The MILP Solver and Solutions at SAS

Yan Xu & SAS Optimization Team
SAS Institute Inc.

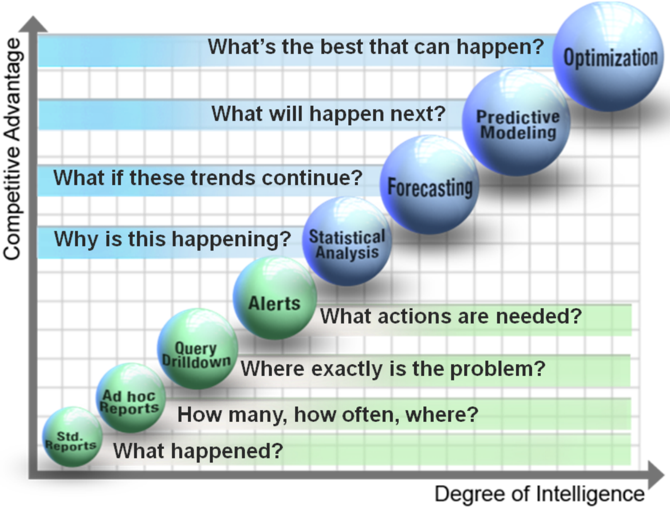
Fields Institute, University of Toronto, April 3, 2012

Outline

About SAS

- The leader in business analytics software and services, and the largest independent vendor in the business intelligence market
- $2.73 billion worldwide revenue in 2011; an unbroken track record of revenue growth every year since 1976
- Continuous reinvestment in research and development, including 24% of revenue in 2011
- Ranked No. 1 on the FORTUNE 100 Best Companies to Work For list for 2010 and 2011
- SAS Canada is No. 3 on 2011 Best Workplaces in Canada list
- About 12,600 employees, 400 offices and 600 alliances globally
- Has offices in 56 countries, and has customers in 129 countries

SAS Analytics



SAS/OR

Enterprise Miner

SAS/ETS,
Forecast Server

SAS/STAT,
SAS/QC,
SAS/IML

SAS Optimization Tools

- Algebraic Modeling Language
- Solvers and Algorithms
  - Linear Programming (primal/dual/interior/network)
  - Quadratic Programming
  - Mixed-Integer Linear Programming
  - Nonlinear Programming
  - Local Search Optimization
  - Graph Algorithms and Social Network Analysis
- SAS/OR tools are offered on 10 different platforms
  - Windows x86/x64, Linux x86/x64, Solaris x64/SPARC, HP–UX PA–RISC/Itanium, AIX Power and z/OS

SAS Optimization Solutions & Services

- Solutions
  - SAS Inventory Optimization
  - SAS Marketing Optimization
  - SAS Markdown Optimization
  - SAS Pack Optimization
  - SAS Revenue Optimization
  - SAS Service Parts Optimization
  - ...
- Services
  - SAS/OR Center of Excellence (COE)
  - SAS Professional Services
  - SAS Technical Support
  - SAS Training

Outline

- Introduction to SAS Optimization
- Improving SAS MILP Solver
  - Presolve
  - Continuous variables
  - Heuristics framework
- Solving Real World Problems
- Challenges and Directions

SAS MILP Solver

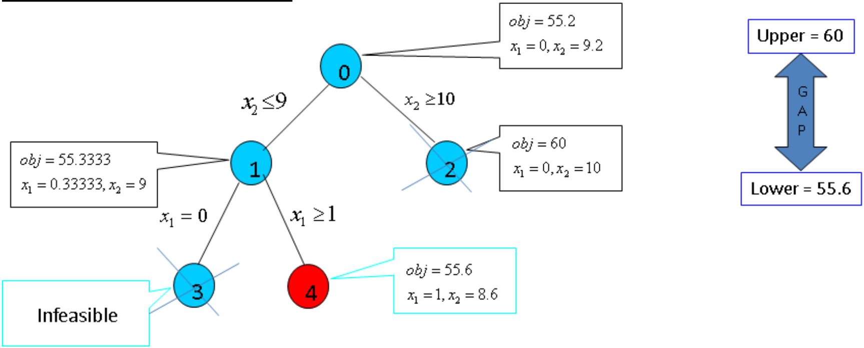
- A mixed integer linear program (MILP)

