Compiling and solving 100,000 equations on a PC in (3) minutes

Benjamin A. Allan
Arthur W. Westerberg
Dept. of Chemical Engineering
and the Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213

Paper presented at INFORMS Meeting
Montreal, Quebec, Canada
April 28, 1998

Outline

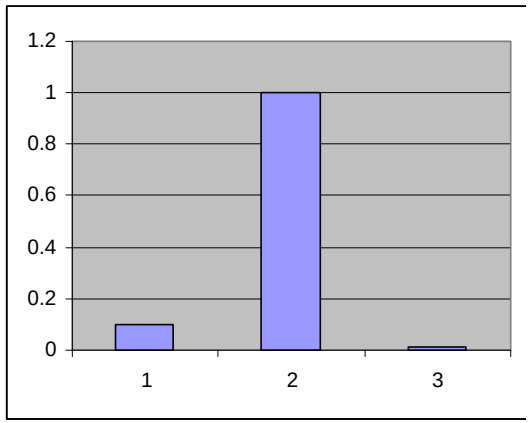
- Speeding up model solving
- Procedural vs. non-procedural modeling
- Directed acyclic graphs (DAGs)
  + Hierarchical modeling
  + Object sharing
- An DAG based algorithm to find unique equation types
- Impact on
  + compiling
  + residual and jacobian evaluation
- A DAG based algorithm to reduce L\U factorization times

Speeding up model solving

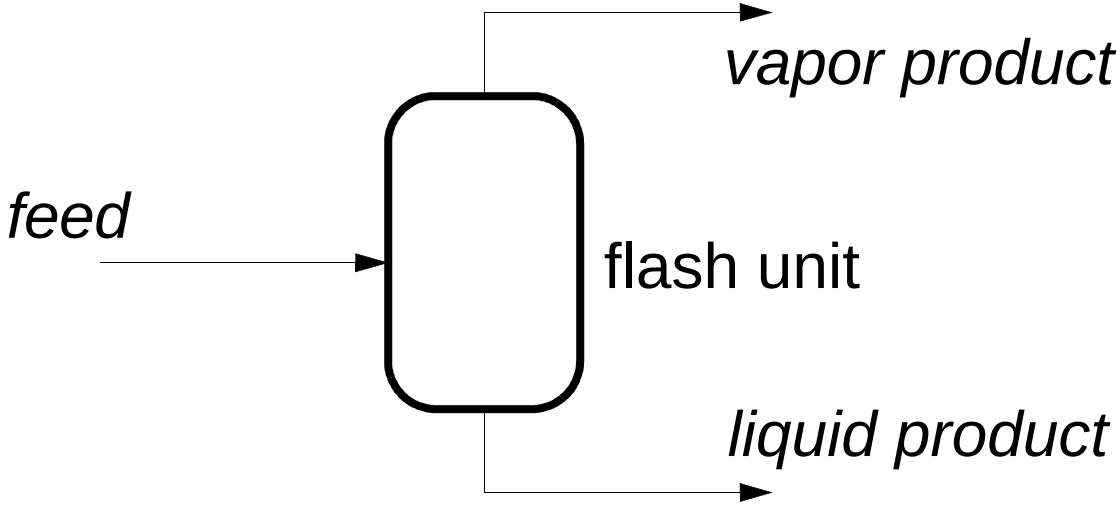
Need much faster

1.compile
2.residual and jacobian evaluation
3.L\U factorization

Failure on any one leaves a bottleneck



Equation-based vs. procedural modeling



Procedural

Given *feed* and *T, P*, write subroutine to compute *vapor* and *liquid products*

