Implementation and Use Cases of a Commercial Decomposition Solver

CO@Work Summer School 2020

Philipp M. Christophel and Matt Galati

Scientific Computing, R&D, SAS Institute Inc.

Outline

- A Quick Overview of DECOMP
- Implementation Challenges
- Use Cases
  - The Optimal Wedding Seat Assignment
  - The Kidney Exchange Problem
  - ATM Cash Management
- Conclusions

Optimization with SAS

- SAS Optimization includes
  - LP, QP, NLP, MILP, constraint programming, black-box, network algorithms, and the algebraic modeling language OPTMODEL
- Callable on SAS Viya from SAS, Python, Lua, Java, R, and REST API
- Also available: sasoptpy, a modeling package for Python

What is DECOMP?

- SAS DECOMP is the first/only commercial, generalized and automated branch-and-price solver
  - Automated Dantzig-Wolfe decomposition
  - User defined or automatic detection of blocks

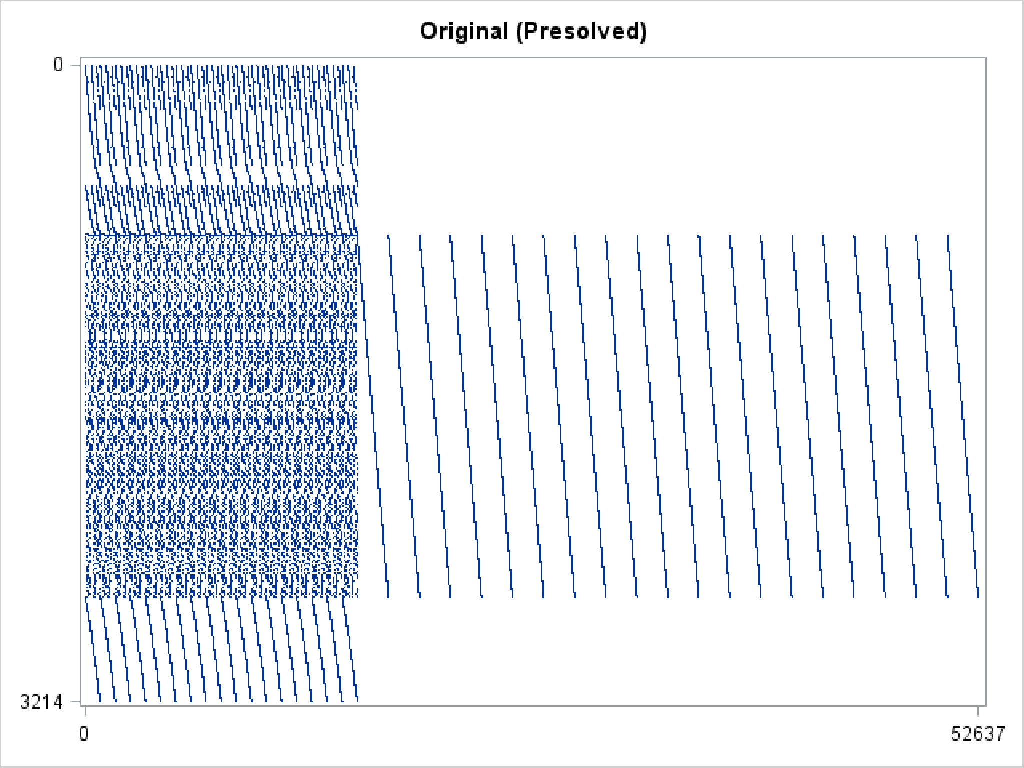
$$\begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^K \\ A^1 & A^2 & \cdots & A^K \end{pmatrix}$$

Singly-bordered block-diagonal form (SBDF)

