Session „**RTE Mechanisms** “

Chair: *Marco Di Natale, Scuola Superiore Sant'Anna, Pisa, Italy*

Explicit Preemption Placement for Real-Time Conditional Code

Bo Peng, Nathan Fisher and Marko Bertogna

Multi Sloth: An Efficient Multi-Core RTOS using Hardware-Based Scheduling

Rainer Müller, Daniel Danner, Wolfgang Schröder-Preikschat and Daniel Lohmann

Explicit Preemption Placement for Real-Time Conditional Code via Graph Grammars and Dynamic Programming

Bo Peng[1], Nathan Fisher[1], and Marko Bertogna[2]

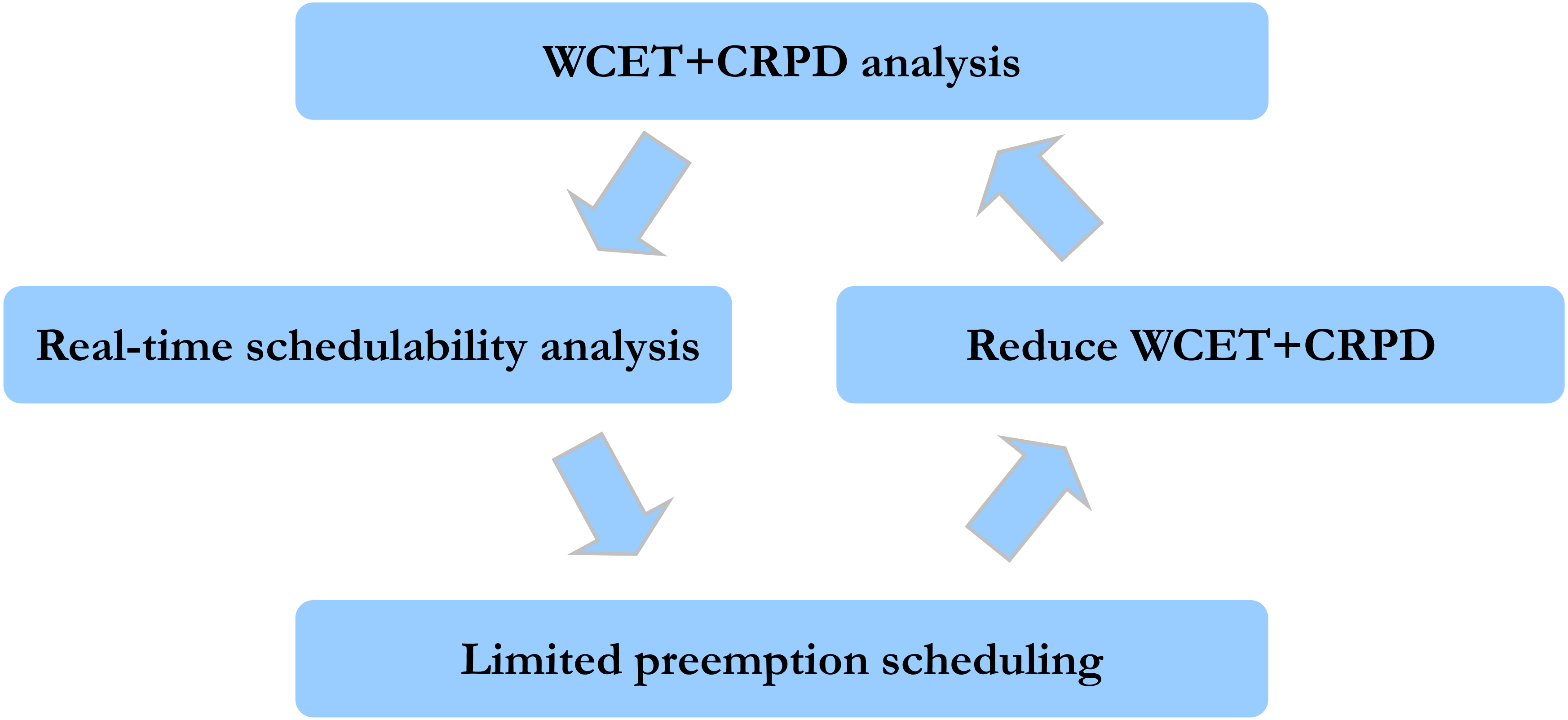
[1] Department of Computer Science, Wayne State University, USA
[2] Algorithmic Research Group, University of Modena, Italy

This research has been supported by US National Science Foundation and the European Commission.

Outline

Background	Limited-preemption scheduling model in real -time code.
Model	Series-parallel flowgraphs.
Problem Statement	Optimize the WCET+CRPD of the flowgraphs.
Solution	Graph grammars; Dynamic programming.

Introduction



WCET+CRPD analysis

Real-time schedulability analysis

Reduce WCET+CRPD

Limited preemption scheduling

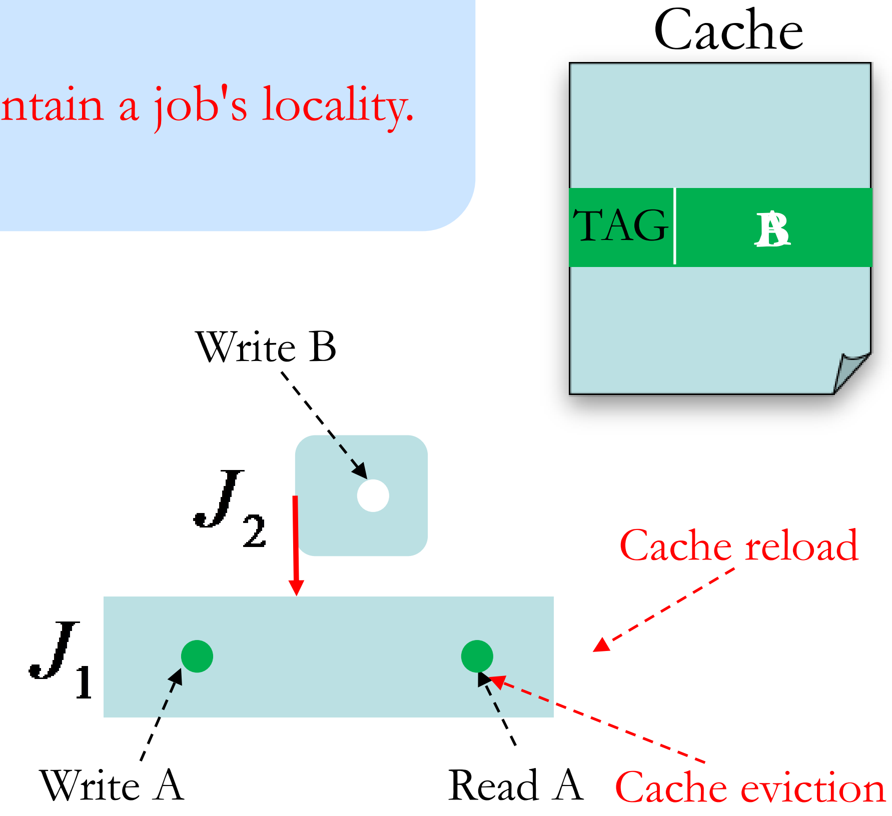
Limited Preemption Scheduling

Reduce WCET+CRPD

- Precise upper bounds on the cache-related preemption delays (CRPD).
- Delay the preemption to maintain a job's locality.

CRPD

◆Cache evictions by preempting higher-priority tasks.



Limited Preemption Scheduling

Reduce WCET+CRPD when arbitrarily preempt

- Precise upper bounds on the cache-related preemption delays (CRPD).
- Delay the preemption to maintain a job's locality.

Our goal

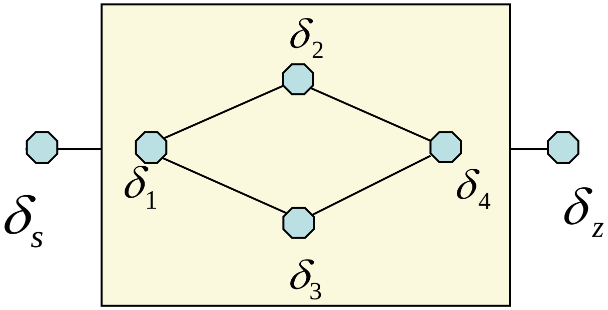
Reduce CRPD via limited preemption while preserving system schedulability.