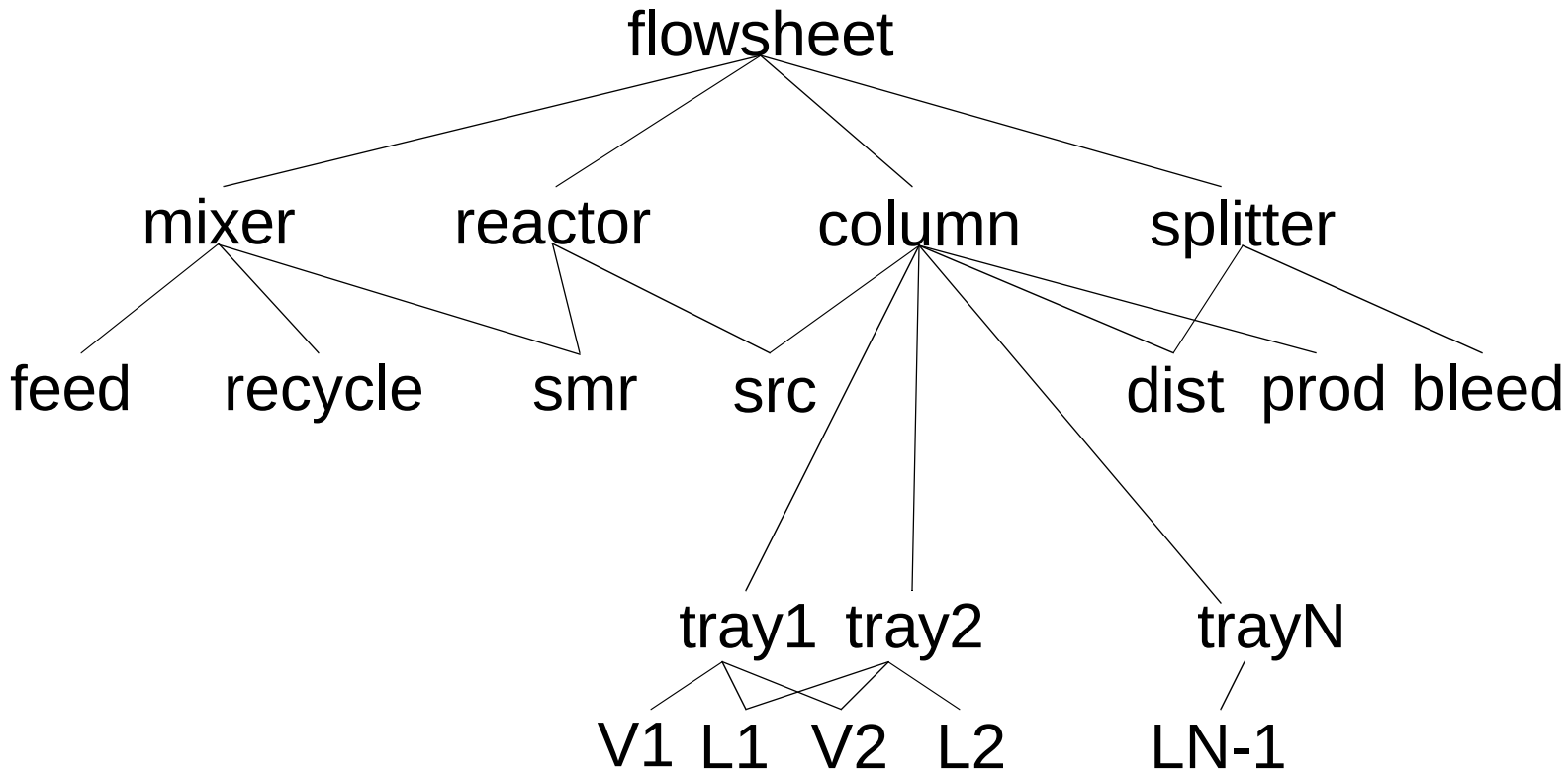
Equation-based

Fix a sufficient set of variables so model can compute the remaining *n* variables using the *n* model equations

Directed Acyclic Graphs (DAGs)

