SoPlex

and

Zimpl

Thorsten Koch, ZIB

$$\min\ c^T x$$

subject to $Ax \geqslant b$

with
$$c \in \mathbb{R}^n$$
$$x \in \mathbb{R}^n$$
$$A \in \mathbb{R}^{k \times n}$$
$$b \in \mathbb{R}^k$$

and $k \geqslant n$ and A has full rank

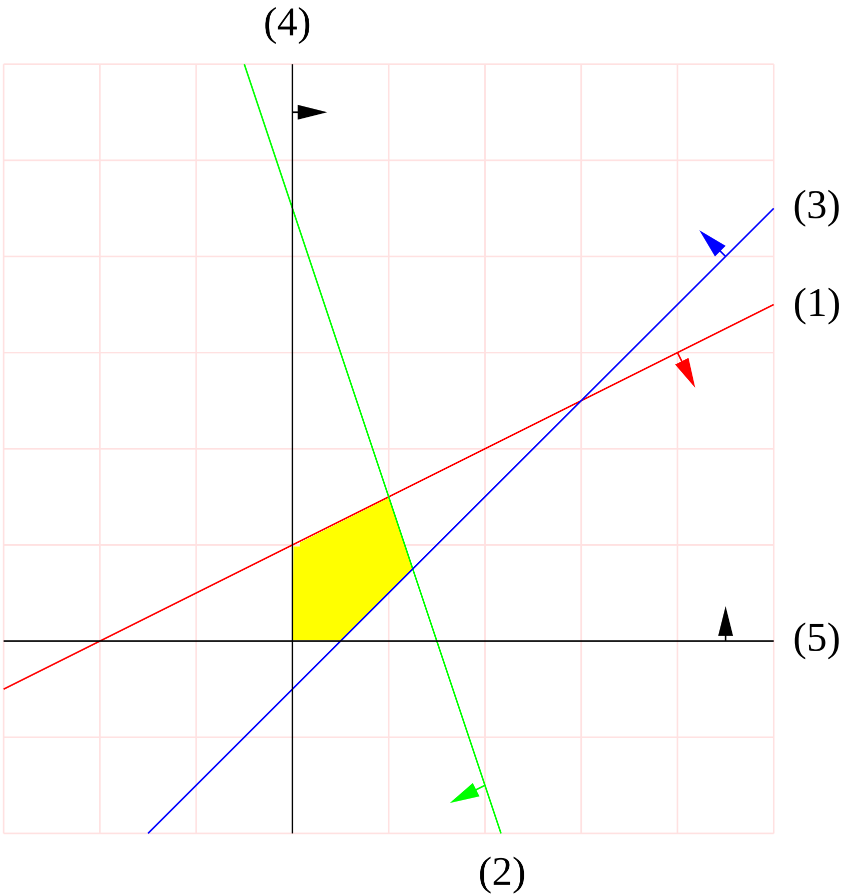
$$\max\ c^T x$$

$$\text{subject to} \quad Ax \leqslant b$$

$$x \geqslant 0$$

with $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$.

Linear program example



$$\min \; -x \; -y$$

subject to

$$\begin{array}{rrcrl}
x_1 & -2x_2 & \geqslant & -2 & (1)\\
-6x_1 & -2x_2 & \geqslant & -9 & (2)\\
-x_1 & +2x_2 & \geqslant & -1 & (3)\\
x_1 & & \geqslant & 0 & (4)\\
 & x_2 & \geqslant & 0 & (5)
\end{array}$$

The **S**equential **o**bject-oriented sim**plex** class library

- Implementation of the revised simplex algorithm
- Primal and dual solving routines
- Row and column based basis representation
- Can solve instances with a million non-zero elements
- C++ class library and standalone program
- Very portable: compiles with C++ compilers from GNU, Compaq, Intel, SUN, HP, SGI, IBM, and even M$
- Licensed to several commercial companies
- Free for non commercial academic use

SoPlex is a linear program solver.

It can not solve integer programs,

i.e. $x \in \mathbb{Z}^n$.

For this we have SCIP, please wait for the next talk.

Status

- Currently one of the top free LP-Solvers
- Initially developed in 1996 by Roland Wunderling, as part of his PhD thesis *Paralleler und Objektorientierter Simplex-Algorithmus*
- SoPlex is under continous development
- While slower than, for example, CPLEX, adaquate for many tasks

Available at

http://www.zib.de/Optimization/Software/Soplex

Zuse Institute Mathematical Programming Language

- Algebraic modeling language, like AMPL, GAMS, etc.
- Transforms mathematical descriptions of linear mixed integer models into solver input
- Easy to use
- Has been used in several industry projects and lectures
- Can generate models with more than 10 million variables
- Released under GNU GPL

Example

A Zimpl model consists of

- **Sets**
- **Parameters**
- **Variables**
- **Objective**
- **Constraints**

Zimpl example: TSP model

Let $G = (V, E)$ be a complete graph, with V being the set of cities and E being the set of links between the cities. Introducing binary variables x_{ij} for each $(i, j) \in E$ indicating if edge (i, j) is part of the tour, the TSP can be written as:

$$
\begin{array}{rll}
\min \displaystyle\sum_{(i,j)\in E} d_{ij} x_{ij} & & \text{subject to} \\
\displaystyle\sum_{(i,j)\in \delta_v} x_{ij} & = 2 & \text{for all } v \in V \\
\displaystyle\sum_{(i,j)\in E(U)} x_{ij} & \leqslant |U| - 1 & \text{for all } U \subseteq V, \emptyset \neq U \neq V \\
x_{ij} & \in \{0, 1\} & \text{for all } (i, j) \in E
\end{array}
$$

