Overview of Continuous Optimization Advances and Applications to Circuit Tuning

Andrew R. Conn, Chandu Visweswariah

IBM Thomas J. Watson Research Center

Yorktown Heights, NY

Outline

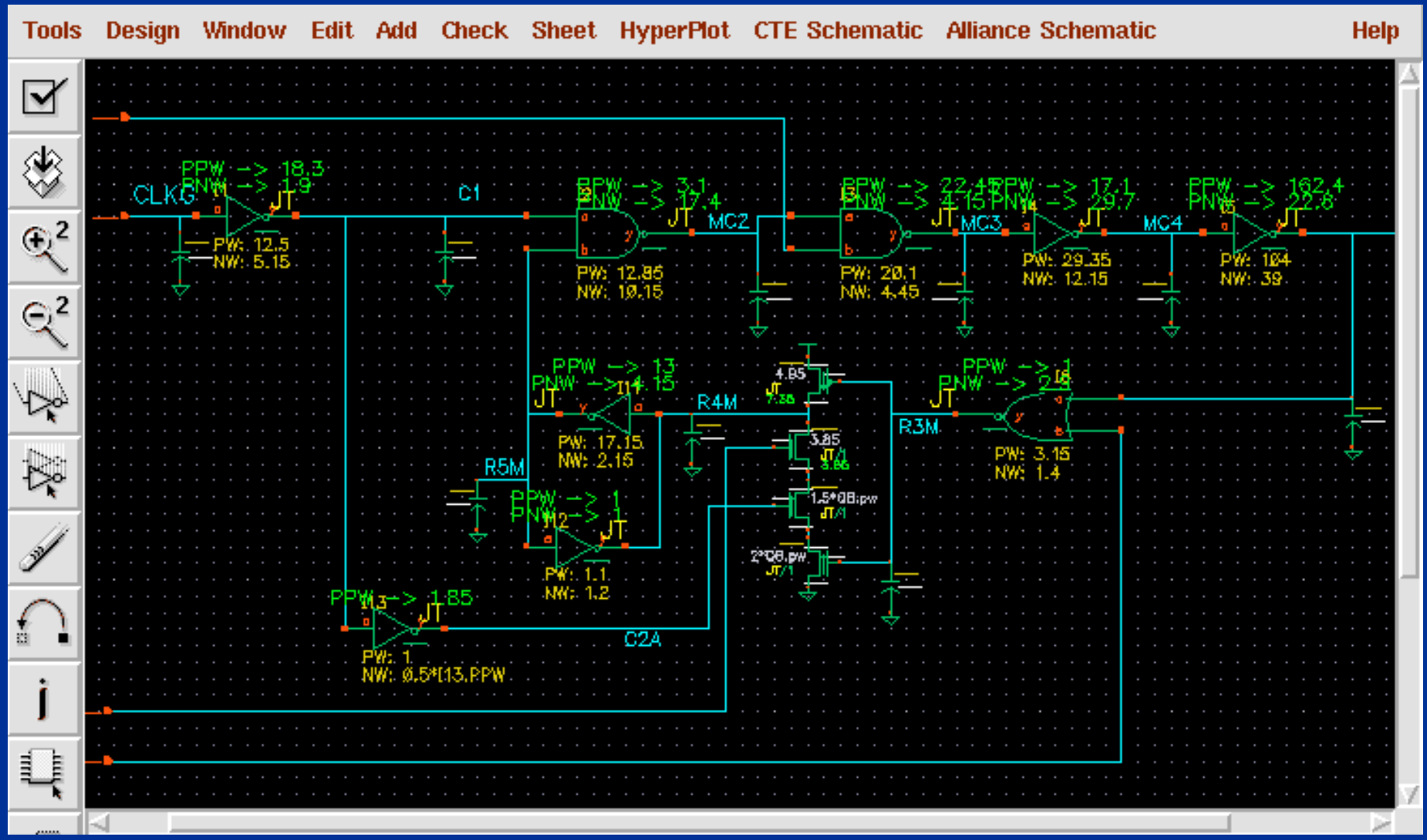
- Part I: Nonlinear optimization (Andy Conn)
- Part II: Application to circuit tuning (Chandu Visweswariah)

Part II: Application to circuit tuning

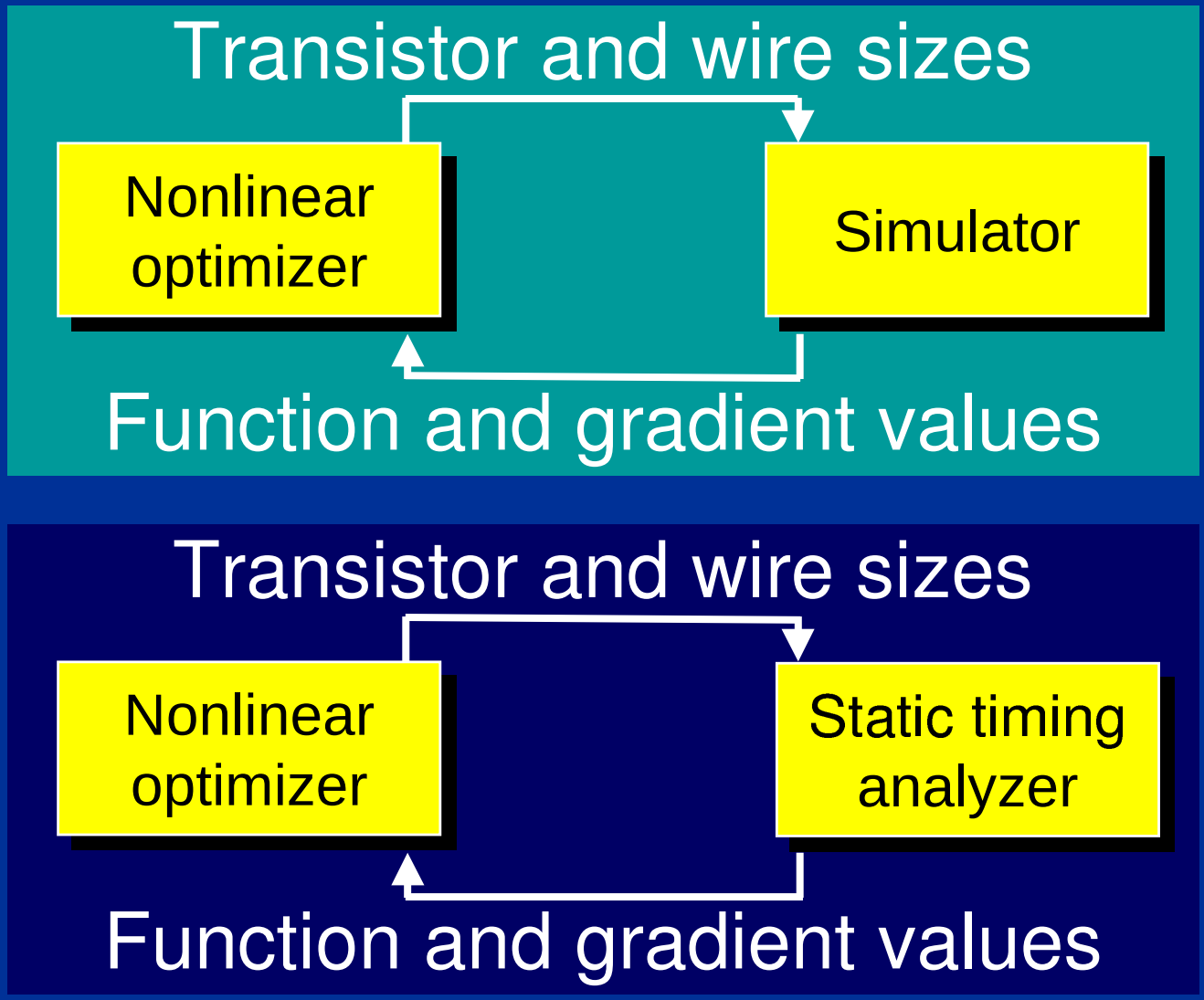
- Motivation
- Dynamic tuning: JiffyTune
- Static tuning: EinsTuner
- Conclusions

- Acknowledgments: the entire JiffyTune and EinsTuner teams at IBM, and Phillip Restle for the algorithm animations

What is circuit tuning?



Dynamic vs. static tuning



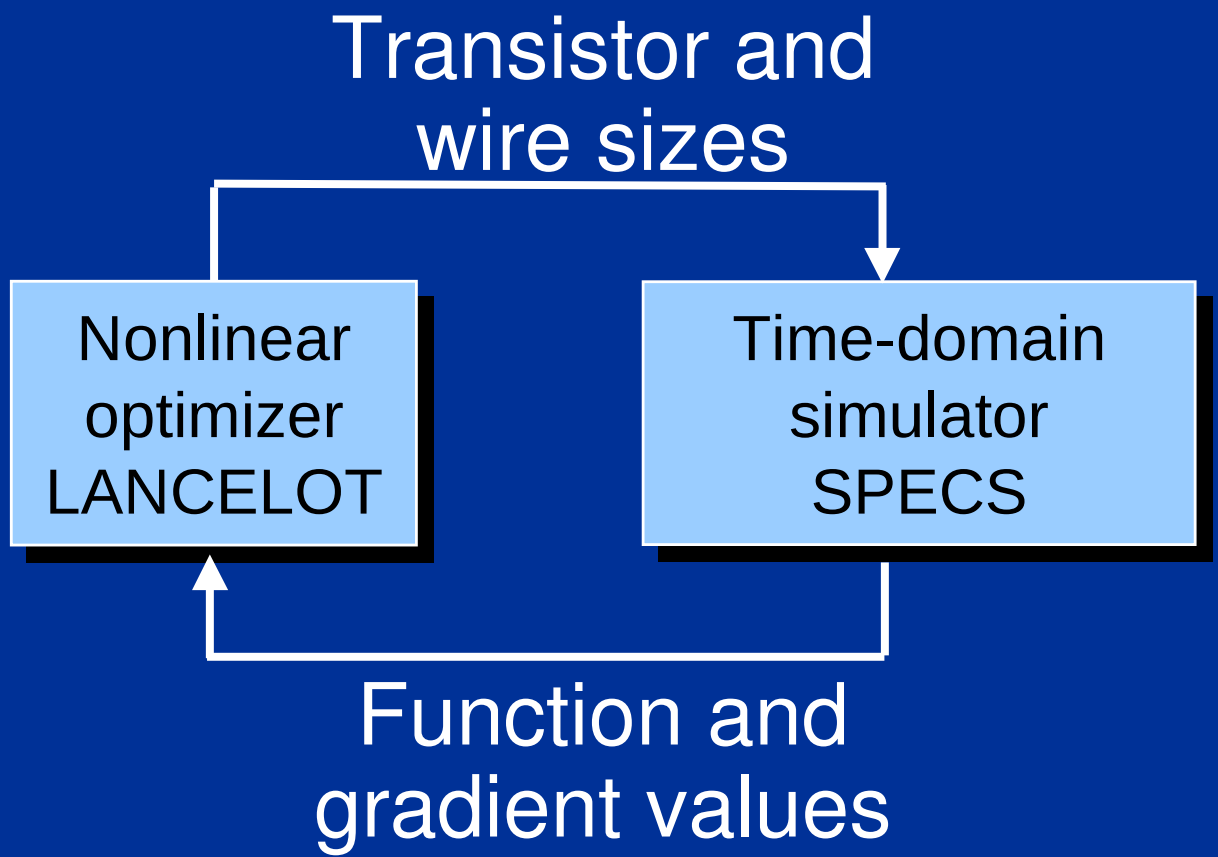
Dynamic vs. static tuning

Dynamic	Static
Example: DELIGHT.SPICE, JiffyTune	Example: TILOS, Contrast, EinsTuner
Optimizes specified paths	Optimizes all paths
Needs input vectors	None required
Requires carefully thought-out problem statement	Automatic
Hard to use	Easy
Not susceptible to false paths, pessimism	Susceptible
Easier to add noise/power considerations	Harder

Application to circuit tuning

- Most nonlinear optimizers benchmarked by performance on analytic problems
- But custom circuit tuning implies simulation (noisy and expensive function and gradient values), with important implications
  - all reasonable steps to reduce iterations (e.g., two-step updating, SIAM J Opt. 9/99)
  - noise considerations
  - failure recovery
- Do not treat the optimizer as a black box; rather, customize to the application