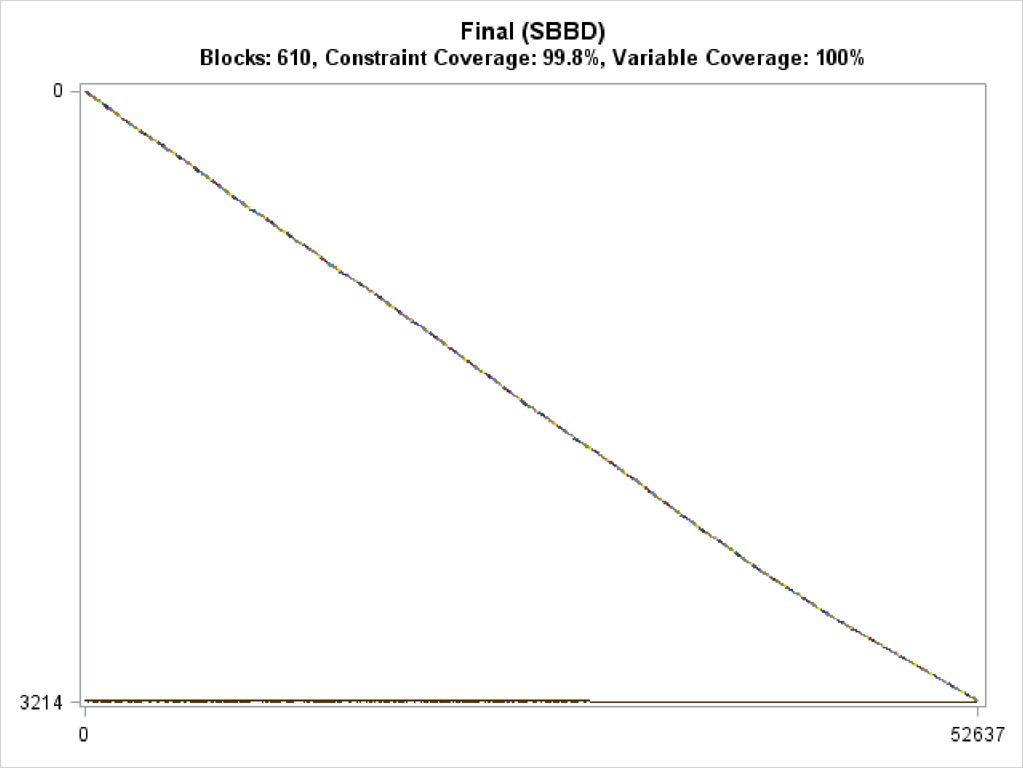
$$\begin{pmatrix} D^1 & & & & F^1 \\ & D^2 & & & F^2 \\ & & \ddots & & \vdots \\ & & & D^K & F^K \\ A^1 & A^2 & \cdots & A^K & G \end{pmatrix}$$

Doubly-bordered block-diagonal form (DBDF)

A Real-world Example in Pictures



A Real-world Example in Pictures



A Real-world Example

Branch-and-cut

```
NOTE: The presolved problem has 52638 variables, 3215 constraints, and 131250 constraint coefficients.
          Node    Active    Sols     BestInteger       BestBound       Gap     Time
             0         1       3    6151.1464478    8590.4503506    28.40%        0
-- snip --
             0         1       4    6151.1466160    7045.9724210    12.70%      782
-- snip --
        175772    173251      11    6871.8766247    7044.1201668     2.45%     3599
NOTE: Real time limit reached.
```

