
Efficient Memoization for Dynamic Programming with Ad-Hoc Constraints

Joxan Jaffar, Andrew E. Santosa, Răzvan Voicu

{joxan, andrews, razvan}@comp.nus.edu.sg

School of Computing

National University of Singapore

Introduction

Some combinatorial optimization problems can be represented as *dynamic programs*, but not necessarily resulting in efficient solving.

Some characteristics of dynamic programming:

- *Memoization* to reuse solved subproblems
- Memoization requires *identification* of similarity

Our Solution: *Generalize* the memoed subproblem using *interpolation* such that more subproblems can be identified similar *without losing precision*

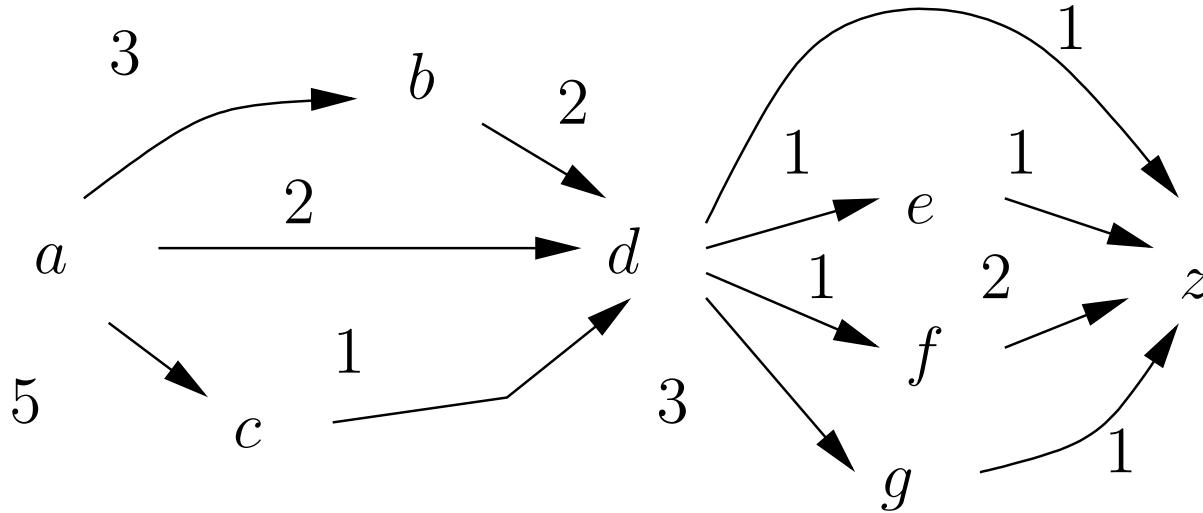
Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Shortest Path



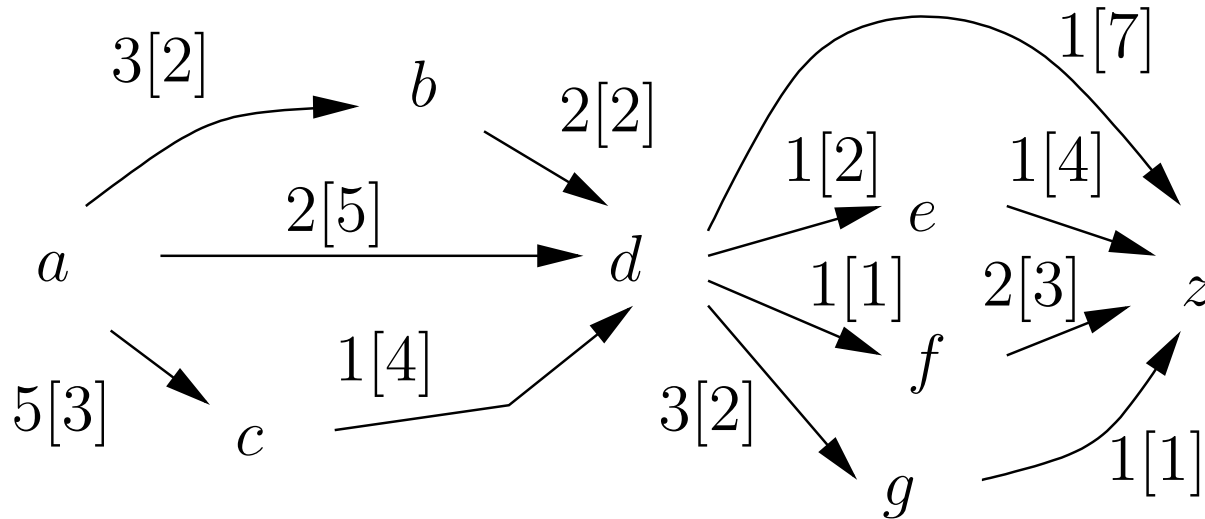
- Edge has a “cost”
- Find the path from a to z with least total cost

Shortest Path

$$f(X) = \begin{cases} \text{if } \textit{destination}(X) \text{ then } 0 \\ \text{else } \min\{f(Y) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\} \end{cases}$$

- \mathcal{E} : Set of edges.
- $c_{X,Y}$: Cost of edge (X, Y) .
- Dynamic programming is efficient: search tree size that is linear to the number of graph vertices.
- Uses memoization: solution to $f(v)$ is reused when $f(v)$ is subsequently encountered.

RCSP



- $c[r]$: $c = \text{cost}$, $r = \text{resource consumed}$.
- Find the shortest path from a to z where resource consumption ≤ 10 .

RCSP

Resource-Constrained Shortest Path: NP-Complete

$$f(X, \tilde{R}) = \mathbf{if} \ \tilde{R} > \tilde{u} \ \mathbf{then} \ \infty \\ \mathbf{else if} \ \mathit{destination}(X) \ \mathbf{then} \ 0 \\ \mathbf{else} \ \mathit{min}\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$$

- \tilde{R} : Vector of resource values.
- \tilde{u} : Resource bounds.
- $\tilde{r}_{X,Y}$: Resource consumed along edge (X, Y) .
- Resource bound is an *ad-hoc constraint* added to the original shortest path problem.

Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Identifying Similarity

$f(X, \tilde{R}) =$ **if** $\tilde{R} > \tilde{u}$ **then** ∞
else if $\textit{destination}(X)$ **then** 0
else $\min\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

Identifying Similarity

$f(X, \tilde{R}) = \mathbf{if} \tilde{R} > \tilde{u} \mathbf{then} \infty$
 $\mathbf{else if} \textit{destination}(X) \mathbf{then} 0$
 $\mathbf{else} \min\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

- Obviously so, when $v = w$ and $\tilde{r} = \tilde{s}$
Same set of solutions

Identifying Similarity

$f(X, \tilde{R}) = \mathbf{if} \tilde{R} > \tilde{u} \mathbf{then} \infty$

$\mathbf{else if} \textit{destination}(X) \mathbf{then} 0$

$\mathbf{else} \min\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

- Obviously so, when $v = w$ and $\tilde{r} = \tilde{s}$

Same set of solutions

- More generally:

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$

$f(v, \tilde{r})$ violates bounds more often, hence

Solutions of $f(v, \tilde{r}) \subseteq$ solutions of $f(w, \tilde{s})$

$\rightarrow f(v, \tilde{r})$ has no better optimal

Identifying Similarity

$f(X, \tilde{R}) = \mathbf{if} \tilde{R} > \tilde{u} \mathbf{then} \infty$

else if *destination*(X) **then** 0

else $\min\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

- Obviously so, when $v = w$ and $\tilde{r} = \tilde{s}$

Same set of solutions

- More generally:

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$

$f(v, \tilde{r})$ violates bounds more often, hence

Solutions of $f(v, \tilde{r}) \subseteq$ solutions of $f(w, \tilde{s})$

$\rightarrow f(v, \tilde{r})$ has no better optimal

2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$

Sharing the same optimal

Generalizing Context

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

Generalizing Context

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

Generalize “ $f(w, \tilde{s})$ ” in the memo table to “ $f(w, \tilde{S}), \tilde{S} \geq \tilde{t}$ ”
having \tilde{s} for \tilde{S} satisfying $\tilde{S} \geq \tilde{t}$ (that is, $\tilde{s} \geq \tilde{t}$) **while ensuring the same set of solutions**

Expected result:

More $f(v, \tilde{r})$ can reuse the result of $f(w, \tilde{S}), \tilde{S} \geq \tilde{t}$ due to satisfying condition 1: $v = w$ and $\tilde{r} \geq \tilde{t}$

Generalizing Context

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

Generalize “ $f(w, \tilde{s})$ ” in the memo table to “ $f(w, \tilde{S}), \tilde{S} \geq \tilde{t}$ ”
having \tilde{s} for \tilde{S} satisfying $\tilde{S} \geq \tilde{t}$ (that is, $\tilde{s} \geq \tilde{t}$) **while ensuring the same set of solutions**

Expected result:

More $f(v, \tilde{r})$ can reuse the result of $f(w, \tilde{S}), \tilde{S} \geq \tilde{t}$ due to satisfying condition 1: $v = w$ and $\tilde{r} \geq \tilde{t}$

HOW TO GENERALIZE?