JiffyTune: dynamic circuit tuner

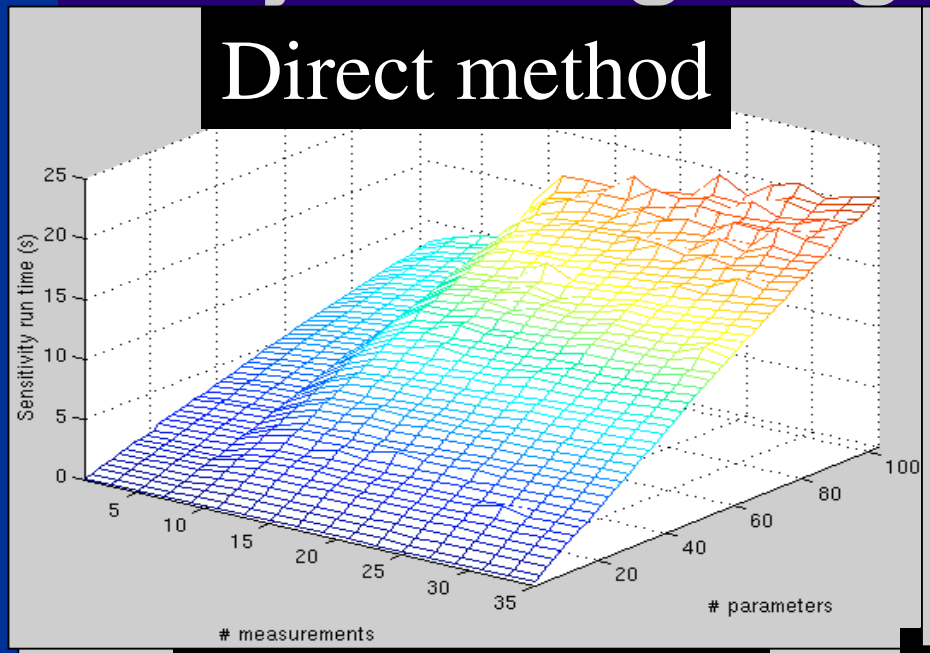


JiffyTune features

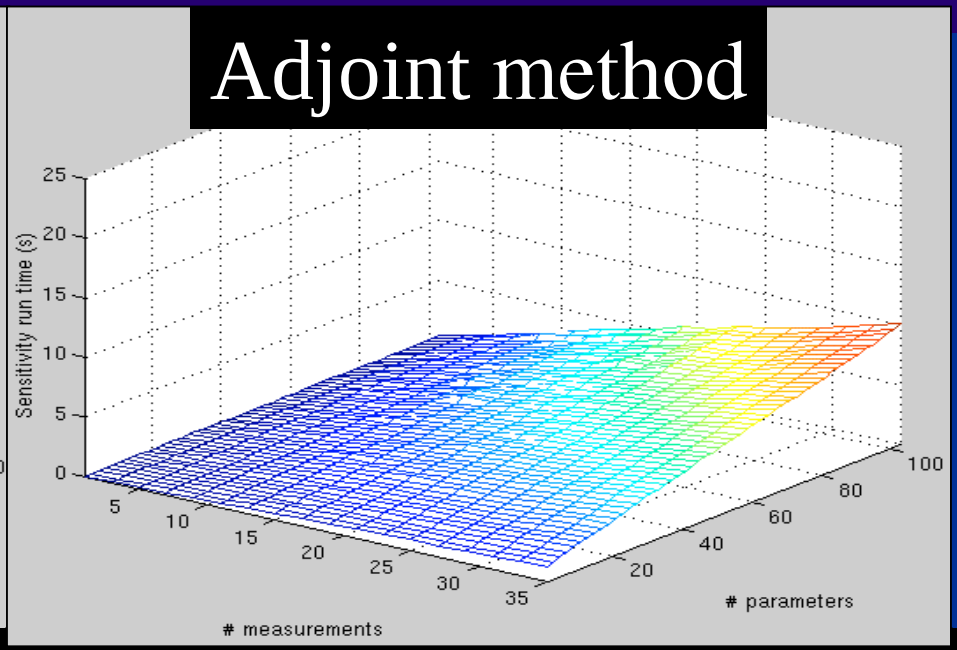
- Delay, slew (rise/fall time), area, noise, power measurements
- Any objective function and constraints
- Minimax optimization
- Ratio-ing and grouping
- Graphical problem specification; graphical back-annotation of tuned results and timing information
- 20,000+ transistor capacity

Adjoint Lagrangian formulation

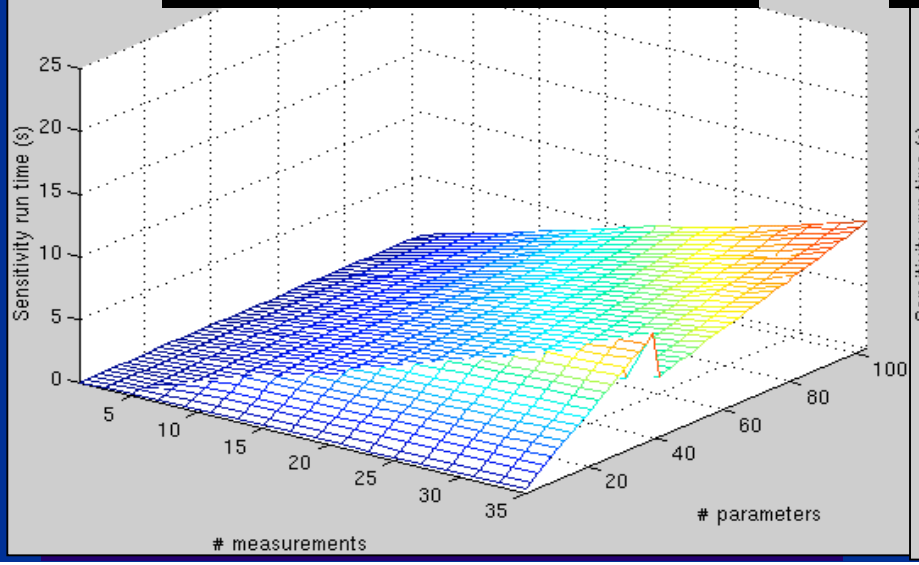
Direct method



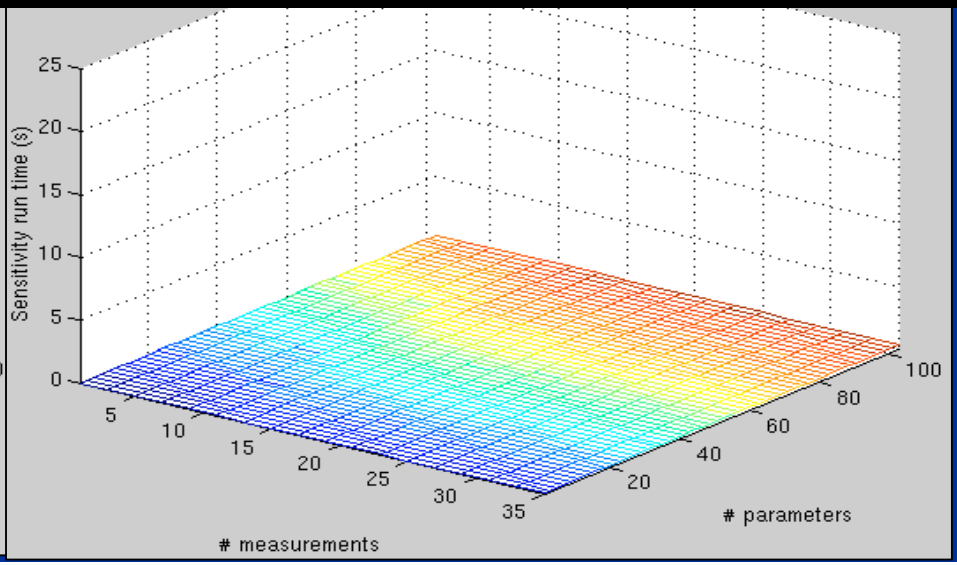
Adjoint method



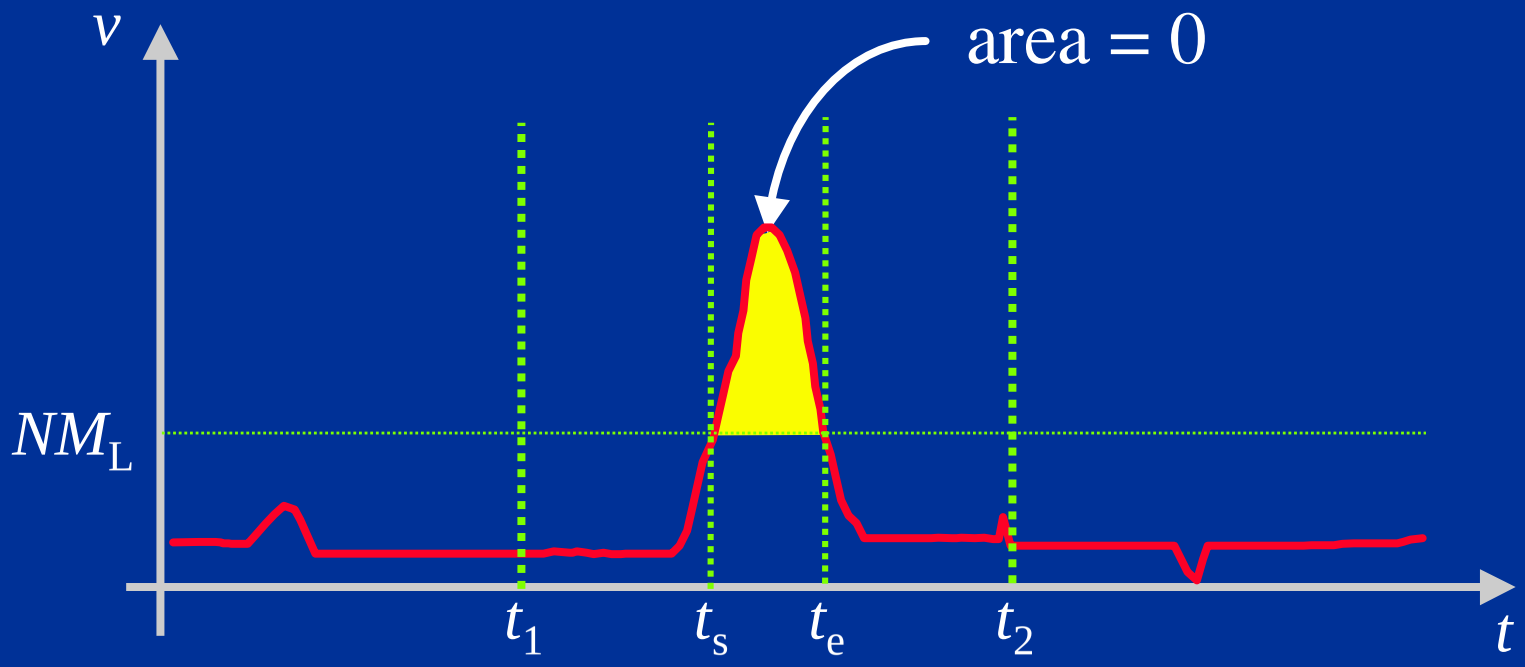
Heuristic choice



Adjoint Lagrangian method



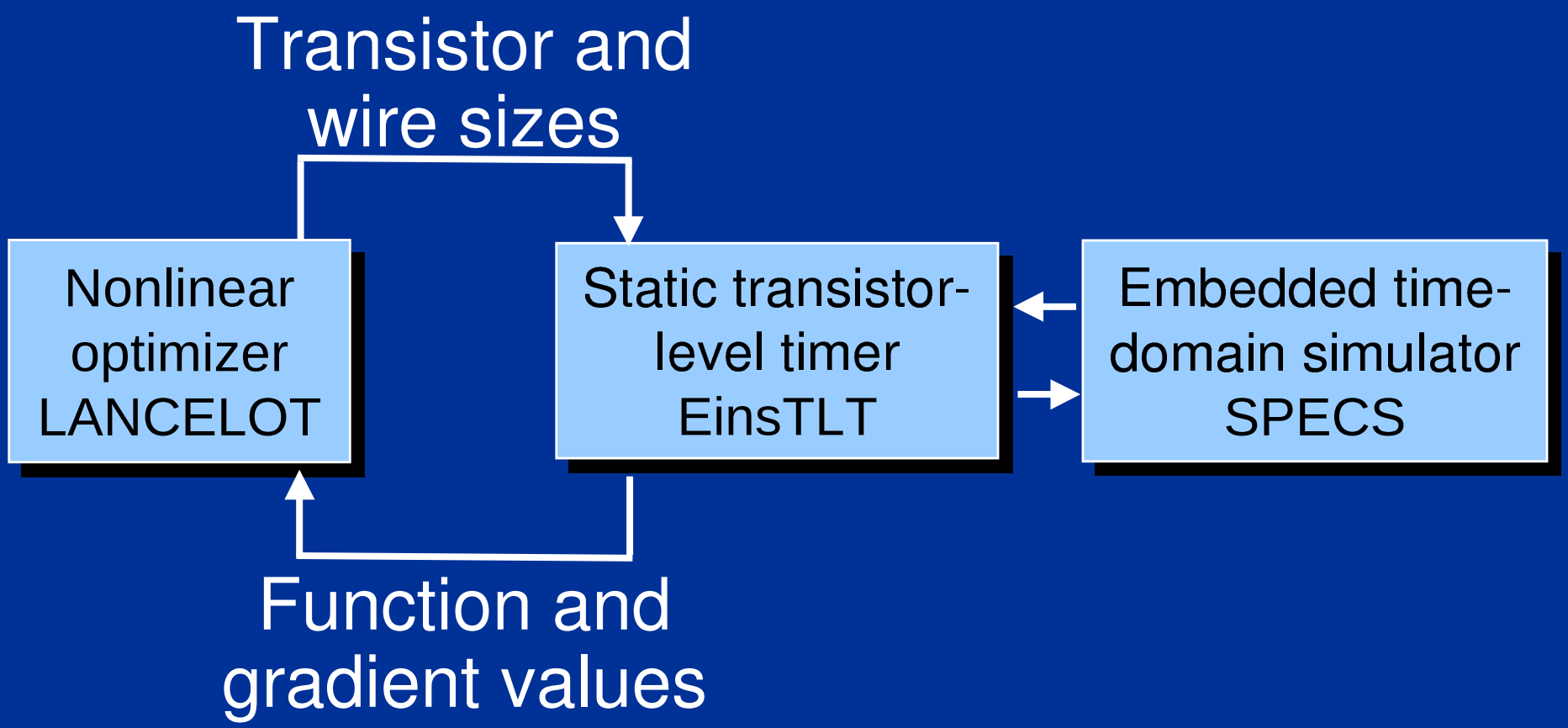
Noise



$$v(x,t) \le NM_L \text{ for all } t \text{ in } [t_1, t_2]$$

$$\int_{t_s}^{t_e} \{v(x,t) - NM_L\}\, dt = 0$$

EinsTuner: formal static optimizer



Components of EinsTuner

1 Read netlist; create timing graph (EinsTLT)

Formulate pruned optimization problem 2

Feed problem to nonlinear optimizer (LANCELOT)

