DM826 – Spring 2014

Modeling and Solving Constrained Optimization Problems

Lecture 1
Course Introduction
Hybrid Modeling

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

[*partly based on slides by Stefano Gualandi, Politecnico di Milano*]

Outline

1. Course Introduction

2. Modeling in MP and CP

Outline

1. Course Introduction

2. Modeling in MP and CP

Schedule and Material

- Schedule:
  - Monday 12.15-14
  - Wednesday 16.15-18
  - Thursday 16.15-18
  - Break in week 9!
  - Officially last lecture in Week 13, Thursday, 27th March, 2014
- Communication tools
  - Public Course Webpage (Wp)
    http://www.imada.sdu.dk/~marco/DM826/
  - In Blackboard (Bb):
    - Announcements
    - Documents (Photocopies)
  - Discussion board in Bb
  - Personal email
  - You are welcome to visit me in my office in working hours.

Evaluation

- Two obligatory assignments (50% of final grade)
  - Model
  - Implementation
  - Report (3 pages)
- Third final assignment (50% of final grade)
  - Model
  - Implement
  - Report (Max 10 pages)

References

- Main References:

  - B1 F. Rossi, P. van Beek and T. Walsh (ed.), Handbook of Constraint Programming, Elsevier, 2006
  - B2a C. Schulte, G. Tack, M.Z. Lagerkvist, Modelling and Programming with Gecode 2013
  - B2b MiniZinc tutorial

- Photocopies (Bb)
- Articles from the Webpage
- Lecture slides
- Assignments

- Active participation

Software

Under development:
http://www.minizinc.org/challenge2013/results2013.html

Here, we will use free and open-source software:

- Gecode (C++) – MIT license
- OR-tools (C++) – Apache license 2.0
- Python vs MiniZinc – BSD-style license

Outline

Computational Models

Three main Computational Models to solve (combinatorial) constrained optimization problems:

- Mathematical Programming (LP, ILP, QP, SDP, ...)
- Constraint Programming (CSP as a model, SAT as a very special case)
- Local Search (... and Meta-heuristics)

- Others? Dynamic programming, dedicated algorithms, satisfiability modulo theory, etc.

Modeling

Modeling:

1. identify:
   - variables and domains
   - constraints
   - objective functions

   that formulate the problem

2. express what in point 1) in a way that allows the solution by available software

Variables

In MILP: real and integer variables

In CP:

- finite domain integer (including Booleans),
- continuos with interval constraints
- structured domains: finite sets, multisets, graphs, ...

Constraint Programming

- In MILP we formulate problems as a set of linear inequalities
- In CP we describe substructures (so-called global constraints) and combine them with various combinators.
- Substructures capture building blocks often (but not always) comptuationally tractable by special-purpose algorithms
- CP models can:
  - solved by the constraint engine
  - be linearized and solved by their MIP solvers;
  - be translated in CNF and sovled by SAT solvers;
  - be handled by local search
- In MILP the solver is often seen as a black-box
  In CP and LS solvers leave the user the task of programming the search.
- CP = model + propagation + search
  constraint propagation by domain filtering $\rightsquigarrow$ inference
  search = backtracking, branch and bound, local search

Example: Sudoku

How can you solve the following Sudoku?

	4	3		8		2	5	
6								
					1		9	4
9					4		7	
			6		8			
	1		2					3
8	2		5					
								5
	3	4		9		7	1	

Sudoku: ILP model

Let y_{ijt} be equal to 1 if digit t appears in cell (i, j). Let N be the set $\{1, \ldots, 9\}$, and let J_{kl} be the set of cells (i, j) in the 3×3 square in position k, l.