Related Work

Preemption model		Fixed preemption point models	Floating preemption point models
Scheduling algorithm	**EDF scheduled systems**	Burns ['94]	Baruah ['05] Bertogna and Baruah ['10]
	Fixed Priority scheduled systems	Burns ['94], Bril et al. ['09], Bertogna et al. ['11], Davis et al. ['12]	Yao et al. ['09]
Code structure	**Linear code structure**	Bertogna et al.['10, '11]	
	Conditional code structure	?	

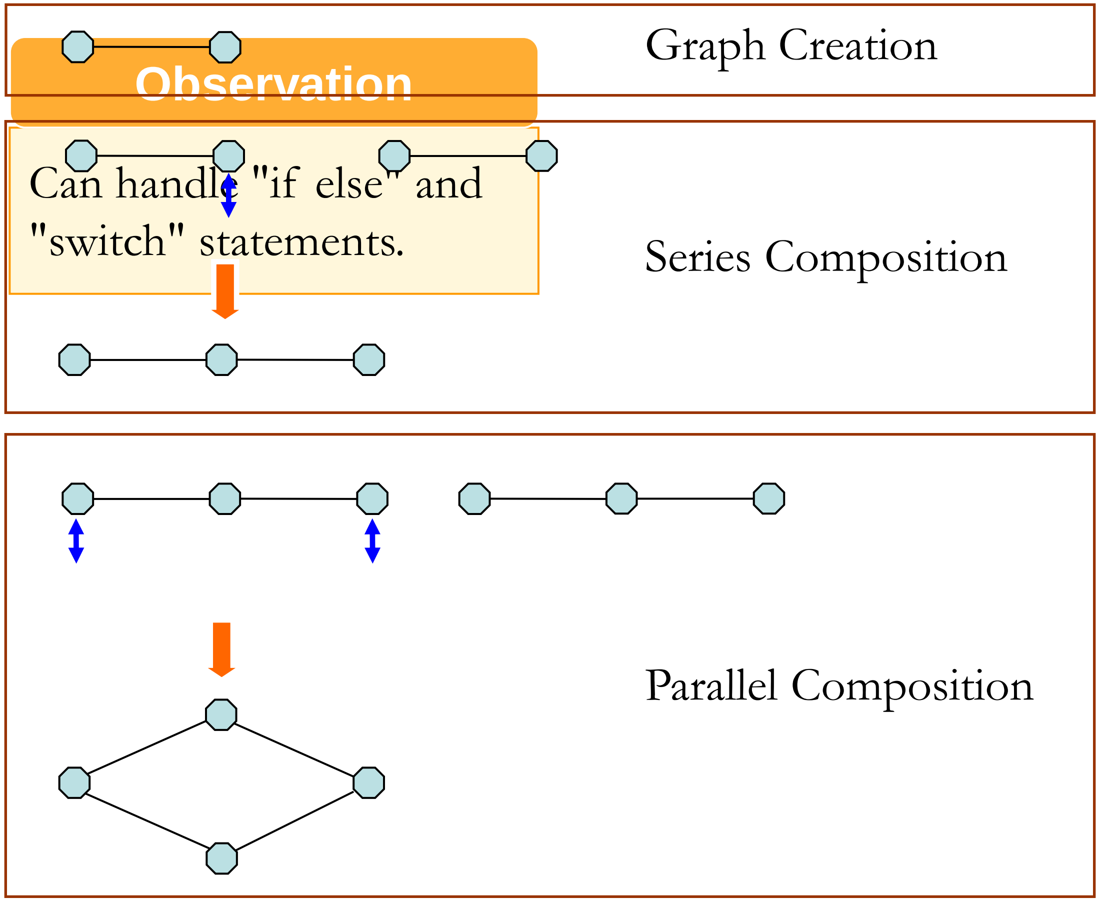
Model

- Control flowgraph: $G_P = (V, E, \delta_s, \delta_z)$
- $V = \{\delta_1, \delta_2, \ldots, \delta_n\}$: set of basic blocks (BBs)
- $(\delta_1, \delta_2) \in E \subseteq V \times V$: set of edges
- $C : V \mapsto \Re \geq 0$: WCET function of BBs
- $\xi : E \mapsto \Re \geq 0$: CRPD function of edges
- PPP : Potential Preemption Point
- EPP : Effective Preemption Point



Model

- Series-parallel graphs:



Graph Creation

Series Composition

Parallel Composition

Observation

Can handle "if else" and "switch" statements.

Problem Statement

Problem statement:
Given $G_{\mathcal{P}} \in \mathcal{G}$ and associated functions ξ and C, find $S \subseteq E$ that minimizes

Series-parallel graphs

Choose the path with max WCET+CRPD

Sum of BBs' WCET

Sum of selected edges' CRPD

$$\Phi(G_{\mathcal{P}}, S) \overset{def}{=} \max_{p \in \mathsf{paths}(G_{\mathcal{P}}, \delta_s, \delta_z)} \left\{ \sum_{\delta_u \in p} C(\delta_u) + \sum_{\substack{\delta_u, \delta_v \in p \\ (\delta_u, \delta_v) \in S}} \xi(\delta_u, \delta_v) \right\} \quad (1)$$

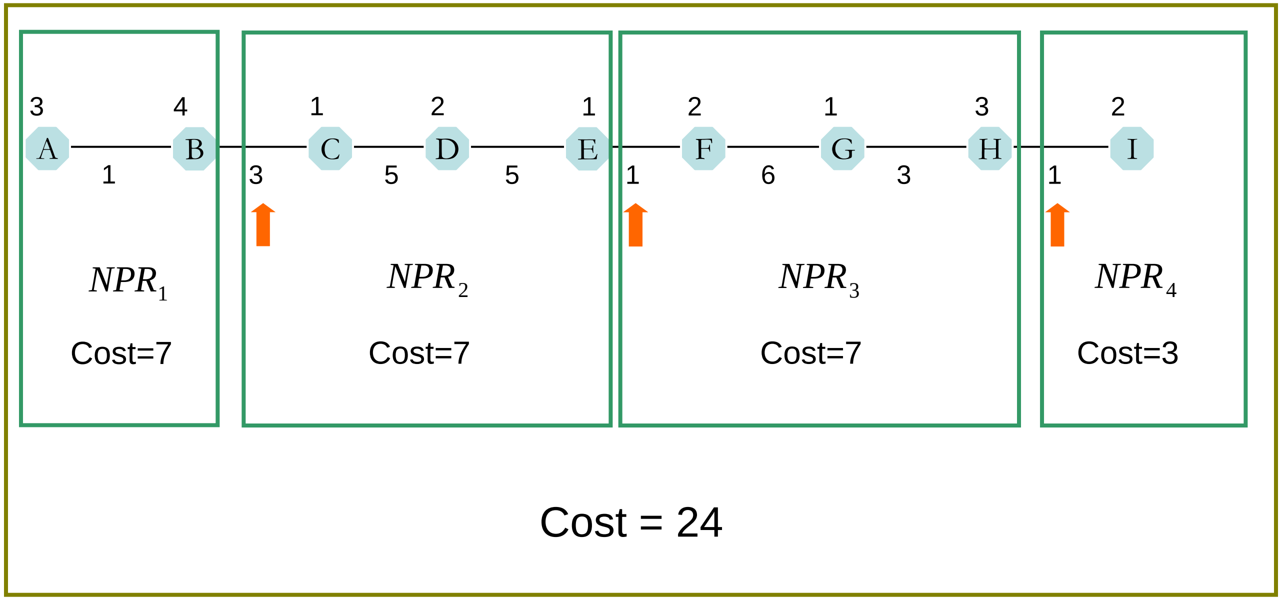
subject to the constraint that $\forall p \in \mathsf{paths}(G_{\mathcal{P}}, \delta_s, \delta_z), \delta_i \in p$: $\exists e_1 = (\delta_u, \delta_v), e_2 = (\delta_x, \delta_y) \in S ::$

Upper bound of NPR from schedulability analysis

$$(\delta_u \preceq_p \delta_i \preceq_p \delta_y) \quad \wedge \quad \left(\overbrace{\xi(e_1) + \sum_{\substack{\delta_j \in p \\ \delta_v \preceq_p \delta_j \preceq_p \delta_x}} C(\delta_j)}^{\text{NPR from } e_1 \text{ to } \delta_x} \leq Q \right). \quad (2)$$

Problem Statement

- Find a selection of EPPs that minimizes the WCET+CRPD of a flowgraph.
- The cost of any non-preemptive region should less than Q.