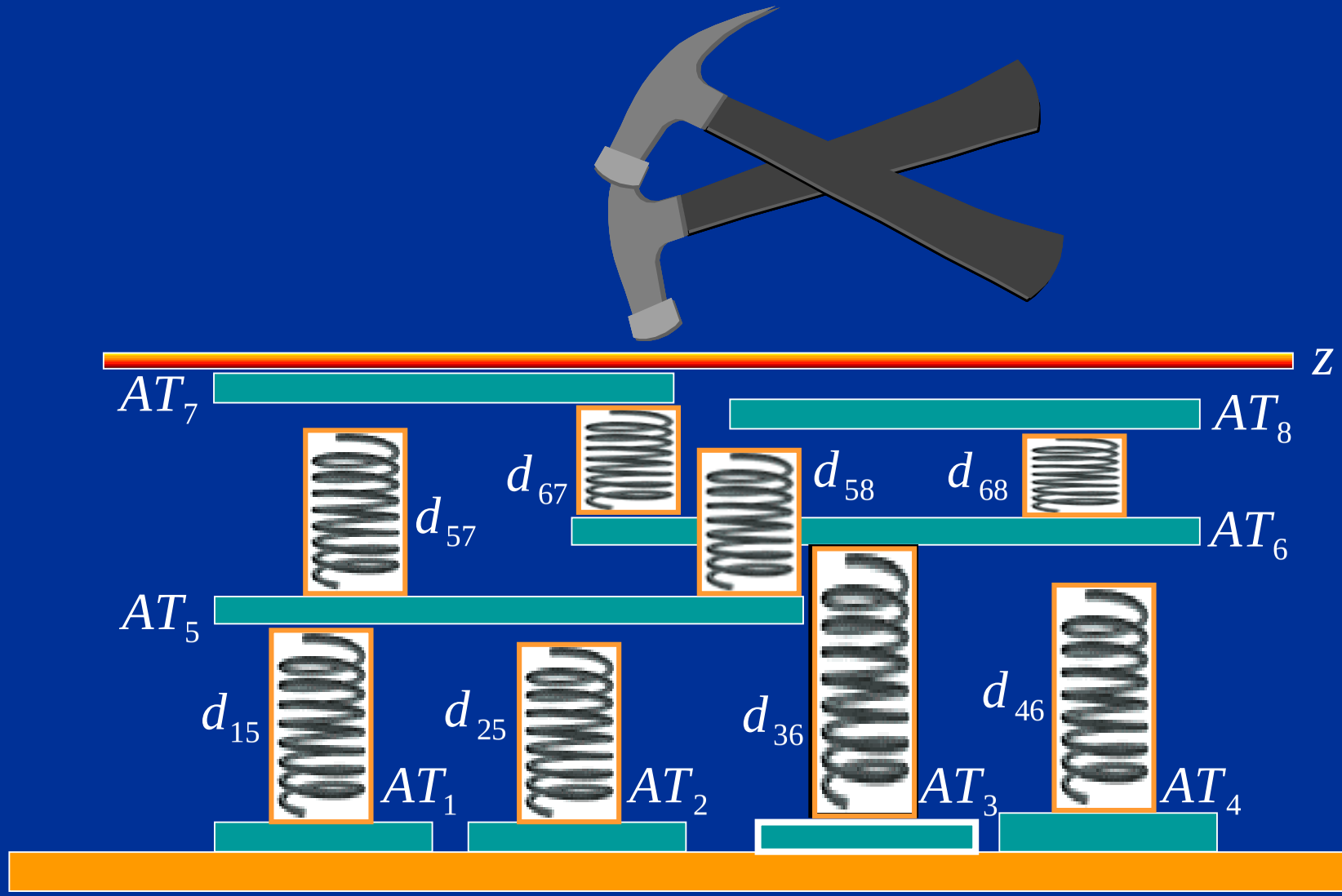
Solve optimization problem, call simulator for delays/slews and gradients thereof

Fast simulation and incremental sensitivity computation (SPECS)

