Business Computing

Decision Support & OR Lab

mops®

mathematical optimization systems

MOPS Overview

Prof. Dr. Uwe H. Suhl

CEO MOPS Optimierungssysteme GmbH & Co KG

Universität Paderborn

(Freie Universität Berlin)

Progress of LP/IP-engines in MOPS:

- Shell - a mixed integer optimization model unsolvable on mainframes 1985 with MPSX/MIP370; today its an easy problem!

Before LP-preprocessing

Rows: 5563
Columns: 6181
0-1: 74
Nonzeros: 39597

After LP-preprocessing with V10

Rows: 1427
Columns: 1877
0-1: 71
Nonzeros: 15512

- Solving Shell as LP-model

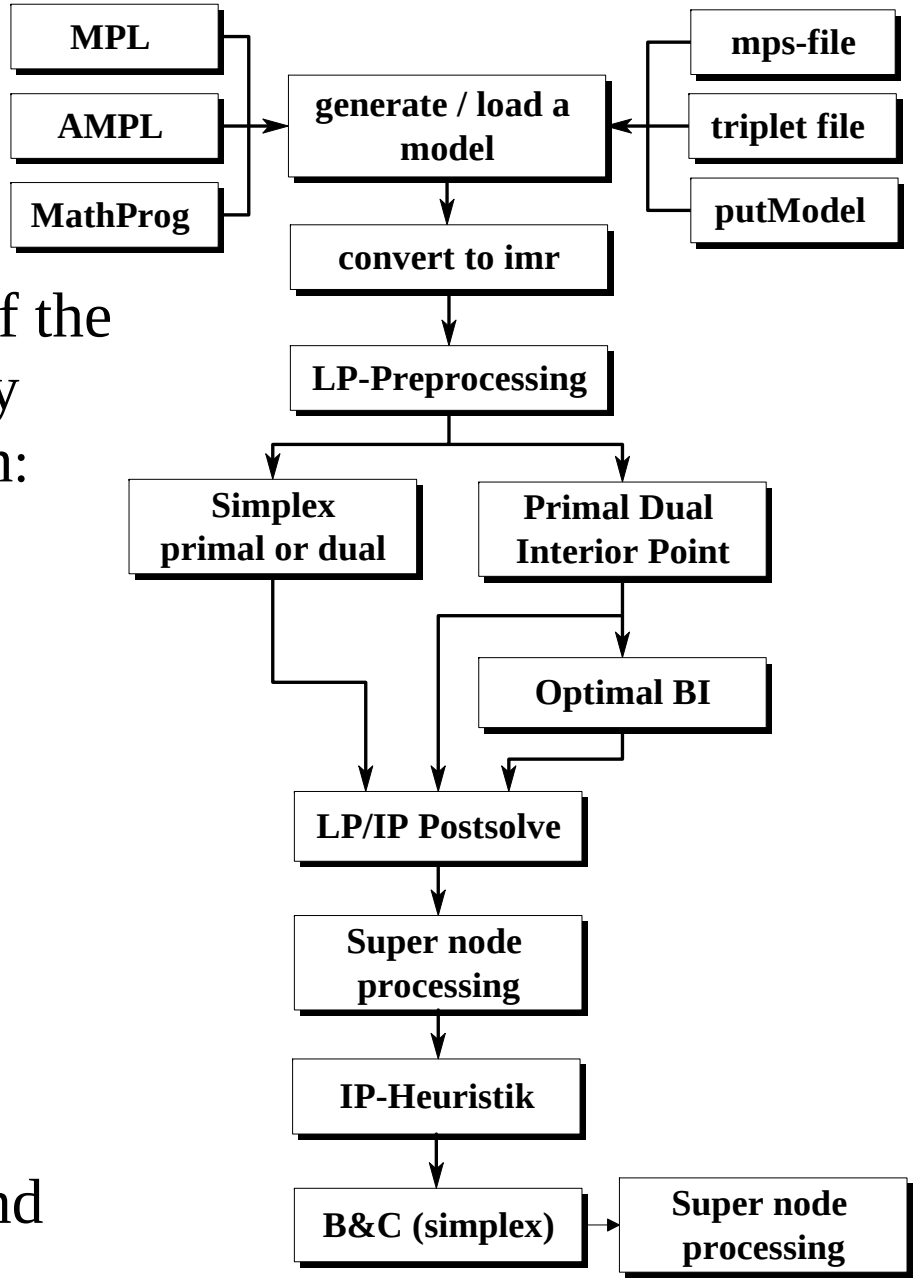
year	version	Shell LP	secs
1991	1.4	I486 (25Mhz), DOS	612.4
1996	2.8	HP-735, HP-UX	17.6
2001	5.0	PIII (500), Win98	3.9
2003	6.3	PIV (2.2) Win 2000	0.8
2007	8.0	PIV (3.4) Win XP	0.6
2009	10.x	IntelCore2Duo (2.66)	0.2