$$\begin{array}{ll} Minimize & cx \\ Subject\ to & Ax \leq b \\ & x \geq 0 \\ & Some\ x_i\ are\ integers \end{array}$$

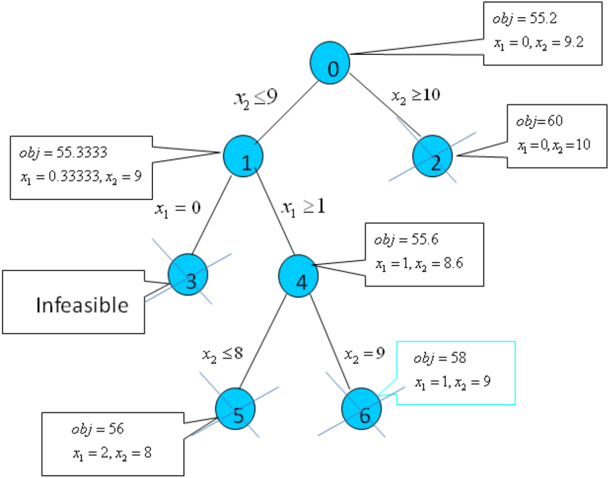
- LP relaxation-based branch and cut algorithm
- Accessible through
  - » the OPTMODEL modeling language, and
  - » the OPTMILP with data sets (MPS files) as input

A Branch and Bound Example

Minimize	$4x_1 + 6x_2$
Subject to	$3x_1 + 5x_2 \geq 46$
	$5x_1 + 10x_2 \geq 50$
	$x_1, x_2 \geq 0$, integer



A Branch and Bound Example (cont'd)



Upper = 56

Lower = NONE

SAS MILP Solver: A Bag of Techniques

- **Preprocessing and node presolve**
  - Many small ideas to reduce problem size and strengthen model
- **Node Selection**
  - Best first, Best estimate, Depth first, etc.
- **Variable selection**
  - Pseudo-cost, Strong-branching, Max infeasibility, etc.
- **Heuristics**
  - A variety of rounding and diving heuristics
  - Feasibility pump, local search, etc.
- **Cutting Planes**
  - Formulation: MIR, Knapsack, GUB, Flow Cover, Flow Path, Cliques, Implied, Zero-Half
  - Tableaux: Gomory, Mixed Lifted 0-1, Lift-and-Project

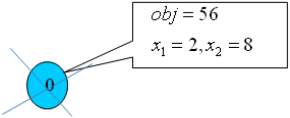
The Example Revisit

Minimize $4x_1 + 6x_2$

Subject to

$$3x_1 + 5x_2 \geq 46$$
$$5x_1 + 10x_2 \geq 50$$
$$x_1, x_2 \geq 0, \text{ integer}$$

Upper = 56



Lower = NONE

1. Rounding heuristic: $x_1 = 0,\ x2 = 10,\ obj = 60$
2. Local search heuristic: $x_1 = 1,\ x2 = 9,\ obj = 58$
3. MIR cut: $x_1 + x_2 \geq 10$
4. Resolve root LP, and LP solution $(x_1 = 2,\ x_2 = 8,\ obj = 56)$ happens to be integer feasible

SAS MILP Solver: Recent Developments

- **Presolve enhancement**
  - Reduction based on logic implications
- **Techniques based on continuous variables**
  - Implied integer variables
  - Mixed lifted 0-1 cuts
- **Heuristics**
  - Heuristics framework
  - Handle time consuming heuristics

