DM204, 2010
SCHEDULING, TIMETABLING AND ROUTING

Lecture 5
Mixed Integer Programming
Models and Exercises

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

1. An Overview of Software for MIP
2. ZIBOpt
3. Models

How to solve MIP programs

- Use a mathematical workbench like MATLAB, MATHEMATICA, MAPLE, R.
- Use a modeling language to convert the theoretical model to a computer usable representation and employ an out-of-the-box general solver to find solutions.
- Use a framework that already has many general algorithms available and only implement problem specific parts, e. g., separators or upper bounding.
- Develop everything yourself, maybe making use of libraries that provide high-performance implementations of specific algorithms.

Thorsten Koch
"Rapid Mathematical Programming"
Technische Universität, Berlin, Dissertation, 2004

How to solve MIP programs

- Use a mathematical workbench like MATLAB, MATHEMATICA, MAPLE, R.

Advantages: easy if familiar with the workbench

Disadvantages: restricted, not state-of-the-art

How to solve MIP programs

- Use a modeling language to convert the theoretical model to a computer usable representation and employ an out-of-the-box general solver to find solutions.

Advantages: flexible on modeling side, easy to use, immediate results, easy to test different models, possible to switch between different state-of-the-art solvers

Disadvantages: algoritmical restrictions in the solution process, no upper bounding possible

How to solve MIP programs

- Use a framework that already has many general algorithms available and only implement problem specific parts, e.g., separators or upper bounding.

Advantages: allow to implement sophisticated solvers, high performance bricks are available, flexible

Disadvantages: view imposed by designers, vendor specific hence no transferability,

How to solve MIP programs

- Develop everything yourself, maybe making use of libraries that provide high-performance implementations of specific algorithms.

Advantages: specific implementations and max flexibility

Disadvantages: for extremely large problems, bounding procedures are more crucial than branching

Modeling Languages

Name		URL	Solver	State
AIMMS	Advanced Integrated Multi-dimensional Modeling Software	www.aimms.com	open	commercial
AMPL	A Modeling Language for Mathematical Programming	www.ampl.com	open	commercial
GAMS	General Algebraic Modeling System	www.gams.com	open	commercial
LINGO	Lingo	www.lindo.com	fixed	commercial
LPL	(Linear|Logical|Literate) Programming Language	www.virtual-optima.com	open	commercial
MINOPT	Mixed Integer Non-linear Optimizer	titan.princeton.edu/MINOPT	open	mixed
MOSEL	Mosel	www.dashoptimization.com	fixed	commercial
MPL	Mathematical Programming Language	www.maximalsoftware.com	open	commercial
OMNI	Omni	www.haverly.com	open	commercial
OPL	Optimization Programming Language	www.ilog.com	fixed	commercial
GNU-MP	GNU Mathematical Programming Language	www.gnu.org/software/glpk	fixed	free
ZIMPL	Zuse Institute Mathematical Programming Language	www.zib.de/koch/zimpl	open	free

Thorsten Koch
"Rapid Mathematical Programming"
Technische Universität, Berlin, Dissertation, 2004

LP-Solvers

CPLEX	http://www.ilog.com/products/cplex
XPRESS-MP	http://www.dashoptimization.com
SOPLEX	http://www.zib.de/Optimization/Software/Soplex
COIN CLP	http://www.coin-or.org
GLPK	http://www.gnu.org/software/glpk
LP_SOLVE	http://lpsolve.sourceforge.net/

"Software Survey: Linear Programming" by Robert Fourer
http://www.lionhrtpub.com/orms/orms-6-05/frsurvey.html

MIP-Solvers

CPLEX	http://www.ilog.com/products/cplex
SCIP	http://zibopt.zib.de/
GUROBI	http://www.gurobi.com/