Zimpl example: data

The data is read in from a file that lists for each city the name and the x and y coordinates. Distances between cities are assumed Euclidean.

```
# City         X    Y
Berlin      5251 1340
Frankfurt   5011  864
Leipzig     5133 1237
Heidelberg  4941  867
Karlsruhe   4901  840
Hamburg     5356  998
Bayreuth    4993 1159
Trier       4974  668
Hannover    5237  972
Stuttgart   4874  909
Passau      4856 1344
Augsburg    4833 1089
Koblenz     5033  759
Dortmund    5148  741
Bochum      5145  728
Duisburg    5142  679
Wuppertal   5124  715
Essen       5145  701
Jena        5093 1158
```

Zimpl example: code


```
set      V          := { read "tsp.dat" as "<1s>" comment "#" };
set      E          := { <i,j> in V * V with i < j };
set      P[]        := powerset(V);
set      K          := indexset(P);
param    px[V]      := read "tsp.dat" as "<1s> 2n" comment "#";
param    py[V]      := read "tsp.dat" as "<1s> 3n" comment "#";
defnumb  dist(a,b):= sqrt((px[a]-px[b])^2 + (py[a]-py[b])^2);
var      x[E] binary;

minimize cost: sum <i,j> in E : dist(i,j) * x[i, j];

subto two_connected: forall <v> in V do
   (sum <v,j> in E : x[v,j]) + (sum <i,v> in E : x[i,v]) == 2;
subto no_subtour:
   forall <k> in K with
      card(P[k]) > 2 and card(P[k]) < card(V) - 2 do
      sum <i,j> in E with <i> in P[k] and <j> in P[k] : x[i,j]
      <= card(P[k]) - 1;
```

Zimpl example: result

The resulting linear program has

171	variables,
239,925	constraints, and
22,387,149	non-zero entries

in the constraint matrix, giving an MPS-file size of 936 mb.

An optimal tour for the data on the previous slide is **Berlin, Hamburg, Hannover, Dortmund, Bochum, Wuppertal, Essen, Duisburg, Trier, Koblenz, Frankfurt, Heidelberg, Karlsruhe, Stuttgart, Augsburg, Passau, Bayreuth, Jena, Leipzig, Berlin.**

Benefits

- Modeling languages make it much easier to rapidly experiment with models and ideas
- The reproducibility of the results is increased
- Higher solver independency
- Solver dependent transformation of special functions are possible

Extended functions

For $a, b, c \in \mathbb{Z}$, $a \in [-15, 15], b \in [-10, 20], c \in [-20, 10]$,

maximize $5a + 3b + c$ subject to:

$$\begin{aligned}
\text{if } & (a \neq b \text{ and } (|a - b| = 3c \text{ or } c - a \geqslant 0)) \\
\text{then } & a + b + c \geqslant 7 \\
\text{else } & a + b \leqslant 1
\end{aligned}$$

can be formulated as an IP. But it is rather difficult…

Extended functions: Zimpl

For $a, b, c \in \mathbb{Z}$, $a \in [-15, 15]$, $b \in [-10, 20]$, $c \in [-20, 10]$,

maximize $5a + 3b + c$ subject to:

$$\text{if } (a \neq b \text{ and } (|a - b| = 3c \text{ or } c - a \geqslant 0))$$

$$\text{then } a + b + c \geqslant 7$$

$$\text{else } a + b \leqslant 1$$

```
var a integer >= -15 <= 15;
var b integer >= -10 <= 20;
var c integer >= -20 <= 10;
maximize obj: 5 * a + 3 * b + c;
subto c1: vif (a!=b and (vabs(a-b) == 3*c or c-a >= 0))
      then a + b + c >= 7
      else a + b <= 1 end;
```

Extended functions: result

```
- xp0  + bp2                           <= 0
- b    + a        - xp0  + bm3         <= 0
- xp0  + 25 bp2                        >= 0
- b    + a        - xp0  + 35 bm3      >= 0
- bp2  - bm3      + re4                 = 0
- b    + a        + xm6  - xp5          = 0
- xp5  + 25 bp7                        >= 0
+ xm6  + 35 bp7                        <= 35
- xm6  - xp5      + re8                 = 0
- xp9  + bp11                          <= 0
- 3 c  + re8      - xp9  + bm12        <= 0
- xp9  + 95 bp11                       >= 0
- 3 c  + re8      - xp9  + 30 bm12     >= 0
+ bp11 + bm12     + re13                = 1
- xp14 + bp16                          <= 0
+ c    - a        - xp14 + bm17        <= 0
- xp14 + 25 bp16                       >= 0
+ c    - a        - xp14 + 35 bm17     >= 0
+ bp16 + bm17                          <= 1
+ re13 - re19                          <= 0
- bm17 - re19                          <= -1
+ re13 - bm17     - re19               >= -1
+ re4  - re20                          >= 0
+ re19 - re20                          >= 0
+ re4  + re19     - re20               <= 1
+ c    + b        + a    - 52 re20     >= -45
```

IP after CPLEX presolve:
27 rows, 20 columns, 77 NZE.

Optimal solution:

c	2
b	20
a	14
xm1	6
bm3	1
re4	1
xm6	6
re8	6
re13	1
xm15	12
bm17	1
re19	1
re20	1

Where?

www.zib.de/koch/zimpl

THANK YOU!

QUESTIONS?