Constraint Programming for Compiler Optimizations

Peter van Beek, Abid Malik

October 2003

Optimization problems in compilers

Instruction selection

Register allocation

Instruction scheduling

- basic-block instruction scheduling
- software pipelining & loop unrolling
- trace scheduling

Interprocedural optimizations

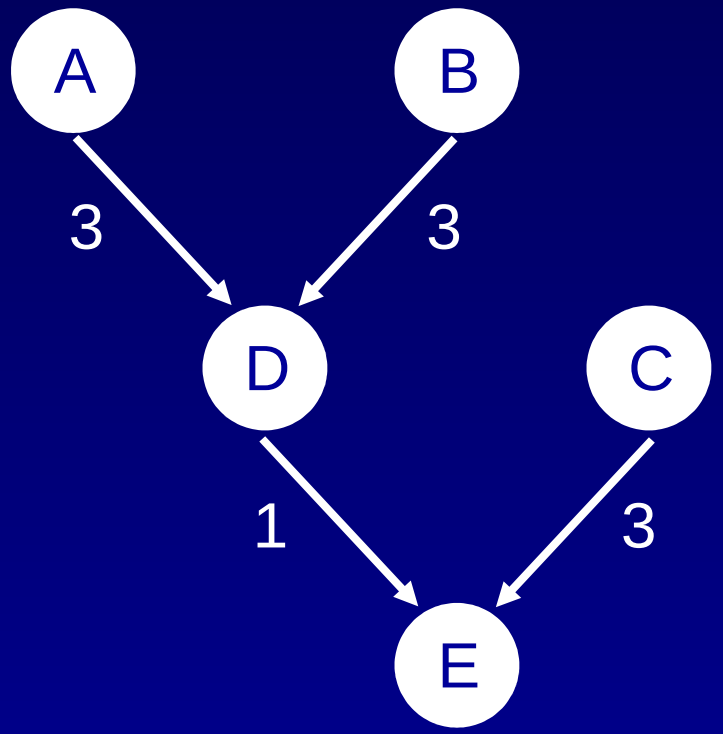
Memory hierarchy optimizations

Example: $(a + b) + c$

instructions

A	r1 $\leftarrow a$
B	r2 $\leftarrow b$
C	r3 $\leftarrow c$
D	r1 $\leftarrow$ r1 + r2
E	r1 $\leftarrow$ r1 + r3

dependency DAG

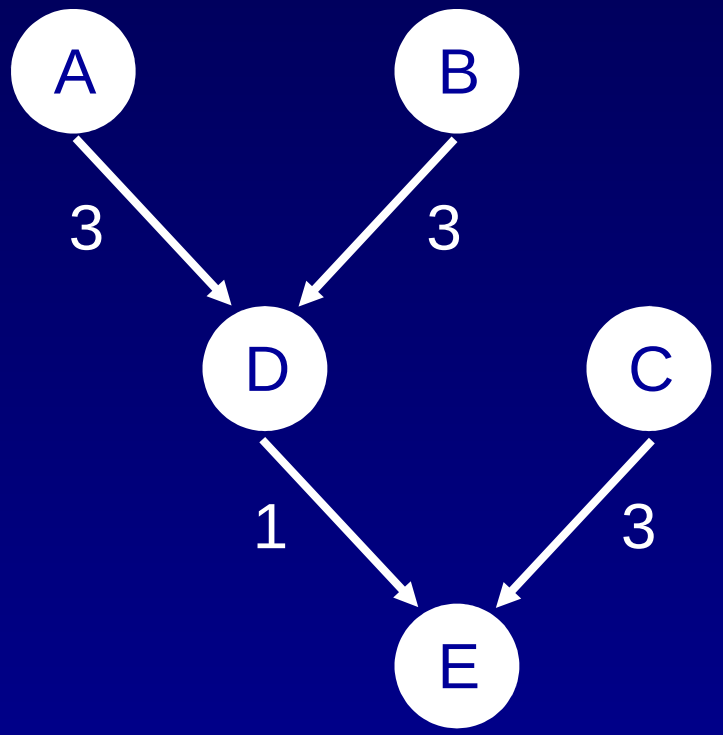


Single-issue pipelined processor

optimal schedule

A	r1 $\leftarrow a$
B	r2 $\leftarrow b$
C	r3 $\leftarrow c$
	nop
D	r1 $\leftarrow$ r1 + r2
E	r1 $\leftarrow$ r1 + r3

dependency DAG



Production compilers

“ At the outset, note that basic-block scheduling is an NP-hard problem, even with a very simple formulation of the problem, so we must seek an effective heuristic, rather than exact, approach.”