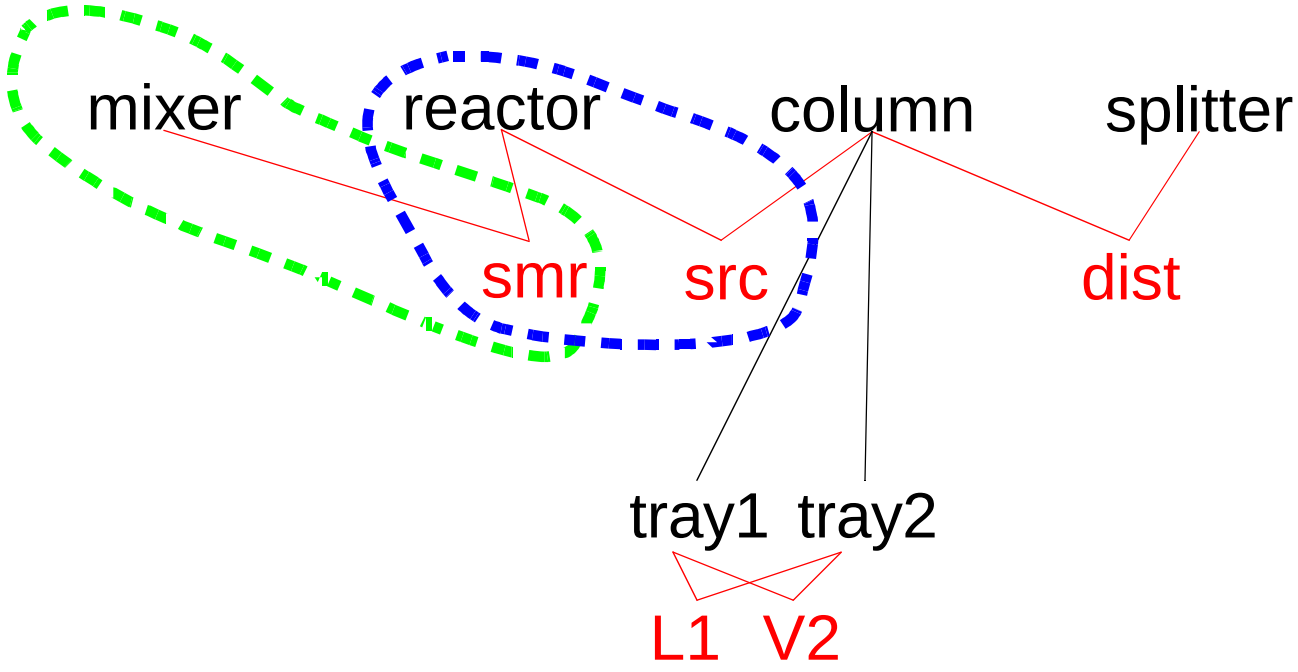
Hierarchical part/whole modeling

- to control complexity

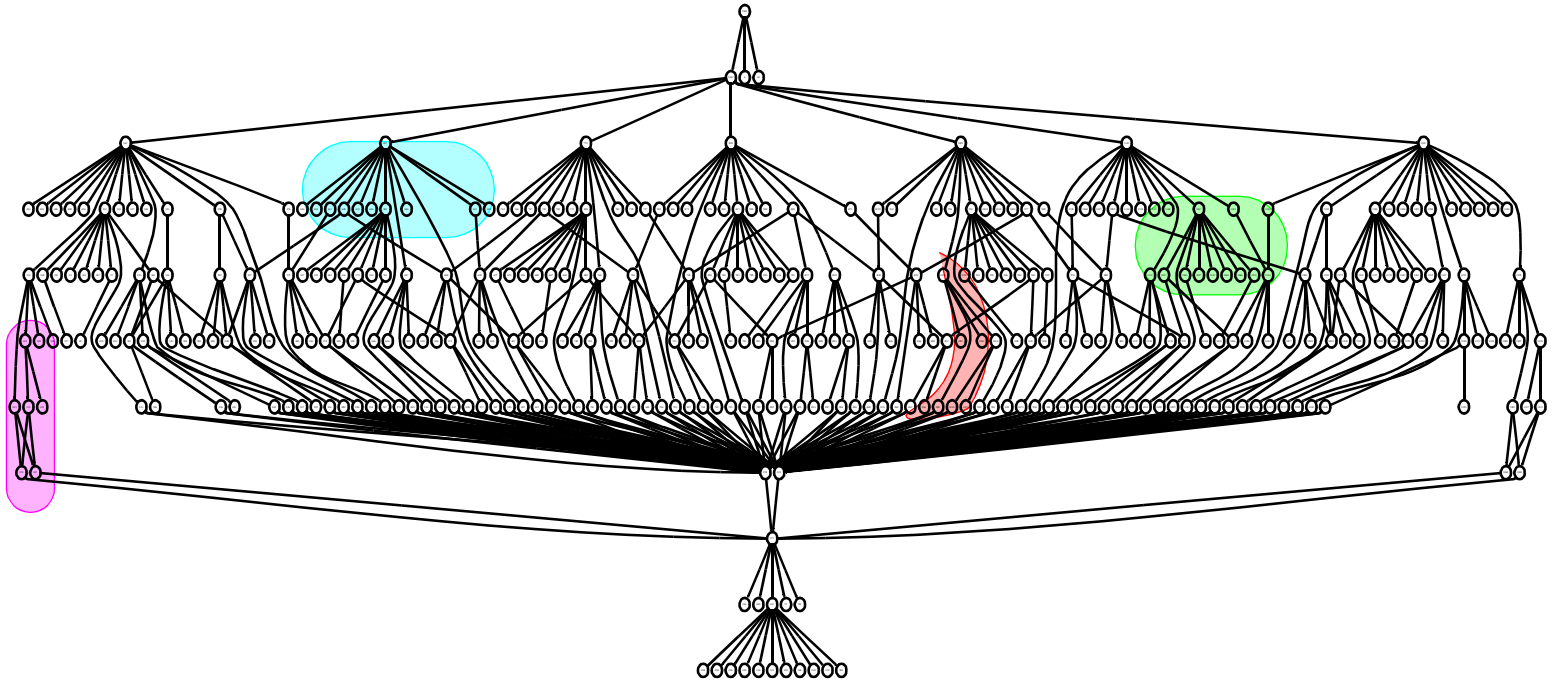


Sharing of parts to

- configure complex models in a natural way
- permit solving of isolated parts



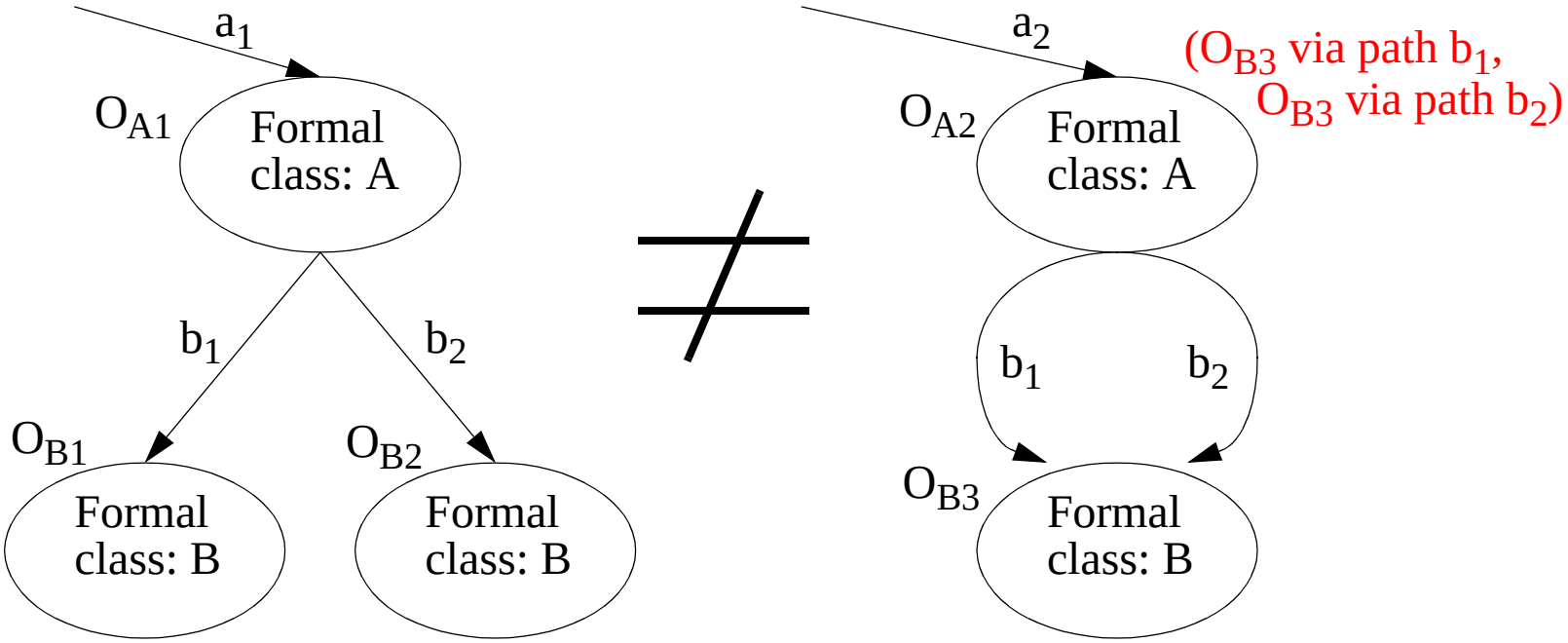
The DAG for a seven tray column



Much repeated structure

- Can we find it?
- Can we then use that information?