Solving Shell as IP-model

year	version	Shell IP	secs
1994	2.0	PII (500), LIFO-MIP	1794.3
1995	2.5	PII (500), non lifo	450.1
2003	6.3	PIV (2.2) refinements	39.6
2006	7.0	PIV (3.0) Dual, new cuts	9.8
2007	8.0	PIV (3.4) MIR cuts	6.7
2009	10.x	IntelCore2Duo (2.66)	1.8

Computational framework of MOPS

- State-of-the-art LP-Preprocessing
- IPM with x-over, primal or dual simplex
- Dual and IPM rank among the best engines in the world
- Supernode Processing (SNP) - specific nodes of the branch-and-bound tree are processed with many different algorithms to tighten the LP relaxation:
  - logical tests and probing of 0-1-variables implications are stored in implication table
  - identification of cliques from individual rows
  - bound and simple coefficient reduction
  - Extended coefficient reduction based on probing
  - Mixed integer rounding cuts (c-Mir)
  - cliques, implications, cover and flow (path) cuts
  - Some cut types are lifted (cliques, covers, ...)
  - euclidean reduction - rows scaled by gcd
  - Gomory mixed integer cuts
- IP-Heuristic
  - Various generalized rounding procedures with b&b
- Branch-and-Bound / Cut uses restricted SNP and Simplex engines (warm start)



LP Preprocessing Reductions on some models

Name	model	rows	columns	nonzeros
Ken-18	Original size	105127	154699	358171
	Reduced Model V9	46887	96783	233288
	Reduced Model V10	39549	89445	208276
Tough	Original size	217238	172158	997990
	Reduced Model V9	8633	99725	129119
	Reduced Model V10	5429	86156	112873
Watson 2	Original size	352013	671861	1841028
	Reduced Model V9	116812	310324	1110099
	Reduced Model V10	114535	308047	1097355
rsc_miso168	Original size	508946	724096	1431278
	Reduced Model V9	7299	142371	149502
	Reduced Model V10	6408	81528	87768
Ulm	Original size	954021	5061135	15547105
	Reduced Model V9	499160	4606274	13985012
	Reduced Model V10	212137	2187806	7074412
Wien	Original size	2039724	1798971	7654866
	Reduced Model V9	1272249	934449	5671513
	Reduced Model V10	1137194	788484	5156118
Gas	Original size	4341028	1958061	11369098
	Reduced Model V9	1781001	673015	6623012
	Reduced Model V10	1600001	804015	4709512

Algorithms for solving LPs

- There are three computational competitive algorithms
  - Primal simplex (xlptyp = 0, 1)
  - Dual simplex (xlptyp = 2) which is the default in MOPS
  - Interior point (barrier) (xlptyp = 4) with x-over
- There are problem classes where each algorithm is best
- Simplex (primal, dual)
  - computes a *basic solution* – very important for IP to exploit strong LP relaxation
  - excellent warm start capabilities important for b&B/C in solving IPs
  - lower main memory requirement, important for very large models and 32 Bit OS / software
- interior point
  - Main kernel is the symbolic factorization of a sparse positive definite symmetric system
  - Number of expensive iterations is between 30-50; after 180 iters. MOPS switches to dual
  - Iteration speed depends mainly on the size (fill-in) of the sparse cholesky factorization
  - Needs a crossover from an optimal LP-solution to an optimal basic solution
  - In most cases the fastest algorithm – even with crossover
  - No efficient warm start – rules out the use in branch-and-bound
- During last decade dual simplex has become a strong contender to IPM