Reduction Based on Logic Implications

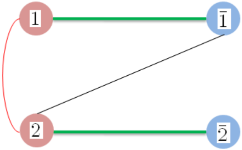
- Business applications
  - Airline crew scheduling: $\sum_{j=1}^{n} a_{ij} x_j = 1$
  - Sport scheduling: $\sum_{j=1}^{n} y_j \leq 1$

- Four possible logic relationship between two binary variables

$$
\boxed{
\begin{array}{lcrcr}
x_1 = 1 \Rightarrow x_2 = 0 & \Longleftrightarrow & x_1 & + & x_2 \leq 1 \\
x_1 = 0 \Rightarrow x_2 = 0 & \Longleftrightarrow & (1 - x_1) & + & x_2 \leq 1 \\
x_1 = 1 \Rightarrow x_2 = 1 & \Longleftrightarrow & x_1 & + & (1 - x_2) \leq 1 \\
x_1 = 0 \Rightarrow x_2 = 1 & \Longleftrightarrow & (1 - x_1) & + & (1 - x_2) \leq 1
\end{array}
}
$$

Handle Logic implications

- How to get logic implications?
  - Check constraints
  - Probing
- How to store logic implications? (conflict graph)
  - vertex i represents x_i; vertex $\bar{i}$ represents $1 - x_i$
  - An edge represent an logic relationship (edge inequality)



$$\boxed{\begin{array}{rcl} x_1 & + & x_2 \leq 1 \\ (1 - x_1) & + & x_2 \leq 1 \end{array}}$$

How to Use Logic Implication?

- Fix variables

  $x_1 = 0 \Rightarrow x_2 = 0$ and $x_1 = 1 \Rightarrow x_2 = 0$, fix $x_2 = 0$

- Substitute variables

  $x_1 = 0 \Rightarrow x_2 = 0$ and $x_1 = 1 \Rightarrow x_2 = 1$, derive $x_2 = x_1$

- Strengthen (Lift) set packing constraints
  - Given $x_1 + x_2 + x_3 \leq 1$, if edges (1, 4), (2, 4) and (3,4) are in the conflict graph, then we can strengthen it to $x_1 + x_2 + x_3 + x_4 \leq 1$

How to Use Logic Implications? (cont'd)

- Fix variables with set partitioning constraints
  - Given $x_1 + x_2 + x_3 = 1$, if edges (1, 4), (2, 4) and (3,4) are in the conflict graph, then we know $x_4 = 0$
- Remove dominated constraints
  - Derive initial cliques and store them in the clique table
    - A clique is $\sum_{j \in S^+} x_j - \sum_{j \in S^-} x_j \leq 1 - |S^-|$
    - How to derive? Weighted vertex packing problem.
  - Lift cliques by using implications in the conflict graph
  - Check if constraints are dominated by cliques
    $x_1 + x_2 \leq 1$ is dominated by $x_1 + x_2 + x_3 + x_4 \leq 1$

Effectiveness of Logic Implications

- Tested on ACC instances (Nemhauser & Trick, Henz, etc.)
- Basketball games scheduling problem.
- Have a lot of constraints of this type $\sum_{i \in S} x_i \leq 1$

Effectiveness of Logic implications (cont'd)

- Problem size after presolve

Instance	Without Logic Reduction		With Logic Reduction		
	Variables	Constraints	Variables	Constraints	Coef Lifted
acc1	1620	2286	1620	990	162
acc2	1620	2520	1620	1224	162
acc3	1620	3249	1620	1953	162
acc4	1620	3285	1620	1989	162
acc5	1308	3052	1308	1972	372
acc6	1308	3047	1308	1983	438

Effectiveness of Logic implications (cont'd)

- Nodes and solution time comparison (time in seconds)