Anonymous types

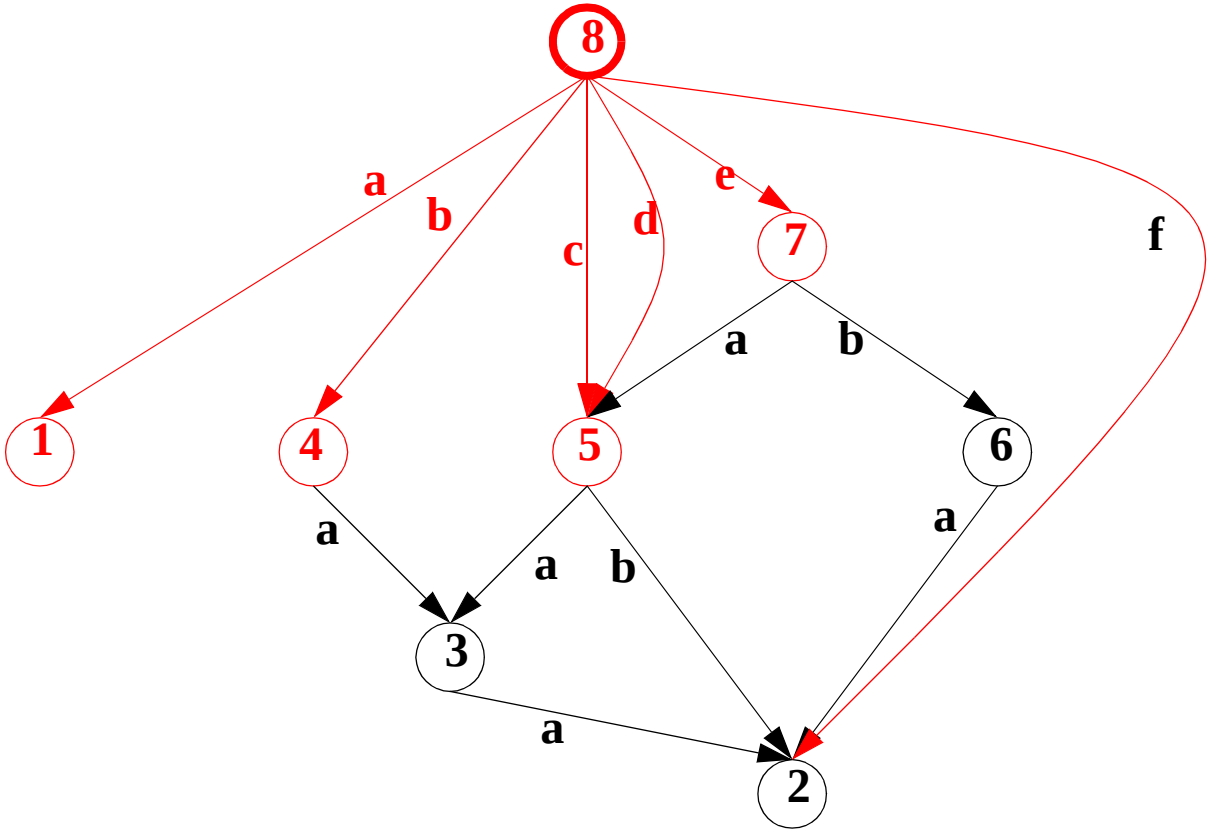


Algorithm

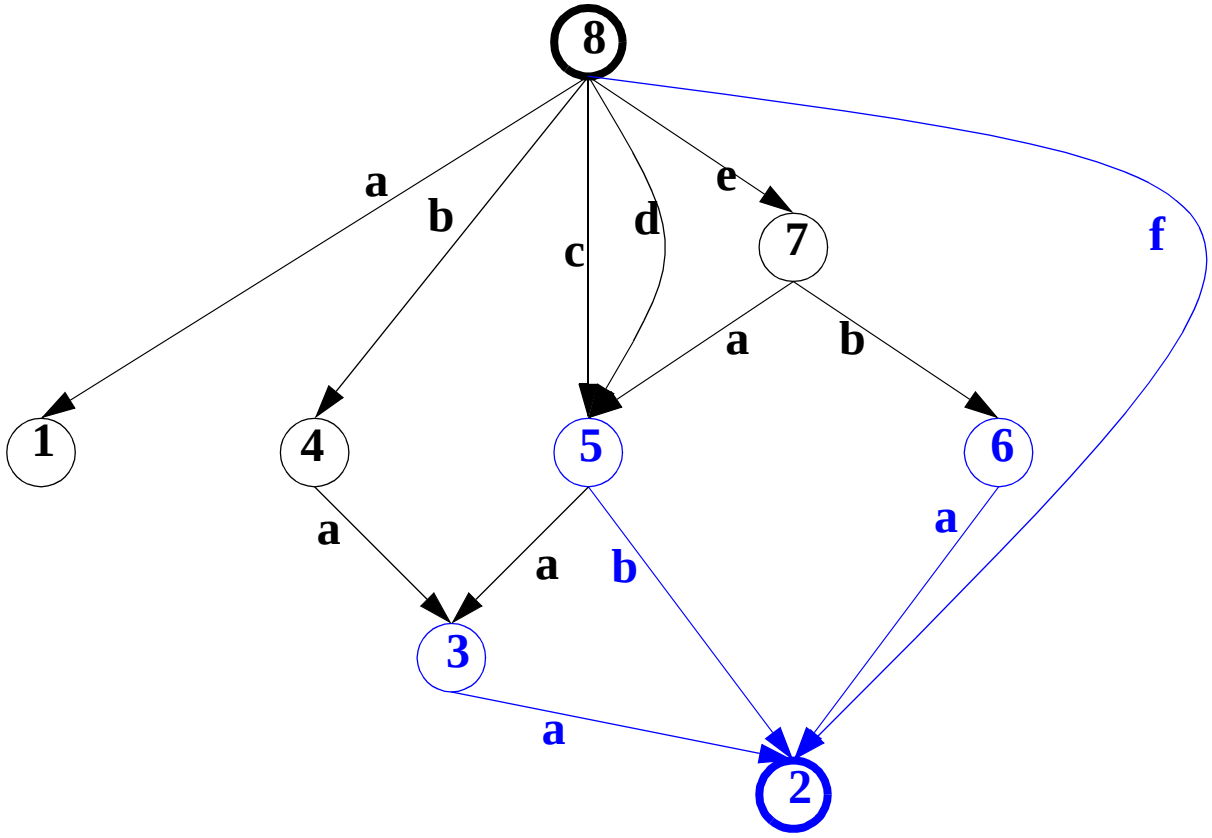
- For all nodes *i*
  + Construct list of visible nodes