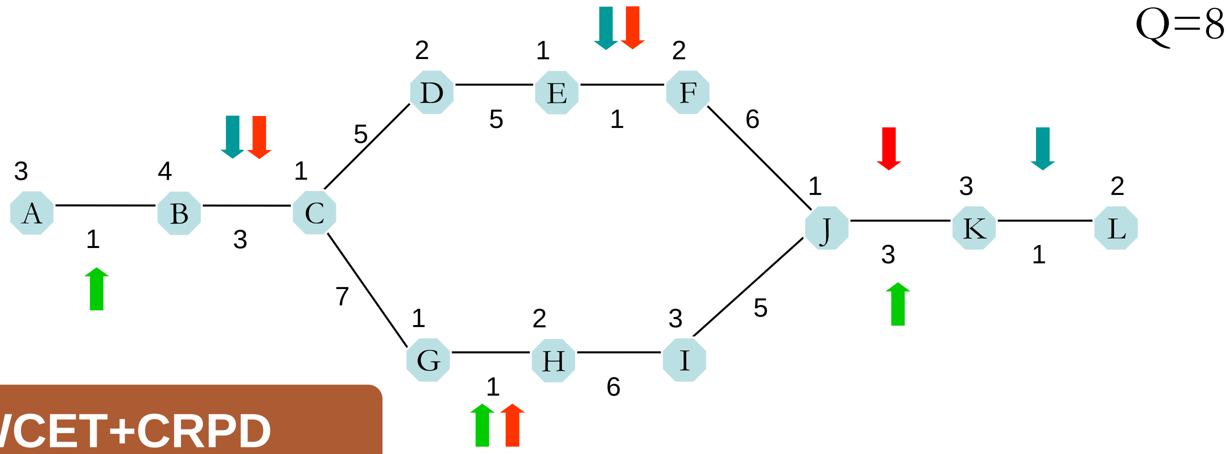
Q=8

Optimal selection of EPPs in sequential flowgraphs

Bertogna et al.['11]

Problem Statement

- How about the previous algorithm for conditional structure?



WCET+CRPD

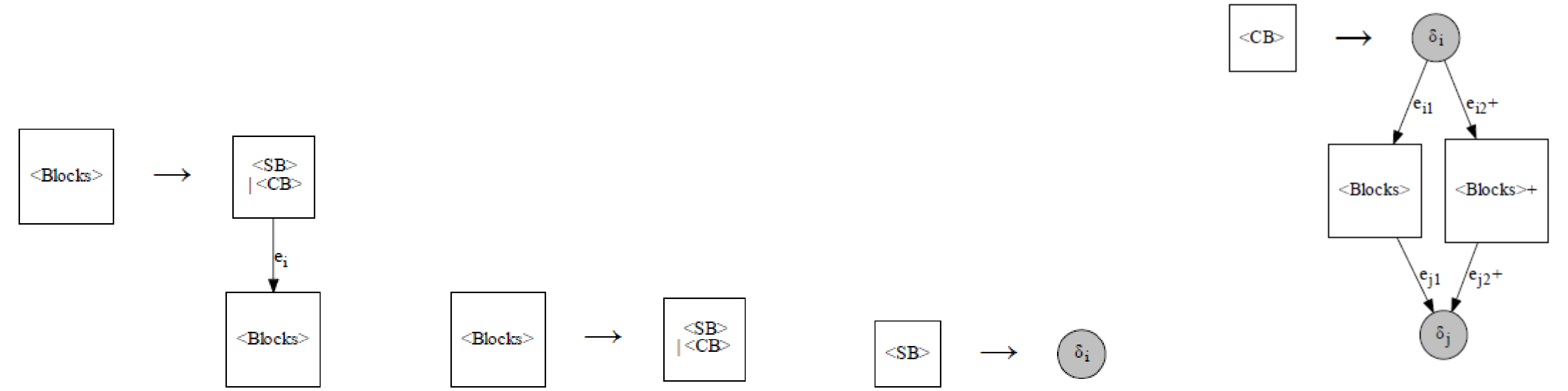
- Upper path: 28
- Lower path: 25
- Combined result: 32(U)

Observation

- Previous algorithm is not optimal for conditional structure.
- Use graph grammar and dynamic programming technique.

Graph Grammar

- Decompose control flowgraphs (Linear time parsing)
- Extended Backus-Naur Form (EBNF)



(a) $\langle\texttt{Blocks}\rangle \rightarrow (\langle\texttt{SB}\rangle \,|\, \langle\texttt{CB}\rangle), e_i, \langle\texttt{Blocks}\rangle$

(b) $\langle\texttt{Blocks}\rangle \rightarrow (\langle\texttt{SB}\rangle \,|\, \langle\texttt{CB}\rangle)$

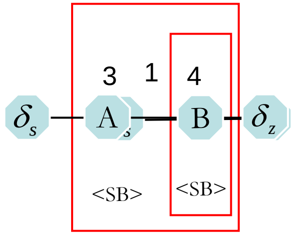
(c) $\langle\texttt{SB}\rangle \rightarrow \delta_i$

(d) $\langle\texttt{CB}\rangle \rightarrow \delta_i, [e_{i_1}, \langle\texttt{Blocks}\rangle, e_{j_1}], \left([e_{i_2}, \langle\texttt{Blocks}\rangle, e_{j_2}]\right)+, \delta_j$

Dynamic Programming:

Sequential Block

Q=8



Cost matrices reflect optimal substructures

Blocks contains δ_A and δ_B : $Cost[\delta_s][\delta_z]$

	0	1	2	3	4	5	6	7
0	7	7	8	8	INF	INF	INF	INF
1	7	8	8	8	INF	INF	INF	INF
2	8	8	8	8	INF	INF	INF	INF
3	8	8	8	8	INF	INF	INF	INF
4	8	8	8	8	INF	INF	INF	INF
5	8	8	8	8	INF	INF	INF	INF
6	INF	INF	INF	INF	INF	INF	INF	INF
7	INF	INF	INF	INF	INF	INF	INF	INF

Blocks only contains δ_B : $Cost[\delta_s][\delta_z]$

	0	1	2	3	4	5	6	7
0	4	4	4	4	4	INF	INF	INF
1	4	4	4	4	INF	INF	INF	INF
2	4	4	4	INF	INF	INF	INF	INF
3	4	4	INF	INF	INF	INF	INF	INF
4	4	INF	INF	INF	INF	INF	INF	INF
5	INF	INF	INF	INF	INF	INF	INF	INF
6	INF	INF	INF	INF	INF	INF	INF	INF
7	INF	INF	INF	INF	INF	INF	INF	INF

$1: Cost[\delta_S][\delta_Z] = C(\delta_A) + \xi(\delta_A, \delta_B) + Cost_{prev} = 3+1+4 = 8$

$2: Cost[\delta_S][\delta_Z] = C(\delta_A) + Cost_{prev} = 3+4 = 7$

$1: Cost[\xi(\delta_A, \delta_B), \delta_B][\delta_Z] = Cost[1][0] = 4$

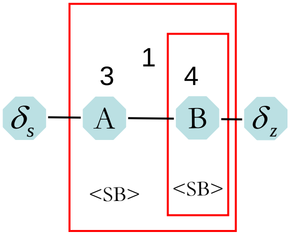
8

INF

$2: Cost[\delta_S + \delta_A][\delta_Z] = Cost[4][0] = 4$

Dynamic Programming:

Conditional Block and Block Union



Theorem

A optimal EPP selection of a larger block is combined by the optimal selections of left block and right block.

Corollary

Theorem

(1) Get the optimal value of when preempt.