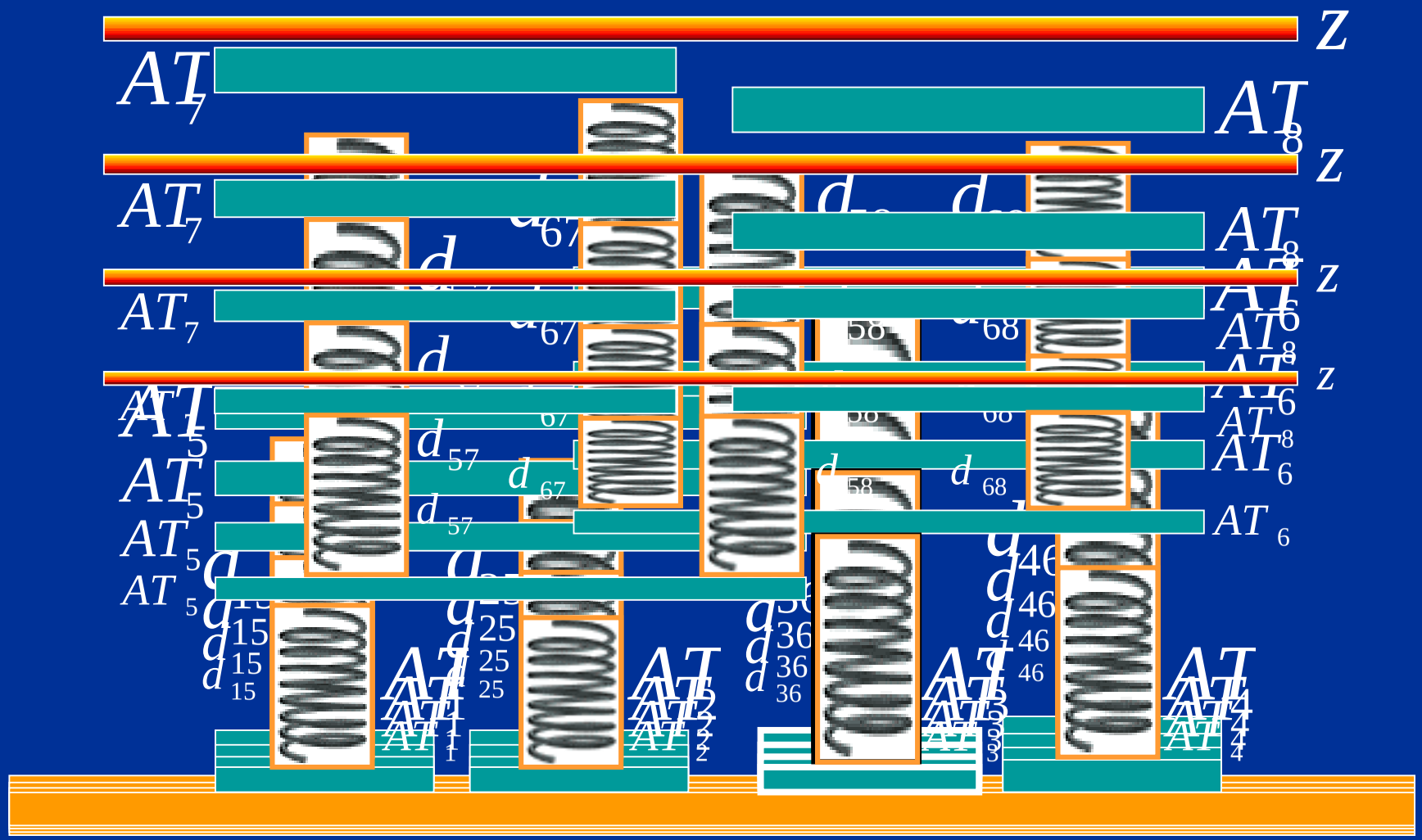
Obtain converged solution

Snap-to-grid; back-annotate; re-time

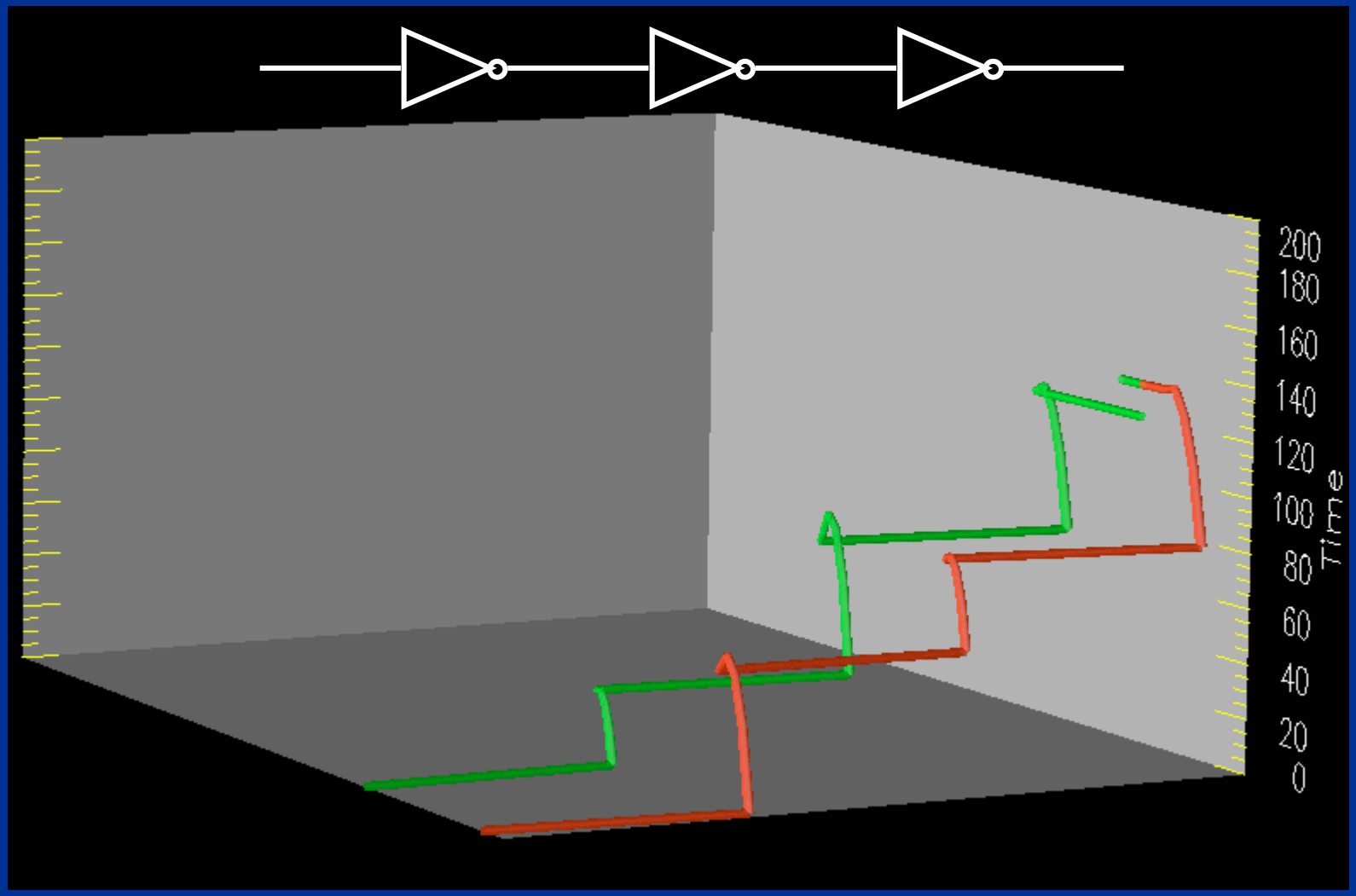
Springs and planks analogy



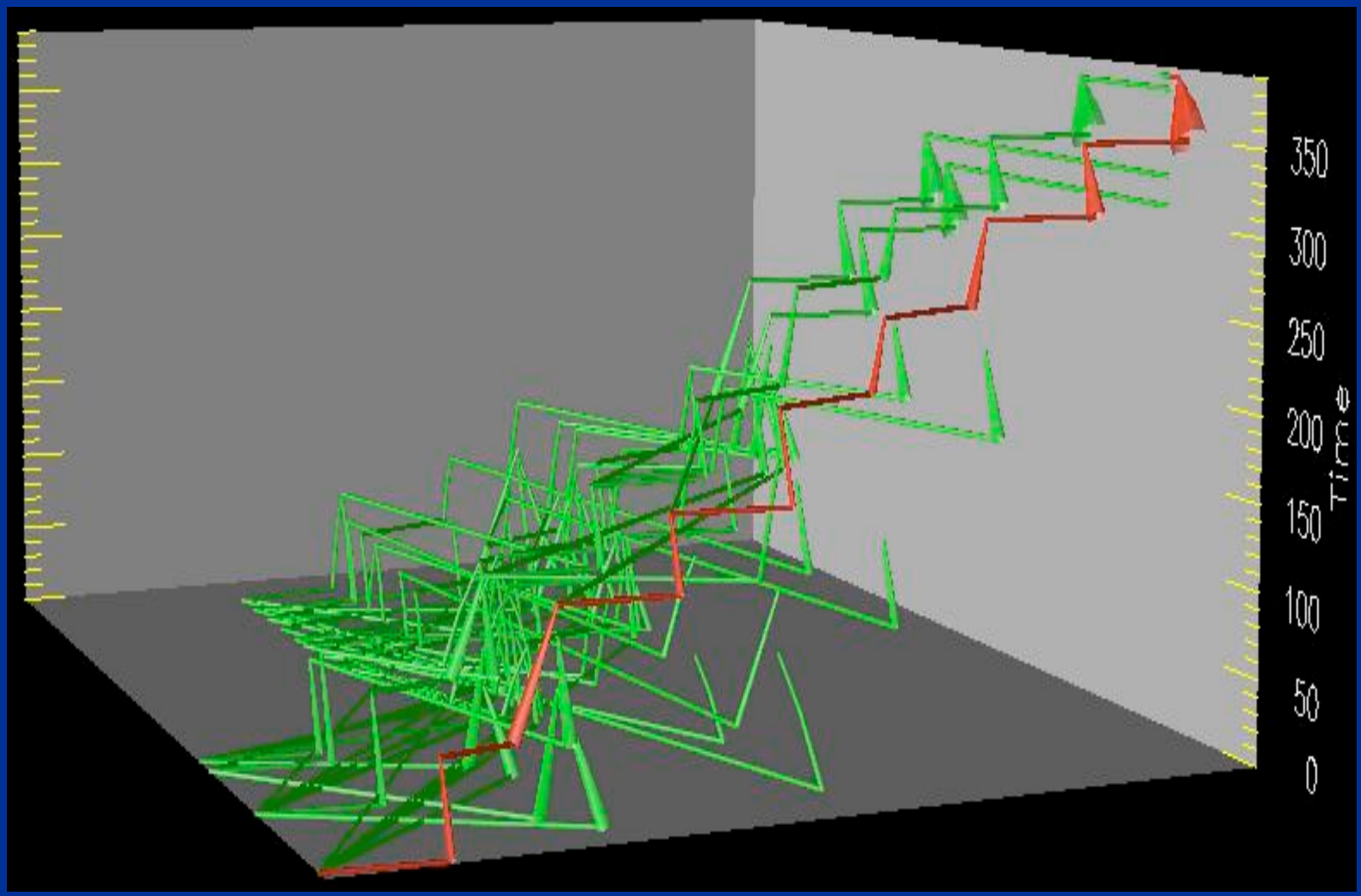
Springs and planks analogy



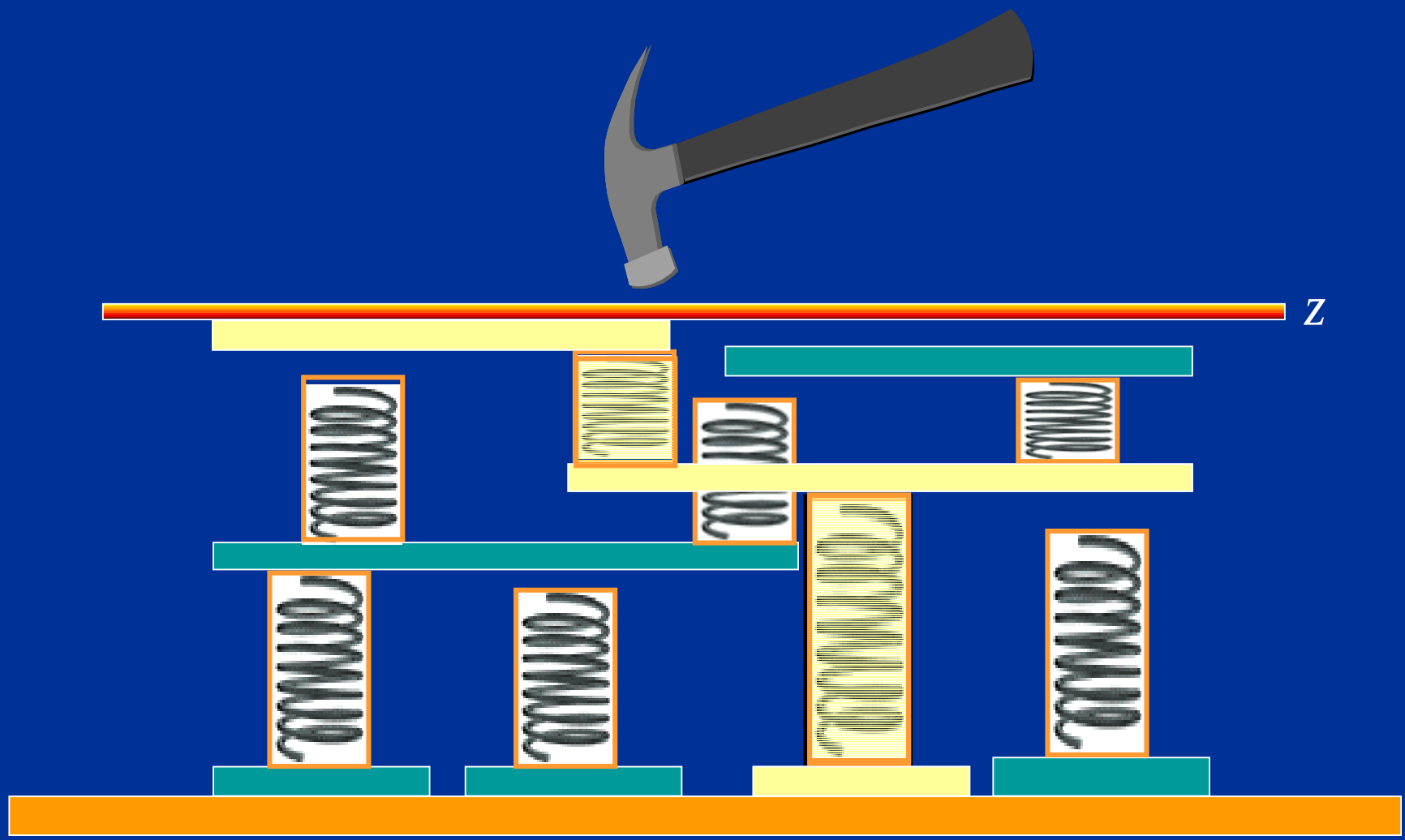
Algorithm demonstration: inv3



Algorithm demonstration: 1b ALU

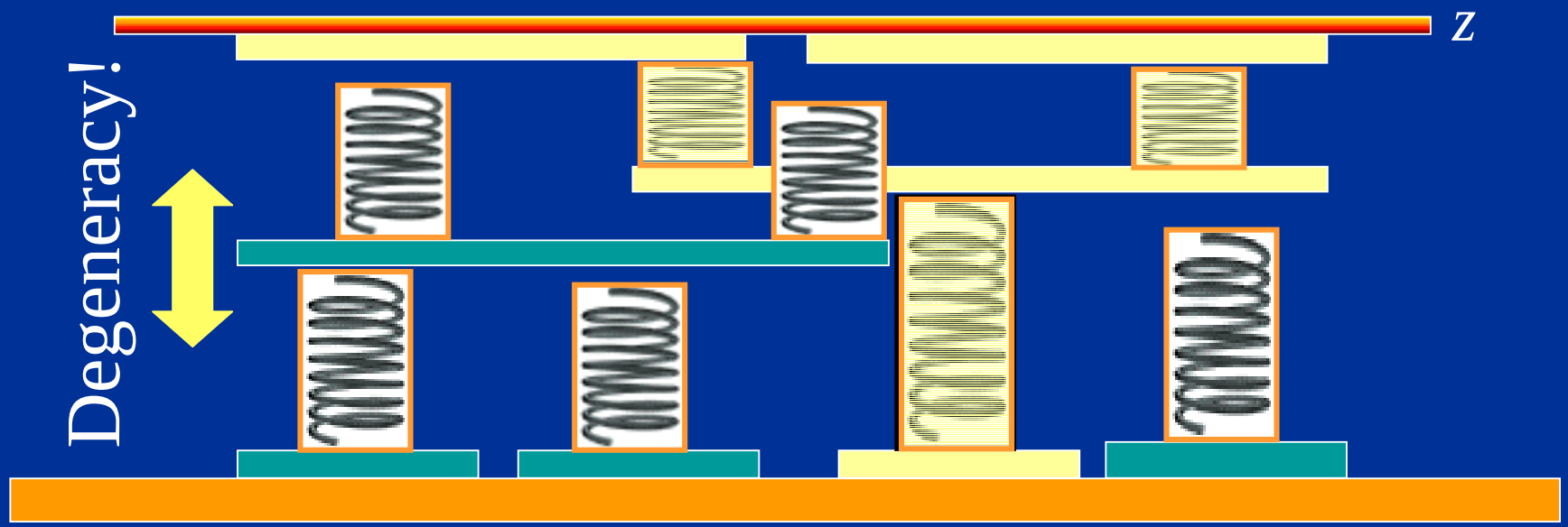


Demonstration of degeneracy

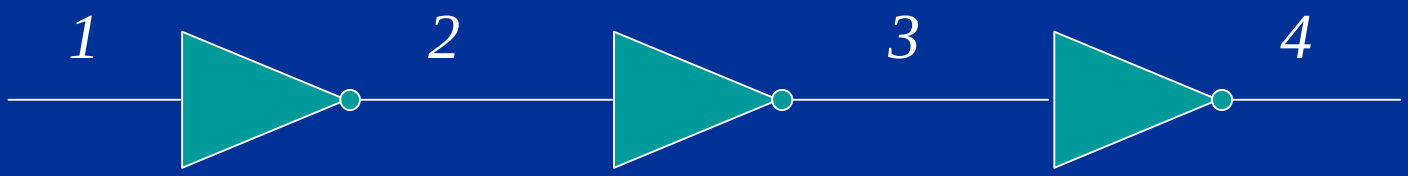


Demonstration of degeneracy

- We need pruning to combat redundancy and degeneracy
- We need pruning to reduce size