- For all node pairs *i* and *j*
  + Find all independent paths *i* to *j*
  + Label node *i* with such paths

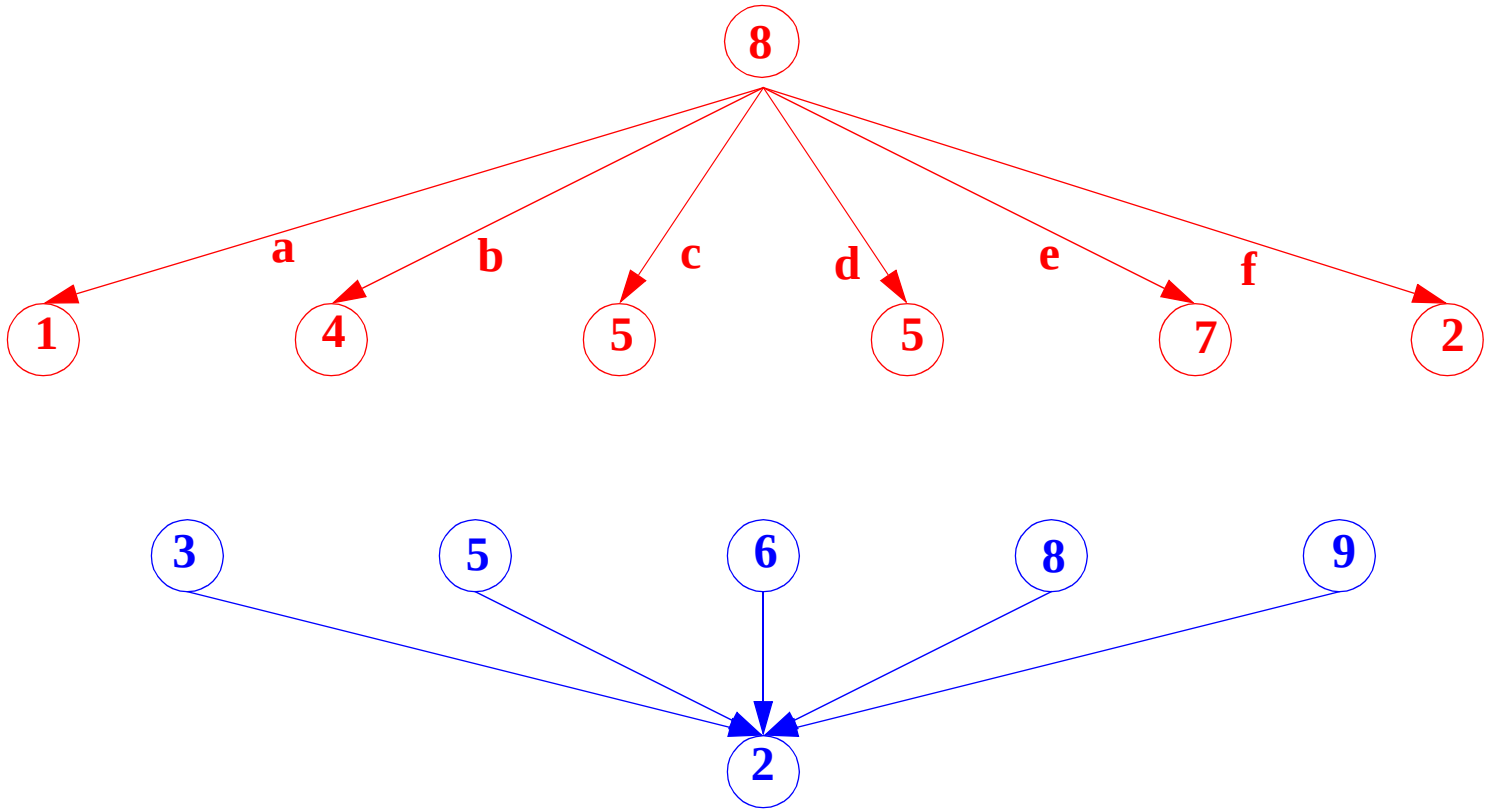
Provably order n^4; experience linear growth