Interpolant (Definition)

If F and G are formulas such that F entails G , then there exists an *interpolant* $int(F, G)$ which is a formula such that

$$F \Rightarrow int(F, G) \Rightarrow G,$$

and each parameter of $int(F, G)$ is a parameter of both F and G . [Craig 1955]

DP as Transition System

$$f(X, \tilde{R}) = \text{if } \tilde{R} > \tilde{u} \text{ then } \infty \\ \text{else if } \textit{destination}(X) \text{ then } 0 \\ \text{else } \min\{f(Y, \tilde{R} + \tilde{r}_{X,Y}) + c_{X,Y} \mid (X, Y) \in \mathcal{E}\}$$

- Transition relation (for $(x, y) \in \mathcal{E}$):

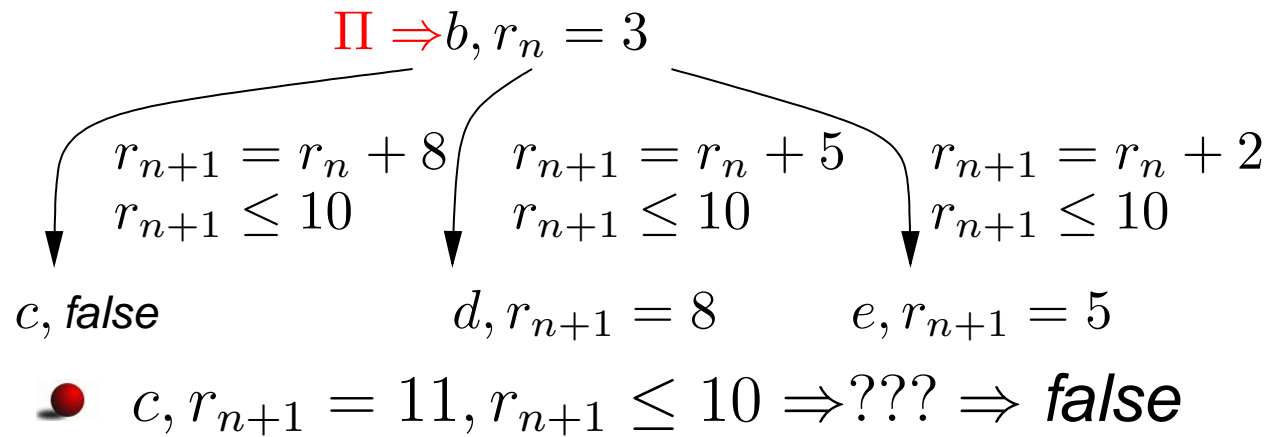
$$\rho(X, \tilde{R}, X', \tilde{R}') \equiv X = x \wedge X' = y \wedge \tilde{R}' \leq \tilde{u} \wedge \tilde{R}' = \tilde{R} + \tilde{r}_{x,y}$$

- Initial state: $\Theta(X, \tilde{R}) \equiv X = s \wedge \tilde{R} = \tilde{0}$ for s the source
- Final condition: $\varphi(X, \tilde{R}) \equiv X = t$ for t the destination

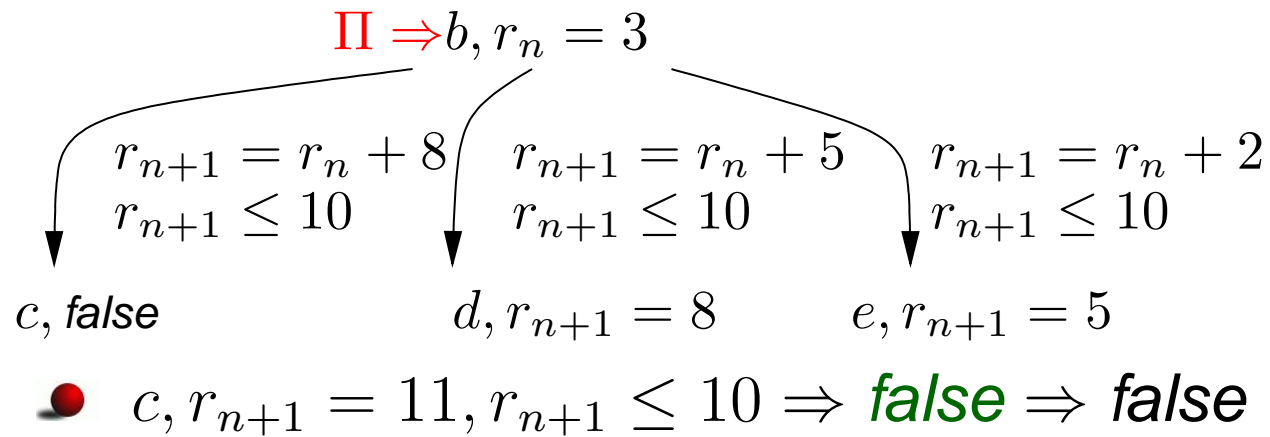
Trace of length m *identifying subproblem* $f(X_m, \tilde{R}_m)$:

$$\Pi \equiv \Theta(X_0, \tilde{R}_0) \wedge \bigwedge_{i=0}^{m-1} \rho_{i+1}(X_i, \tilde{R}_i, X_{i+1}, \tilde{R}_{i+1})$$