Branch-and-price: DECOMP

```
NOTE: The problem has a decomposable structure with 610 blocks.
      The largest block covers 0.2488% of the constraints in the problem.
NOTE: The decomposition subproblems cover 52638 (100%) variables and 3207 (99.75%) constraints.
      Iter        Best       Master         Best        LP        IP   CPU  Real
                 Bound    Objective      Integer       Gap       Gap  Time  Time
         .   7963.9759    6467.2136    6467.2136    18.79%    18.79%    13     8
         2   7267.7239    6467.2136    6467.2136    11.01%    11.01%    26    13
         3   7147.9955    6878.4375    6467.2136     3.77%     9.52%    51    21
         5   6986.1299    6960.5400    6960.5400     0.37%     0.37%    74    30
         6   6986.1299    6965.5335    6965.5335     0.29%     0.29%    84    33
         7   6972.3310    6972.3309    6972.3309     0.00%     0.00%    87    34
         Node  Active   Sols          Best          Best       Gap     CPU   Real
                                   Integer         Bound              Time   Time
            0       0      9     6972.3309     6972.3310     0.00%      87     34
NOTE: The Decomposition algorithm time is 34.61 seconds.
NOTE: Optimal within relative gap.
```

Branch-and-cut vs. DECOMP

- Some non-obvious differences
  - Progress report
  - Parallelization
  - Symmetry
- Direct comparison of branch-and-cut and automatic DECOMP
  - 85% of the time branch-and-cut wins
  - If DECOMP works well it crushes branch-and-cut
- Automatically choosing which solver to use is non-trivial
- By default, the SAS MILP solver uses DECOMP under the hood

Implementation Challenges: LP Reliability

- Both branch-and-cut and DECOMP make heavy use of (simplex) LP solvers
- Branch-and-cut
  - Mostly relies on warm-started dual simplex solves and the occasional primal simplex solve
  - The LPs solved mostly differ in the bounds of the problem, but the matrix is mostly the original problems matrix (plus cuts)
- DECOMP
  - Uses the primal simplex a lot more (when warm-starting after adding columns)
  - The LPs solved contain results of previous solves as coefficients in the matrix which can lead to all kinds of numerical trouble
- DECOMP stress tests the simplex solver implementations!

Implementation Challenges: Block Detection

- User-defined blocks

```
for{t in TABLES} do;
   TableSizeCon[t].block = t;
   for{<g,h> in GUEST_PAIRS}
      TableMeasureCon[t,g,h].block = t;
end;
solve with milp / decomp=(method=user);
```

- Special structures
  - Network structure
  - Connected components
  - Set partitioning structure

Automatic Detection of Blocks

- Automatic detection methods
  - APC
    - Fixed number of blocks
    - Based on graph partitioning of a bipartite graph
    - Cevdet Aykanat, Ali Pinar, and Ümit V. Çatalyürek: *Permuting Sparse Rectangular Matrices into Block-Diagonal Form,* SIAM Journal on Scientific Computing, 2004, Vol. 25, No. 6.
  - KEE
    - Flexible number of blocks
    - Approach based on modularity and community detection
    - Minimize border area while maximizing a quality function for the diagonal
    - Taghi Khaniyev, Samir Elhedhli, Fatih Safa Erenay: *Structure Detection in Mixed-Integer Programs*, INFORMS Journal on Computing, 2018, Vol. 30, No. 3.

Use Case: The Optimal Wedding Seat Assignment

- See the blog post here:

  https://blogs.sas.com/content/operations/2014/11/10/do-you-have-an-uncle-louie-optimal-wedding-seat-assignments/

- Ryan-Foster Branching:

  https://go.documentation.sas.com/?docsetId=casmopt&docsetTarget=casmopt_decomp_details06.htm&docsetVersion=8.5&locale=en

- METHOD=SET or manually setting the blocks with METHOD=USER

```
for{t in TABLES} do;
   TableSizeCon[t].block = t;
   for{<g,h> in GUEST_PAIRS}
      TableMeasureCon[t,g,h].block = t;
end;
solve with milp / decomp=(method=user);
```

Use Case: The Kidney Exchange Problem

- See the blog post here

  https://blogs.sas.com/content/operations/2015/02/06/the-kidney-exchange-problem/

- See the documentation example here

  https://go.documentation.sas.com/?docsetId=casmopt&docsetTarget=casmopt_decomp_examples22.htm&docsetVersion=8.5&locale=en