Children of node *i* (8)

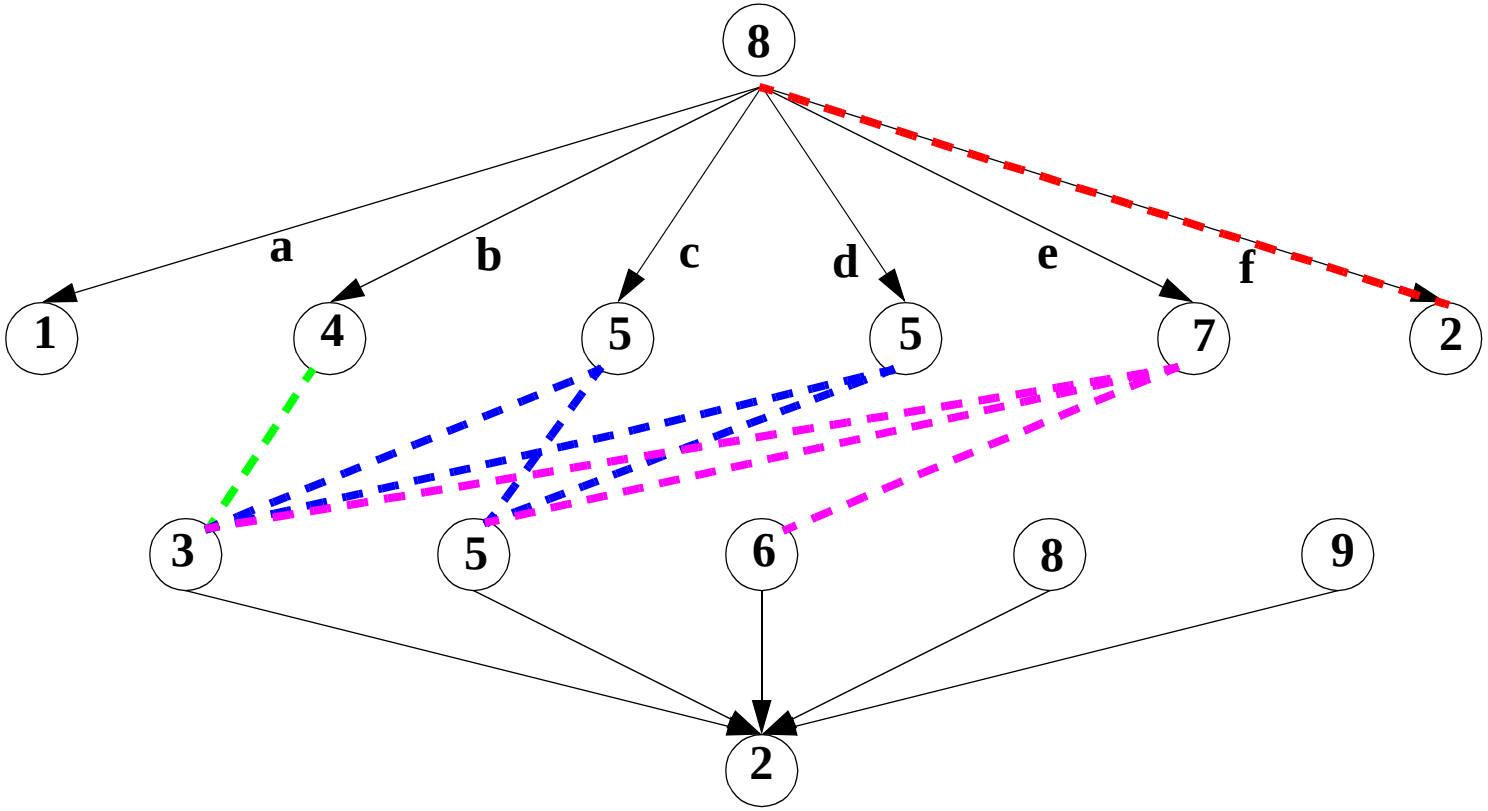


Parents of node *j* (2)