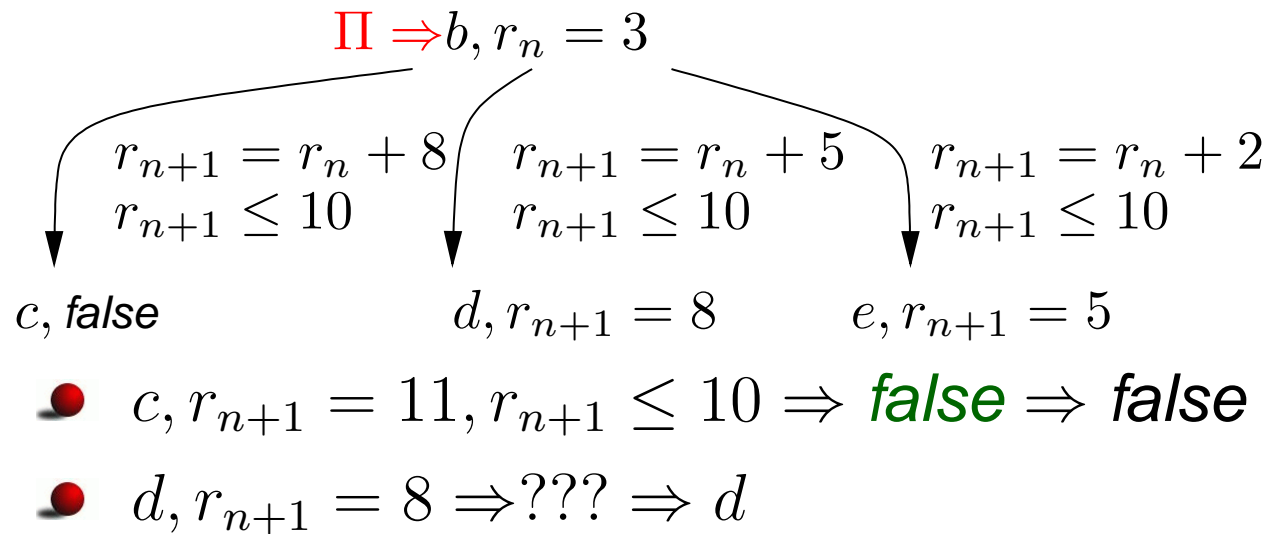
Interpolation



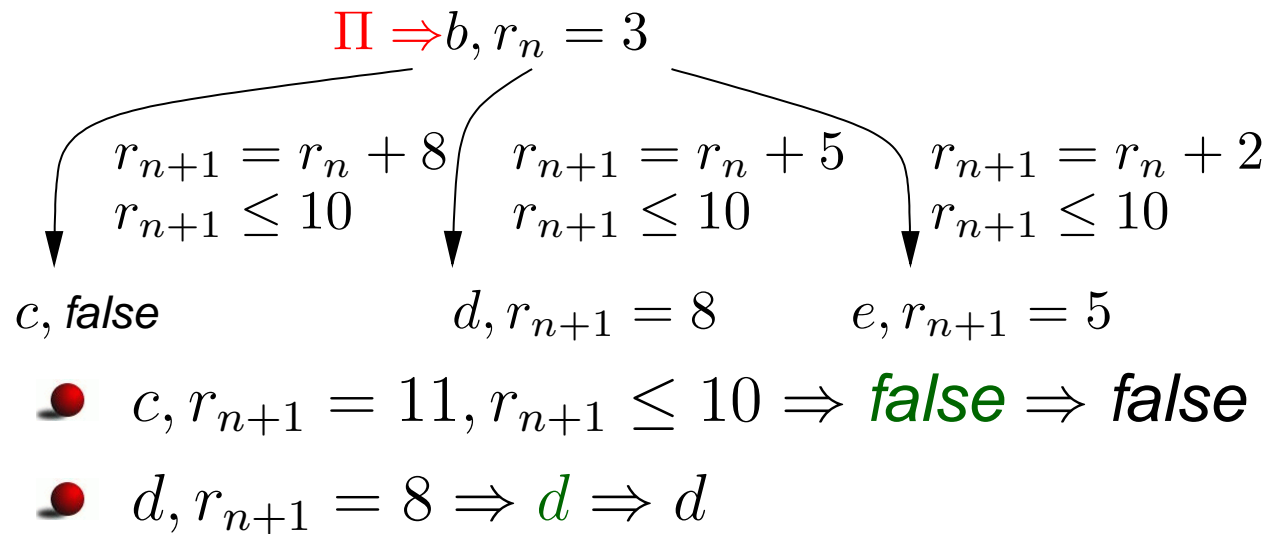
Interpolation



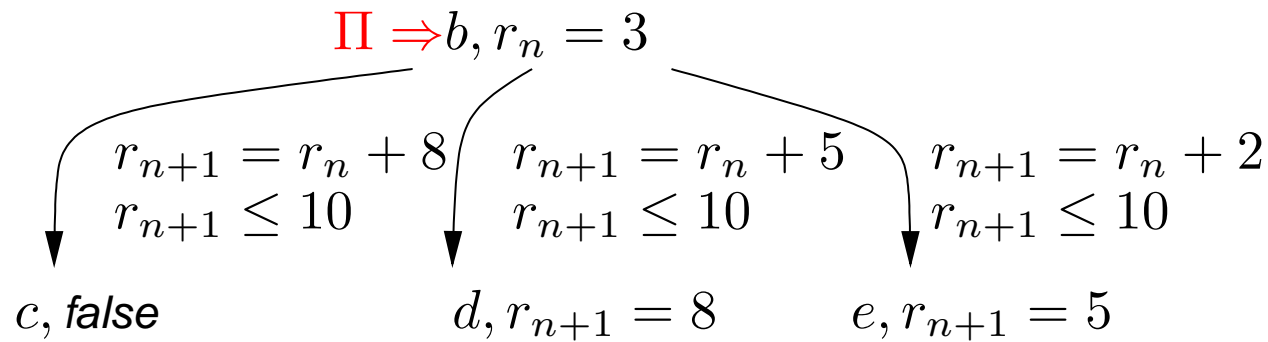
Interpolation



Interpolation



Interpolation

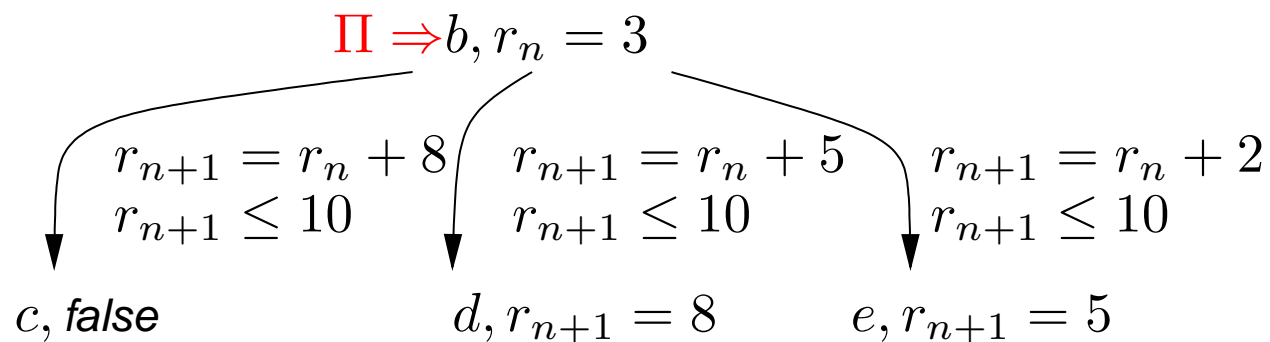


● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow ??? \Rightarrow e, r_{n+1} \geq 3$

Interpolation

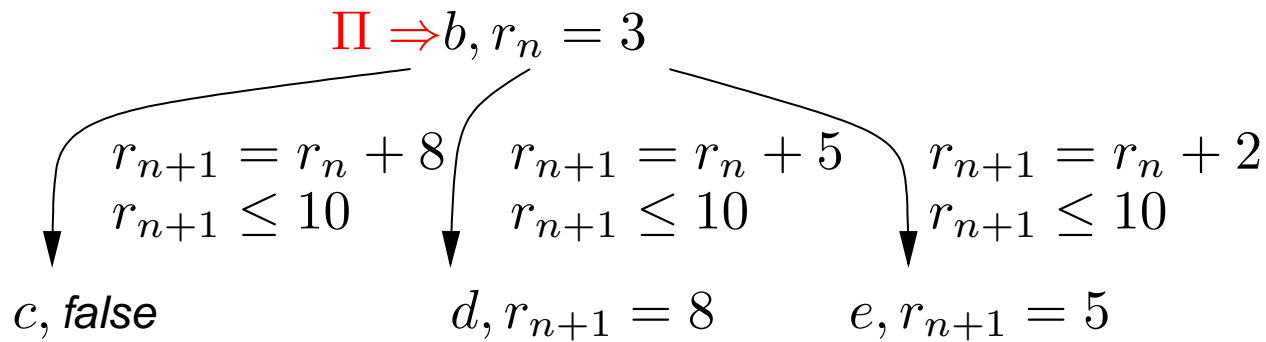


● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

Interpolation



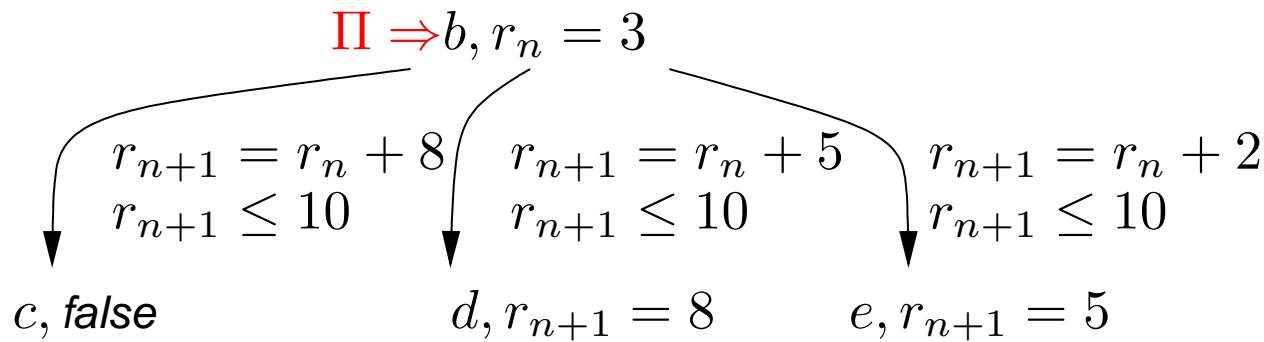
● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow ??? \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

Interpolation



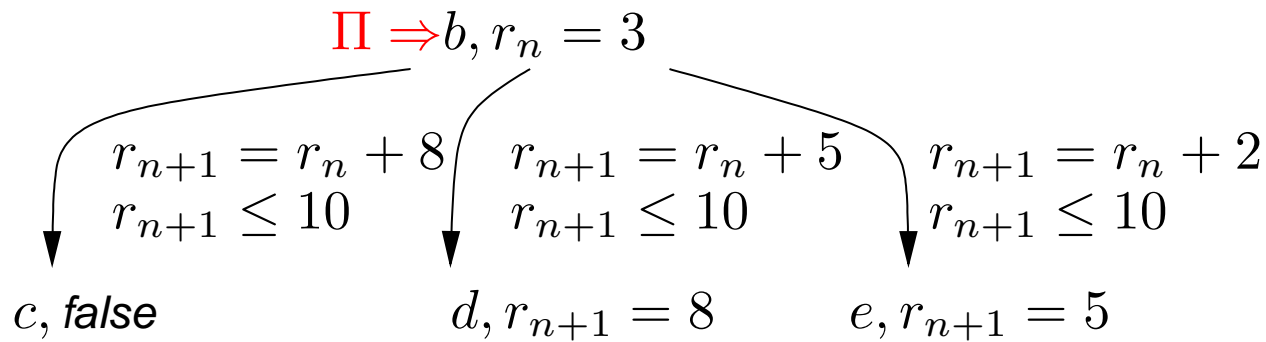
● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

Interpolation



● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

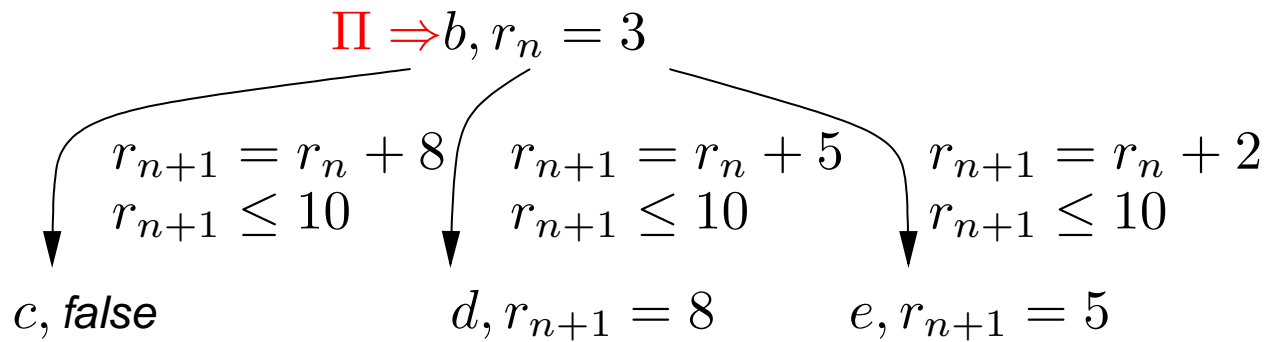
● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

● $b, r_n = 3 \Rightarrow ??? \Rightarrow (b, d, r_{n+1} = r_n + 5, r_{n+1} \leq 10 \Rightarrow d)$

