zero
  SOS2: set of vars within which at most two can be non-zero. The two variables must be adjacent in the ordering

- separable programming and piecewise linear functions (next 5 slides)

- $z \cdot x$

  1) replace zx by z_1
  2) impose:

$$\begin{aligned} x = 0 &\iff z_1 = 0 \\ x = 1 &\iff z_1 = z \end{aligned}$$

$$\begin{aligned} z_1 - Mx &\leq 0 \\ -z + z_1 &\leq 0 \\ z - z_1 + Mx &\leq M \end{aligned}$$

Separable Programming

- Separable functions: sum of functions of single variables:

  $$x_1^2 + 2x_2 + e^{x^3} \quad \text{YES}$$

  $$x_1 x_2 + \frac{x_2}{x_1 + 1} + x_3 \quad \text{NO}$$

  (actually, some non-separable can also be made separable:

  1. $x_1 x_2$ by y
  2. relate y to x_1 and x_2 by:

  $$\log y = \log x_1 + \log x_2$$

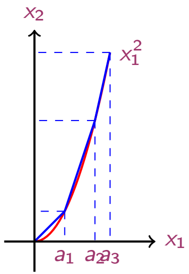
  needs care if x_1 and x_2 close to zero.)

- non-linear separable functions can be approximated by piecewise linear functions
  (valid for both constraints and objective functions)

Convex Non-linear Functions

- We can model convex non-linear functions by piece-wise linear functions and LP

$$\begin{array}{rrrl} \min & x_1^2 & -4x_1 & -2x_2 \\ & x_1 & + \; x_2 & \leq 4 \\ & 2x_1 & + \; x_2 & \leq 5 \\ & -x_1 & + \; 4x_2 & \geq 2 \\ & & x_1, x_2 & \geq 0 \end{array}$$



- LP Formulation

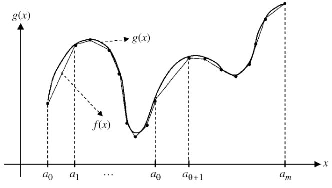
$$\begin{aligned} & x \; = \lambda_0 a_0 + \lambda_1 a_1 + \lambda_2 a_2 + \lambda_3 a_3 \\ & y \; = \lambda_0 f(a_0) + \lambda_1 f(a_1) + \lambda_2 f(a_2) + \lambda_3 f(a_3) \\ & \textstyle\sum_{i=0}^{3} \lambda_i = 1 \\ & \lambda_i \geq 0 \qquad i = 0, \ldots, 3 \\ & \text{at most two adjacent } \lambda_i \text{ can be non zero} \qquad (*) \end{aligned}$$

- To model (*) which are SOS2 we would need binary indicator variables and hence BIP as in next slide.
- However since the problem is convex, an optimal solution lies on the borders of the functions and hence we can skip introducing the binary variables and relax (*)

Non-convex Functions
Piece-wise Linear Functions

- non-convex functions require indicator variables and IP formulation

$$g(x) = \sum_j g_j(x) \quad g_j \text{ non linear}$$



$f(x)$ is a piecewise linear function of $g(x)$

- approximated by $f(x)$ piecewise linear in the disjoint intervals $[a_i, b_i]$
- convex hull formulation (convex combination of points)

$$\bigcup_{i \in I} \begin{pmatrix} x = & \lambda a_i + \mu b_i \\ y = & \lambda f(a_i) + \mu f(b_i) \\ \lambda + \mu = 1 & \lambda, \mu \geq 0 \end{pmatrix}$$

Remember how we modeled disjunctive polyhedra...

(cntd)

- using indicator variables δs we obtain the BIP formulation:

$$\begin{array}{l} x = \sum_{i \in I} (\lambda_i a_i + \mu_i b_i) \\ y = \sum_{i \in I} (\lambda_i f(a_i) + \mu_i f(b_i)) \\ \lambda_i + \mu_i = \delta_i \quad \forall i \in I \\ \sum_{i \in I} \delta_i = 1 \\ \lambda_i, \mu_i \geq 0 \quad \forall i \in I \\ \delta_i \in \{0,1\} \quad \forall i \in I \end{array}$$

the δs are SOS1.

Good/Bad Models

- Number of variables: sometimes it may be advantages increasing if they are used in search tree.

  $0-1$ var have specialized algorithms for preprocessing and for branch and bound. Hence a large number solved efficiently. Good using.

  Binary expansion:

  $$0 \leq y \leq u$$

  $$y = x_0 + 2x_1 + 4x_2 + 8x_3 + \ldots + 2^r x_r \qquad r = \log_2 u$$

- Making explicit good variables for branching:

  $$\sum_j a_j y_j \leq b$$

  $$\sum_j a_j x_j + u = b$$

  u may be a good variable to branch (u is relaxed in LP but must be integer as well)

- Symmetry breaking: Eg machine maintenance (see sol of Assignment 1, Task 4) $y_j \in \mathbb{Z}$ vs $x_j \in \mathbb{B}$
- Difficulty of LP models depends on number of constraints:

$$\min \sum_t |a_t z_t - b_t| \qquad \max \sum_t z'_t \qquad \max \sum_t z_t^+ - z_t^-$$

$$z'_t \geq a_t z_t - b_1 \qquad z_t^+ - z_t^- = a_t z_t - b_t$$

$$z'_t \geq b_t - a_t z_t$$

more variables but less constraints

- With IP it might be instead better increasing the number of constraints.
- Make M as small as possible in IP (reduces feasible region possibly fitting it to convex hull.

Practical Tips

- Units of measure: check them!
  all data should be scaled to stay in $0.1 - 10$
  some software do this automatically
- Write few line of text describing what the equations express and which are the variables, give examples on the problem modeled.
- Try the model on small simple example that can be checked by hand.
- Be diffident of infeasibility and unboundedness, double check.
- Estimate the potential size.
  If IP problem large and no structure then it might be hard.
  If TUM then solvable with very large size
  If other structure, eg, packing, covering also solvable with large size
- Check the output of the solver and understand what is happening
- If all fail resort to heuristics

Interiori-Point Approach to LP

- Karmakar in 1984, design and polinomiality proof
- Today most powerful software packages include at least an interior point algorithm
- It is an itertative algorithm that gets started by identifying a feasible trial solution. Moves from this towards an optimal solution.
- Main difference from simplex: it moves through interiori points, not boundaries.
- Also called barrier method (constraints are treated as barriers)

- The average computation time per iteration is higher than the simplex
- For small problems simplex works therefore best.
- For large problems with many cosntraints (eg, 10.000) the simplex requires many iteraitons (20.000) while iterior-point algorithms less (100) and are therefore prefereable.
- Simplex is well suited for post-optimality analysis, while this is not the case for interior-point algorithms
- Hence complementary roles:
  up to 10.000 constraints or 100.000 constraints and unlimited variables: use simplex
  convert solutions from interior-point to simplex.

Conclusion

- I hope this course has changed your way of thinking when you hear the word "optimization"
- I hope to have transmitted part of my enthusiasm for this subject.
  Look also at the videos and blog linked from the web page.
- Where from here:
  DM811: Heuristics for Combinatorial Optimization
  DM204: Scheduling Timetabling and Routing
  DM208/DM209: Combinatorial Optimization I/II
  DM817: Network Programming: Theory and Applications
- Thesis and ISA
  Applications are really welcome!
  Timetabling at the Faculty of Science, Crew Scheduling and Air Craft Routing with Air Support, Routing with CargoMatch, Fairness issues, Advanced techniques for GCP and sum coloring, Curricula generation, ...
  or come with your own problem!