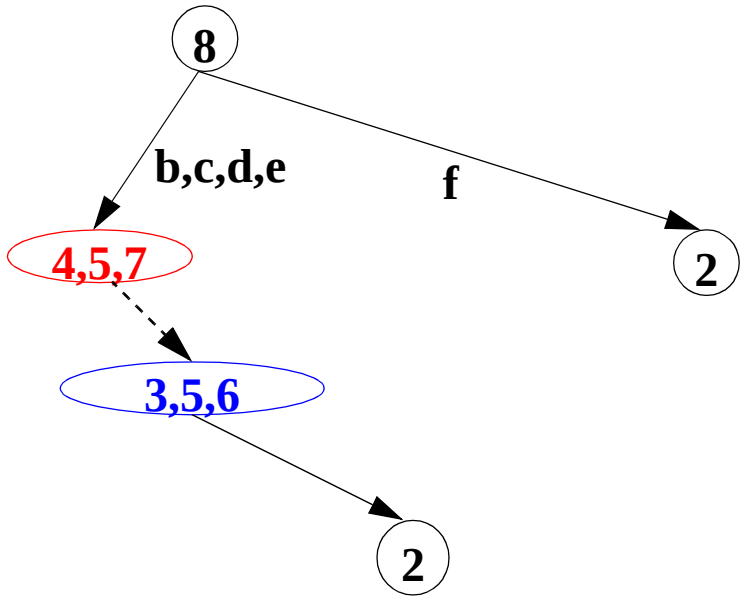



8
a
b
c
d
e
f
1
4
5
5
7
2
3
5
6
8
9
2

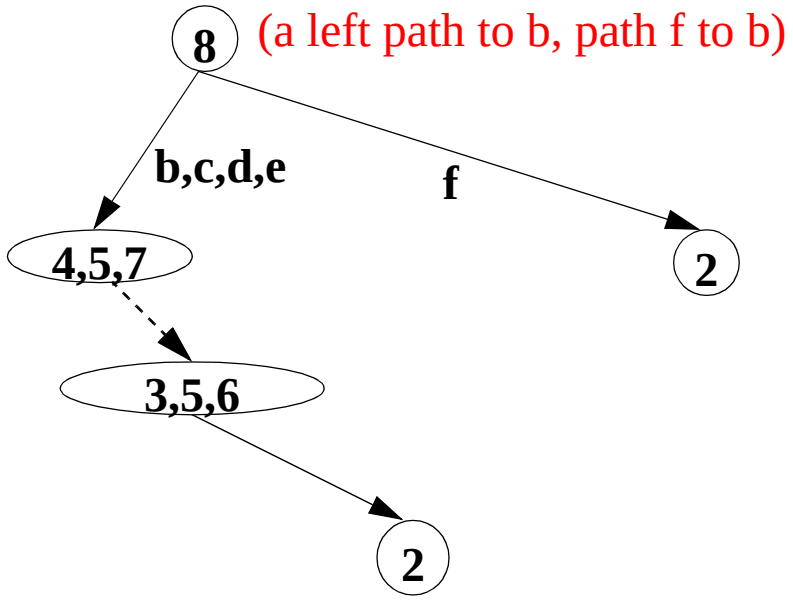
Parents of *j* seen by children of *i*



Collapse graph; spot independent paths



Label node *i*



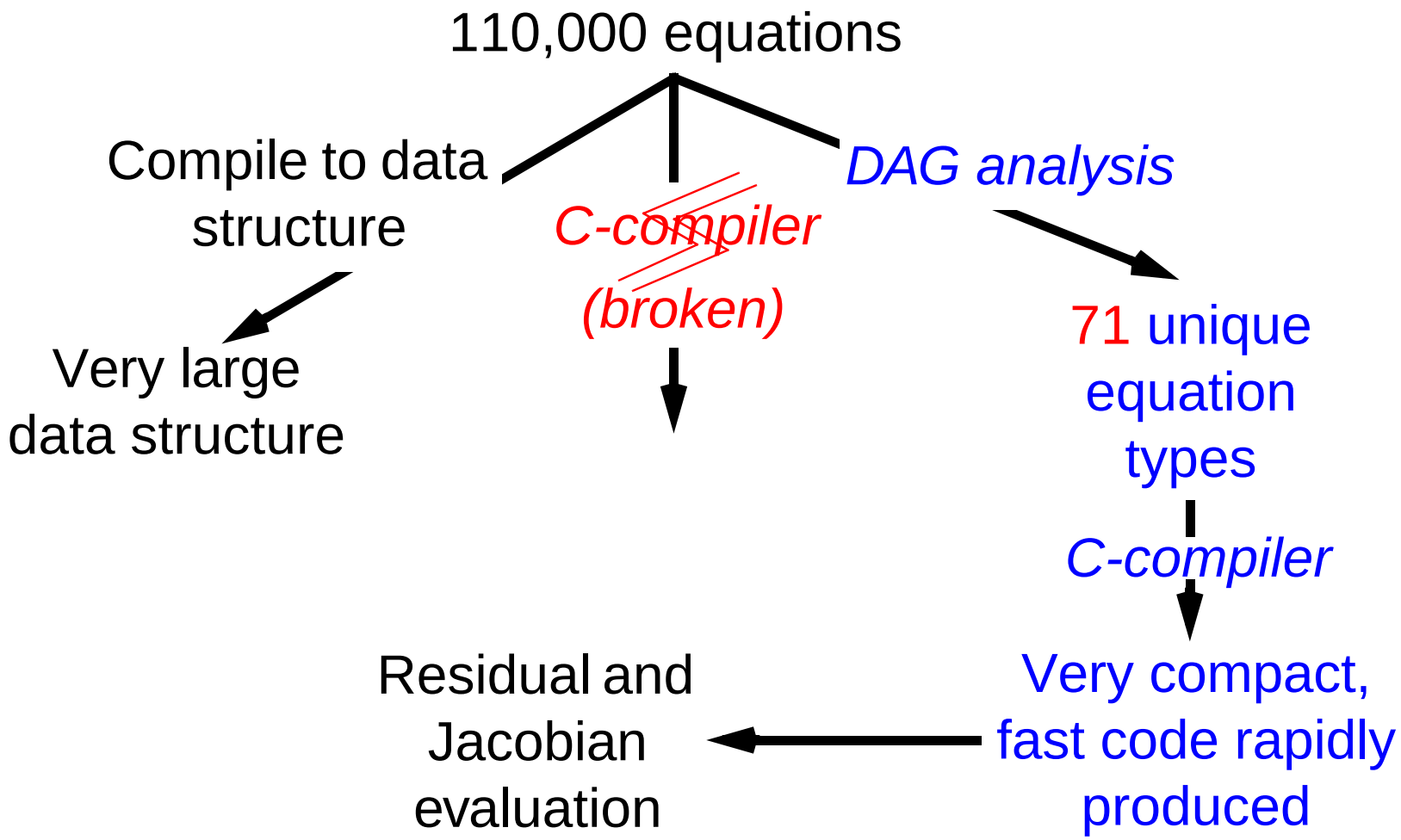
Very fast compile

We now compile at the rate of
120,000 equations/minute on
200 megahertz pentium PC