Interpolation



● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

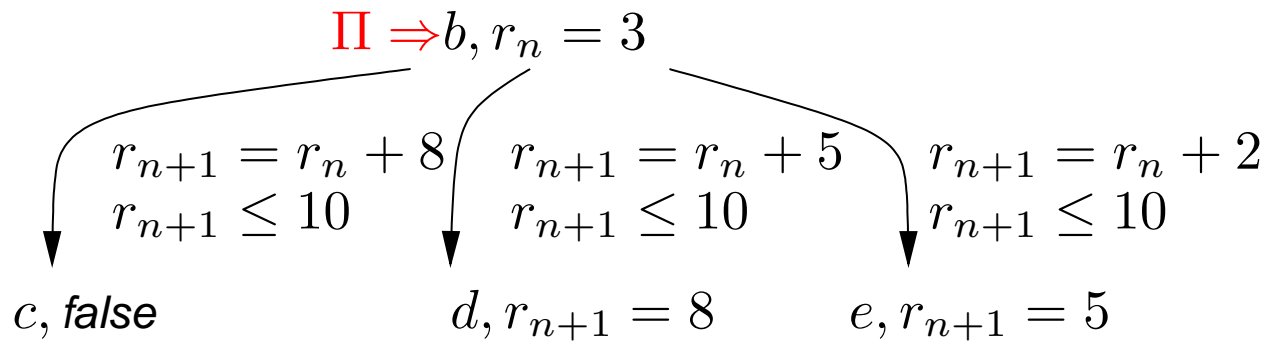
● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

● $b, r_n = 3 \Rightarrow b \Rightarrow (b, d, r_{n+1} = r_n + 5, r_{n+1} \leq 10 \Rightarrow d)$

Interpolation



● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

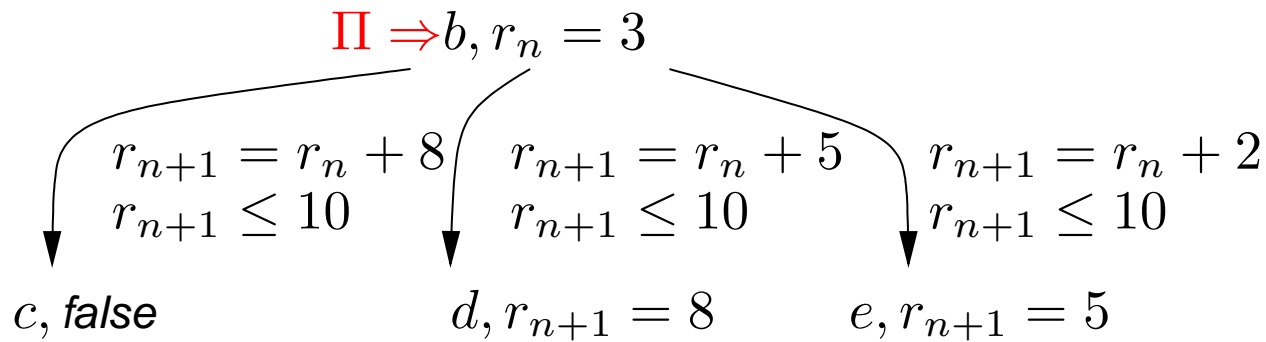
● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

● $b, r_n = 3 \Rightarrow b \Rightarrow (b, d, r_{n+1} = r_n + 5, r_{n+1} \leq 10 \Rightarrow d)$

● $b, r_n = 3 \Rightarrow ??? \Rightarrow (b, e, r_{n+1} = r_n + 2, r_{n+1} \leq 10 \Rightarrow e, r_{n+1} \geq 3)$

Interpolation



● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

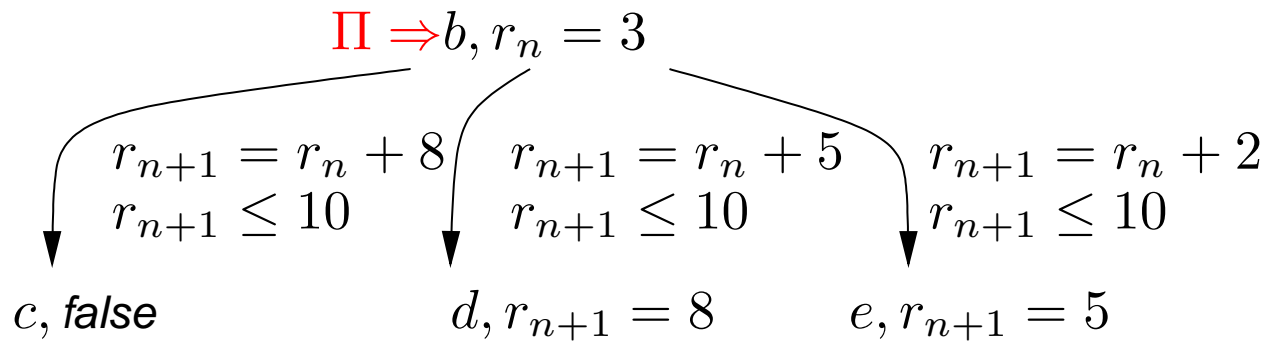
● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

● $b, r_n = 3 \Rightarrow b \Rightarrow (b, d, r_{n+1} = r_n + 5, r_{n+1} \leq 10 \Rightarrow d)$

● $b, r_n = 3 \Rightarrow b, r_n \geq 1 \Rightarrow (b, e, r_{n+1} = r_n + 2, r_{n+1} \leq 10 \Rightarrow e, r_{n+1} \geq 3)$

Interpolation



● $c, r_{n+1} = 11, r_{n+1} \leq 10 \Rightarrow \text{false} \Rightarrow \text{false}$

● $d, r_{n+1} = 8 \Rightarrow d \Rightarrow d$

● $e, r_{n+1} = 5 \Rightarrow e, r_{n+1} \geq 3 \Rightarrow e, r_{n+1} \geq 3$

● $b, r_n = 3 \Rightarrow b, r_n \geq 3 \Rightarrow (b, c, r_{n+1} = r_n + 8, r_{n+1} \leq 10 \Rightarrow \text{false})$

● $b, r_n = 3 \Rightarrow b \Rightarrow (b, d, r_{n+1} = r_n + 5, r_{n+1} \leq 10 \Rightarrow d)$

● $b, r_n = 3 \Rightarrow b, r_n \geq 1 \Rightarrow (b, e, r_{n+1} = r_n + 2, r_{n+1} \leq 10 \Rightarrow e, r_{n+1} \geq 3)$

● $b, r_n \geq 3 \wedge b \wedge b, r_n \geq 1 \equiv b, r_n \geq 3$

Storing optimal

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

We have discussed about Condition 1.

Storing optimal

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

We have discussed about Condition 1.

For Condition 2, we simply store (at most) some k best solutions of $f(w, \tilde{s})$ (typically $k = 1$)

Storing optimal

Can $f(v, \tilde{r})$ reuse the result for $f(w, \tilde{s})$?

1. when $v = w$ and $\tilde{r} \geq \tilde{s}$,
2. the optimal for $f(w, \tilde{s})$ is feasible for $f(v, \tilde{r})$.

We have discussed about Condition 1.

For Condition 2, we simply store (at most) some k best solutions of $f(w, \tilde{s})$ (typically $k = 1$)

Check that some $l \leq k$ best solutions are feasible for $f(v, \tilde{r})$

$f(v, \tilde{r})$ inherits l feasible best solutions

Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Summarization

A triple (Ψ, S, C) where:

- Ψ a generalized context of a subproblem (interpolant)
- S set containing 0 up to k solutions
- C boolean flag, true iff S has all solutions
($S = \emptyset \Rightarrow C = \text{true}$ holds)

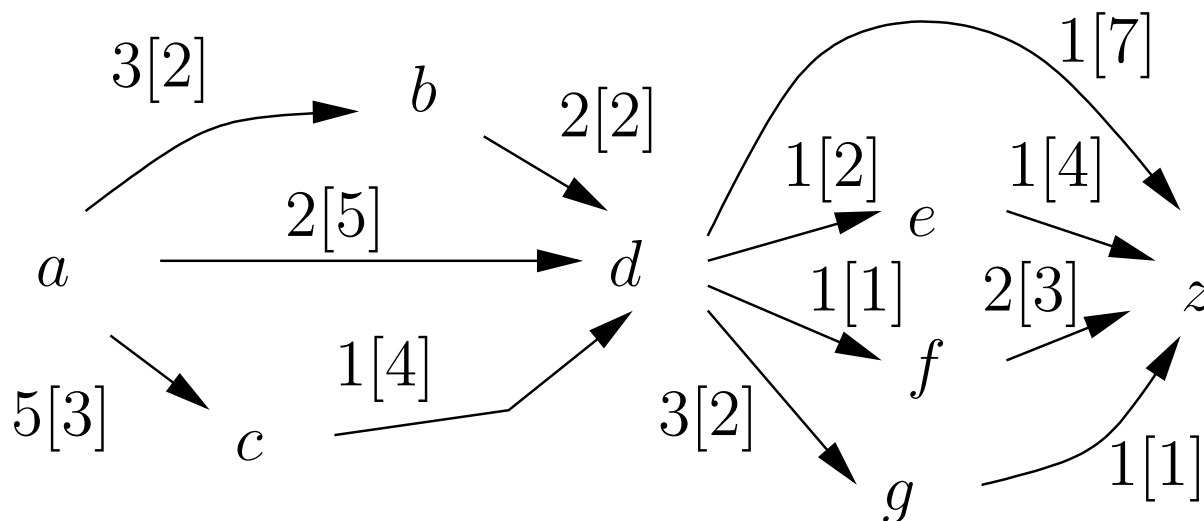
The Main Procedure

```
func summarize( $\Pi$  of length  $m$ ) do  
  if ( $\Pi \Rightarrow false$ ) then return ( $false, \emptyset, true$ )  
  else if ( $\Pi \Rightarrow \varphi(\tilde{X}_m)$ ) then return ( $int(\Pi, \varphi(\tilde{X}_m)), \{true\}, true$ )  
  else if (There is  $(\Psi, S, C) \in Table$  s.t.  $\Pi \Rightarrow \Psi(m)$ ) then  
    ... return ( $\Psi'(\tilde{X}_m), S', C'$ )  
  else  
     $(\Psi', S', C') := (true, \emptyset, true)$   
    foreach (Transition relation  $\rho(\tilde{X}_m, \tilde{X}_{m+1})$ ) do  
      ...  $(\Psi(m+1), S, C) := summarize(\Pi \wedge \rho(\tilde{X}_m, \tilde{X}_{m+1}))$  ...  
    endfor  
     $Table := Table \cup \{(\Psi', S', C')\}$   
    return ( $\Psi', S', C'$ )  
  endif  
endfunc
```