Instance	Without Logic Reduction		With Logic Reduction	
	Nodes	CPU Time	Nodes	CPU Time
acc1	74	13.5	1	1.5
acc2	1	7.4	1	2.0
acc3	94	21.8	31	24.2
acc4	12748	1634.6	1729	446.7
acc5	1711	232.3	1750	151.6
acc6	1347	469.6	396	62.2

Implied Integer Variables (IIV)

- Some variables are declared as continuous, but can be treated as integers
- Two types (y is declared as a continuous variable)
  - Implied by feasibility conditions (primal)

$$\boxed{y + x_1 + x_2 + x_3 = 1000,\ x \text{ are integers}}$$

  - Implied by optimality conditions (dual)

$$\begin{array}{lll}
\text{min} & 5y + 3x_2 + 4x_3 & \\
\text{s.t.} & y \geq x_2 + 2x_4 + 2 & (1) \\
 & y \geq 2x_2 + x_3 + x_4 & (2) \\
 & x \geq 0 \text{ and integers},\ y \geq 0 &
\end{array}$$

Do Implied Integers happen often?

- Percentage of instances that have implied integers

	percentage
Implied Integer	14.2%
– Feasi-implied only	7.5%
– Opti-implied only	5.0%
– Both	1.7%

- Several MIPLIB 3 instances

Instance	Num Cont Vars	Feas-Implied	Opti-Implied
blend	89	88	0
flugpl	7	1	6
rentacar	9502	1241	2

Effectiveness of Using Implied Integers

- Tested the 152 instances with implied integers
- Number of instances solved in 2 hours

	Not Use IIV	Use IIV
Solved	99	105

- Assume a 2 hours solution time for unsolved instances
  - Using implied integers is 14% faster

Mixed Lifted 0-1 Inequalities (MLI)

- Reference:
  - Narisetty, Richard and Nemhauser, Lifted tableaux inequalities for 0-1 Mixed Integer Programs: A Computational Study, 2010
  - Marchand and Wolsey, The 0-1 Knapsack problem with a single continuous variable, 1999
- Single row relaxation of 0-1 mixed integer knapsack problem

$$S = \left\{ (x,y) \in \{0,1\}^m \times [0,1]^n \;\middle|\; \sum_{j \in M} a_j x_j + \sum_{j \in N} b_j y_j \leq d \right\},$$

1. $M = \{1,\ldots,m\}$, $N = \{1,\ldots,n\}$
2. $a_j \in \mathbb{Z}, 0 < a_j \leq d \quad \forall j \in M$,
3. $b_j \in \mathbb{Z}, 0 < b_j \leq d \quad \forall j \in N$,
4. $d \in \mathbb{Z}$

Four Families of MLI

- Lifted 0-1 Covers (LC)
  - Starting with a seed inequality (0-1 cover); lifted rest variables
- Lifted 0-1 Packings (LP)
  - Starting with a seed inequality (0-1 packing); lifted rest variables
- Lifted 0-1 Lifted Continuous Covers (LCC)
  - Starting with a seed inequality (continuous cover); lifted rest variables
- Lifted Continuous Packings (LCP)
  - Starting with a seed inequality (continuous packing); lifted rest variables

Lifted 0-1 Lifted Continuous Covers (LCC)

- LCC is given by

$$\sum_{j\in C} b_j y_j + \sum_{j\in M_0} \Phi(a_j)x_j + \sum_{j\in M_1} \min\{a_j,\mu\}x_j$$
$$\leq \sum_{j\in C} b_j - \mu + \sum_{j\in M_1} \min\{a_j,\mu\} \qquad (LCC)$$

where $M_1 = \{1,\ldots,l\}$ with $a_1 \geq \ldots a_c \geq \mu \geq a_{c+1} \geq \ldots \geq a_l$, $A_j = \sum_{i=1}^{j} a_i$, $A_0 = 0$ and

$$\Phi(a) = \begin{cases} k\mu & \text{if} \quad A_k \leq a \leq A_{k+1} - \mu \quad \text{for} \quad k = 0,\ldots,c \\ k\mu + a - A_k & \text{if} \quad A_k - \mu \leq a \leq A_k \quad \text{for} \quad k = 1,\ldots,c-1 \\ c\mu + a - A_c & \text{if} \quad A_c - \mu \leq a \end{cases}$$