Pruning: an example



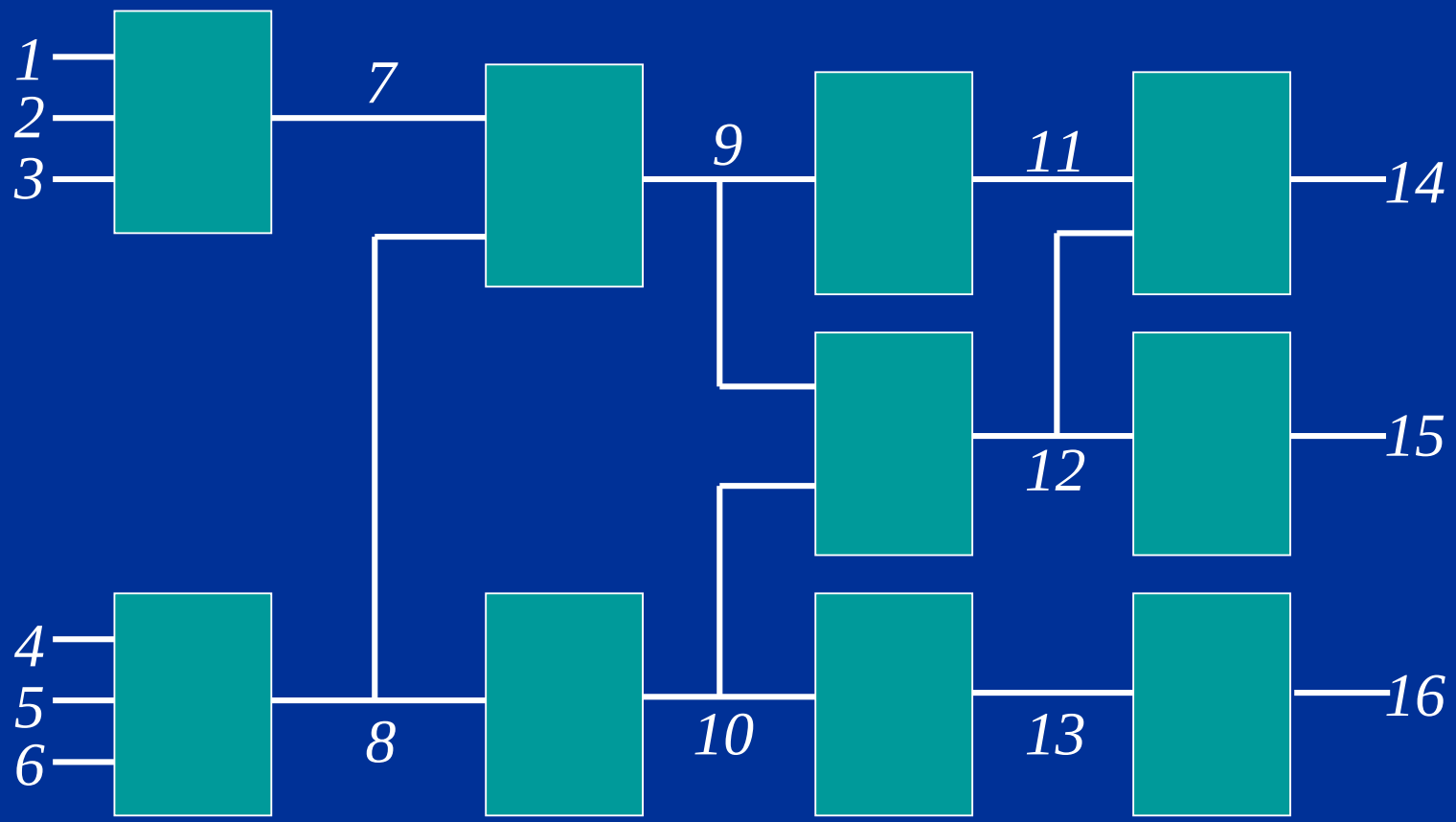
$$\min z$$
$$\text{s.t. } z \geq AT_4^r$$
$$\text{s.t. } z \geq AT_4^f$$
$$\text{s.t. } AT_4^r \geq AT_3^f + d_{34}^r$$
$$\text{s.t. } AT_4^f \geq AT_3^r + d_{34}^f$$
$$\text{s.t. } AT_3^r \geq AT_2^f + d_{23}^r$$
$$\text{s.t. } AT_3^f \geq AT_2^r + d_{23}^f$$
$$\text{s.t. } AT_2^r \geq AT_1^f + d_{12}^r$$
$$\text{s.t. } AT_2^f \geq AT_1^r + d_{12}^f$$

$$\Rightarrow$$

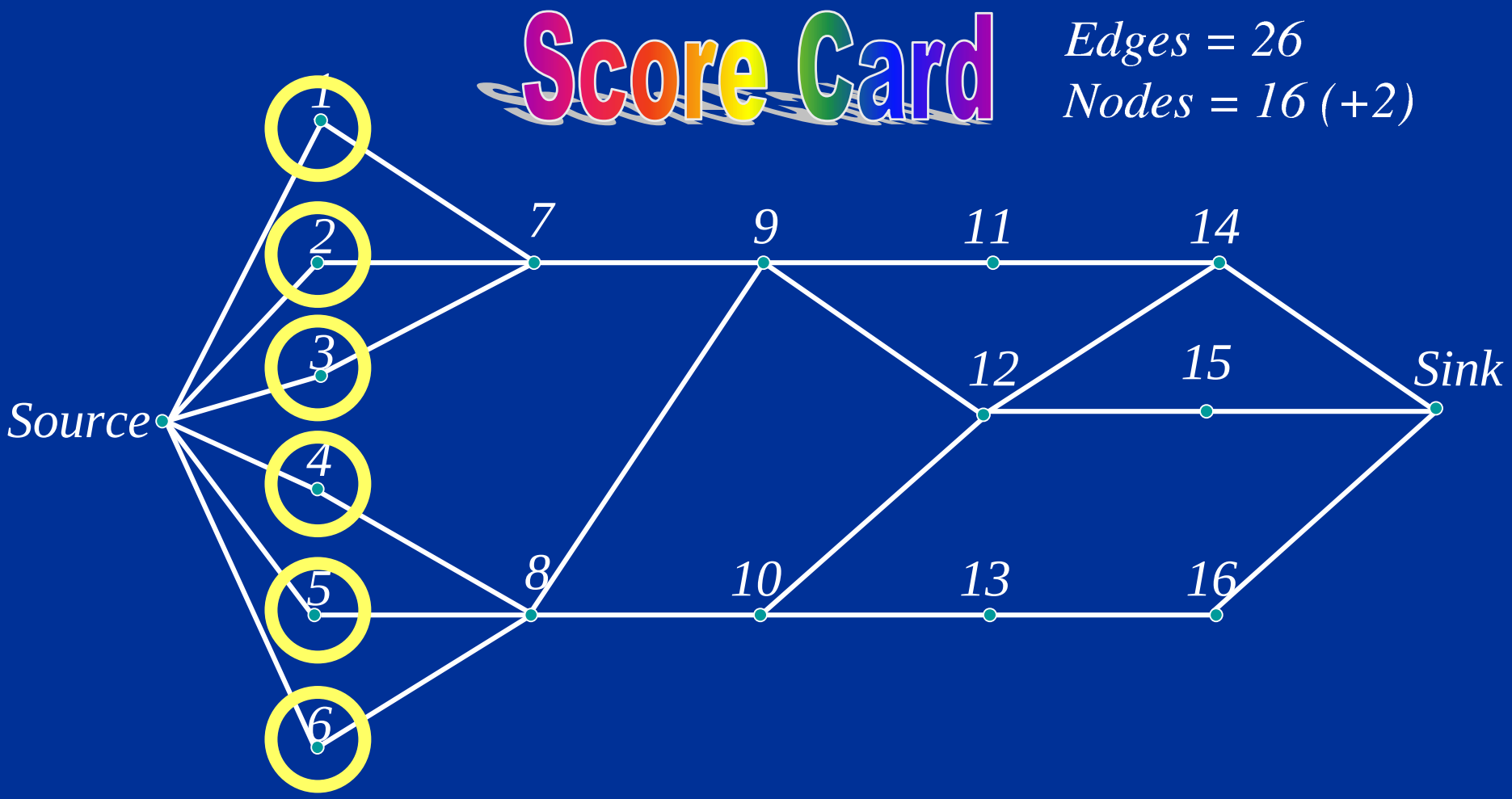
$$\min z$$
$$\text{s.t. } z \geq AT_1^r + d_{12}^f + d_{23}^r + d_{34}^f$$
$$\text{s.t. } z \geq AT_1^f + d_{12}^r + d_{23}^f + d_{34}^r$$