Outline

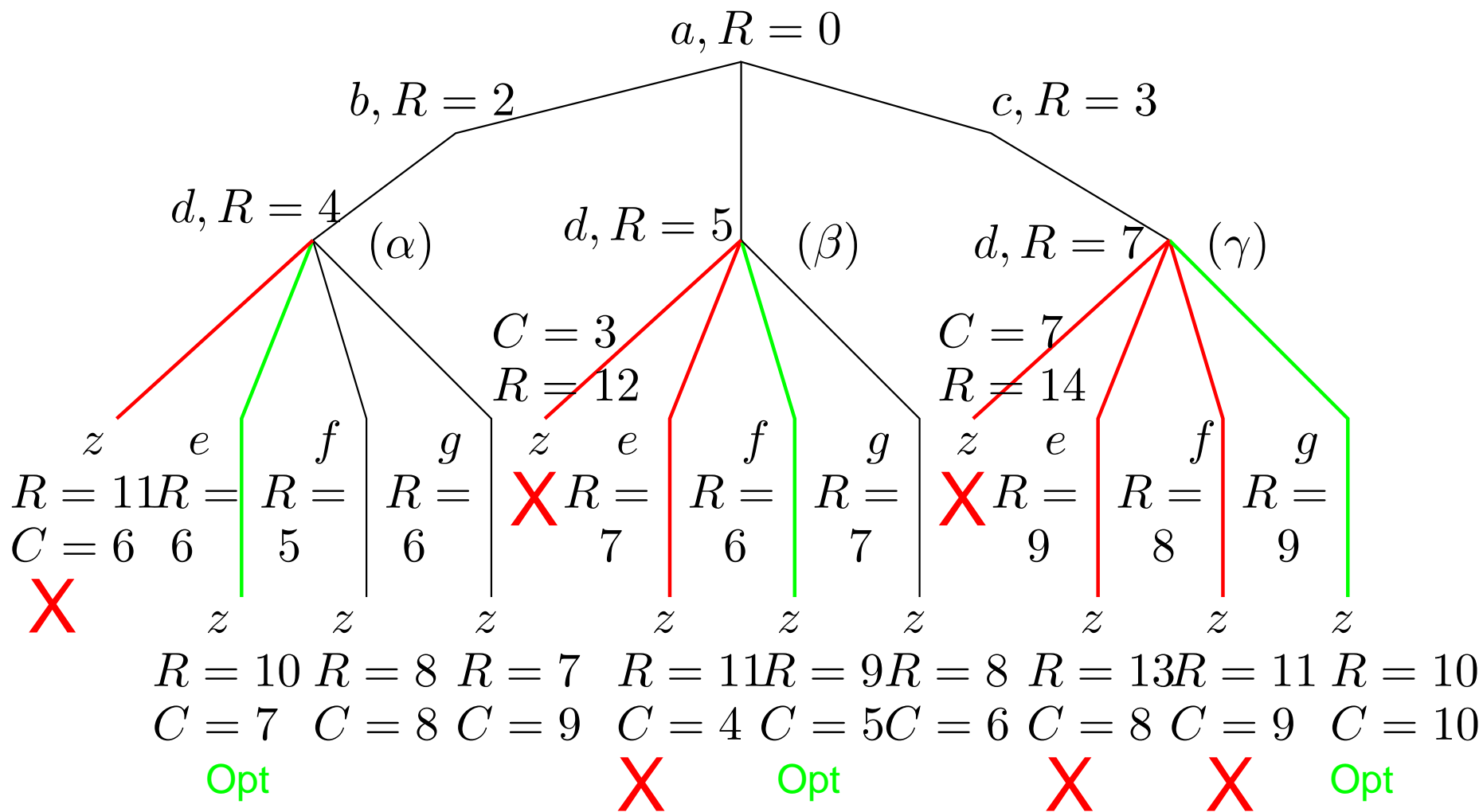
1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Example: RCSP

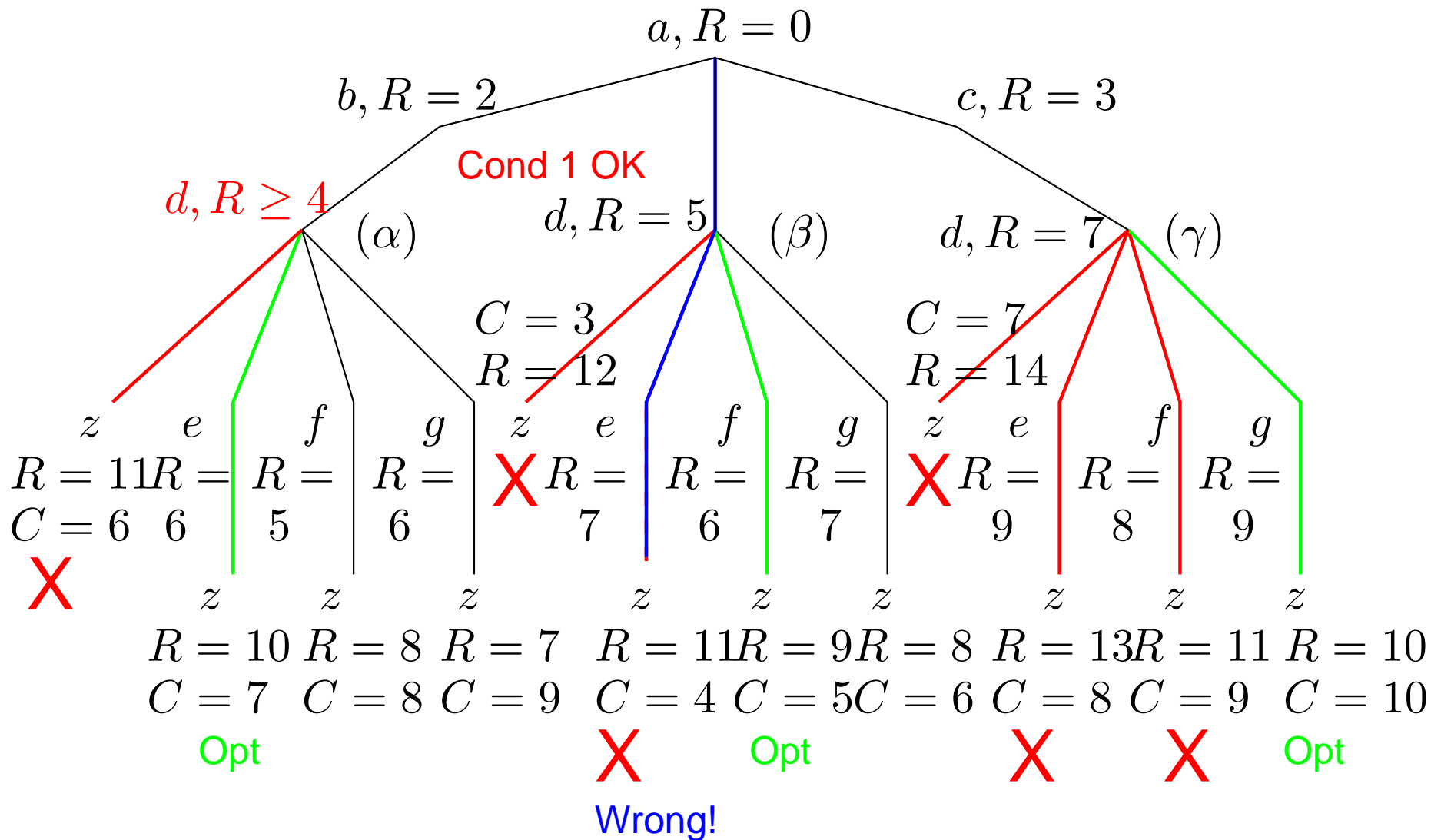


- $c[r]$: c = cost, r = resource consumed.
- Find the shortest path from a to z where resource consumption ≤ 10 .