- Derivation
  - Start with seed inequality.
  - Lift 0-1 variables in M_1.
  - Lift 0-1 variables in M_0.
  - Lift continuous variables in $N \setminus C$.
  - To yield a strong nontrivial inequality, at least one variable in M_1 should have a large coefficient.

Computational Results

- Test on 89 instances that have both binary and continuous variables
- Number of instances solved in 2 hours

	Not Use MLI	Use MLI
Solved	53	56

- Solution time comparison (use vs. not use)

	Geometic Mean
SolTime $\leq$ 100 s	4% faster
100 s $<$ SolTime $\leq$ 7200 s	36% faster

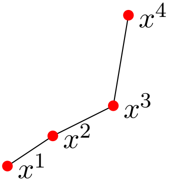
Types of Heuristics

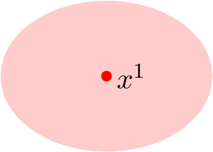
- Integer Solution Requirements:

$$
\begin{aligned}
cx &< c\bar{x} && (1)\\
Ax &\leq b && (2)\\
x_i \in \mathbb{Z} \quad &\forall i \in I && (3)
\end{aligned}
$$

- Starting heuristics
  - input satisfies: (1) and (2) but not (3)
- Improvement heuristics
  - input satisfies: (2) and (3) but not (1)
- Repair heuristics
  - input satisfies: (3) and (1) but not (2)
- Special heuristics
  - No requirements

Underlying Principles of Heuristics





Line Search

- Rounding
- Diving
- Feasibility Pump
- Pivoting
- ...

Neighborhood Search

- MIPing
- Genetic Algorithms
- ...

The New Heuristics Framework

- All heuristics are categorized by
  - Type: starting, improvement, repair and special
  - Speed: very fast, fast, moderate, slow
- Two solution pools store solutions
  - Improvement pool
  - Repair pool
- The framework manages all the heuristics
- The framework usually manipulates categories of heuristics
  - Independent from which heuristics are actually used
- The framework is controlled by the heuristics strategy

Handling Time-consuming Heuristics

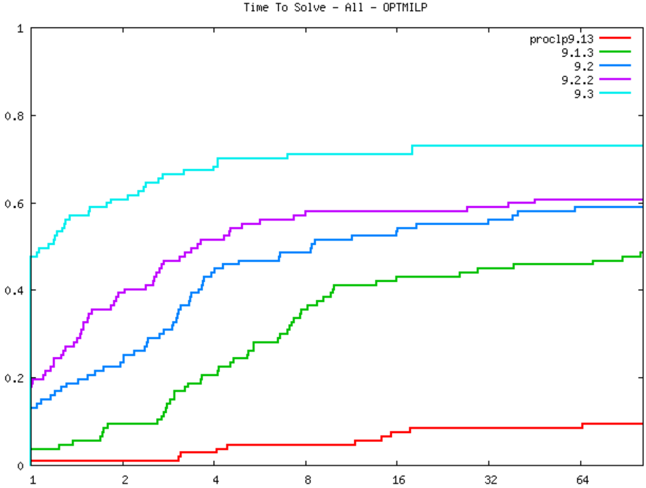
The Problem

- Some of the most powerful heuristics are also the most time-consuming
- Many instances can be solved very quickly without using any heuristics

One Solution

- Slow heuristics are only called if $\rho = \frac{t_h}{t}, \rho \leq \bar{\rho}$ where t_h is the time spent in heuristics and t is the total time spent in the solver.
- The parameter $\bar{\rho}$ can be interpreted as: *Don't spent more than X percent of the time on heuristics*
- Since using runtime to compute ρ would make the solver non-deterministic, we estimate ρ using SIE (simplex iteration equivalent) units