and solve

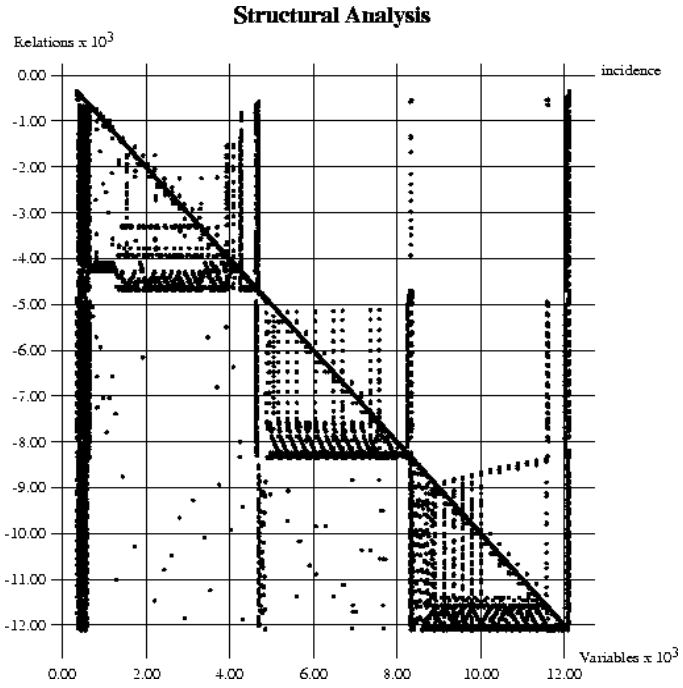
Residual evaluation is at
230,000 equations/second on
200 megahertz pentium PC

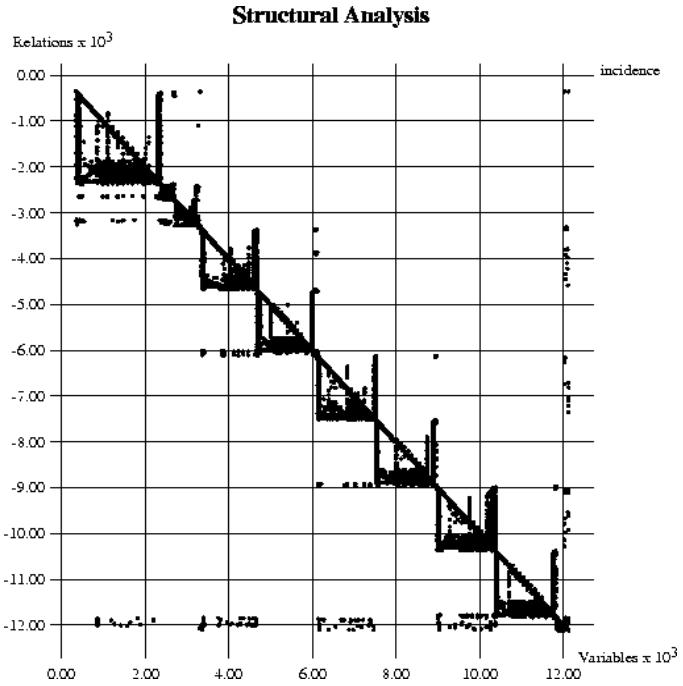
Why so fast?

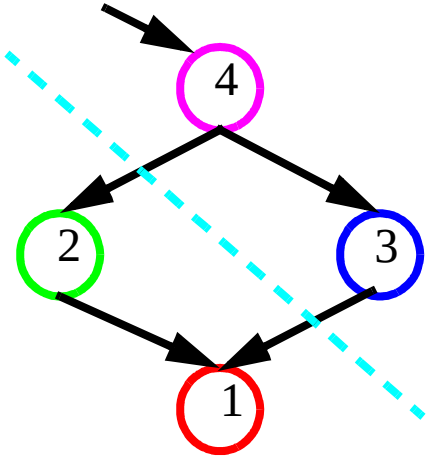

110,000 equations
Compile to data structure
C-compiler
(broken)
DAG analysis
Very large data structure
71 unique equation types
C-compiler
Very compact, fast code rapidly produced
Residual and Jacobian evaluation

Reducing fill for L\U factorization

- DAG-based algorithm pre-orders equations
- Recursive bisection at object boundaries







	1	2	3	4	***5***	6	7	8
1	***X***	x		x				
2	***x***	**X**	x					
3			**X**	x				
4				**X**	***x***			
5						**X**	x	x
6						x	**X**	
7	***x***						x	**X**
8					***X***		x	

L\U factorization speeds

Times reduced by factors of 5 to 10

Ex:

83,000 equations (53,000 equation partition) preordered in 15.5 sec and factored in 4 sec on 150 megahertz UNIX workstation

In conclusion

DAG-based algorithms to speed up

- compiling
- evaluating residuals and jacobians
- finding BBD structure for L\U factoring

to meet the goal to

Compile and solve 100,000 equations (flow-sheet modeling) in three minutes on a PC