- Path-based vs. block-based formulation

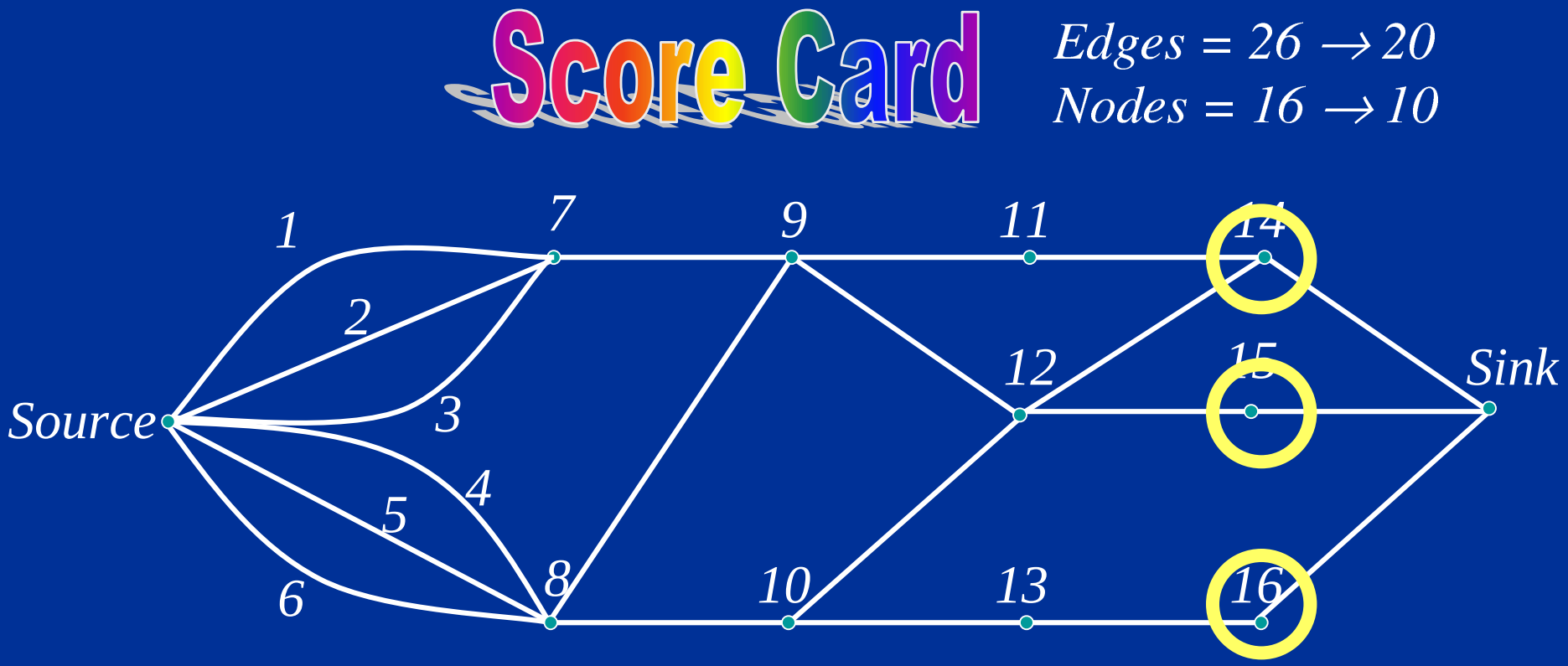
Detailed pruning example



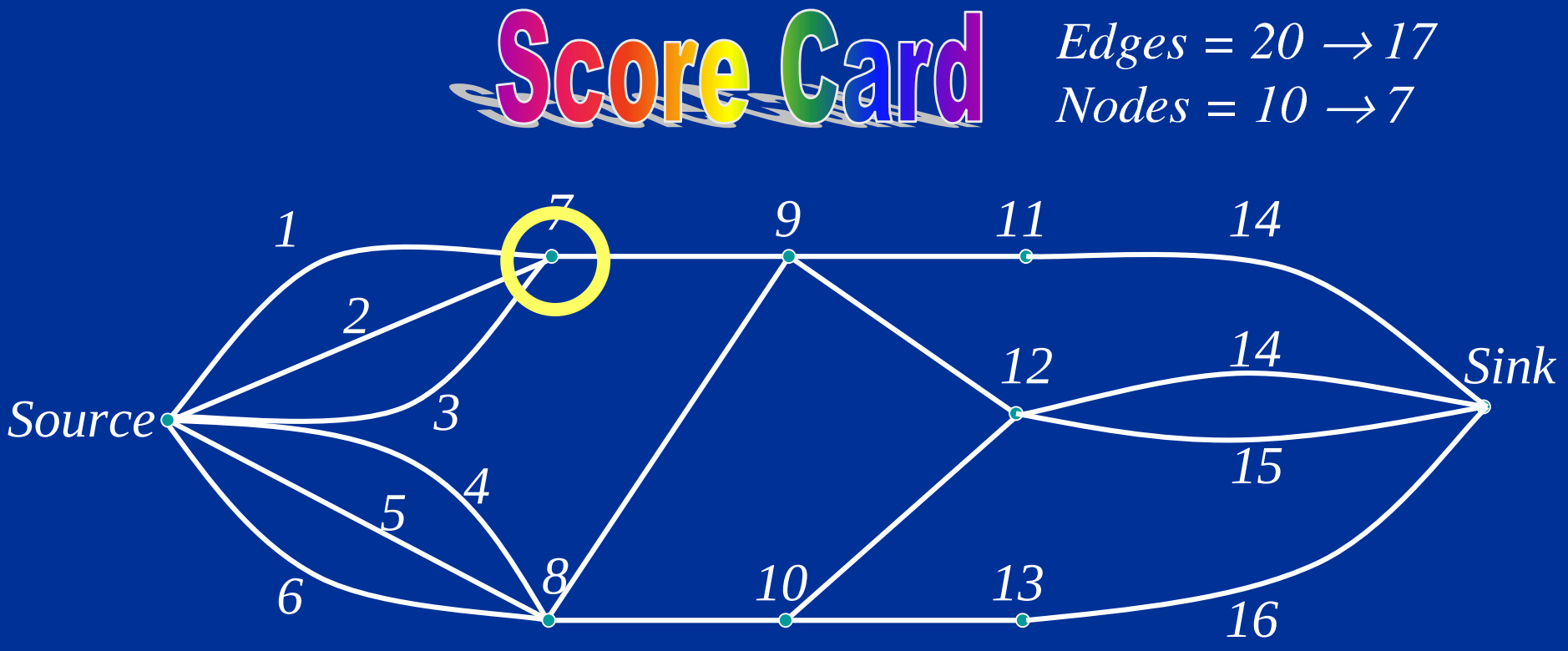
Detailed pruning example



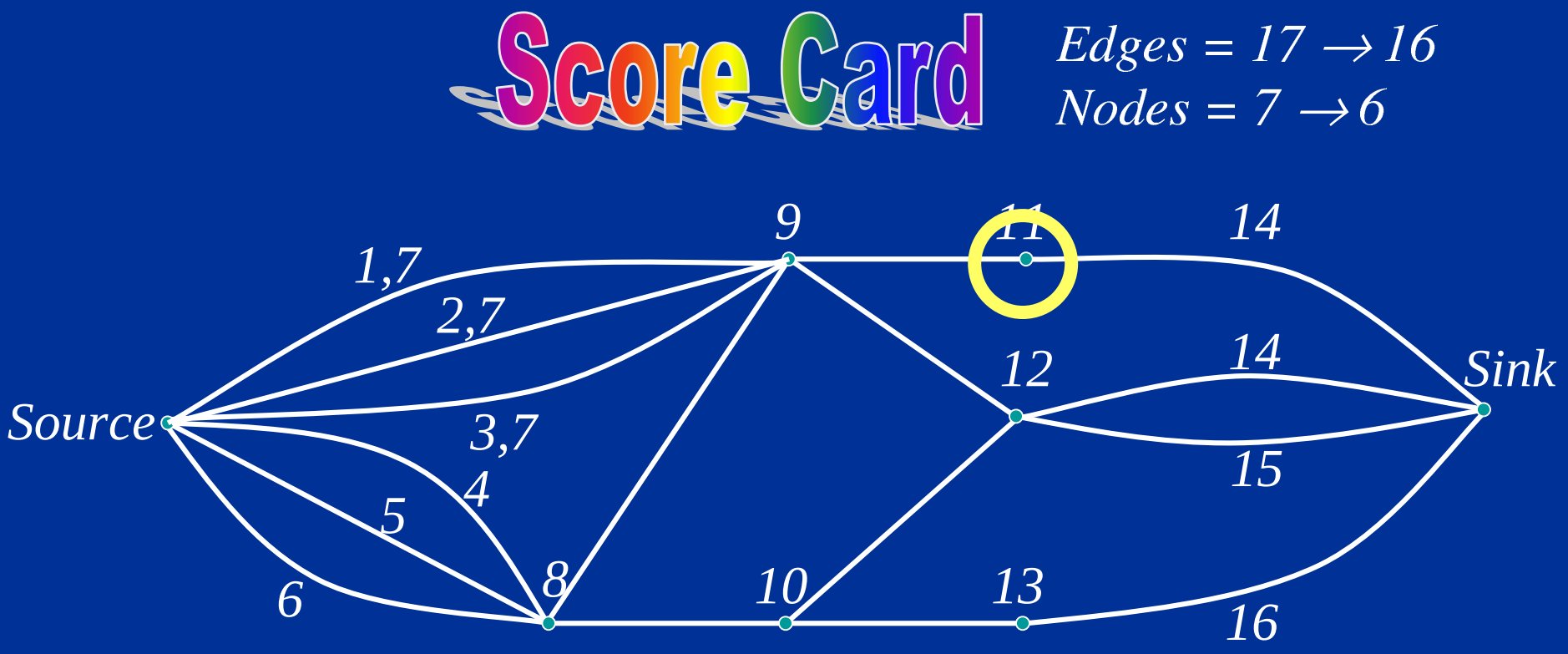
Detailed pruning example



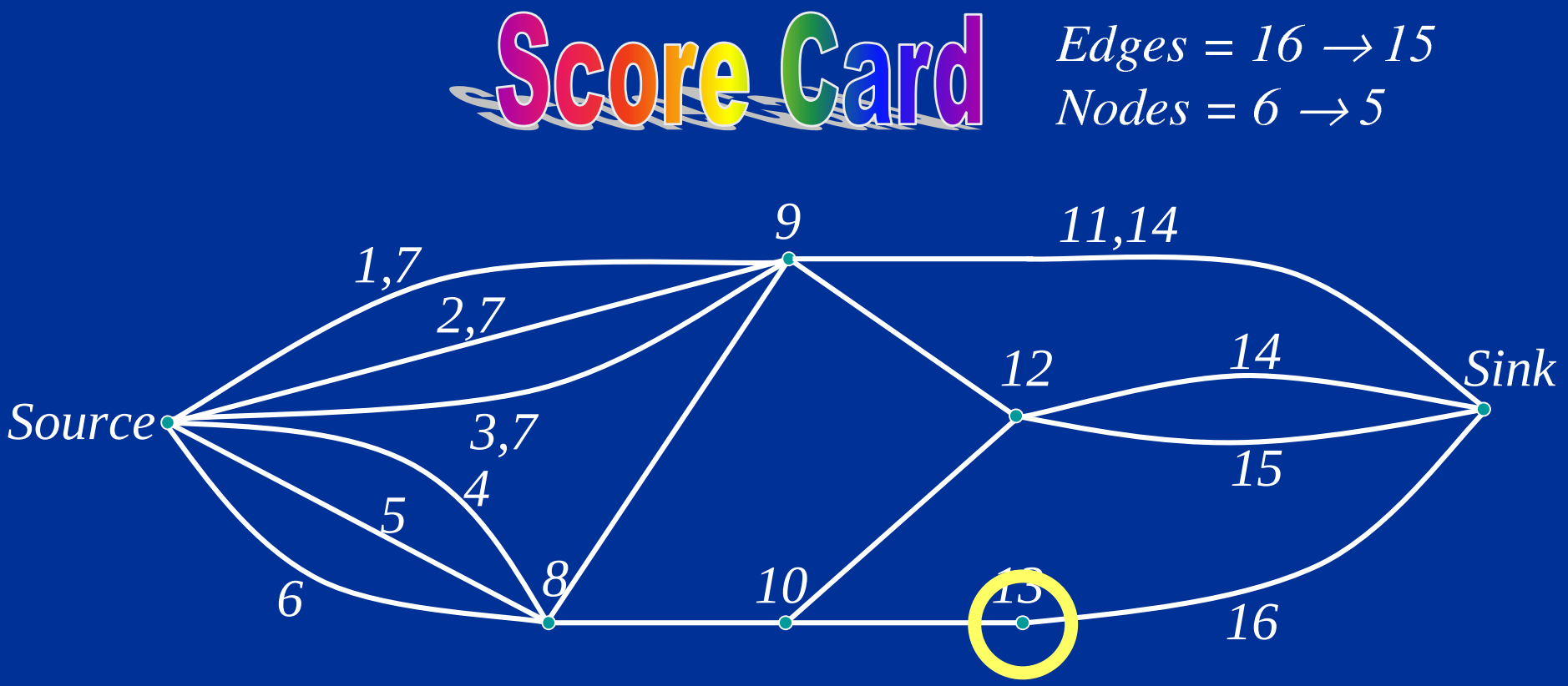
Detailed pruning example