Performance of Different Versions of SAS MILP Solver

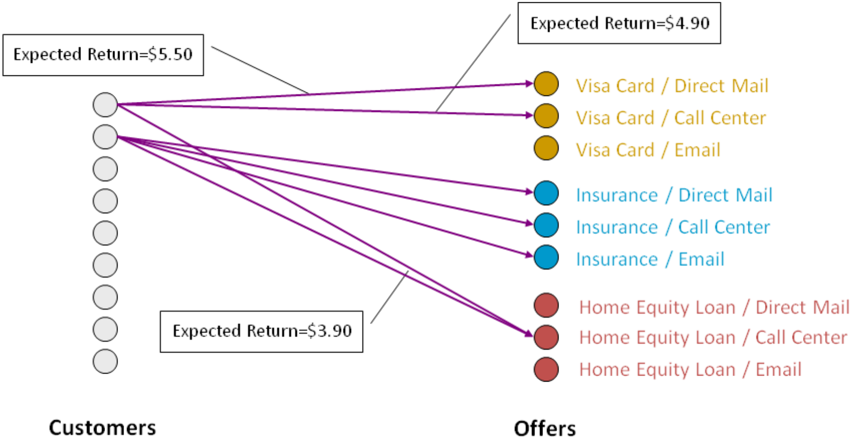


Outline

- Introduction to SAS Optimization
- Improving SAS Optimization Tools
- Solving Real World Problems
  - Solution: Marketing Optimization
  - Consulting service: ATM Replenishment Problem
- Challenges and Directions

Marketing Optimization (MO)

- MO is an offer assignment problem



MO: Business Applications

- Finance services
  - Cross-sell and up-sell in retail banking: savings accounts, home equity loans, credit cards, lines of credit, etc.
  - Insurance policy offers
  - Deciding credit line increases
  - Deciding what APR to offer on balance transfer offers
- Other Industries
  - Hotels & Casinos: loyalty offers
  - Retail: personalized coupons
  - Telecom: cell phone or calling plan offers

MO: MILP Formulation

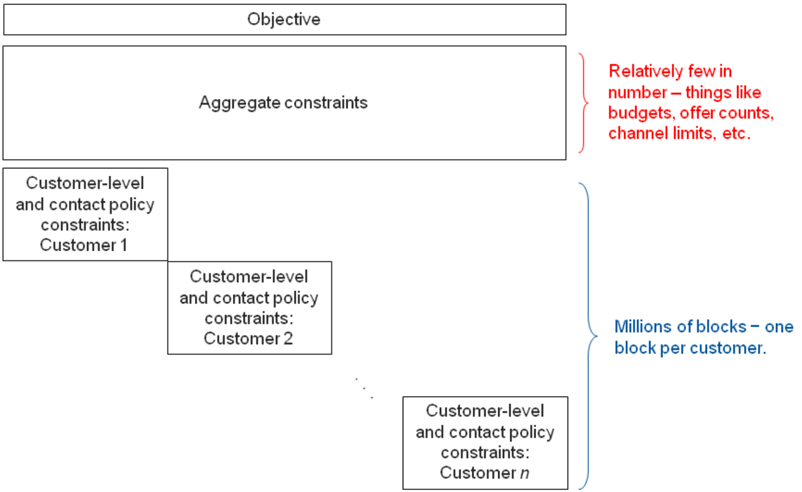
$$
\begin{array}{llll}
\text{maximize} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} & & \\
\text{subject to} & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ijk}x_{ij} \leq A_k, & & k = 1,\ldots,\#\text{ of constraints} \\
& \displaystyle\sum_{j=1}^{m} b_{ijp}x_{ij} \leq B_{ip}, & & i = 1,\ldots,n; \\
& & & p = 1,\ldots,\#\text{ of contact policies} \\
& x_{ij} \in \{0,1\}, & & i = 1,\ldots,n; \quad j = 1,\ldots,m
\end{array}
$$

- Variable x_{ij} is binary, and is 1 if customer i receives offer j

MO: What Make it So Hard?