Steven Muchnick,

Advanced Compiler Design & Implementation, 1997

Constraint programming methodology

Model problem

- specify in terms of constraints on acceptable solutions
- define/choose constraint model:
  variables, domains, constraints

Solve model

- define/choose search algorithm
- define/choose heuristics

Verify and analyze solution

Minimal constraint model

variables

A, B, C, D, E

domains

{1, .., *m*}

constraints

D ≥ A + 3

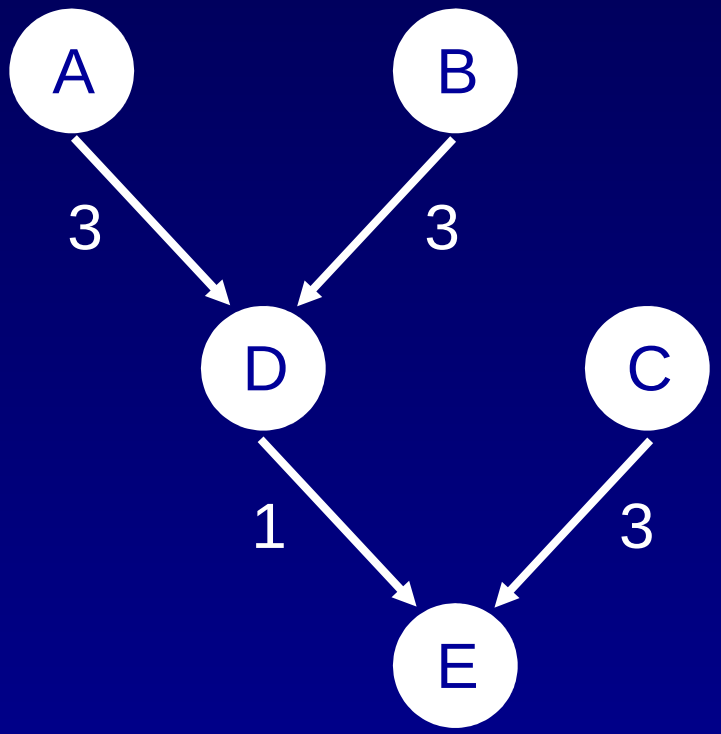
D ≥ B + 3

E ≥ C + 3

E ≥ D + 1

all-diff(A, B, C, D, E)