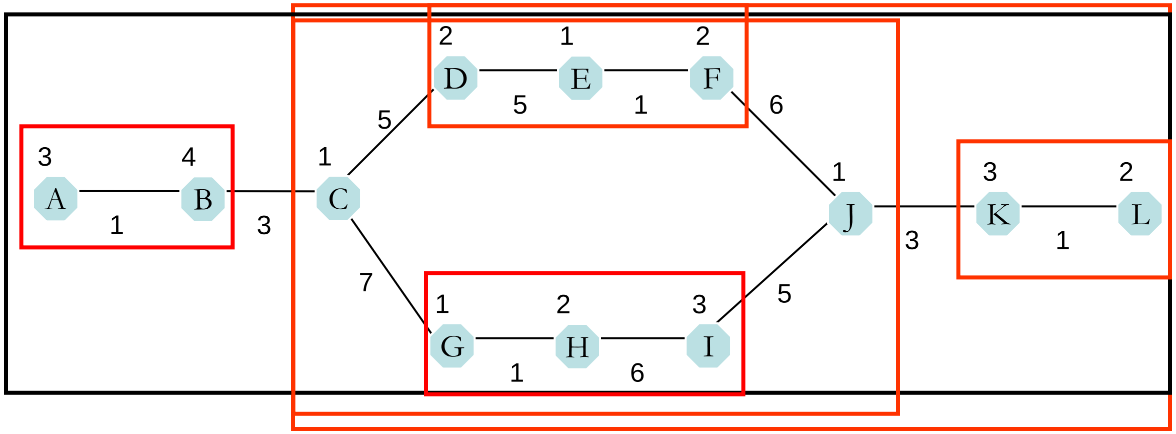
Q=8

Q

$Cost[0][0]$

Q

$O(|V|Q^3)$



Dynamic Programming:

Conditional Block and Block Union

Contribution

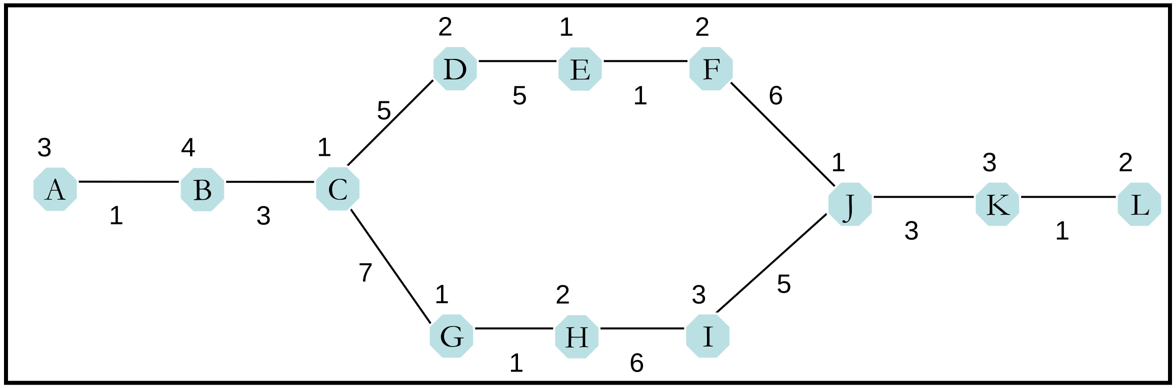
In conditional structure:

Optimal EPPs selection in pseudo-polynomial time.

Q=8

$Cost[0][0]$

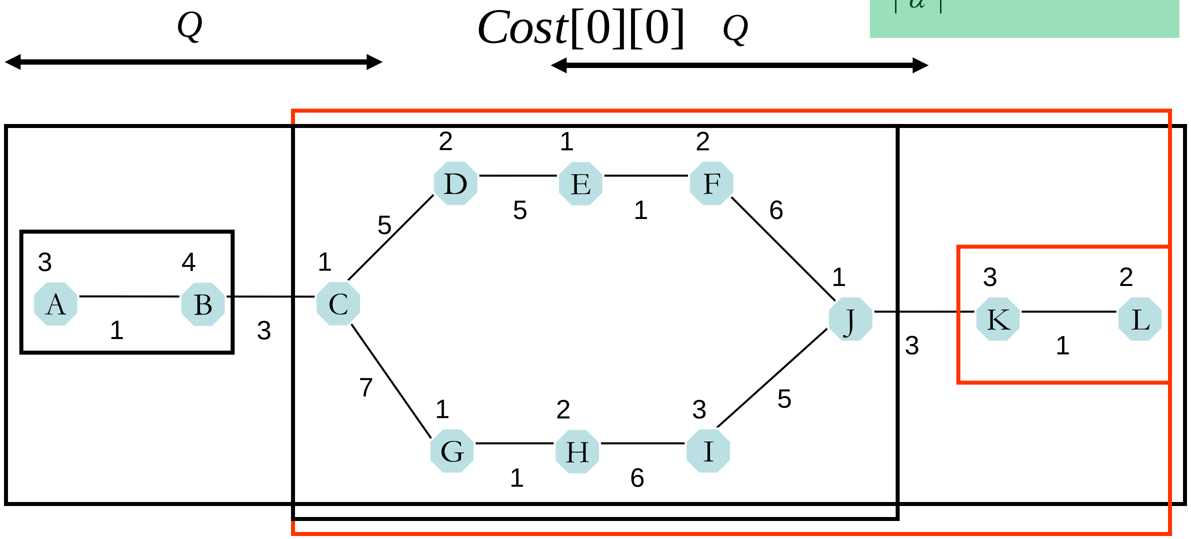
$O(|V|Q^3)$



An Alternative Heuristic

- Approximate flexible $\alpha \times \alpha$ matrix
  - The WCET+CRPD will be larger than the optimal one.
  - Reduce running time.
  - Time complexity: $O(|V|\alpha^3)$

$\left\lceil \frac{Q}{\alpha} \right\rceil$ step each time



$i = 0 : \alpha$
$j = 0 : \alpha$

	0	$\left\lceil \frac{Q}{\alpha} \right\rceil$	$2 \cdot \left\lceil \frac{Q}{\alpha} \right\rceil$	…	$i \cdot \left\lceil \frac{Q}{\alpha} \right\rceil$	…
0						
$\left\lceil \frac{Q}{\alpha} \right\rceil$						
$2 \cdot \left\lceil \frac{Q}{\alpha} \right\rceil$						
…						
$i \cdot \left\lceil \frac{Q}{\alpha} \right\rceil$						
…						

Computational Complexity

- Consider parse tree and cost matrix:
  - Optimal solution: $O(|V|Q^3)$
  - Heuristic solution: $O(|V|\alpha^3)$

Simulations

- Randomly generate control flowgraphs:
  - number of BBs.
  - number of CBs.
- WCET of BBs:
  - Gaussian distribution.
- CRPD:
  - Correlates adjacent EPPs.
  - Randomly generated with a gaussian factor.

Similar to a realistic distribution

Bertogna et al.['11]