- It is a simple MILP, but typical problem scale is
  - Millions of customers
  - Tens to hundreds of offers
- Many millions of binary decision variables and millions of constraints
- Impossible for general purpose optimization solvers
- Contact policy constraints and eligibility are hard constraints
- Specialized algorithms are needed

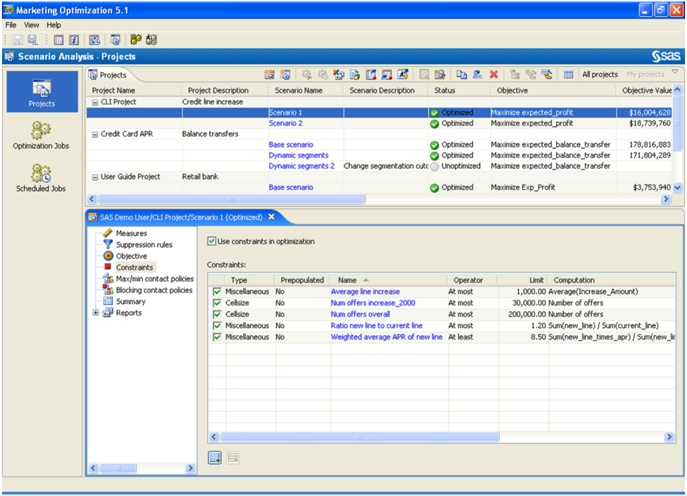
MO: Problem Structure



MO: Solution Methodology

- Dualize aggregate constraints
- Decompose into blocks
- Solve block problems and get new columns
- Solve master problem and update Lagrange multipliers
- Iterate until the objective value doesn't change much
- Techniques used include
  - General purpose LP and MILP solvers
  - Special decomposition algorithm
  - Special subgradient algorithm
  - Special heuristics to find feasible integer solutions
  - Parallel computing
- Results: Can find good solutions in several minutes to hours

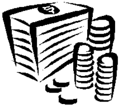
MO: Make It Easy for Users



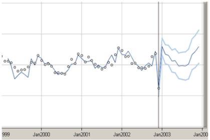
ATM Replenishment Problem

- Given transactional data (withdrawals, deposits, replenishments) for past 3 months
- Forecasting problem: estimate hourly demand for each ATM for the next month
- Optimization problem: determine which hours to replenish each ATM over the next month to avoid cashouts
- "replenish" means fill to capacity
- "cashout" means ATM inventory $<$ next 4 hours of demand

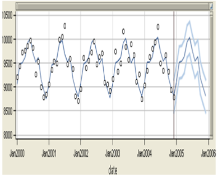
ATM Cash Flow Management Process



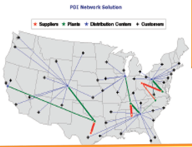
Data Extraction from different transactional systems



Demand Forecasting by ATM / Branch



Determine which ATM, when and how much to replenish



Manage replenish plans and identify changes

Hartgeld (in EUR)

Stückelung	Vorschlag	Bestellung
2,00	9.000,00	
1,00	4.750,00	
0,50	1.800,00	
0,20	1.520,00	
0,10	560,00	
0,05	200,00	
0,02	80,00	
0,01	120,00	
davon Sonderbestellung für Kunden		