dependency DAG



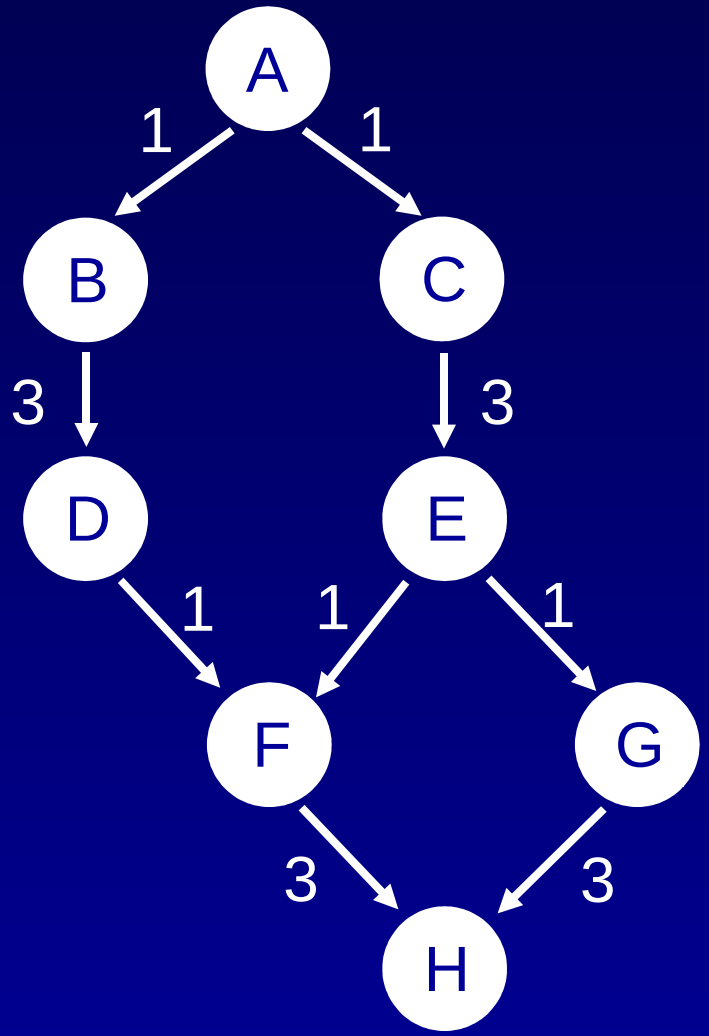
Constraint propagation

variable	*domain*			*constraints*
A	[1, 6]	⇒ [1, 3]	⇒ [1, 2]	D ≥ A + 3
B	[1, 6]	⇒ [1, 2]		D ≥ B + 3
C	[1, 6]	⇒ [3, 3]		E ≥ C + 3
D	[1, 6]	⇒ [4, 6]	⇒ [4, 5]	E ≥ D + 1
E	[1, 6]	⇒ [6, 6]		all-diff(A, B, C, D, E)

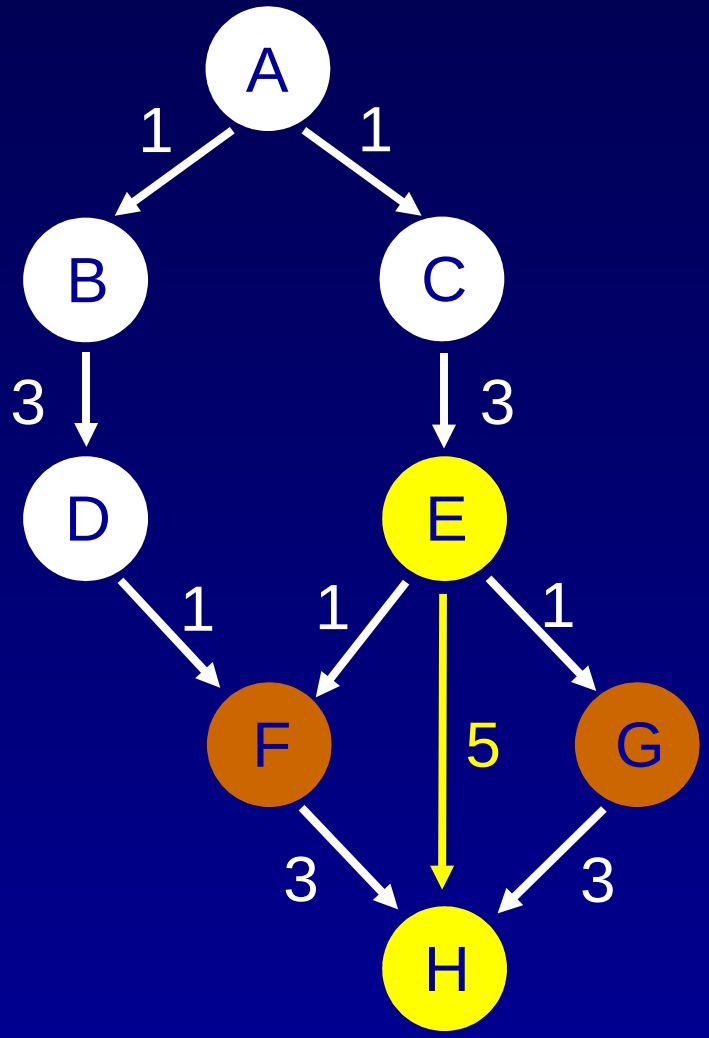
Two improvements to minimal model

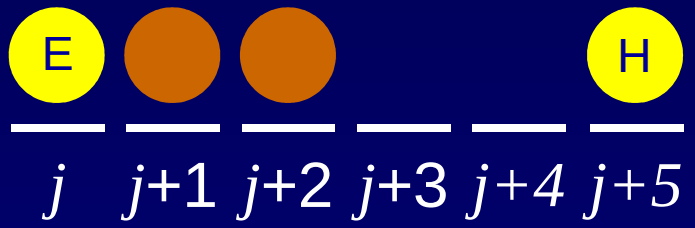
1. Distance constraints
   - defined over nodes which define regions
2. Predecessor and successor constraints
   - defined over nodes with multiple predecessors or multiple successors