Simulations: WCET+CRPD

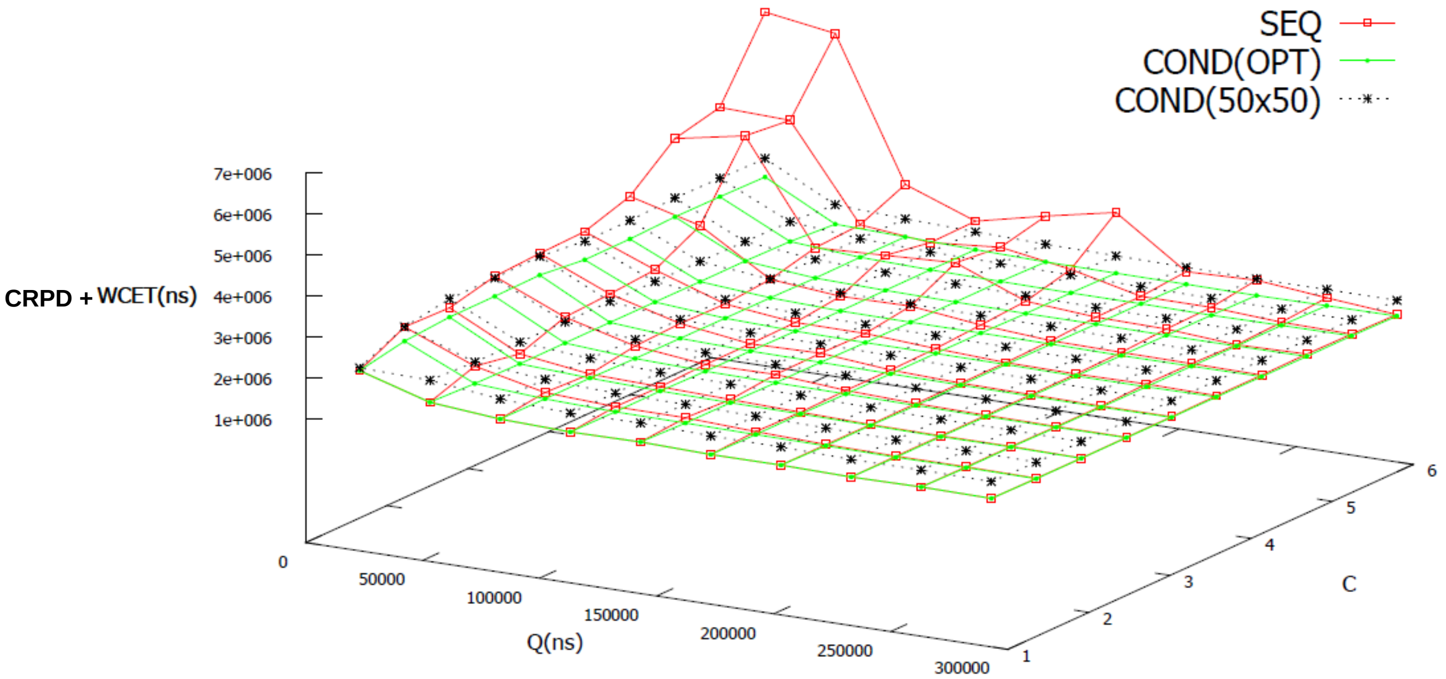


Fig. 4. Comparison of WCET over Different Values of Q and Number of Conditional Blocks (C) for SEQ, COND(OPT), and COND(50×50).

WCET trend

OPT (green mesh)<Alternative heuristic (black mesh)<SEQ (red mesh)

Simulations: Running time

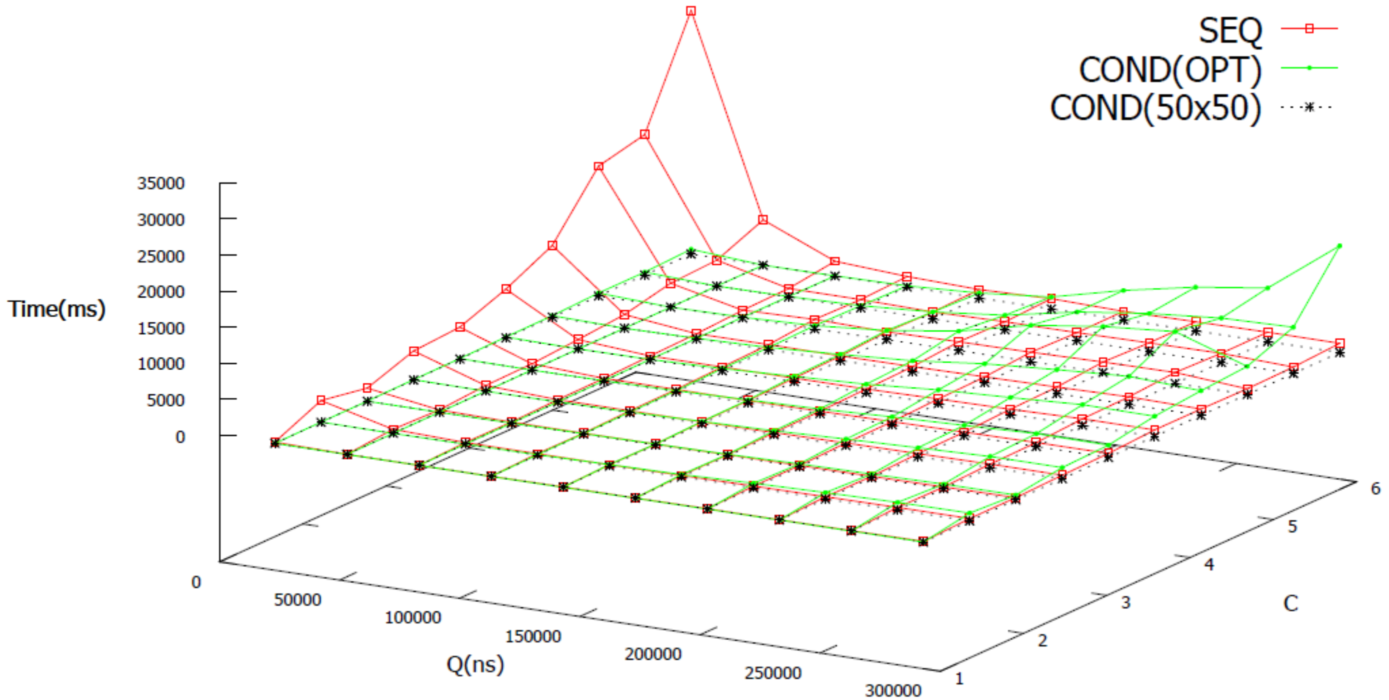


Fig. 5. Comparison of Algorithm Running Times over Different Values of Q and Number of Conditional Blocks (C) for SEQ, COND(OPT), and COND(50×50)

Time trend

Alternative heuristic (black mesh) dominates over OPT and SEQ

Simulations: WCET+CRPD

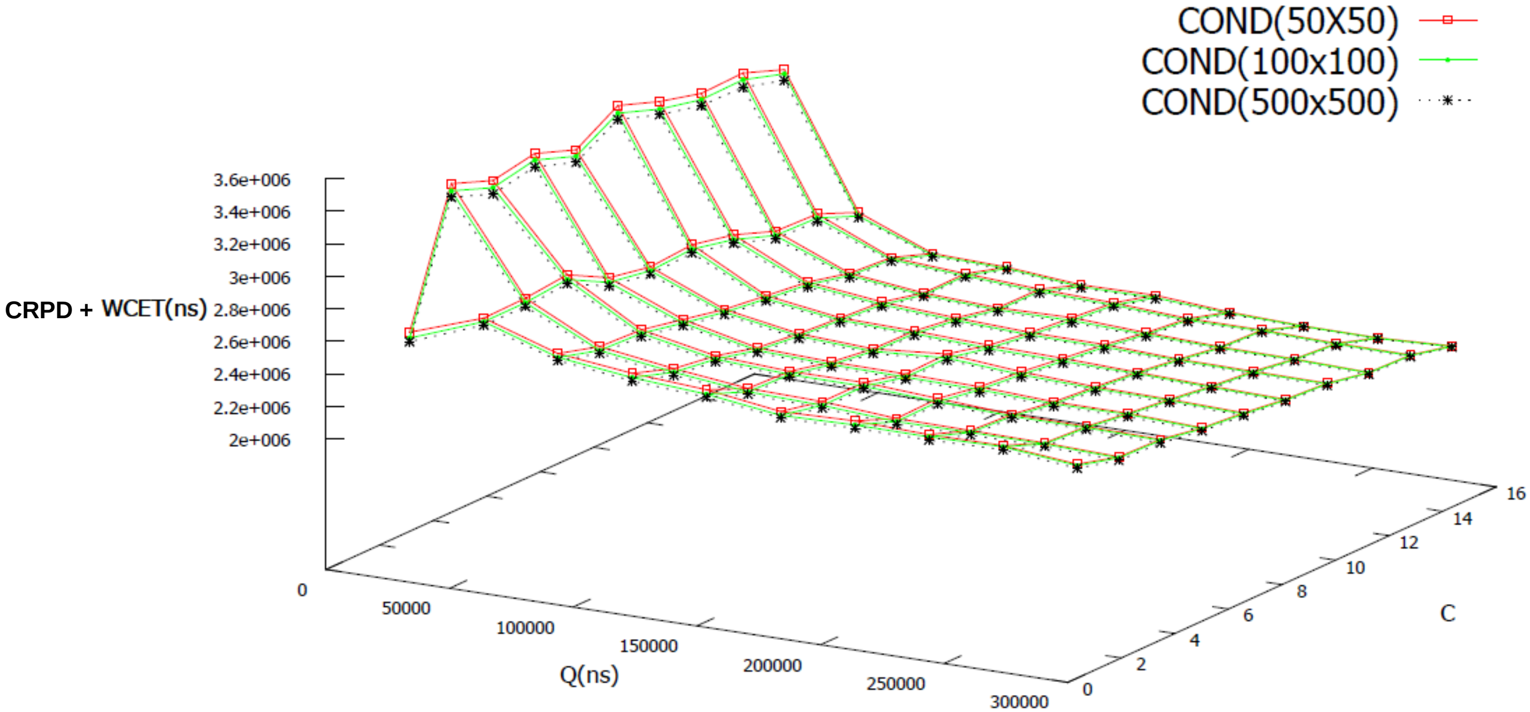


Fig. 6. Comparison of WCET over Different Values of Q and Number of Conditional Blocks (C) for heuristics.