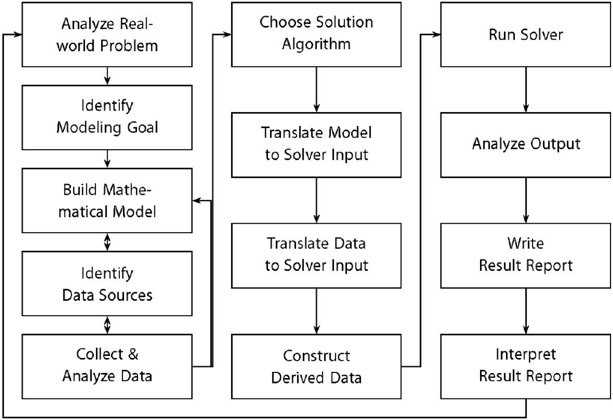
Outline

1. An Overview of Software for MIP
2. ZIBOpt
3. Models

ZIBOpt

- Zimpl is a little algebraic Modeling language to translate the mathematical model of a problem into a linear or (mixed-) integer mathematical program expressed in .lp or .mps file format which can be read and (hopefully) solved by a LP or MIP solver.
- Scip is an IP-Solver. It solves Integer Programs and Constraint Programs: the problem is successively divided into smaller subproblems (branching) that are solved recursively. Integer Programming uses LP relaxations and cutting planes to provide strong dual bounds, while Constraint Programming can handle arbitrary (non-linear) constraints and uses propagation to tighten domains of variables.
- SoPlex is an LP-Solver. It implements the revised simplex algorithm. It features primal and dual solving routines for linear programs and is implemented as a C++ class library that can be used with other programs (like SCIP). It can solve standalone linear programs given in MPS or LP-Format.

Modeling Cycle



H. Schichl. "Models and the history of modeling".
In Kallrath, ed., Modeling Languages in Mathematical Optimization, Kluwer, 2004.

Outline

Modeling

- Min cost flow
- Shortest path
- Max flow
- Assignment and Bipartite Matching
- Transportation
- Multicommmodies

Modeling

Set Covering

$$\min \sum_{j=1}^{n} c_j x_j$$
$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i$$
$$x_j \in \{0,1\}$$

Set Partitioning

$$\min \sum_{j=1}^{n} c_j x_j$$
$$\sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i$$
$$x_j \in \{0,1\}$$

Set Packing

$$\max \sum_{j=1}^{n} c_j x_j$$
$$\sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall i$$
$$x_j \in \{0,1\}$$

Traveling Salesman Problem

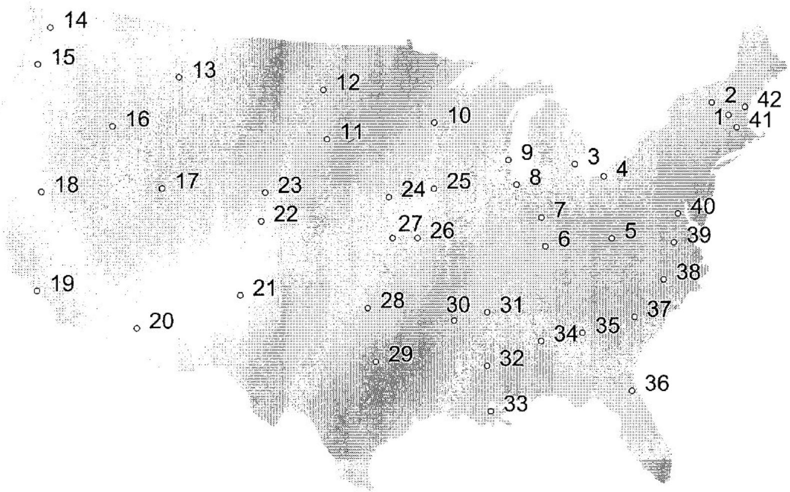


Figure 3.1 Locations of the 42 cities.

Traveling Salesman Problem

Assignment problem - easy, naturally integer.

Indices: i = teacher, j = course.