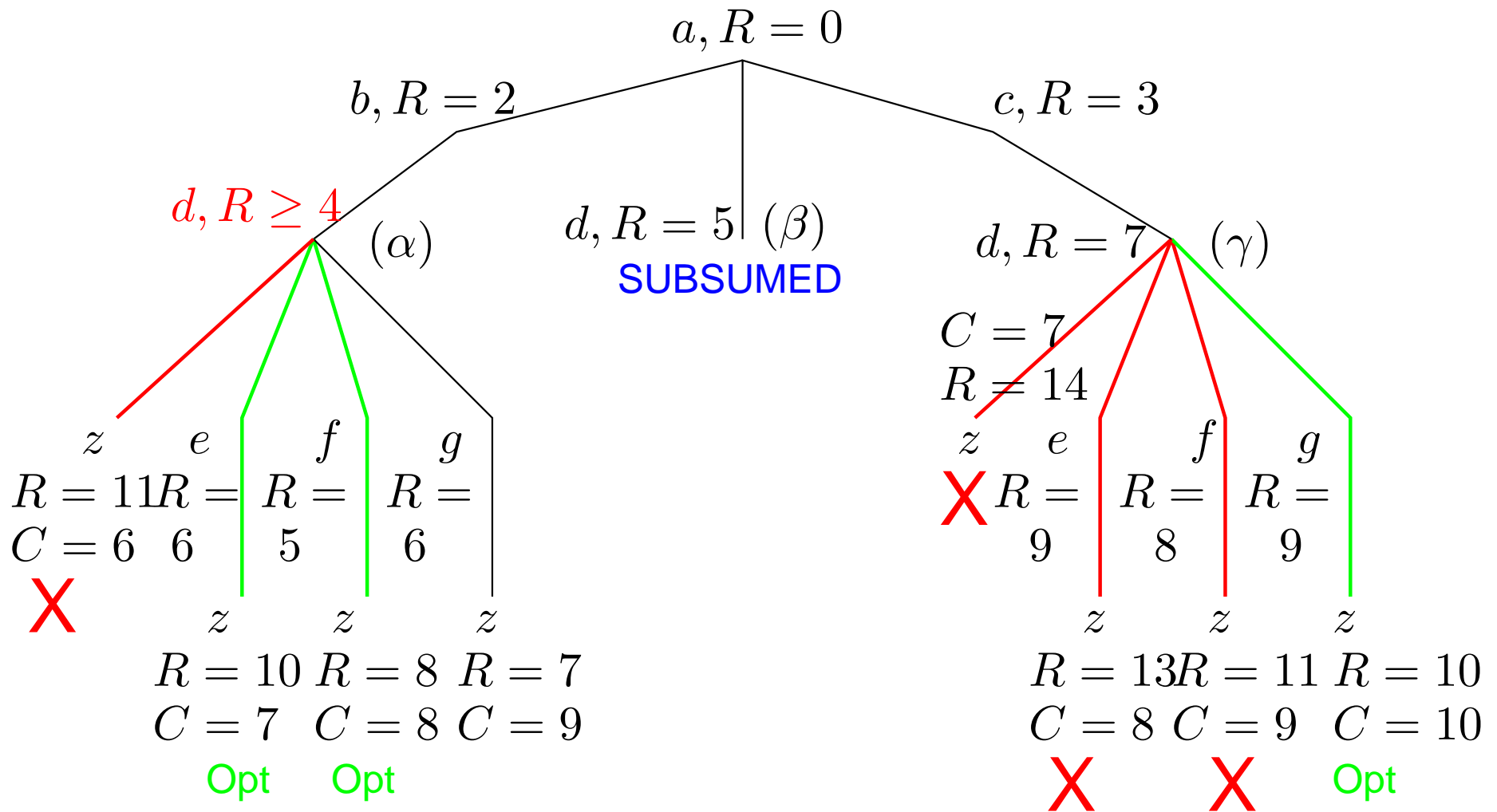
The Search Tree



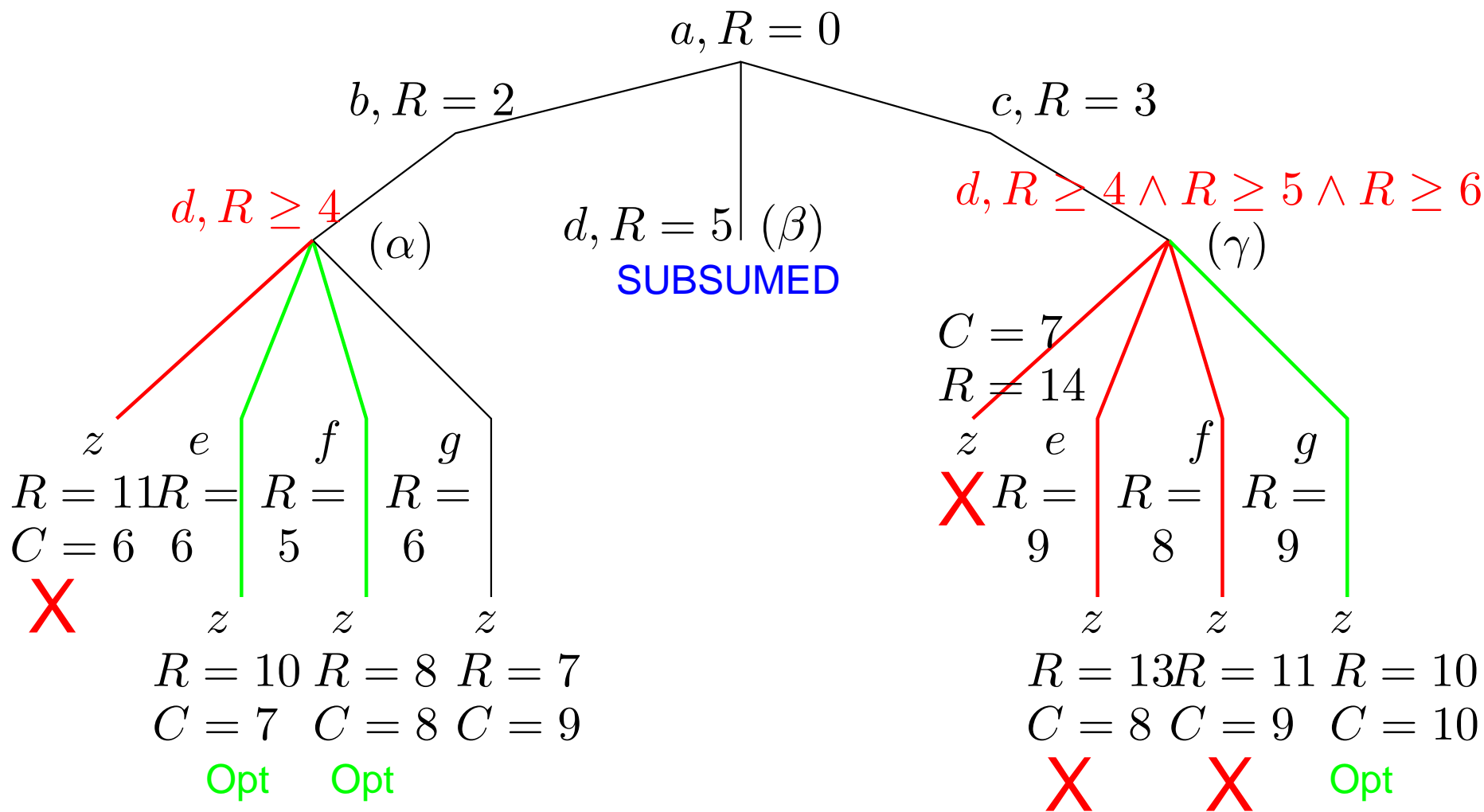
The Search Tree



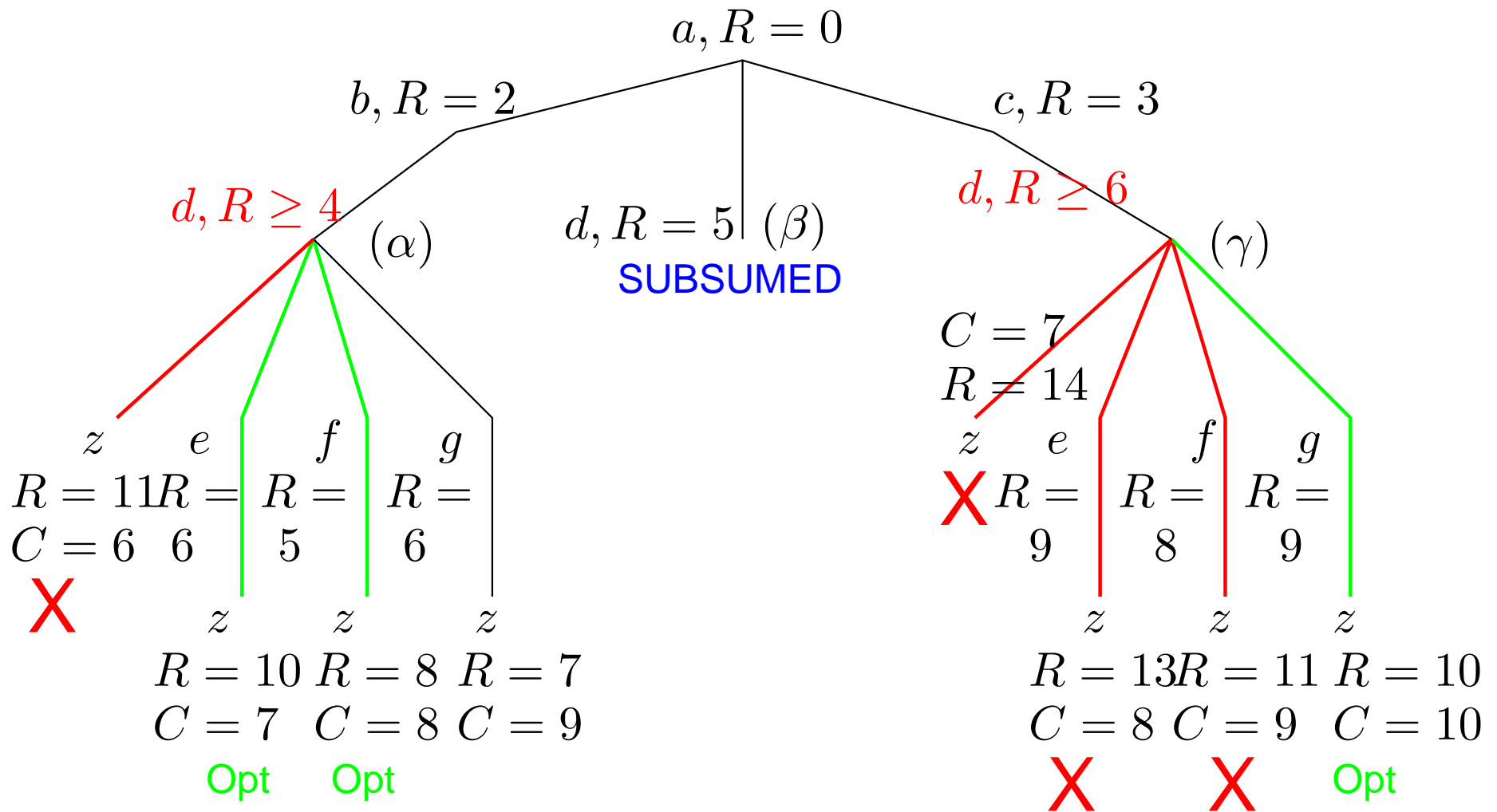
The Search Tree



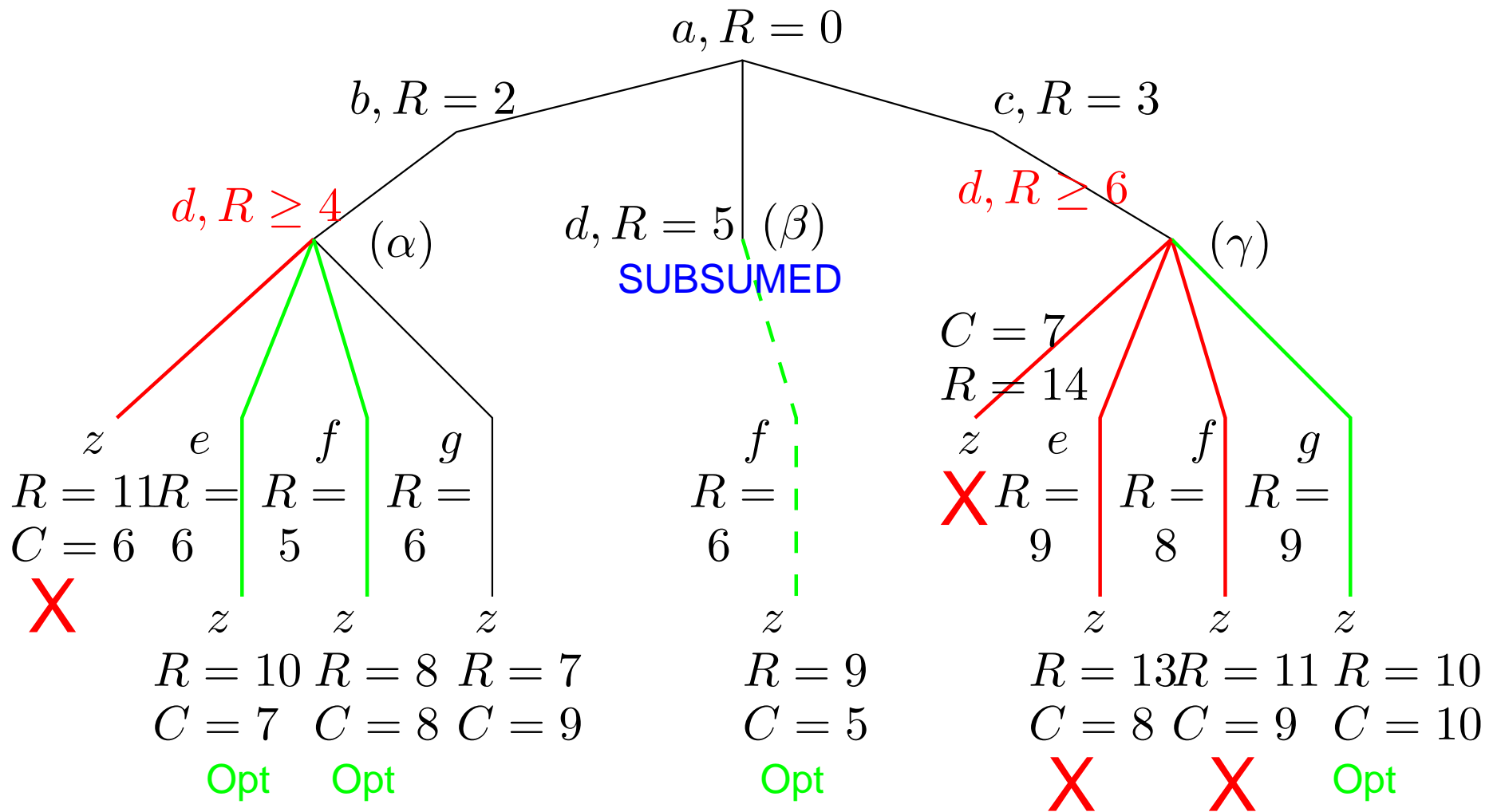
The Search Tree



The Search Tree



The Search Tree



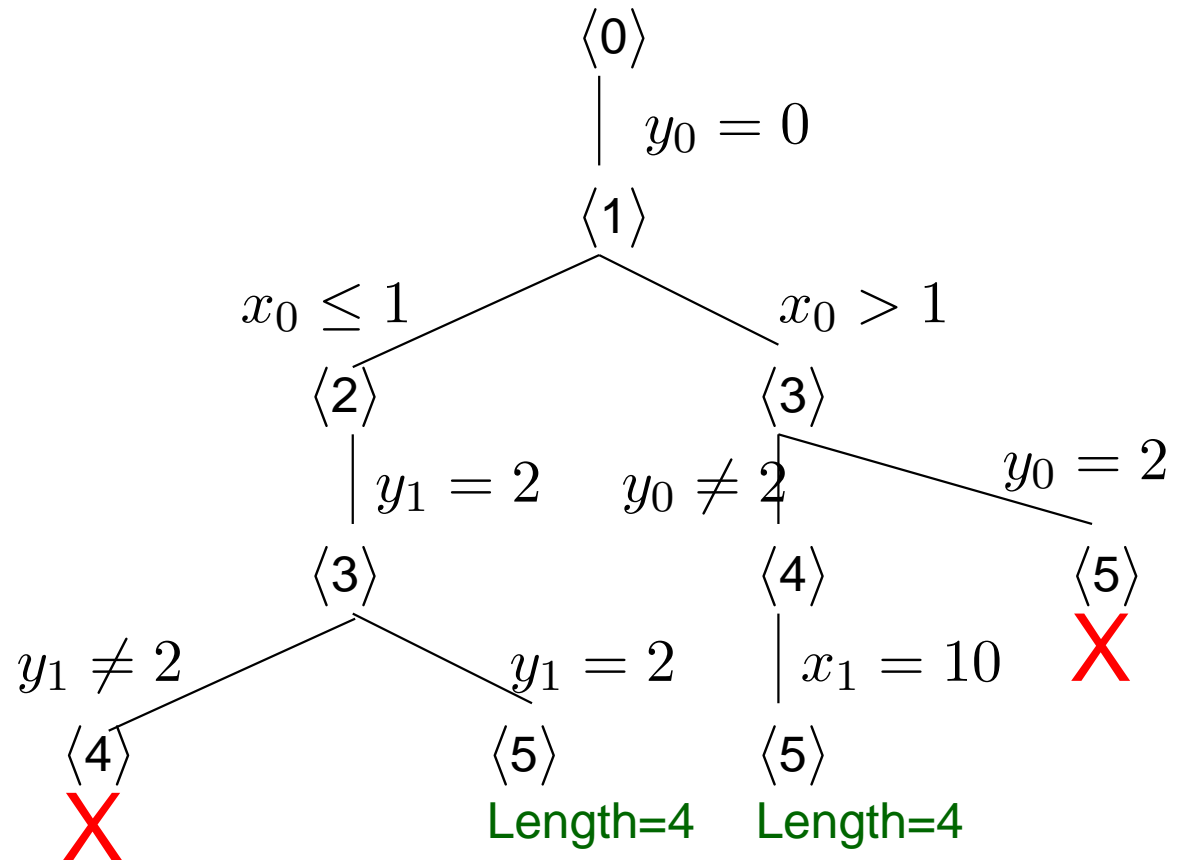
Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Example: WCET

Finding longest path of computer program.

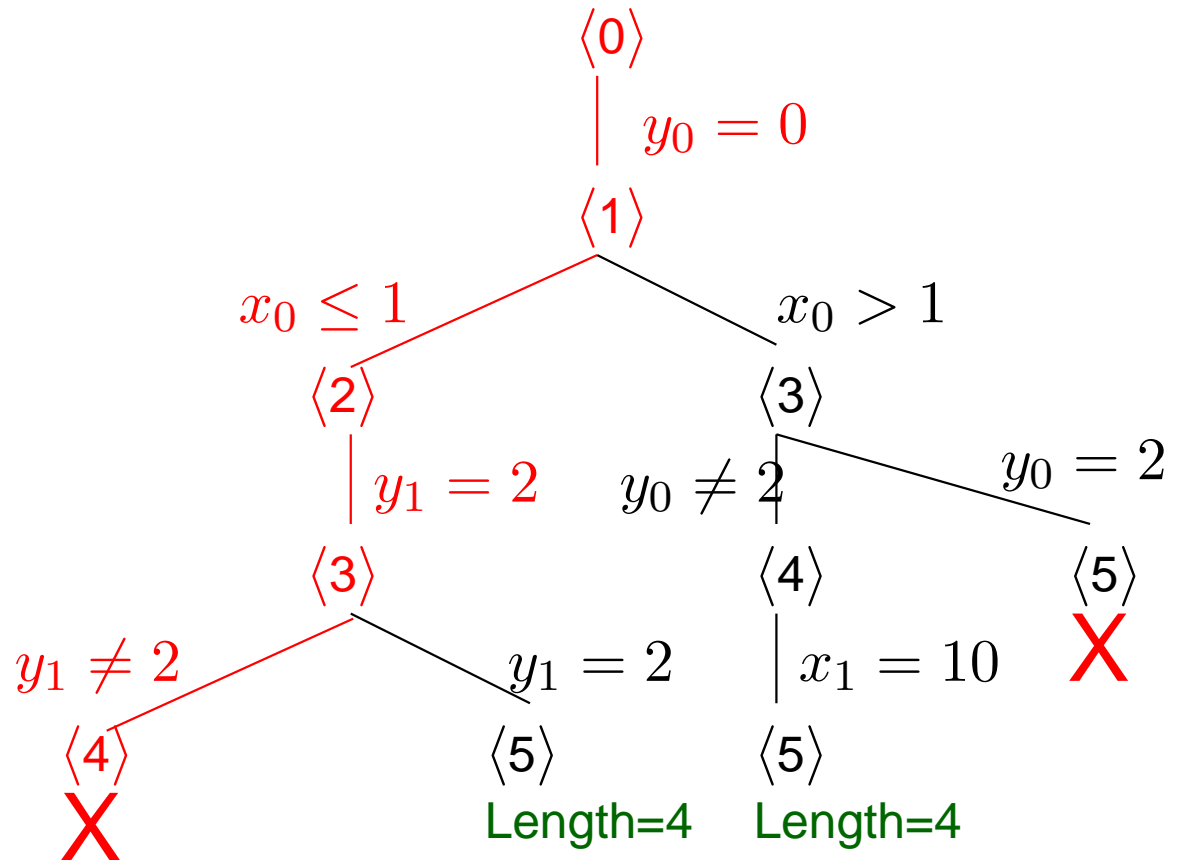
```
<0>   $y := 0$   
<1>  if ( $x \leq 1$ ) then  
<2>     $y := 2$   
      endif  
<3>  if ( $y \neq 2$ ) then  
<4>     $x := 10$   
      endif <5>
```