Comparison of different setting for heuristics

Smaller size of matrices does not significantly increase WCET+CRPD

Simulations: Running time

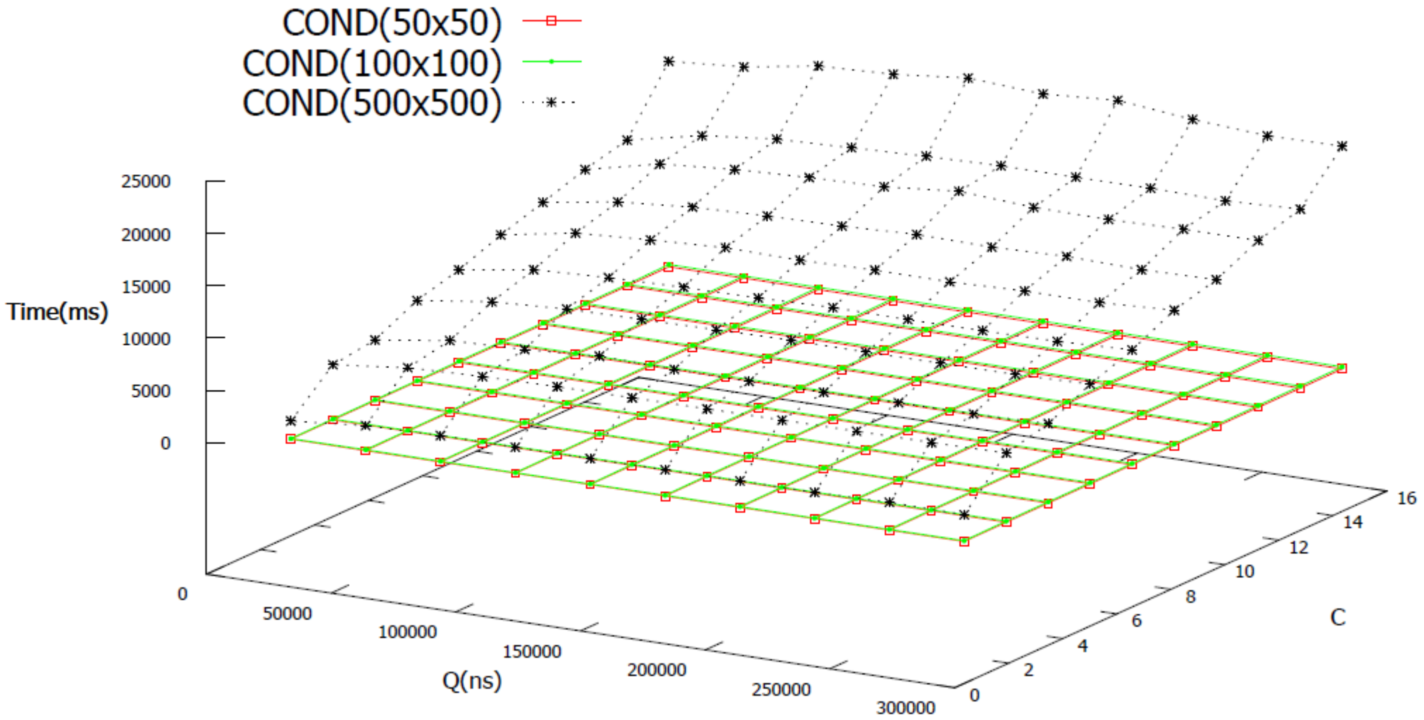


Fig. 7. Comparison of Algorithm Running Times over Different Values of Q and Number of Conditional Blocks ((C) for heuristics.

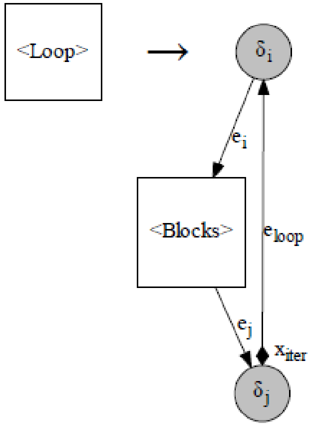
Comparison of different setting for heuristics

Smaller size of matrices significantly decreases running time

Additional Structure

(a) Loops:

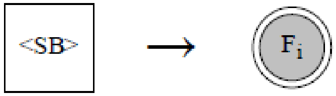
- Unrolled Loop
- Non-Unrolled Loop



(a) $\langle \texttt{Loop} \rangle \rightarrow \{\delta_i, e_i, \langle \texttt{Blocks} \rangle, e_j, \delta_j, e_{loop}, x_{iter}\}$

(b) Function call:

- Calculate the cost matrix of F_i
- Embed F_i to the main control flowgraph



(b) $\langle \texttt{SB} \rangle \rightarrow F_i$

Conlusions and Future Work

- Conlusions:
  - Extend the structure to conditional blocks (also loops and function call)
  - Optimal algorithm for selection of EPPs
  - An alternative heuristic to reduce running time
  - Exhaustive simulation
- Future work:
  - Planar separator theory
  - Parameterized theory
  - NP-Completeness
  - Implement this technique in automatic code generation

Do optimal algorithms using polynomial time exist?

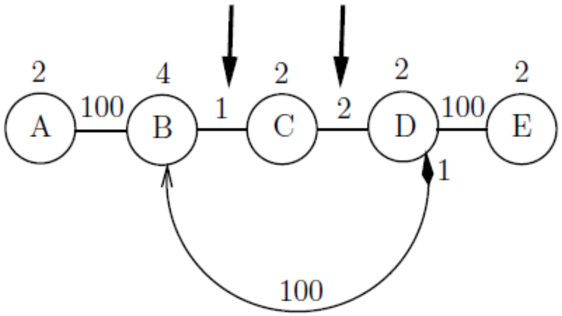
Or, is the problem NP-Complete?

Thanks!

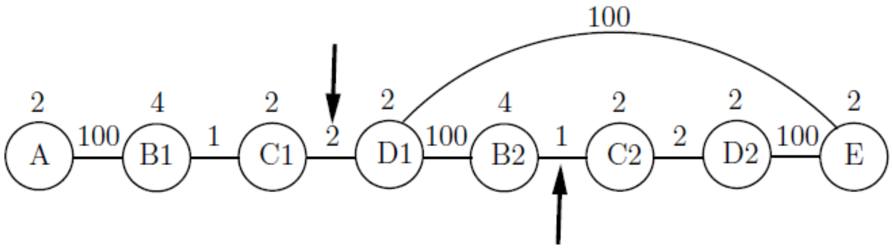
Related Work

EDF scheduled systems **Fixed preemption point models**	Burns ['94] : [11] A. Burns. Preemptive priority based scheduling: An appropriate engineering approach
EDF scheduled systems **Floating preemption point models**	Baruah ['05] : [4] S. Baruah. The limited-preemption uniprocessor scheduling of sporadic task systems. Bertogna and S. Baruah ['10] : [5] M. Bertogna and S. Baruah. Limited preemption EDF scheduling of sporadic task systems.
Fixed Priority scheduled system **Fixed preemption point models**	Burns ['94] : [11] A. Burns. Preemptive priority based scheduling: An appropriate engineering approach Bril et al.['12]: [10]Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption. Bertogna et al. ['11]: [7] M. Bertogna et al. Improving feasibility of fixed priority tasks using non-preemptive regions. Davis et al. ['12] : [13] R. Davis and M. Bertogna. Optimal fixed priority scheduling with deferred preemption.
Fixed Priority scheduled systemFloating preemption point models	Yao et al. ['09]: [19] G. Yao, G. Buttazzo, and M. Bertogna. Bounding the maximum length of non-preemptive regions under fixed priority scheduling.
linear code structure **Fixed preemption point models**	Bertogna et al. ['10]: [6] M. Bertogna et al. Preemption points placement for sporadic task sets. Bertogna et al. ['11]: [8] M. Bertogna et al. Optimal selection of preemption points to minimize preemption overhead.