Parameters: c_{ij} = value if teacher i is assigned to course j.

Variables: $x_{ij} = 1$ if teacher i is assigned to course j, else 0.

Model AP: 1) Max $\sum_i \sum_j c_{ij} x_{ij}$ subject to

2) $\sum_j x_{ij} = 1$, for all i,

3) $\sum_i x_{ij} = 1$, for all j,

4) $x_{ij} \in \{0,1\}$, for all i,j.

Explanation: 1) Maximise value of assignments.

2) Assign each teacher i to one course.

3) Assign each course j to one teacher.

Almost the TSP. Is AP a possible formulation for the TSP?

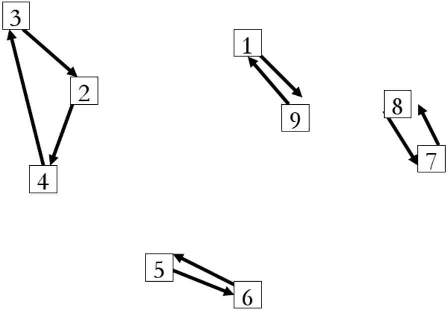
Indices: i, j = city.

Parameter: c_{ij} = cost to go from city i to city j.

Variables: $x_{ij} = 1$ if we drive from city i to city j, else 0.

Traveling Salesman Problem

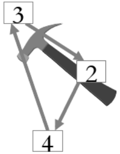
We have subtours.



Oops. How do we get rid of these?

Traveling Salesman Problem

Ways to break subtours: 2^n subtour constraints.



The Dantzig, Fulkerson & Johnson (DFJ) model.

Indices, parameters, & decision variables as before.

Minimise total cost:	$\min \sum_i \sum_j c_{ij} x_{ij}$,
Enter each city once:	$\sum_i x_{ij} = 1$ for all j.
Leave each city once:	$\sum_j x_{ij} = 1$ for all i.
Subtour breaking constraints:	$\sum_{i,j \in S} x_{ij} \leq \lvert S \rvert - 1$, for every subset S.
Binary integrality:	$x_{ij} \in \{0, 1\}$ for all i, j.

For the subtour shown, add: $x_{3,2} + x_{2,4} + x_{4,3} \leq 2$. What are the others?

After solving again with the new constraints, more subtours appear.

For a large TSP, we may need many subtour breaking constraints.

In the worst case, we may need 2^n subtour breaking constraints.

Next week, we will see a way to generate these constraints.

The solution becomes fractional, so we also need to do B&B.

However, every solution gives a lower bound on the optimum.

Traveling Salesman Problem

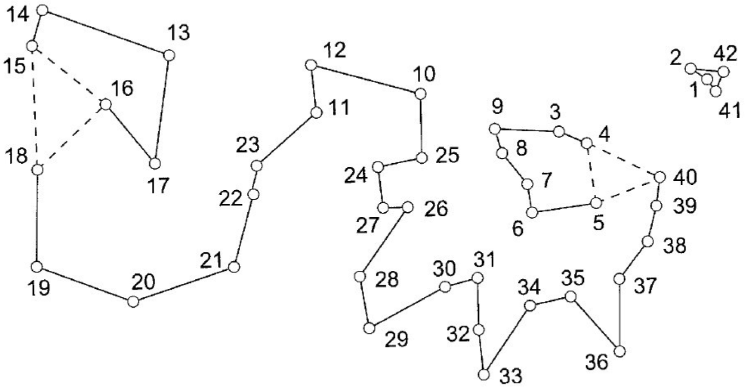


Figure 3.2 Solution of the initial LP relaxation.