Distance constraints

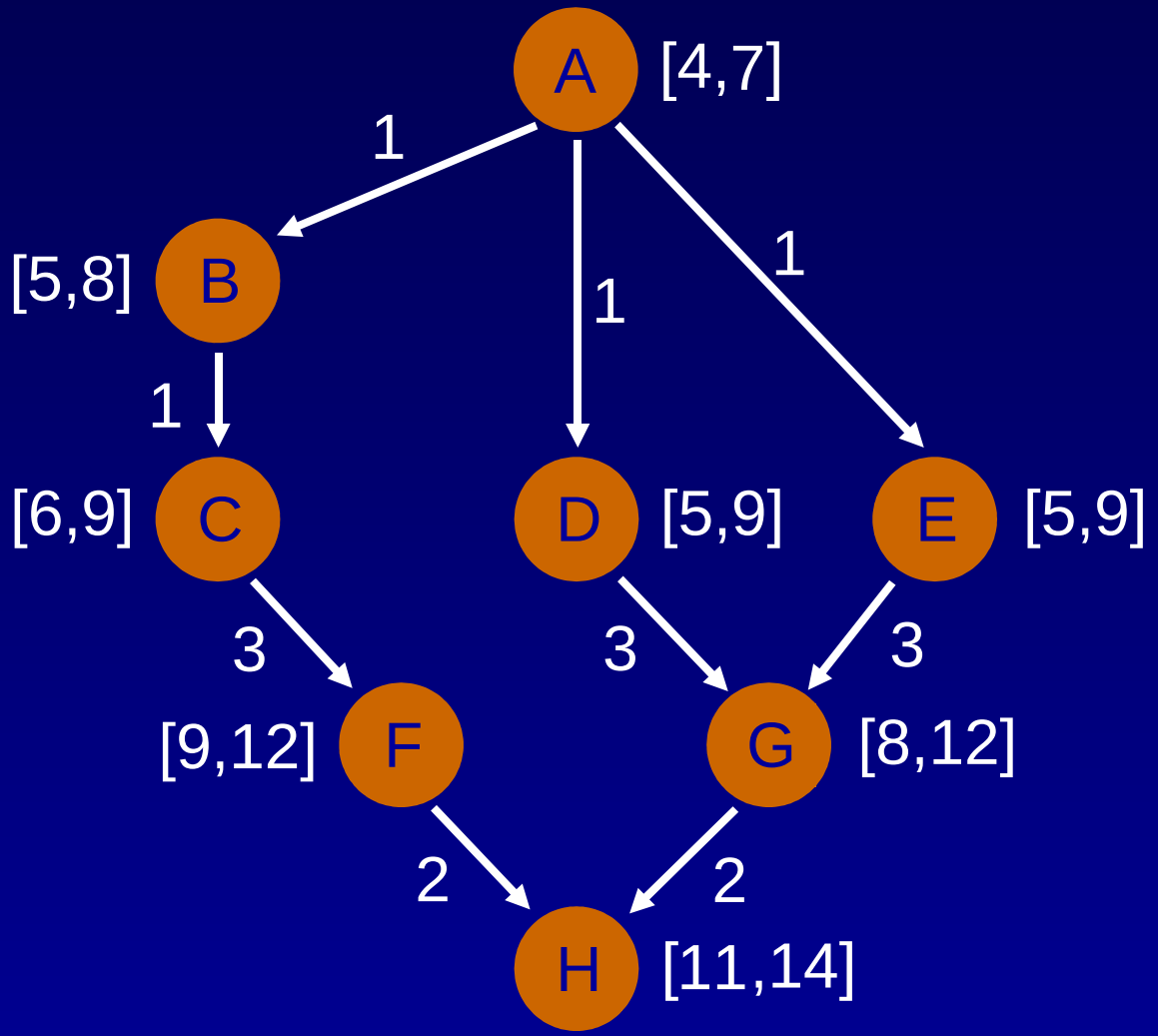


Distance constraints

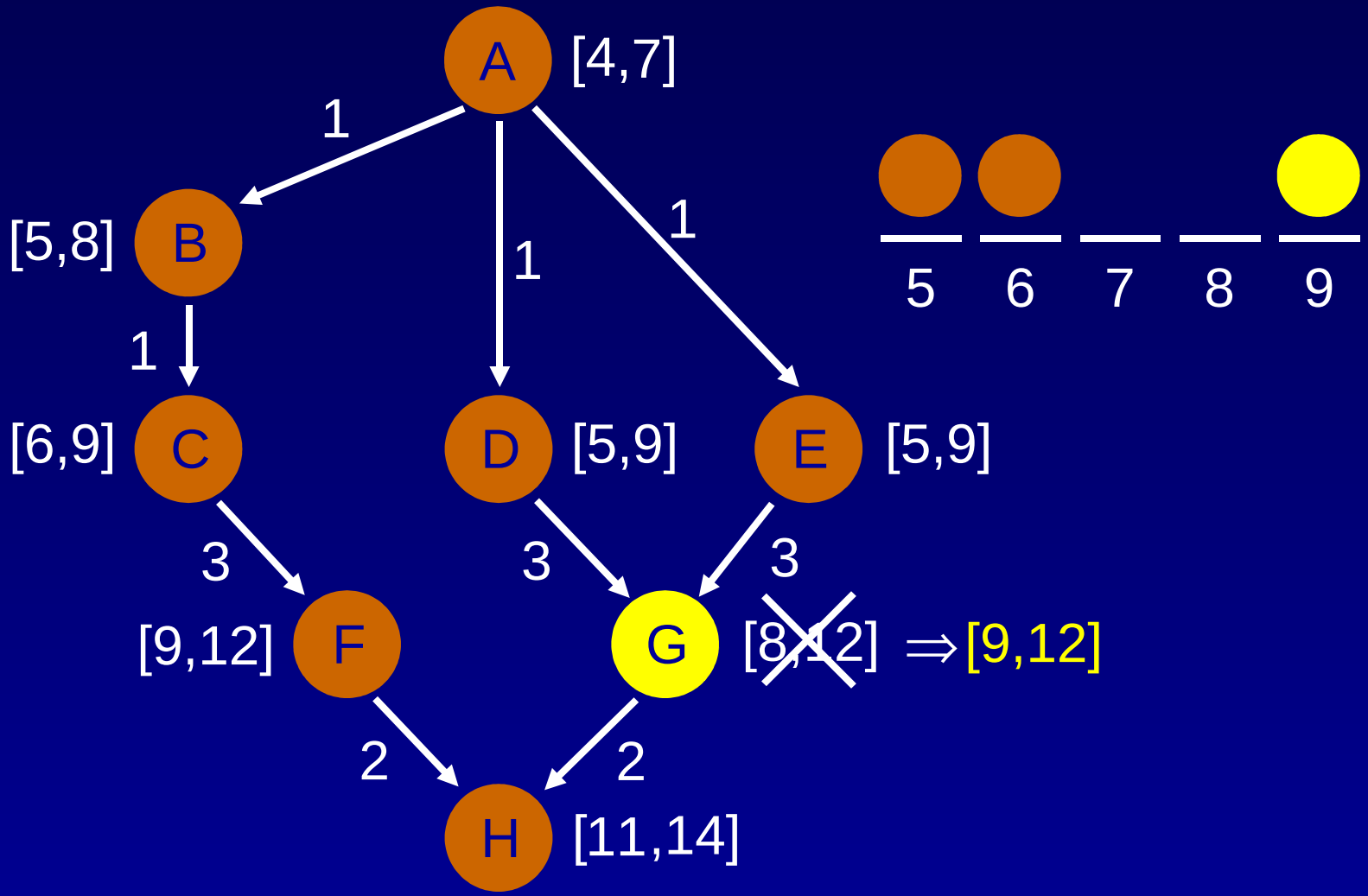




Predecessor constraints



Predecessor constraints



Solving instances of the model

Use constraints to establish:

- lower bound on length m of optimal schedule
- min and max of domains of variables

Backtracking search

- branches on $min(x)$, $min(x)+1$, …
- interleave with constraint propagation

If no solution found, increment m and repeat search

Putting it all together: Experimental results

Time (sec.) to solve instruction scheduling problems for various widths; model includes all constraints.

n	1	1+ 1	1+2	2+1	2+2
69	0.00	0.02	0.02	0.02	0.00
70	0.00	0.02	0.02	0.00	0.00
111	0.03	0.02	0.02	0.02	0.02
211	0.03	0.03	0.02	0.03	0.03
214	0.03	0.03	0.03	0.03	0.03
216	0.03	0.03	0.02	0.02	0.02
220	0.02	0.02	0.03	0.03	0.03
377	0.11	0.14	0.17	0.06	0.06
381	0.05	0.05	0.05	0.03	0.03
394	0.09	0.20	0.44	0.05	0.05
556	0.25	0.23	0.34	0.13	0.13
690	0.17	0.19	0.19	0.17	0.17
691	0.28	0.52	0.66	0.23	0.19
856	0.66	392.53	287.78	362.11	239.84
1006	0.39	0.48	0.41	243.17	0.39