Additional Structure: Loops



(a) Non-Unrolled Loop Example



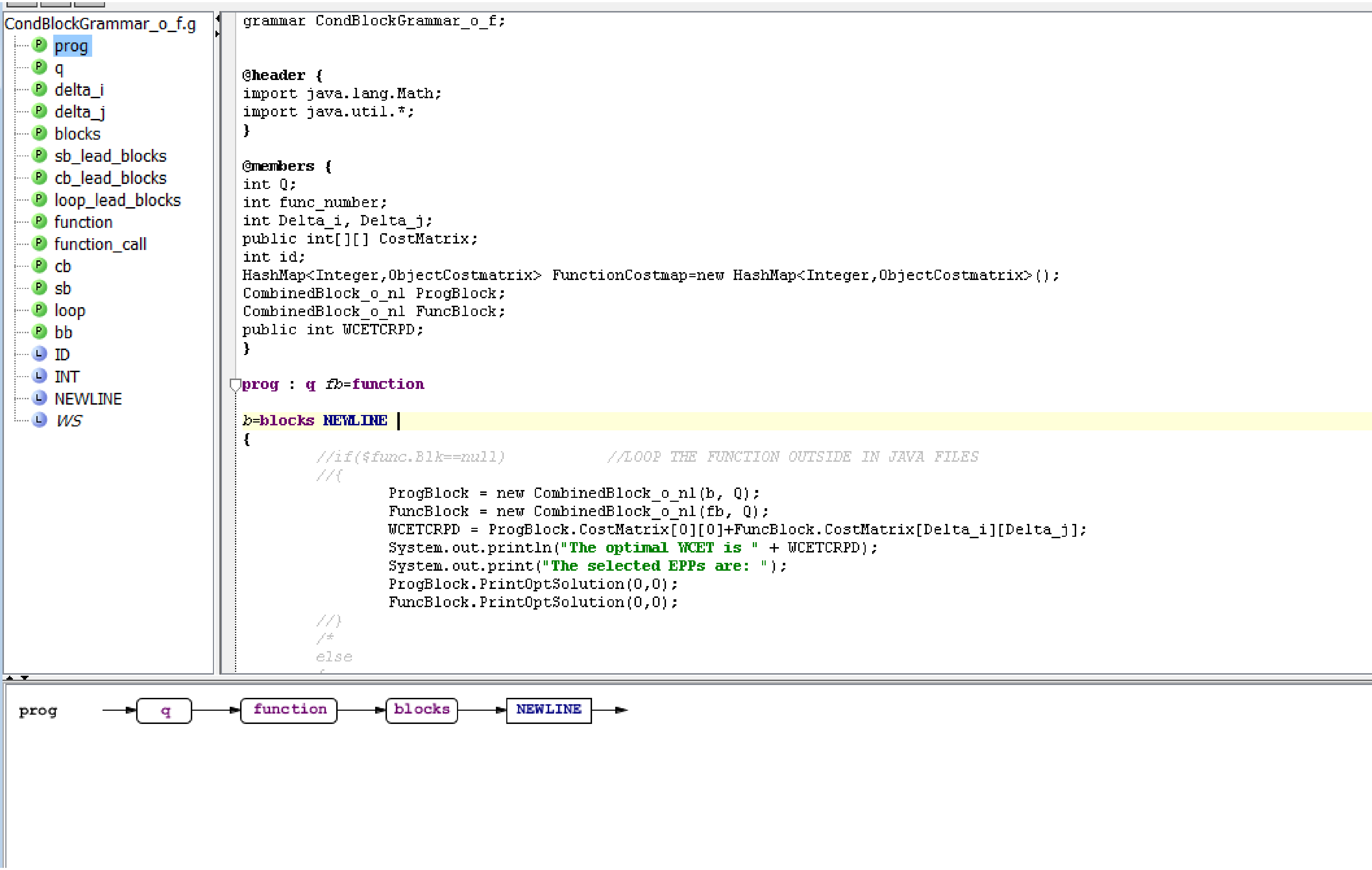
(b) Unrolled Loop Example

(a) Non-Unrolled Loop:

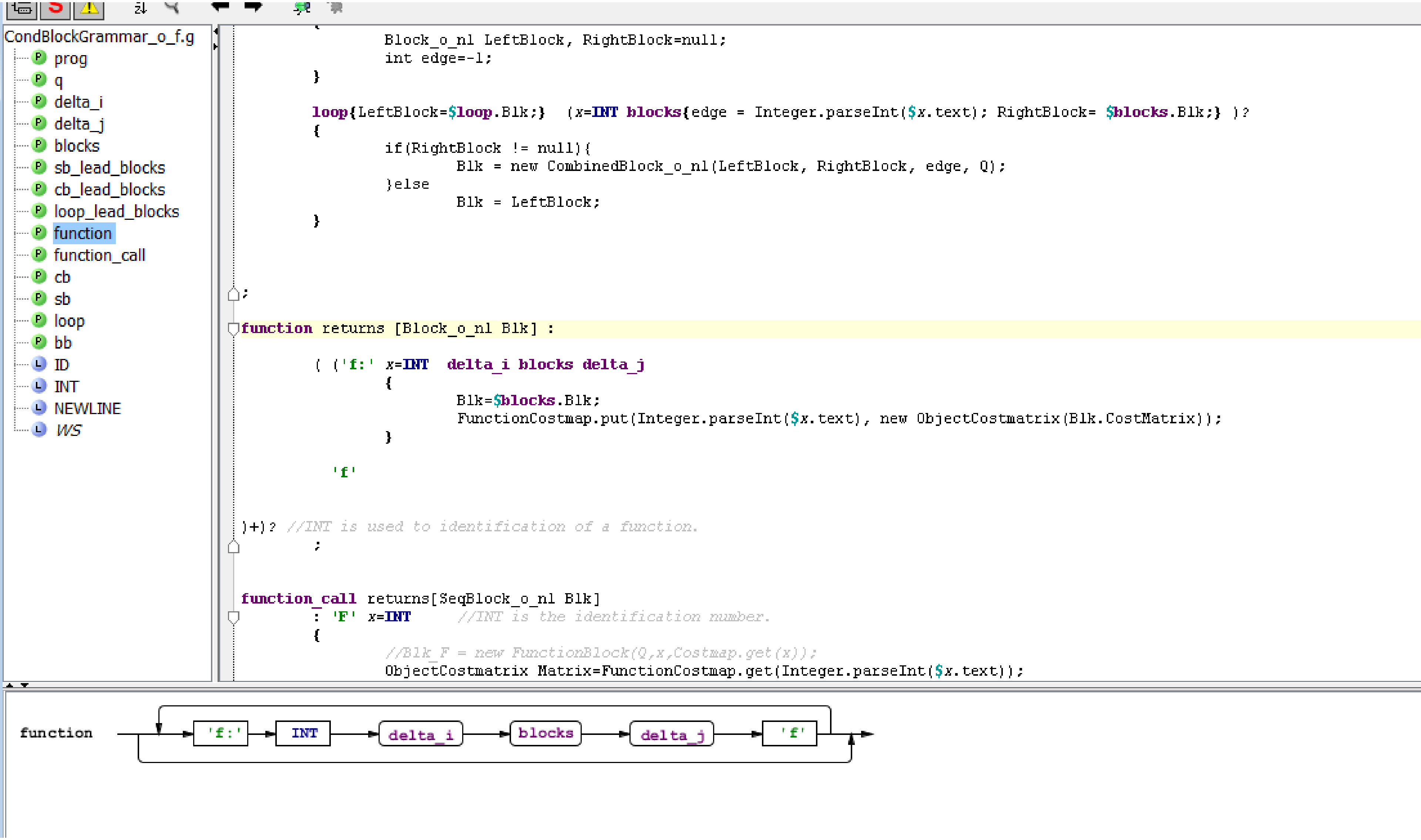
- Whether preempt at e_i, e_j, e_{loop} : eight possible situations
- Choose the smallest one as the value of the cost matrix