Reporting

Optimization Problem

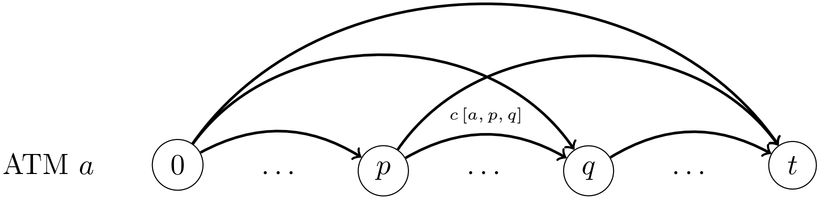
- Four possible objectives to minimize:
  - Cashout hours
  - Cashout events (consecutive cashout hours at same ATM)
  - Lost demand (in dollars)
  - Number of replenishments
- Budget limits total number of replenishments
- Limit on number of simultaneous replenishments varies throughout the day
- Eligible replenishment hours depend on ATM:
  - all day: 4am-noon, 1pm-11pm
  - overnight: 9pm-7am
- Run replenishment scheduling every two weeks for one-month rolling horizon

Initial MILP Formulation

- $\text{Replenish}\,[a,p] = \begin{cases} 1 & \text{if ATM } a \text{ is replenished in period } p \\ 0 & \text{otherwise} \end{cases}$
- Linear constraints among these variables
- Problem is hard to solve
  - » Symmetry
  - » Problem size

Network-Based MILP Formulation

- For $p < q$,
  $$\text{Replenish}[a,p,q] = \begin{cases} 1 & \text{if ATM } a \text{ is replenished in periods } p \text{ and } q \text{ but not between} \\ 0 & \text{otherwise} \end{cases}$$
- Arc cost $c[a,p,q]$ could be number of cashout hours between periods p and q, or lost demand, etc.



Network-Based MILP Formulation (cont'd)

- Integer network flow problem with few side constraints
- Complicated side constraints in initial formulation correspond to removal of arcs:
  - minimum number of hours between replenishments
  - maximum number of consecutive cashout hours
- Typically solves at root node of branch-and-bound tree
  - LP relaxation yields nearly integer solution (fractional in a small percentage of components)

Optimization Results and Business Impact

Objective	Baseline	Optimized
Cashout Hours	?	15
Cashout Events	391	15
Lost Demand	?	$0
Number of Replenishments	11,424	9,828

- 2-hr runtimes well within overnight requirements
- *Significantly increased customer satisfaction* (main goal)
- $1.4 million projected annual savings
- Similar results using historical demands

Outline

- Introduction to SAS Optimization
- Improving SAS Optimization Tools
- Solving Real World Problems
- Challenges and Directions

Challenges and Directions

- MILP is still difficult
  - can not solve half of our internal benchmark instances
  - What can we do?
    - Develop new or revisit old theories and methods
    - Cross disciplines: Artificial Intelligence, Constraint Programming, etc.
    - Better implementation
- Problem size explosion
  - Customers have a lot of data
  - Large companies start to optimize their business
  - What can we do?
    - Decomposition
    - High performance computing

Challenges and Directions (cont'd)

- Enormous Computing Power Everywhere
  - Multi-core PCs or servers, clusters, blades ...
  - GPUs: NVIDIA©Tesla: 448 cores (1.15GHz), 6 GB RAM
  - Intel©MIC: > 60 cores (1.2GHz), 8GB RAM
  - Clouds: Amazon EC2, IBM Cloudburst, Microsoft Azure ...
  - What can we do?
    - Parallel computing (thread or grid)
    - GPU computing
    - Cloud computing
    - Software as a Service (SaaS)
- High Performance Optimization Products
  - Decomposition algorithms for LPs and MILPs
  - Algorithms for big LPs for statistics and solutions
  - Local search optimization
  - ...

SAS OnDemand for Academics

- It provides an online delivery model for teaching and learning data management and analytics.
- It is available at no cost for professors and students registered in classes with those professors.
- Support for many disciplines, including
  - SAS programming
  - statistics
  - data mining
  - Forecasting
  - Math computation
  - Operations research
  - ...

Thank you for your attention.

yan.xu@sas.com