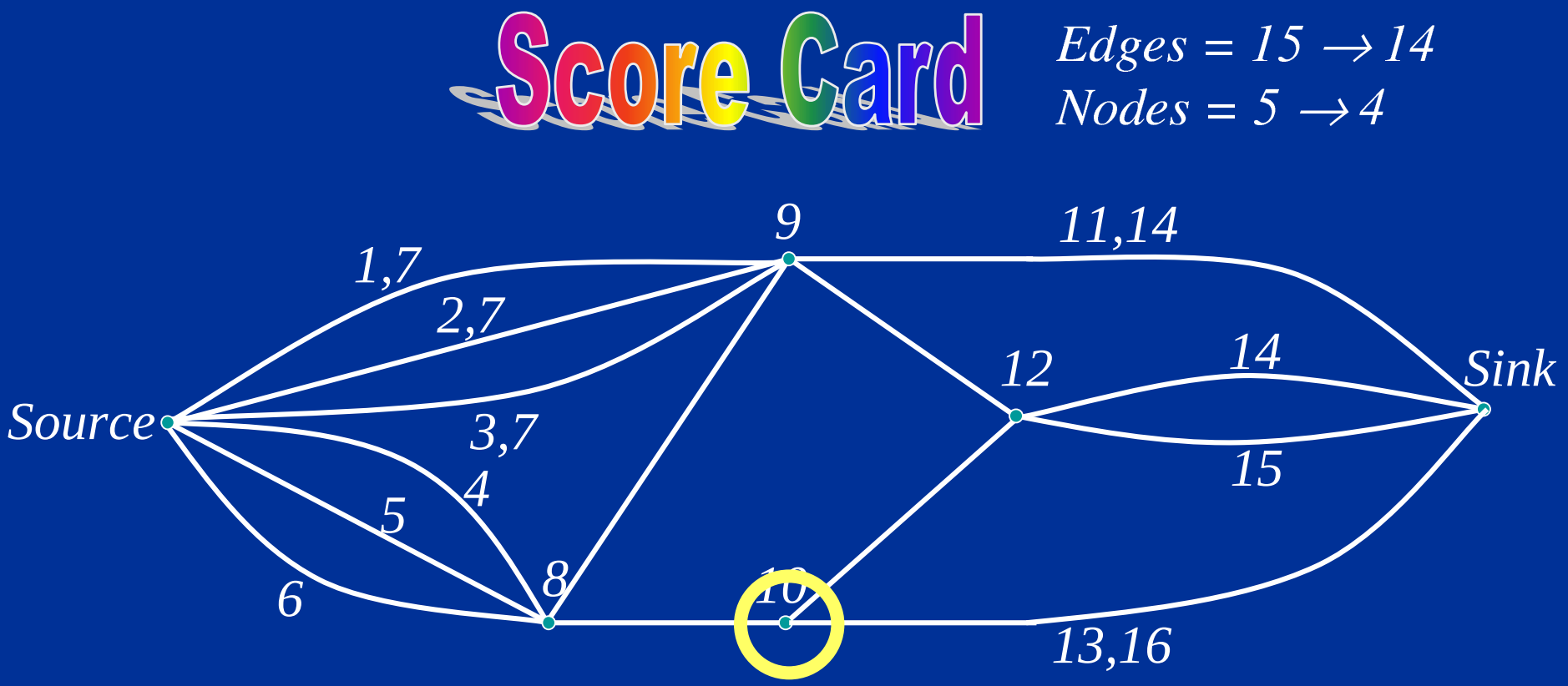
Detailed pruning example



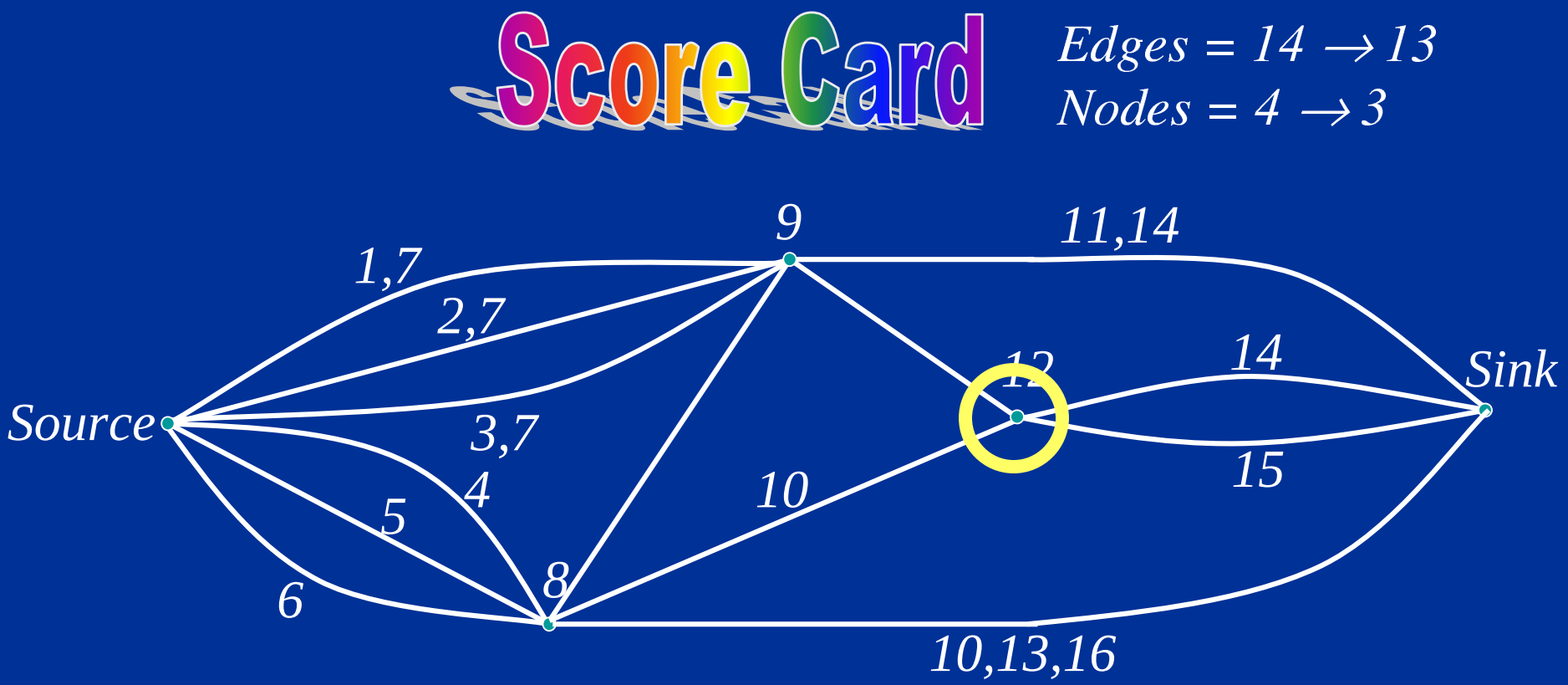
Detailed pruning example



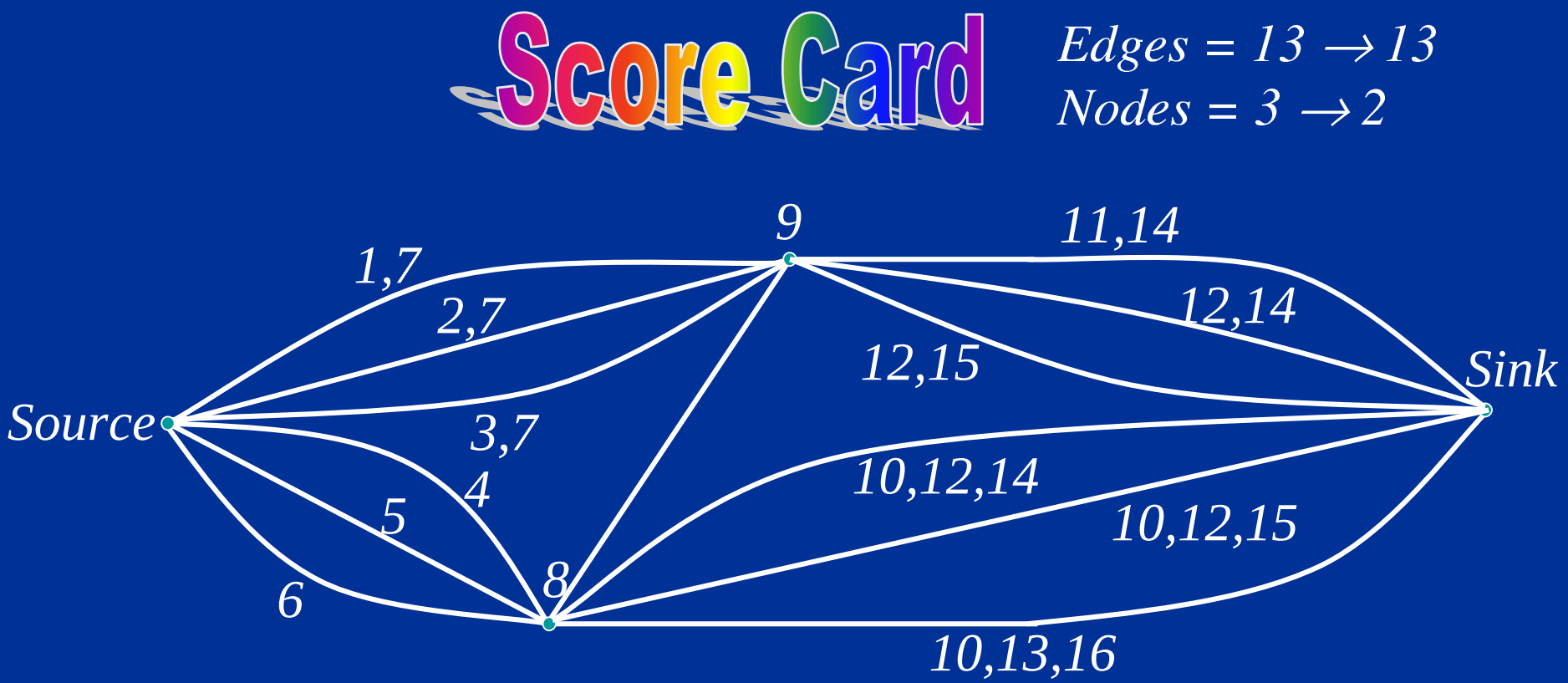
Detailed pruning example



Detailed pruning example



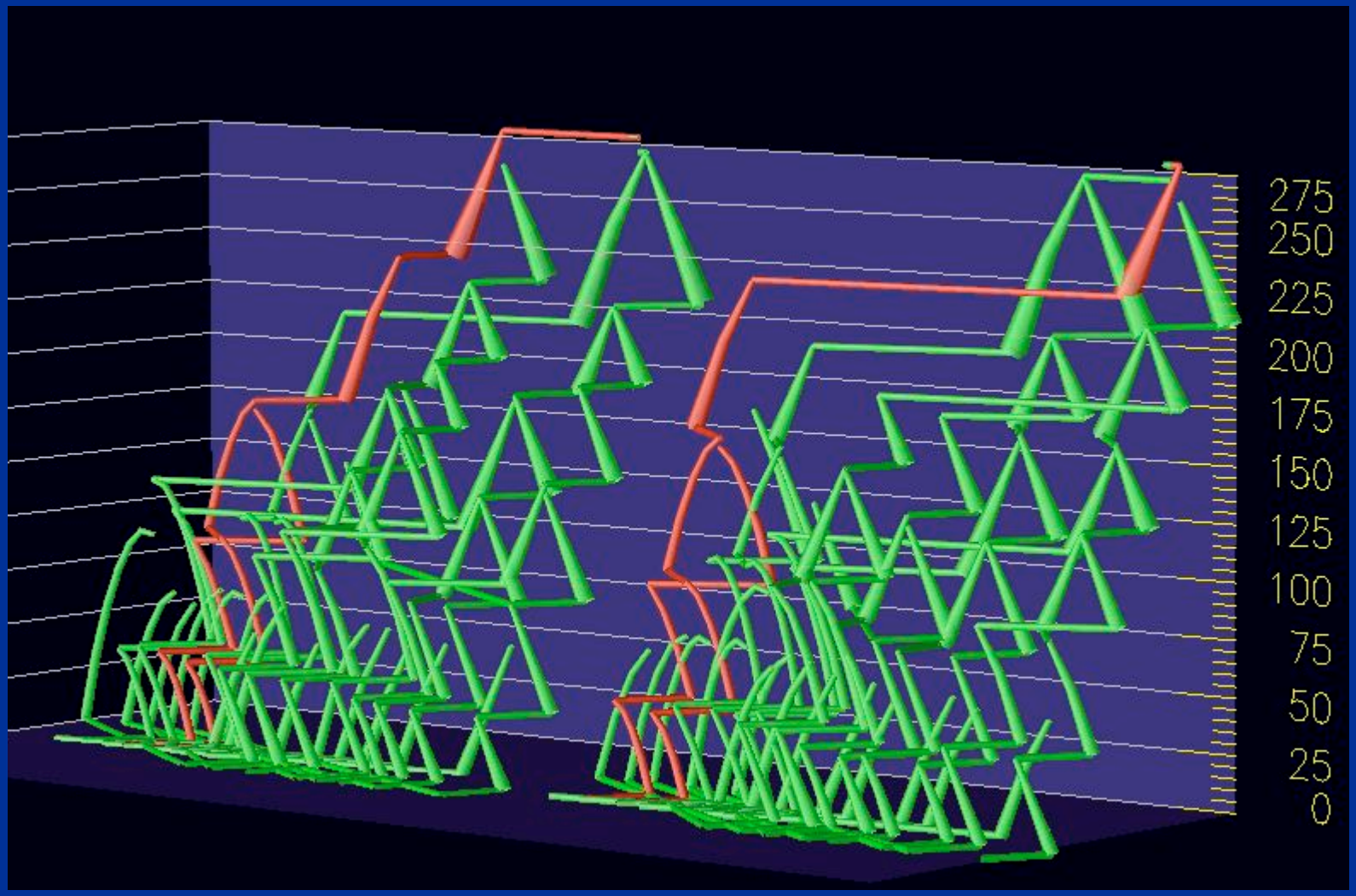
Detailed pruning example



Edges: 26 to 13 (2x)

Nodes: 16 to 2 (8x)