(b) Unrolled Loop:

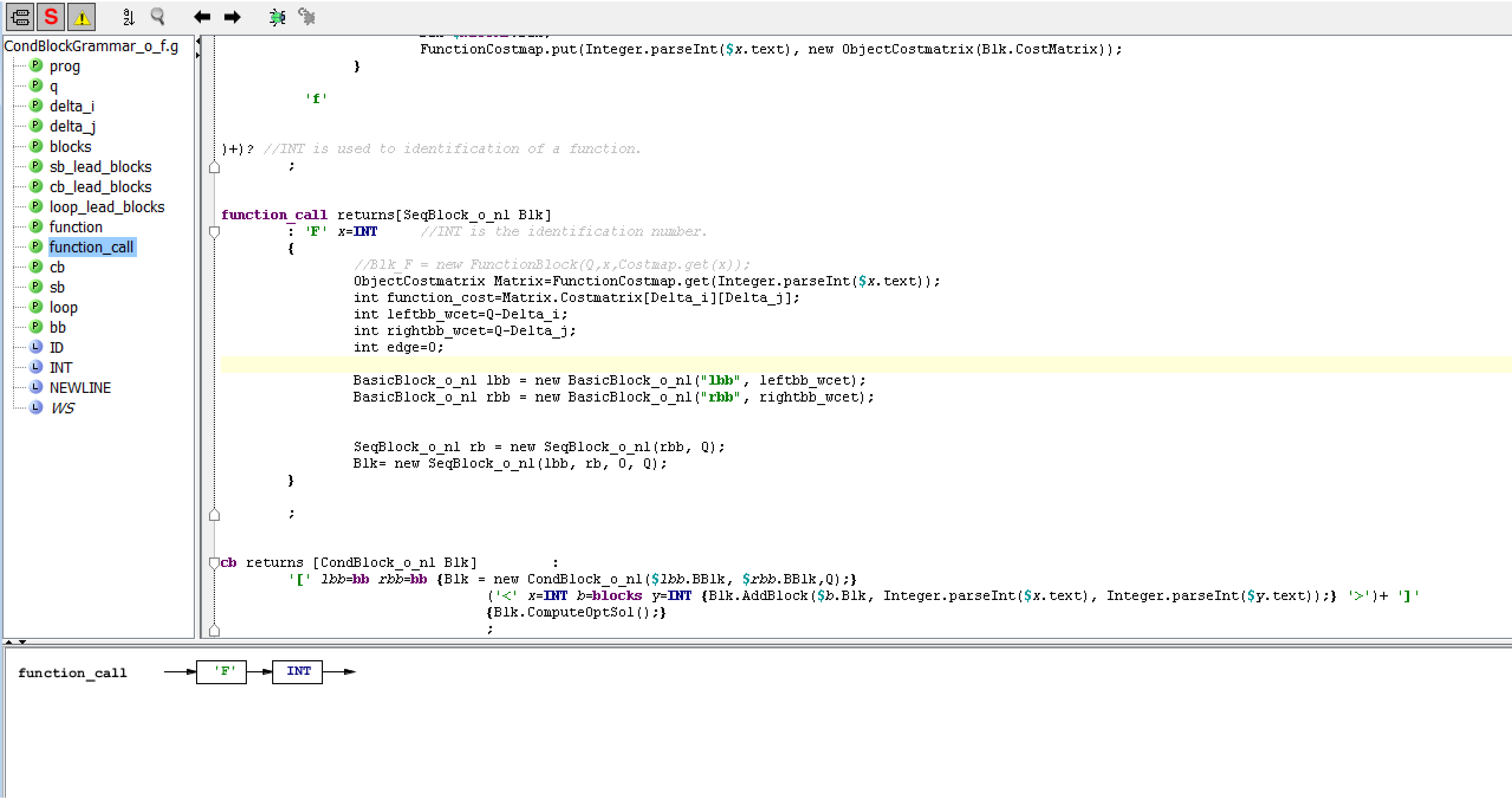
- Preemption places inside the loop is not fixed
- Integrate this structure to conditional structure



```
grammar CondBlockGrammar_o_f;


@header {
import java.lang.Math;
import java.util.*;
}

@members {
int Q;
int func_number;
int Delta_i, Delta_j;
public int[][] CostMatrix;
int id;
HashMap<Integer,ObjectCostmatrix> FunctionCostmap=new HashMap<Integer,ObjectCostmatrix>();
CombinedBlock_o_nl ProgBlock;
CombinedBlock_o_nl FuncBlock;
public int WCETCRPD;
}

prog : q fb=function

b=blocks NEWLINE
{
        //if($func.Blk==null)           //LOOP THE FUNCTION OUTSIDE IN JAVA FILES
        //{
                ProgBlock = new CombinedBlock_o_nl(b, Q);
                FuncBlock = new CombinedBlock_o_nl(fb, Q);
                WCETCRPD = ProgBlock.CostMatrix[0][0]+FuncBlock.CostMatrix[Delta_i][Delta_j];
                System.out.println("The optimal WCET is " + WCETCRPD);
                System.out.print("The selected EPPs are: ");
                ProgBlock.PrintOptSolution(0,0);
                FuncBlock.PrintOptSolution(0,0);
        //}
        /*
        else
```



```
                Block_o_nl LeftBlock, RightBlock=null;
                int edge=-1;
        }

        loop{LeftBlock=$loop.Blk;}  (x=INT blocks{edge = Integer.parseInt($x.text); RightBlock= $blocks.Blk;} )?
        {
                if(RightBlock != null){
                        Blk = new CombinedBlock_o_nl(LeftBlock, RightBlock, edge, Q);
                }else
                        Blk = LeftBlock;
        }



;

function returns [Block_o_nl Blk] :

        ( ('f:' x=INT  delta_i blocks delta_j
                {
                        Blk=$blocks.Blk;
                        FunctionCostmap.put(Integer.parseInt($x.text), new ObjectCostmatrix(Blk.CostMatrix));
                }

          'f'


)+)? //INT is used to identification of a function.
        ;


function_call returns[SeqBlock_o_nl Blk]
        : 'F' x=INT     //INT is the identification number.
        {
                //Blk_F = new FunctionBlock(Q,x,Costmap.get(x));
                ObjectCostmatrix Matrix=FunctionCostmap.get(Integer.parseInt($x.text));
```



```
                        FunctionCostmap.put(Integer.parseInt($x.text), new ObjectCostmatrix(Blk.CostMatrix));
                }

          'f'


)+)? //INT is used to identification of a function.
        ;


function_call returns[SeqBlock_o_nl Blk]
        : 'F' x=INT     //INT is the identification number.
        {
                //Blk_F = new FunctionBlock(Q,x,Costmap.get(x));
                ObjectCostmatrix Matrix=FunctionCostmap.get(Integer.parseInt($x.text));
                int function_cost=Matrix.Costmatrix[Delta_i][Delta_j];
                int leftbb_wcet=Q-Delta_i;
                int rightbb_wcet=Q-Delta_j;
                int edge=0;

                BasicBlock_o_nl lbb = new BasicBlock_o_nl("lbb", leftbb_wcet);
                BasicBlock_o_nl rbb = new BasicBlock_o_nl("rbb", rightbb_wcet);


                SeqBlock_o_nl rb = new SeqBlock_o_nl(rbb, Q);
                Blk= new SeqBlock_o_nl(lbb, rb, 0, Q);
        }

        ;


cb returns [CondBlock_o_nl Blk]         :
        '[' lbb=bb rbb=bb {Blk = new CondBlock_o_nl($lbb.BBlk, $rbb.BBlk,Q);}
                                ('<' x=INT b=blocks y=INT {Blk.AddBlock($b.Blk, Integer.parseInt($x.text), Integer.parseInt($y.text));} '>')+ ']'
                                {Blk.ComputeOptSol();}
                                ;
```