- Sometimes the PRESOLVER= option needs to be adjusted to maintain problem structure
- Static column generation can sometimes be the better choice

Use Case: ATM Cash Management

- See the blog post here

  https://blogs.sas.com/content/forecasting/2015/01/24/atm-replenishment-forecasting-optimization/
- See the documentation example

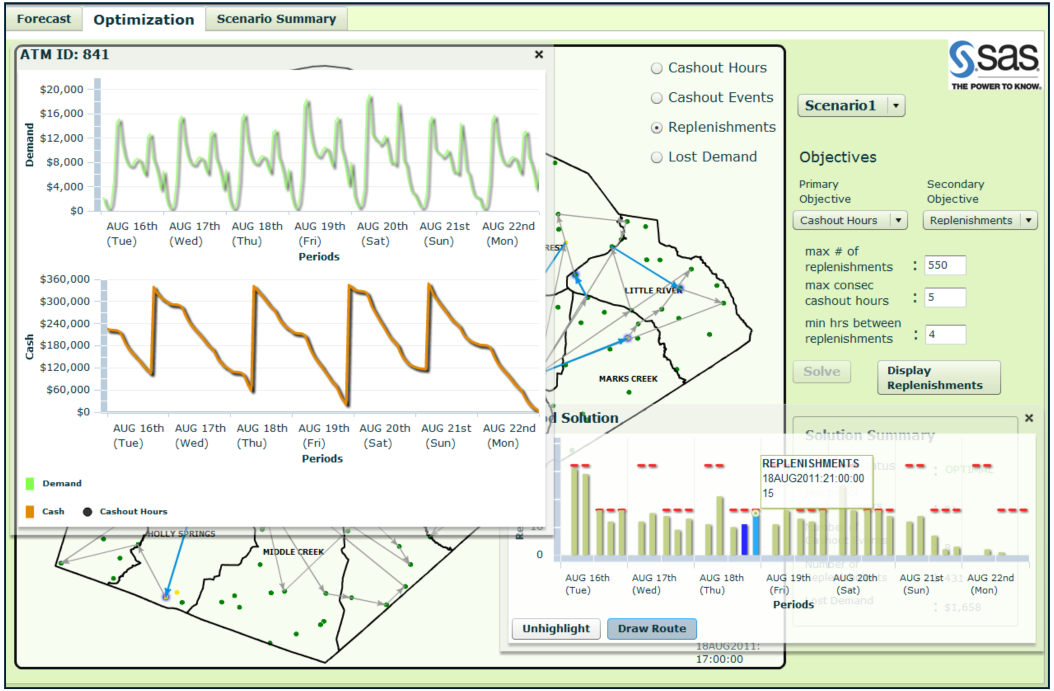
  https://go.documentation.sas.com/?docsetId=casmopt&docsetTarget=casmopt_decomp_examples13.htm&docsetVersion=8.5&locale=en
- And here

  https://go.documentation.sas.com/?docsetId=casmopt&docsetTarget=casmopt_decomp_examples21.htm&docsetVersion=8.5&locale=en

Use Case: ATM Cash Management

- Transactional data for the past 3 months
- Forecasting problem: estimate hourly demand for each ATM for the next month
- Optimization problem: determine at which hours to replenish each ATM over the next month to minimize/avoid cashouts

Use Case: ATM Cash Management



Use Case: ATM Cash Management

Objective	Baseline	Optimized
Cashout Events	391	15
Number of Replenishments	11,424	9,828

- 2-hour runtimes, well within overnight requirements
- Significantly increased customer satisfaction
- Projected annual savings of USD 1.4 million

Conclusions

- Decomposition is not just an academic topic!
- Real-world problems today are solved using advanced optimization techniques
- More can be done when the technology is easier to use and quicker to utilize
- Making advanced optimization techniques accessible to a broad range of customers opens up opportunities

Thank you for your attention!

philipp.christophel@sas.com

sas.com