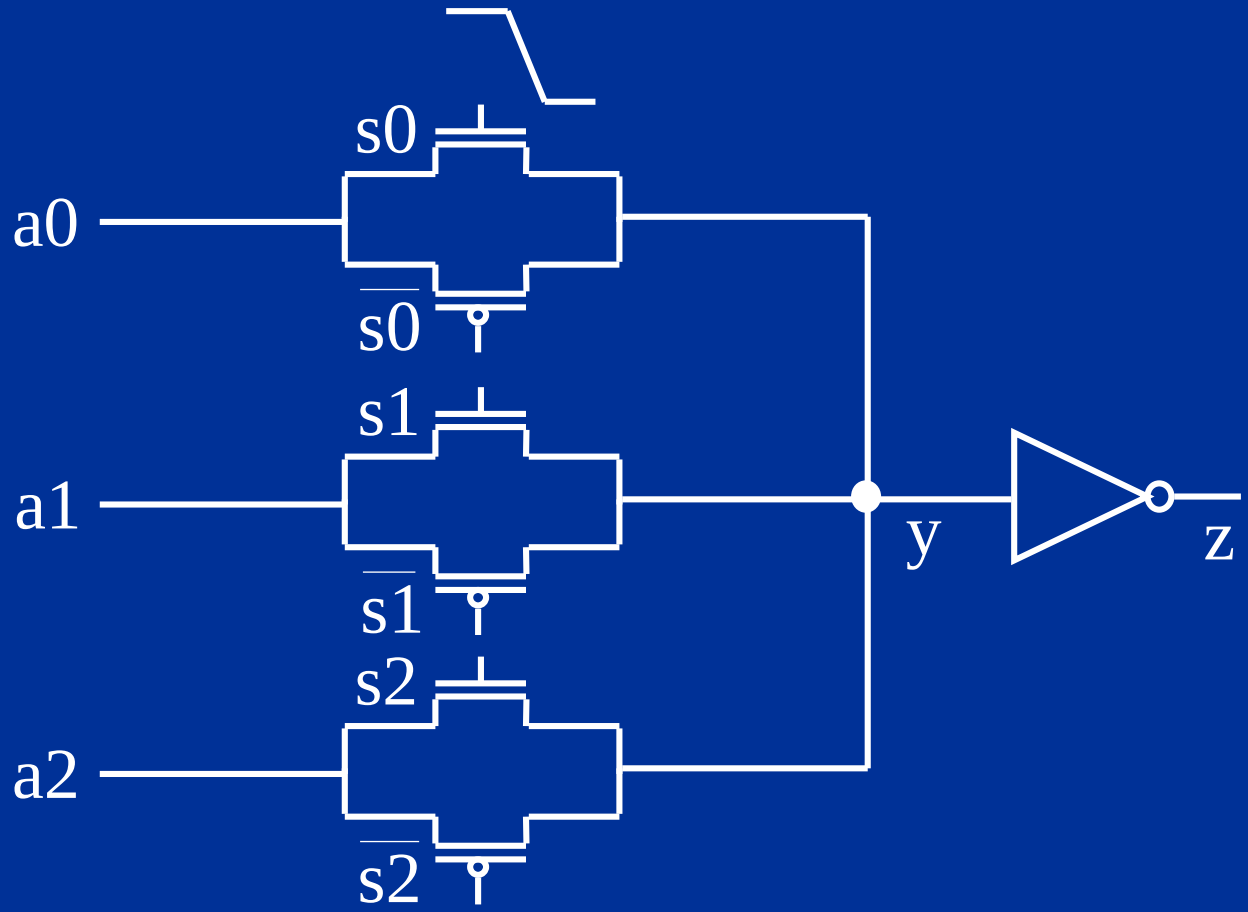
Pruning vs. no pruning



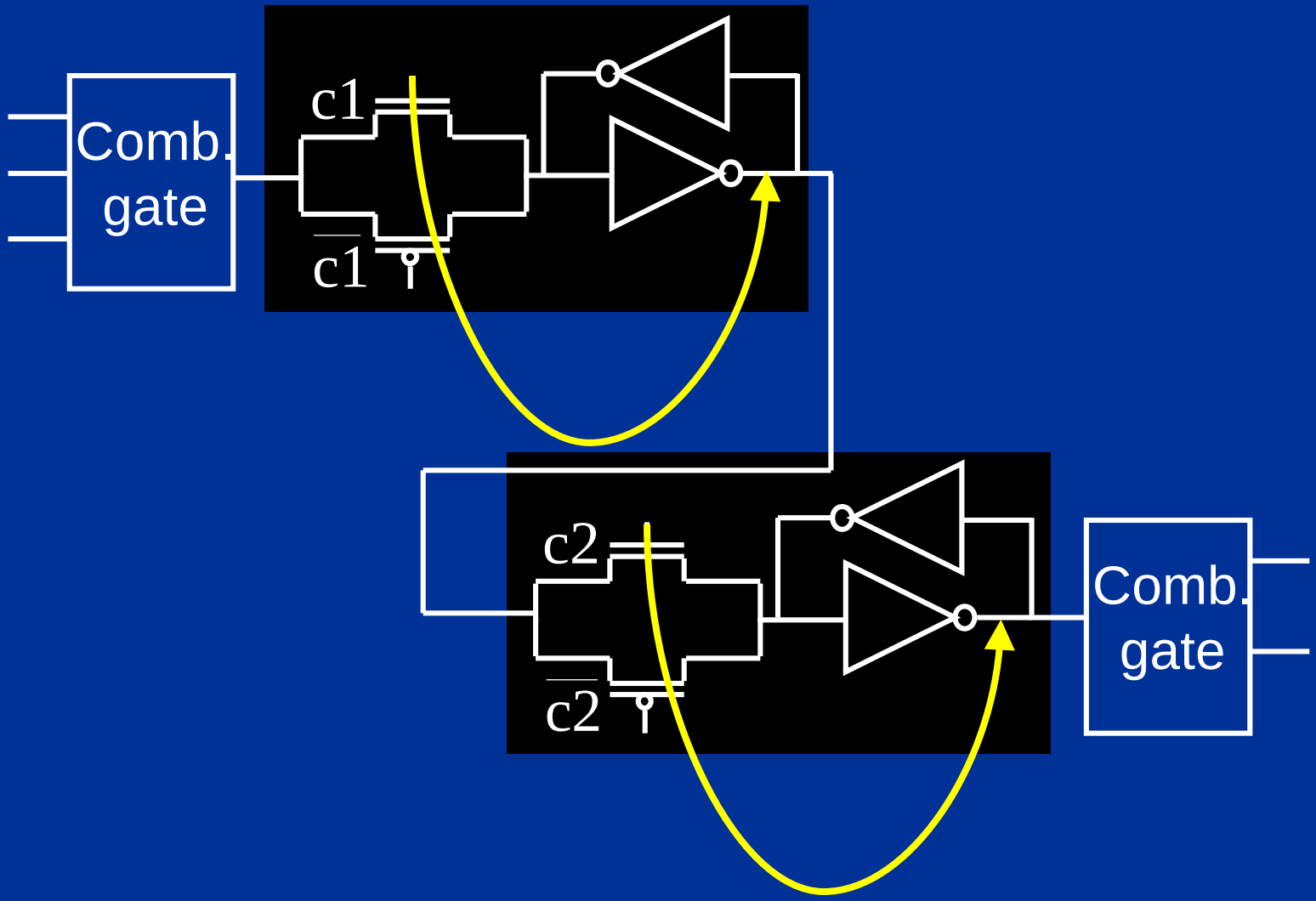
Group partial separability

- Nonlinear elements are:
  - delays
  - slews
  - β ratio constraints
- Each nonlinear element depends on a small number of problem variables
- Gradients of each nonlinear element computed once per iteration in their smaller subspaces
- Updating of approximate Hessian exploits the partial separability

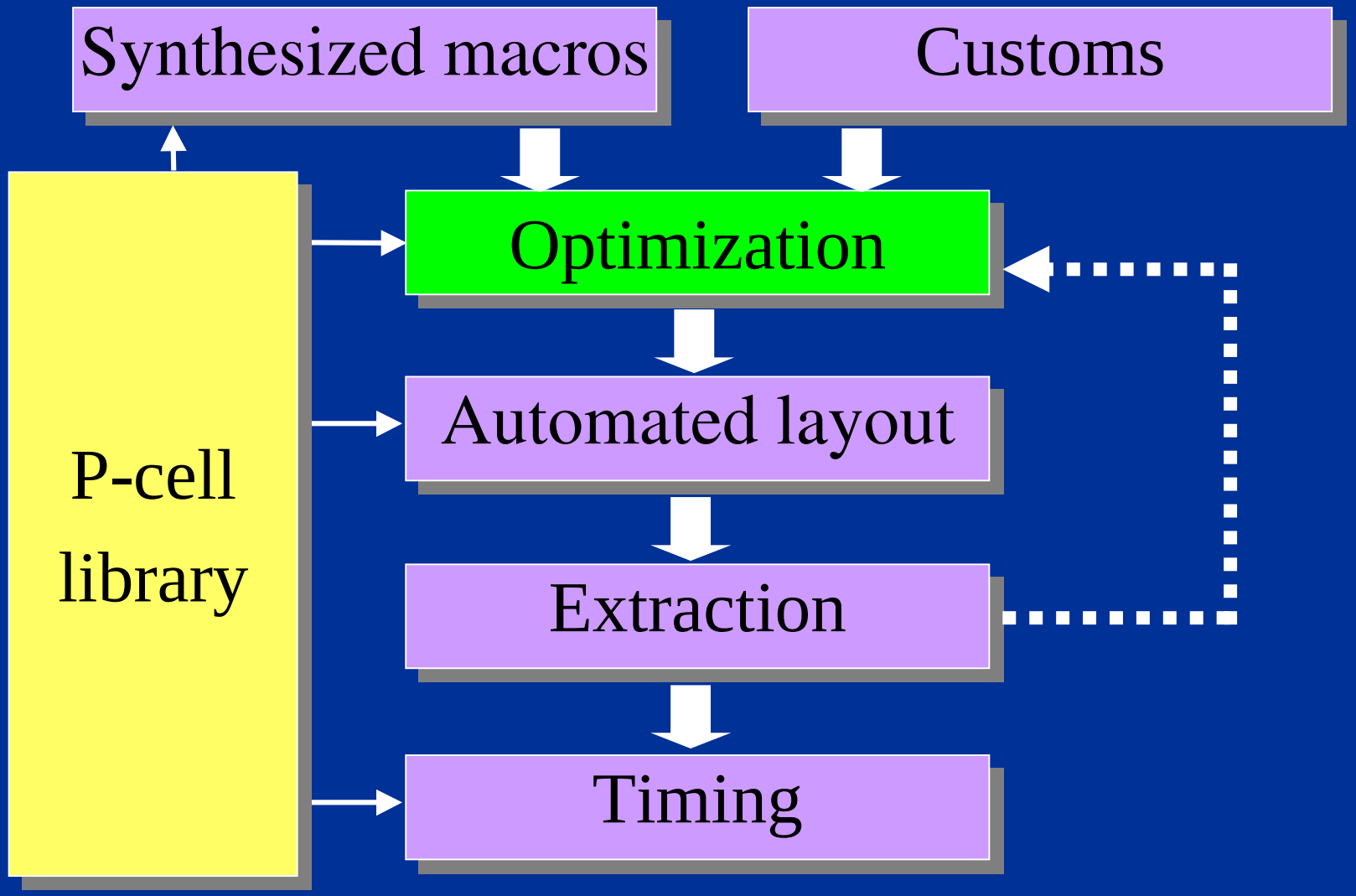
Special circuits: pass-gate MUXes



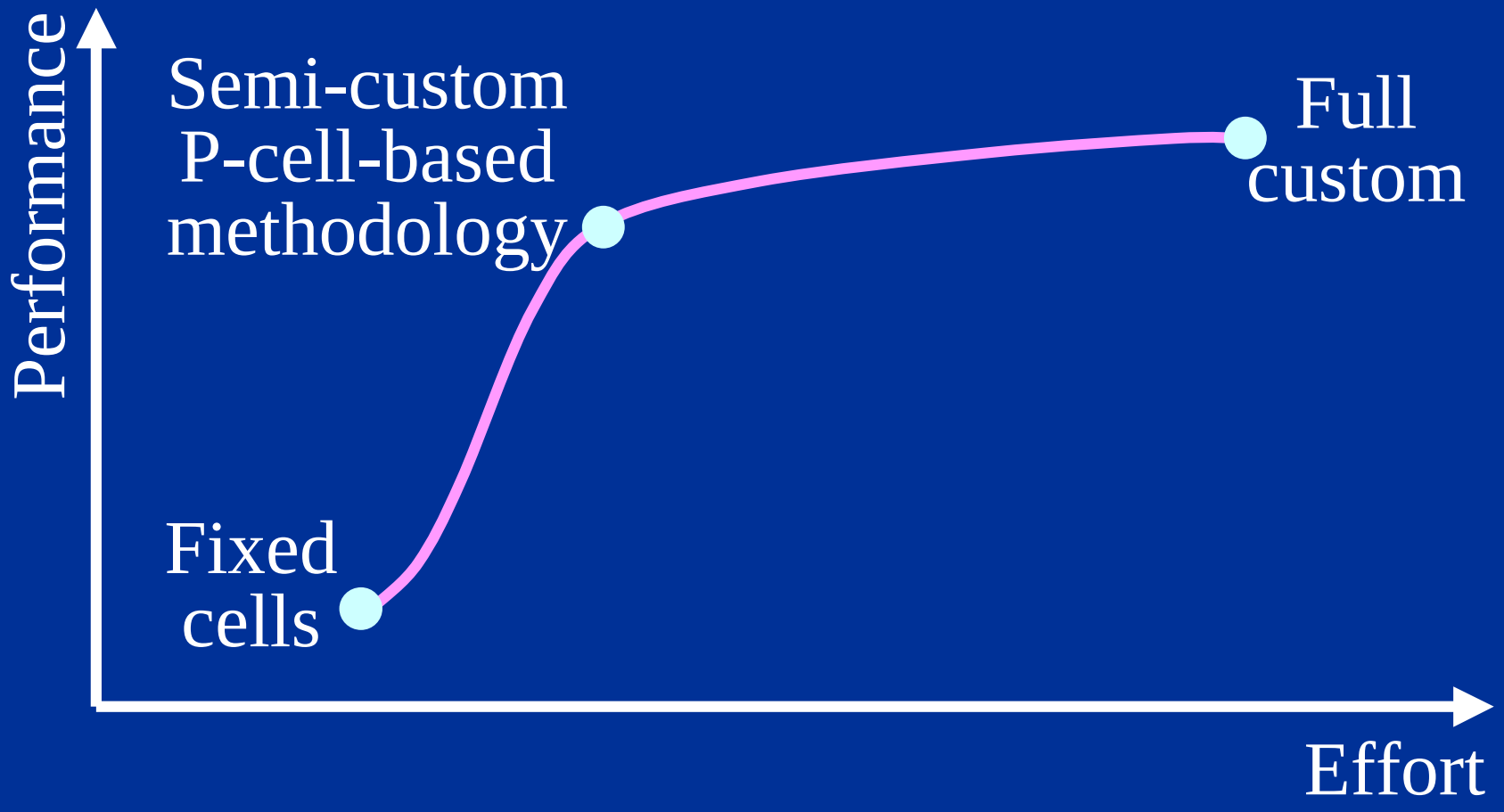
Latches and dynamic circuits



It's the layout, stupid!



Productive high-performance design



Conclusions

- Nonlinear optimization has made remarkable progress in the last 20 years
- Gradients are essential
- Do not treat the optimizer as a black box
- Static and dynamic tuners have been developed
  - 20,000 transistor capacity in either case
  - static tuner has long run times
  - optimal solutions; often 10%+ on previously hand-tuned designs
  - being used for 3rd generation of PowerPC and S/390 (z series) microprocessors