Traveling Salesman Problem

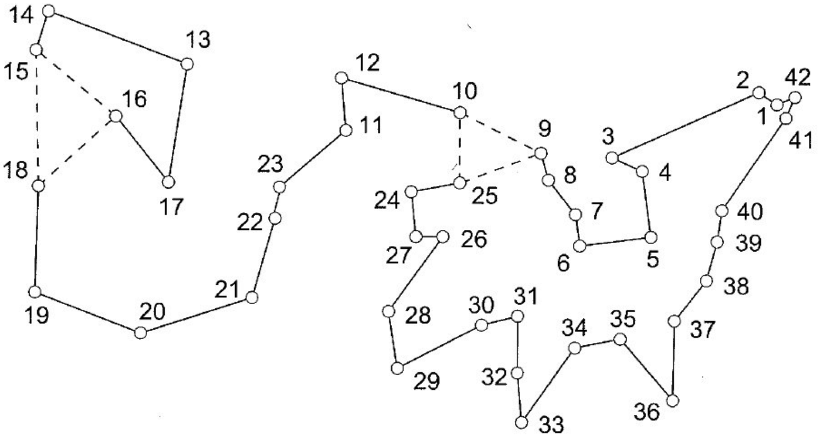


Figure 3.3 LP solution after three subtour constraints.

Traveling Salesman Problem

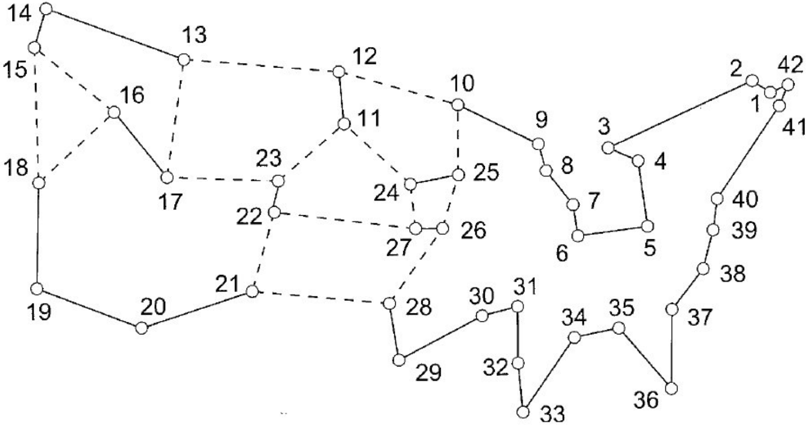


Figure 3.4 LP solution satisfying all subtour constraints.

Traveling Salesman Problem

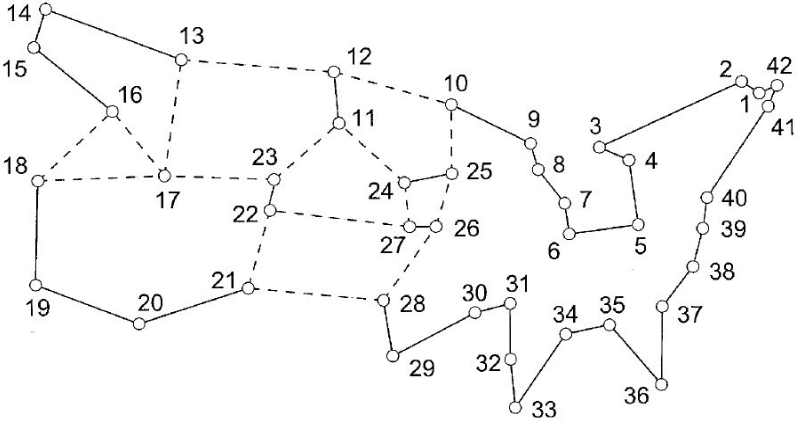


Figure 3.7 What is wrong with this vector?