$$\sum_{j \in N} y_{ijt} = 1, \qquad \forall i, t \in N,$$

$$\sum_{j \in N} y_{jit} = 1, \qquad \forall i, t \in N,$$

$$\sum_{i,j \in J_{kl}} y_{ijt} = 1, \qquad \forall k, l = \{1, 2, 3\}, t \in N,$$

$$\sum_{t \in N} y_{ijt} = 1, \qquad \forall i, j \in N,$$

$$y_{i,j,a_{ij}} = 1, \qquad \forall i, j \in \text{ given instance.}$$

Sudoku: CP model

$$
\begin{aligned}
&X_{ij} \in N, && \forall i, j \in N,\\
&X_{ij} = a_{ij}, && \forall i, j \in \text{ given instance},\\
&\text{alldifferent}([X_{1i}, \dots, X_{9i}]), && \forall i \in N,\\
&\text{alldifferent}([X_{i1}, \dots, X_{i9}]), && \forall i \in N,\\
&\text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), && \forall k, l \in \{1, 2, 3\}.
\end{aligned}
$$

Sudoku: CP model (revisited)

$$X_{ij} \in N, \qquad \forall i, j \in N,$$
$$X_{ij} = a_t, \qquad \forall i, j \in \text{ given instance},$$
$$\text{alldifferent}([X_{1i}, \dots, X_{9i}]), \qquad \forall i \in N,$$
$$\text{alldifferent}([X_{i1}, \dots, X_{i9}]), \qquad \forall i \in N,$$
$$\text{alldifferent}(\{X_{ij} \mid ij \in J_{kl}\}), \qquad \forall k, l \in \{1, 2, 3\}.$$

Redundant Constraint:

$$\sum_{j \in N} X_{ij} = 45, \qquad \forall i \in N,$$
$$\sum_{j \in N} X_{ji} = 45, \qquad \forall i \in N,$$
$$\sum_{ij \in J_{\boldsymbol{kl}}} X_{ij} = 45, \qquad k, l \in \{1, 2, 3\}.$$

Hybrid Methods?

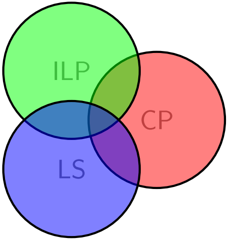
Strengths:

- CP is excellent to explore highly constrained combinatorial spaces quickly
- Math programming is particulary good at deriving lower bounds
- LS is particualry good at derving upper bounds

How to combine them to get better "solvers"?

- Exploiting OR algorithms for filtering
- Exploiting LP (and SDP) relaxation into CP
- Hybrid decompositions:

  1. Logical Benders decomposition
  2. Column generation
  3. Large-scale neigbhrohood search



Integrated Modeling

Models interact with solution process hence models in CP and IP are different.

To integrate one needs:

- to know both sides
- to have available a modelling language that allow integration (python, C++, MiniZinc)

There are typcially alternative ways to formulate a problem. Some may yield faster solutions.

Typical procedure:

- begin with a strightforward model to solve a small problem instance
- alter and refine the model while scaling up the instances to maintain tractability

Linear Programming

Linear Programming

Given A matrix $A \in \mathbb{R}^{m \times n}$ and column vectors $b \in \mathbb{R}^m, c \in \mathbb{R}^n$.

Task Find a column vector $x \in \mathbb{R}^n$ such that $Ax \leq b$ and $c^T x$ is maximum, decide that $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ is empty, or decide that for all $\alpha \in \mathbb{R}$ there is an $x \in \mathbb{R}^n$ with $Ax \leq b$ and $c^T x > \alpha$.

Theory vs. Practice

In theory the Simplex algorithm is exponential, in practice it works.

In theory the Ellipsoid algorithm is polynomial, in practice it is worse than the Simplex.(Interior point methods are polynomial and competitive with the Simplex.)

Integer Programming

Integer Programming

Given A matrix $A \in \mathbb{Z}^{m \times n}$ and vectors $b \in \mathbb{Z}^m, c \in \mathbb{Z}^n$.

Task Find a vector $x \in \mathbb{Z}^n$ such that $Ax \leq b$ and cx is maximum,
or decide that $\{x \in \mathbb{Z}^n \mid Ax \leq b\} = \emptyset$,
or decide that $\sup\{cx \mid x \in \mathbb{Z}^n, Ax \leq b\} = \infty$.