LP Performance aspects

- Examples
  - Ken-18: m = 105127, n = 154699, nz = 358171
  - Ulm: m = 954021, n = 5061132, nz = 15547105

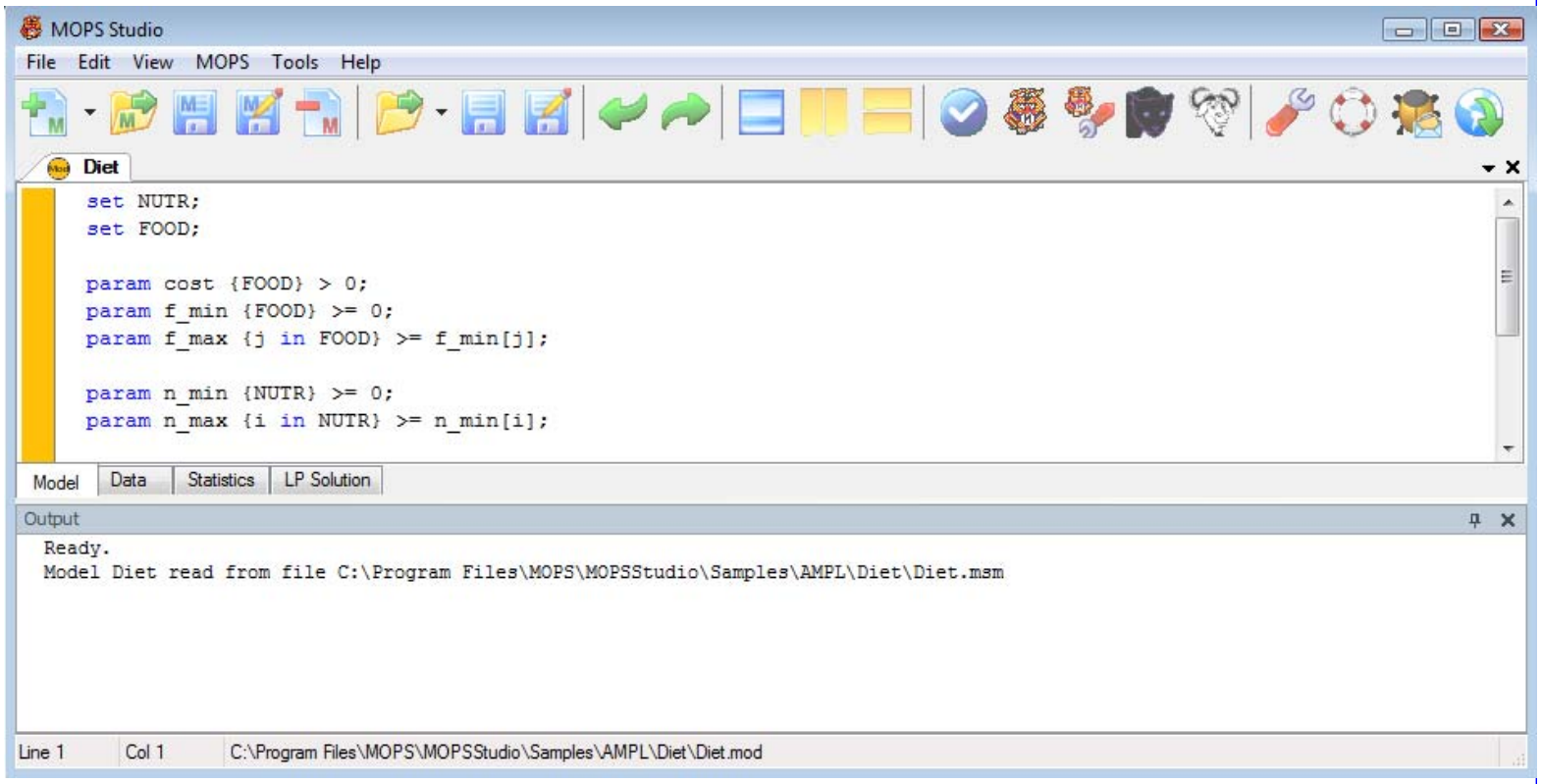
Ulm			Ken-18		
Engine	iterations	secs	Engine	iterations	secs
Primal	-	-	Primal	93506	234.39
Dual	72211	1513.86	Dual	49402	10.49
IPM	48 (6763)	861.80	IPM	20 (2601)	7.66