Traveling Salesman Problem

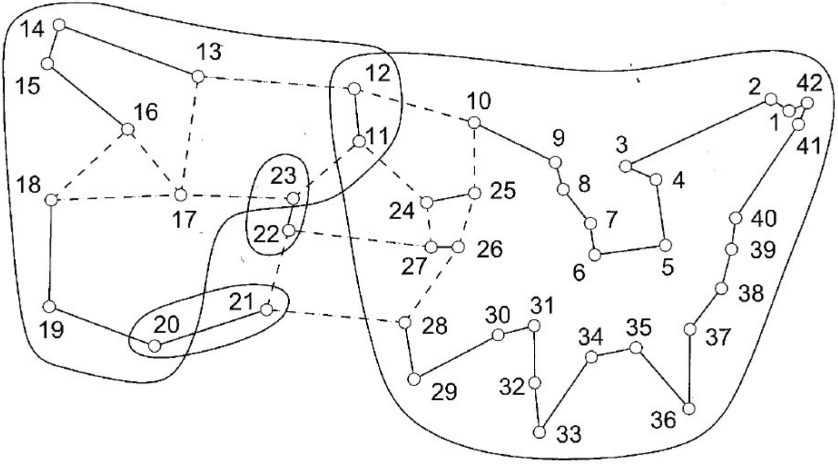


Figure 3.8 A violated comb.

Traveling Salesman Problem

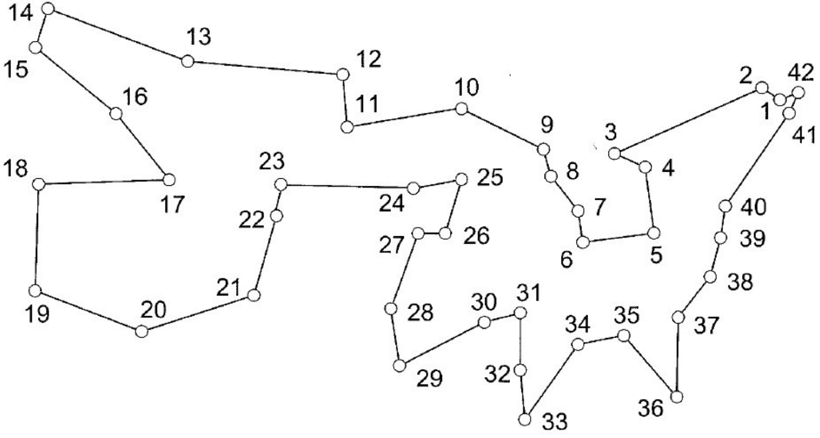


Figure 3.9 An optimal tour through 42 cities.

minimize $c^T x$ subject to

$0 \leq x_e \leq 1$ for all edges e,

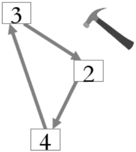
$\sum(x_e \ : \ v$ is an end of $e) = 2$ for all cities v,

$\sum(x_e \ : \ e$ has one end in S and one end not in $S) \geq 2$
for all nonempty proper subsets S of cities,

$\sum_{i=0}^{i=3}(\sum(x_e \ : \ e$ has one end in S_i and one end not in $S_i) \geq 10$,
for any comb

Traveling Salesman Problem

Ways to break subtours: MTZ model



Indices & parameters as before.

Variables: $x_{ij} = 1$ if we drive from city i to city j, else 0.

u_i = number of cities visited at city i.

Minimise total cost: $\min \sum_i \sum_j c_{ij} x_{ij}$,

Enter each city once: $\sum_i x_{ij} = 1$ for all j.

Leave each city once: $\sum_j x_{ij} = 1$ for all i.

Subtour breaking: $u_i + 1 \leq u_j + n(1 - x_{ij})$, for $i = 2, \ldots, n$, $i \neq j$, $j = 2, \ldots, n$,

$x_{ij} \in \{0, 1\}$ for all i, j; $u_i \geq 0$ for all i.

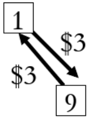
Fewer constraints, but harder to solve! The LP relaxation is not as tight.
Okay for small problems, but is bad for large ones.
Related variations are a bit tighter.

Ref: C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulations and traveling salesman problems," *J. ACM*, 7 (1960), pp. 326–329.