Example: WCET

Finding longest path of computer program.

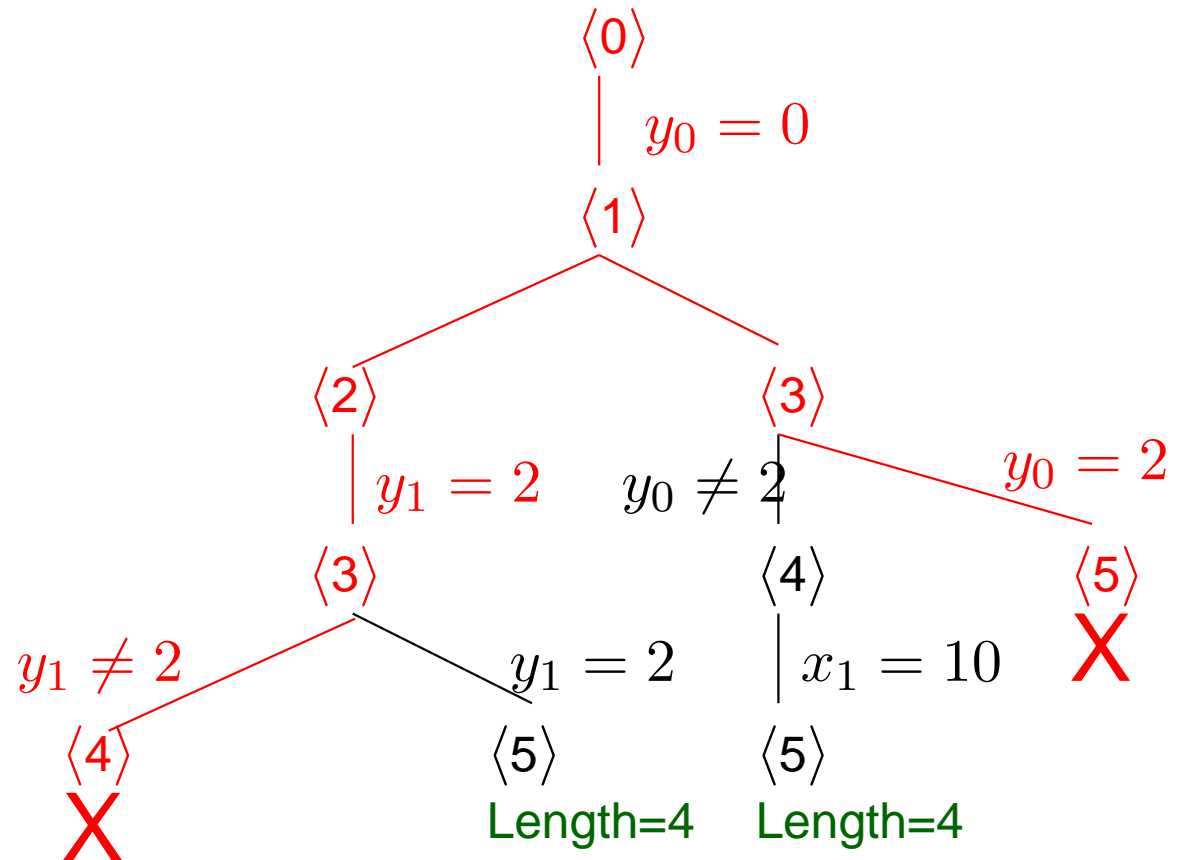
```
<0>   $y := 0$   
<1>  if ( $x \leq 1$ ) then  
<2>     $y := 2$   
      endif  
<3>  if ( $y \neq 2$ ) then  
<4>     $x := 10$   
      endif <5>
```



Example: WCET

Finding longest path of computer program.

```
<0>   $y := 0$   
<1>  if ( $x \leq 1$ ) then  
<2>     $y := 2$   
      endif  
<3>  if ( $y \neq 2$ ) then  
<4>     $x := 10$   
      endif <5>
```



Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
- 6. Experimental results**
7. Related work and conclusion

RCSP Experimental Results

Prob. #	DP		DP+I ($k = 1$)		DP+I ($k = 2$)		DP+I ($k = 10$)	
	Nodes	T	Nodes	T	Nodes	T	Nodes	T
1	55044	94	25021	78	19765	297	19765	281
2	47484	46	22648	31	17724	110	17724	125
3	11448	16	7980	15	8308	32	7659	47
4	9777	16	7237	16	7497	31	7055	47
5	∞		1239214	111623	878879	55804	878879	55640
6	∞		556583	20922	379342	10508	379342	10516
7	178372	687	43888	172	103722	423	103722	454
8	49193	172	18390	94	31759	94	31759	108
9	492	15	459	0	459	0	459	0
10	347	0	321	16	321	0	321	0
11	40346	78	18020	47	26251	93	16941	110
12	35432	62	18973	47	25465	109	17535	110
13	9699	47	7708	16	7266	16	7266	32
14	3678	16	3208	0	3187	32	3187	16
15	111890	375	56273	188	86036	266	86036	281
16	28881	94	20073	62	23698	78	23698	78
17	858743	2234	281101	1031	247821	1110	247821	1234
18	752241	1988	255952	922	226414	954	226414	1106
19	76215	180	42221	93	56806	250	42072	219
20	63592	130	40255	94	47860	204	39684	172
21	590257	2204	60619	266	64461	266	64461	250
22	131905	485	33176	125	35779	110	35779	157
23	3744373	15516	991298	10610	2670921	10258	2670921	10203
24	603421	2234	283042	1203	486856	1673	486856	1625

RCSP Experimental Results

Prob. #	Branch-and-Bound				Hull (Custom)	
	DP+BB		DP+BB+I ($k = 1$)		Relaxation	Closing Gap
	Nodes	T	Nodes	T	T	T
1	17950	63	13030	15	16	16
2	13075	16	9655	15	16	16
3	6195	0	5429	0	16	0
4	5982	0	5613	0	32	0
5	10393	16	8480	15		
6	9751	32	8047	0		
7	9978	31	7565	15		
8	39560	94	17380	31		
9	498	15	432	0	16	0
10	345	0	307	0	16	0
11	15525	31	14309	0	46	0
12	15439	16	14425	16	31	0
13	4678	16	4094	0		
14	3678	32	3208	0		
15	20162	31	15482	31		
16	26416	62	19094	31		
17	1218781	1312	562562	500	46	16
18	1034608	984	512734	422	47	48
19	36789	31	29735	16	78	0
20	44402	31	37463	47	62	0
21	21730	78	16677	15		
22	17381	47	12943	15		
23	100440	610	71837	110		
24	145546	859	103502	157		

WCET Experimental Results

Program	DP		DP+I	
	Nodes	Time (s)	Nodes	Time (s)
bsort(5)	2233	11.22	58	0.05
bsort(10)	∞		218	0.96
bsort(15)	∞		478	7.04
binary(4)	381	0.70	169	0.30
binary(6)	2825	27.47	873	6.54
decoder	344	0.31	132	0.19
sqrt	923	4.25	253	1.43
qurt	1104	14.47	290	2.60
janne_complex	1517	17.93	683	4.36

Outline

1. Dynamic programming with “ad-hoc” constraints
2. Reuse and interpolation
3. The algorithm
4. Example: RCSP
5. Example: WCET
6. Experimental results
7. Related work and conclusion

Related Work

- Formula caching, dominance relation [Kohler 1974]
- Nogood/dead-end learning [Frost & Dechter 1994]
- Conflict-driven/clause learning [Bayardo & Schrag 1997, Moskewicz et al. 2001, Silva & Sakallah 1996]
- Component caching [Kitching & Bacchus 2007]
- Interpolation in program analysis [Henzinger et al. 2004, Jhala & McMillan 2005, McMillan 2003]

Conclusion

- Enhancements to dynamic programming:
 - Generalizing subproblems using *interpolants*
 - Memoing interpolants and up to k optimal as *summarizations*
- Useful in presence of ad-hoc constraints
- Can augment existing memo-based approaches
- Experimental results: RCSP, WCET