- Results on Intel® Core™2 Duo and Core™2Quad Processor and IPM
  - Mun-1: m = 163142, n = 1479833, 0-1 = 1330580, nz = 3031285
  - OS: Windows XP64 (dsx runs also on 32 bit XP, but IPM needs 5 GB)

DSX Xeon (3,4) (secs)	IPM Xeon (3,4) (secs)	DSX Core2Duo (2,66) (secs)	IPM Core2Duo 1 proc. (secs)	IPM Core2Duo 2 proc. (secs)	IPM Core2Quad. (2,66) (secs)
11336.4	3520.8	7940.4	2213.5	1489.1	1136.2

MOPS APIs

- MOPS load module (32 & 64 bit) : *mops.exe, mops64.exe*
  - Windows character executable
  - Data input format mps and triplet format
  - Mops profile allows customizing of optimization
- MOPS Dynamic Link Library (32 & 64 bit): *mops.dll, mops64.dll*
  - DLL functions can be called from nearly all Windows programs such as Visual Basic, C#, Delphi, Java, C++ etc.
- MOPS Studio
  - is an interactive front end to MOPS which allows the development of models in various modeling languages such as *AMPL GNU MathProg MPL.*



Main MOPS Dll functions

- Initialize () initializes MOPSand sets default values
- PutModel (intyp, inf, m, n, nz, ia, ja, a, lb, ub, c, typ) passes a model to MOPS Dll 1. as triplets, 2. rowwise 3. columnwise
- Optimize (dir,status,phase,funct) solves an LP / IP model where dir is the optimization direction and the other parameters are output
- SetParameter (s) where s is a string containing parameter assignments
- GetParameter (Parameter, value) retrieves the value of a MOPS parameter
- GetLPSolution (lpsta, lpFunct, Activity, RedCost, Status) stores status and values of LP-solution of initial LP in user given arrays
- GetIPSolution (ipsta, ipFunct, Activity, RedCost, Status) stores status and values of IP-solution in user given arrays
- FreeMemory () deallocates memory block and closes all (?) files

Sudoku Puzzles

- Given is an incomplete 9x9 matrix with integers in the range 1-9. The matrix has to be completed in such a way that in each row and each column the numbers from 1 to 9 appear exactly once. Moreover in each 3x3 block the numbers 1 to 9 must appear exactly once. We are looking just for a feasible solution.
- Example

incomplete matrix

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

→

completed matrix

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

Modeling Sudoku Puzzles using Integer Programming

- $Z = \{1..,9\}$, $z \in Z$
- I: Set of rows, $I = \{1,..,9]$, $i \in I$
- J: Set of cols, $I = \{1,..,9]$, $j \in J$
- B: Set of 3x3-blocks, $b \in B$
- I_b: Set of rows in block b
- J_b: Set of cols in block b
- M matrix of given numbers $M = (a_{ij})$ $a_{ij} > 0$ if position (i,j) is predetermined, 0 otherwise
- Objective function: min $\underline{0}$‘ y, where y is the vector of decision variables
- decision variables: y_{zij}= 1 if number z is used in row i and column j, $i \in I$, $j \in J$, $z \in Z$
- Constraints
  - *Set predetermined numbers*
    $y_{zij} = 1$ if $z = a_{ij}$ and $a_{ij} > 0$
  - *Each number z must appear exactly once per row*
    $\Sigma_{j \in J}\, y_{zij} = 1$, for all $z \in Z$, $i \in I$
  - *Each number z must appear exactly once per column*
    $\Sigma_{i \in I}\, y_{zij} = 1$, for all $z \in Z$, $j \in J$
  - *Each number z must appear exactly once per 3x3 block*
    $\Sigma_{i \in Ib}\, \Sigma_{j \in Jb}\, y_{zij} = 1$, for all $z \in Z$, $b \in B$
  - *Each number z must appear exactly once per 1x1 block*
    $\Sigma_{z \in Z}\, y_{zij} = 1$, for all $i \in I$, $j \in J$

⇒ Demo in MOPS Studio
Every Sudoku puzzle is solved already in the LP preprocessing of MOPS!