Traveling Salesman Problem

1

The symmetric TSP

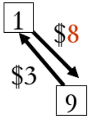


Symmetric TSP: $c_{ij} = c_{ji}$.

Indices: i, j = city.
Parameter: c_{ij} = cost to go from city i to city j.
Variables: x_{ij} = 1 if we drive from city i to city j, else 0, defined only for $i<j$. Half as many variables as the asymmetric!

Minimise total cost: $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$,
Enter each city once: $\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2$ for all i.
Subtour breaking: $\sum_{i,j \in S} x_{ij} \leq |S| - 1$, for each subset S.
Binary integrality: $x_{ij} \in \{0, 1\}$ for all i, j.

The homework is a symmetric TSP.



The asymmetric TSP, $c_{ij} \neq c_{ji}$, is more realistic. Why?

Traveling Salesman Problem

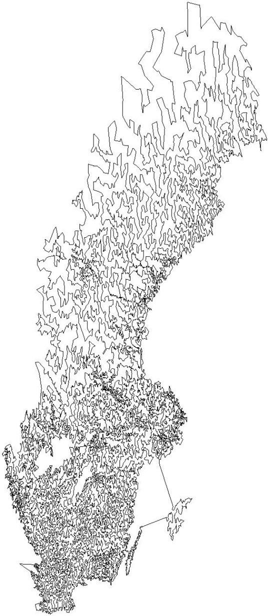
How does the row 2 summation work?

Model: 1. Min $\sum^{n}_{i=1} \sum^{n}_{j=i+1} c_{i,j} x_{i,j}$,
2. $\sum_{i=1}^{j-1} x_{i,j} + \sum^{n}_{i=j+1} x_{j,i} = 2$, for all j.
3. $\sum_{i,j \in S} x_{i,j} \leq |S| - 1$, for every subset S,
4. $x_{i,j} \in \{0,1\}$ for all i, and j: $j > i$.

The variables *into* city 5 are: x15, x25, x35, x45, x65, x75, x85, x95.
The variables *out of* city 5 are: x51, x52, x53, x54, x56, x57, x58, x59.

Since costs are symmetric, $c_{ij} = c_{ji}$, let's drop half the variables.
For x_{ij}, require $i < j$. Allow only the variables going out.
We need only variables x15, x25, x35, x45, x56, x57, x58, x59.
The meaning is not "Go in" or "come out", but "use this arc".

The summation makes sure that we cover only the variables we need.
x15 + x25 + x35 + x45 + x56 + x57 + x58 + x59 = 2.

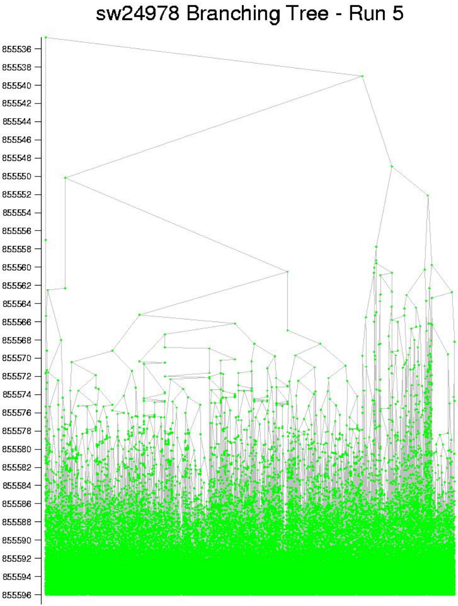


24,978 Cities

solved by LK-heuristic and prooved optimal by branch and cut

10 months of computation on a cluster of 96 dual processor Intel Xeon 2.8 GHz workstations

http://www.tsp.gatech.edu



24,978 Cities

solved by LK-heuristic and prooved optimal by branch and cut

10 months of computation on a cluster of 96 dual processor Intel Xeon 2.8 GHz workstations

http://www.tsp.gatech.edu