Theory vs. Practice

In theory, IP problems can be solved efficiently by exploiting (if you can find-/approximate) the convex hull of the problem.
In practice, we heavily rely on branch&bound search tree algorithms, that solve LP relaxations at every node.

Logical Statements: Frequently (but not always) the integer variables are restricted to be in {0,1} representing Yes/No decisions.

Quadratic Programming

Quadratic Programming

Given: Matrices $A, Q_i \in \mathbb{R}^{n \times n}$, with $i = 0, \ldots, q$, and column vectors $a_i, b, c \in \mathbb{R}^n$.

Task: Find a column vector $x \in \mathbb{R}^n$ such that $x^T Q_i x + a_i^T x \leq b$ and $x^T Q_0 X + c^T x$ is maximum,
or decide that $\{x \in \mathbb{R}^n \mid x^T Q_i x + a_i^T x \leq b\}$ is empty,
or decide that it is unbounded.

Theory vs. Practice

In theory, this is a richer modeling language (quadratic constraints and/or objective functions).
In practice, we linearize all the time, relying on (most of the time linear) cutting plane algorithms.

In Cplex

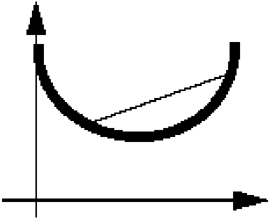
http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r2/topic/ilog.odms.cplex.help/Content/Optimization/Documentation/CPLEX/_pubskel/CPLEX486.html

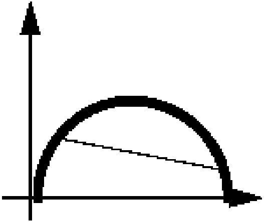
Example
Quadratic programming (QP), quadratically-constrained programming (QCP), mixed integer quadratic programming (MIQP), and mixed-integer quadratically-constrained programming (MIQCP).
Conventionally, a quadratic program is formulated this way:

$$\begin{array}{ll} \min & c^T x + 1/2 x^T Q x \quad (c_1 x_1 + \ldots c_n x_n + q_{11} x_1 x_1 + q_{12} x_1 x_2 + \ldots q_{nn} x_n x_n) \\ s.t. & Ax \sim b \\ & a_i^T x + x^T Q_i x \leq r_i \text{ for } i = 1, ..., q \\ & lb \leq x \leq ub \end{array}$$

Q is a matrix of coefficients. That is, the elements Q_{jj} are the coefficients of the quadratic terms x_j^2, and the elements Q_{ij} and Q_{ji} are summed to make the coefficient of the term $x_i x_j$.
The same for the Q_i in the constraints

In Cplex





The question whether a quadratic objective function is convex (or concave) is equivalent to whether the matrix Q is positive semi-definite (or negative semi-definite).

For convex QPs, Q must be positive semi-definite; that is, $x^T Qx \geq 0$ for every vector x, whether or not x is feasible.

$$
\begin{aligned}
\min\ & x_1 + 2x_2 + 3x_3 + \frac{1}{2}(-33x_1^2 + 12x_1x_2 - 22x_2^2 + 23x_2x_3 - 11x_3) \\
& - x_1 + x_2 + x_3 \leq 20 \\
& x_1 - 3x_2 + x_3 \leq 30 \\
& + x_1^2 + x_2^2 + x_3^2 \leq 1
\end{aligned}
$$

CPLEX Examples:

- quadratic objective function: `qpex1.py` and `qpex1.lp`
- quadratic constraints: `qcpex1.py` and `qcpex1.lp`

Gurobi Examples:

- `qp.py` and `qcp.py`

Example: Quadratic Assignment Problem

- **Given:**
  n units with a matrix $F = [f_{ij}] \in \mathbf{R}^{n \times n}$ of flows between them and
  n locations with a matrix $D = [d_{uv}] \in \mathbf{R}^{n \times n}$ of distances
- **Task:** Find the assignment σ of units to locations that minimizes the sum of product between flows and distances, ie,

$$\min_{\sigma \in \Sigma} \sum_{i,j} f_{ij} d_{\sigma(i)\sigma(j)}$$

Applications: hospital layout; keyboard layout

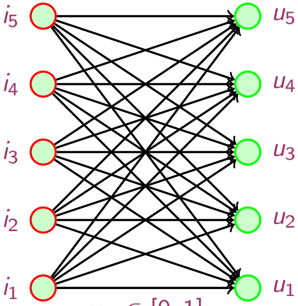
Example: QAP

$$D = \begin{pmatrix} 0 & 4 & 3 & 2 & 1 \\ 4 & 0 & 3 & 2 & 1 \\ 3 & 3 & 0 & 2 & 1 \\ 2 & 2 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 3 & 4 \\ 2 & 2 & 0 & 3 & 4 \\ 3 & 3 & 3 & 0 & 4 \\ 4 & 4 & 4 & 4 & 0 \end{pmatrix}$$

The optimal solution is $\sigma = (1, 2, 3, 4, 5)$, that is,
facility 1 is assigned to location 1,
facility 2 is assigned to location 2, etc.

The value of $f(\sigma)$ is 100.

Quadratic Programming Formulation



indices i, j for units and u, v for locations:
Quadratic 0-1 problem:

$$\min \sum_i \sum_u \sum_j \sum_v f_{ij} d_{uv} x_{iu} x_{jv}$$

$$\sum_i x_{iu} = 1 \qquad \forall u$$

$$\sum_u x_{iu} = 1 \qquad \forall i$$

$$x_{iu} \in \{0, 1\}$$

Largest instances solvable exactly $n = 30$

A possible linearization with $y_{iujv} = x_{iu}x_{jv}$ (Adams-Johnson model)

$$\min \sum_{i,u,j,v} a_{uv} b_{ij} y_{iujv}$$

$$\sum_i x_{iu} = 1 \qquad \forall u$$

$$\sum_u x_{ui} = 1 \qquad \forall i$$

$$\sum_v y_{iujv} = x_{iu} \qquad \forall i, u, j$$

$$\sum_j y_{iujv} = x_{iu} \qquad \forall i, u, v$$

$$y_{iujv} = y_{jviu} \qquad \forall i, u, j, v$$

$$x_{iu} \geq 0 \qquad \forall i, u$$

$$y_{iujv} \geq 0 \qquad \forall i, u, j, v$$

$y_{ijij} = x_{ij}$ for all i and j,
$y_{iuiv} = 0$ for all i and $u \neq v$,
and $y_{iuju} = 0$ for all $i \neq j$
$\rightsquigarrow n^2 + n^2(n-1)/2$ variables.

Constraints
$2n(n-1)2-(n-1)(n-2), n \geq 3$.

In practice

Modeling Languages (e.g., AMPL, Mosel, AIMMS, ZIMPL, MiniZinc, OPL,...)

Write your problem as:

$$\min\{\mathbf{c}^T\mathbf{z} + \mathbf{d}^T\mathbf{y} \mid A\mathbf{z} + B\mathbf{y} \geq b, z \in \mathbb{R}^n, y \in \mathbb{Z}\}$$

push the button solve, and ... cross your fingers!

Theory vs. Practice

In theory, plenty of optimization problem solved in this manner.
In practice, for many real-life discrete (optimization) problems this approach is not suitable (typically, it does not scale well).

The case of Integer Programming

The problem with Integer Programming [Williams, 2010]

IP is essentially concerned with the intersection of two structures:

1. Linear inequalities giving rise to polytopes.
2. Lattices of integer points.

Mathematical and computational methods and results exist for both these structures on their own. Problems arise in both the computation of optimal solutions and the economic interpretation of the results.

Example:

How many times do we really have (an approximation of) the convex hull in our integer problem?

References

Williams H. (2010). **The problem with integer programming**. Tech. Rep. LSE0R 10-118, London School of Economics and Political Science.