Decomposition based Methods for Allocation and Scheduling Problems arising in System Design

Luca Benini
*Michele Lombardi
Michela Milano
Martino Ruggiero

DEIS, University of Bologna

Talk topic & outline

Talk topic

- Context: CAD tools for applications running on MPSoCs...
- ...solution techniques for a core optimization problem (resource allocation & scheduling)
- Practical example: a CAD tool for the Cell BE platform

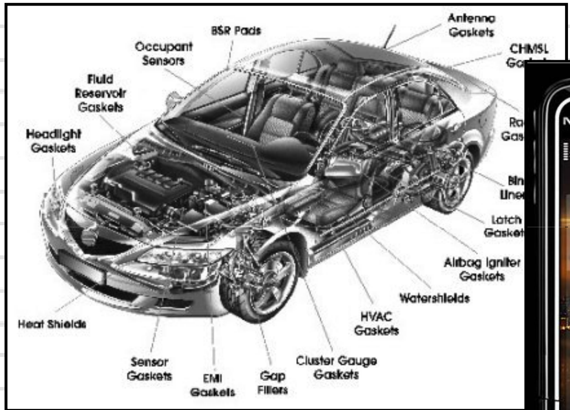
Outline

- Context & Problem description
- Solution ingredients
- Solution methods
  - Multi Stage Logic based Benders' Decomposition
  - Constraint Programming
  - A hybrid solver

1. Context & problem

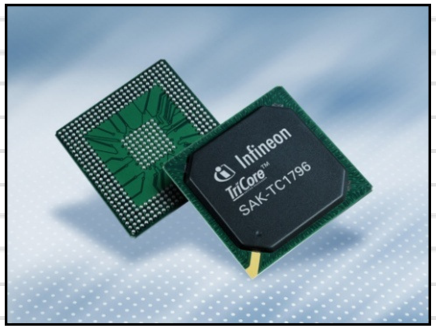
Context

Embedded System:
an information processing device embedded into another product



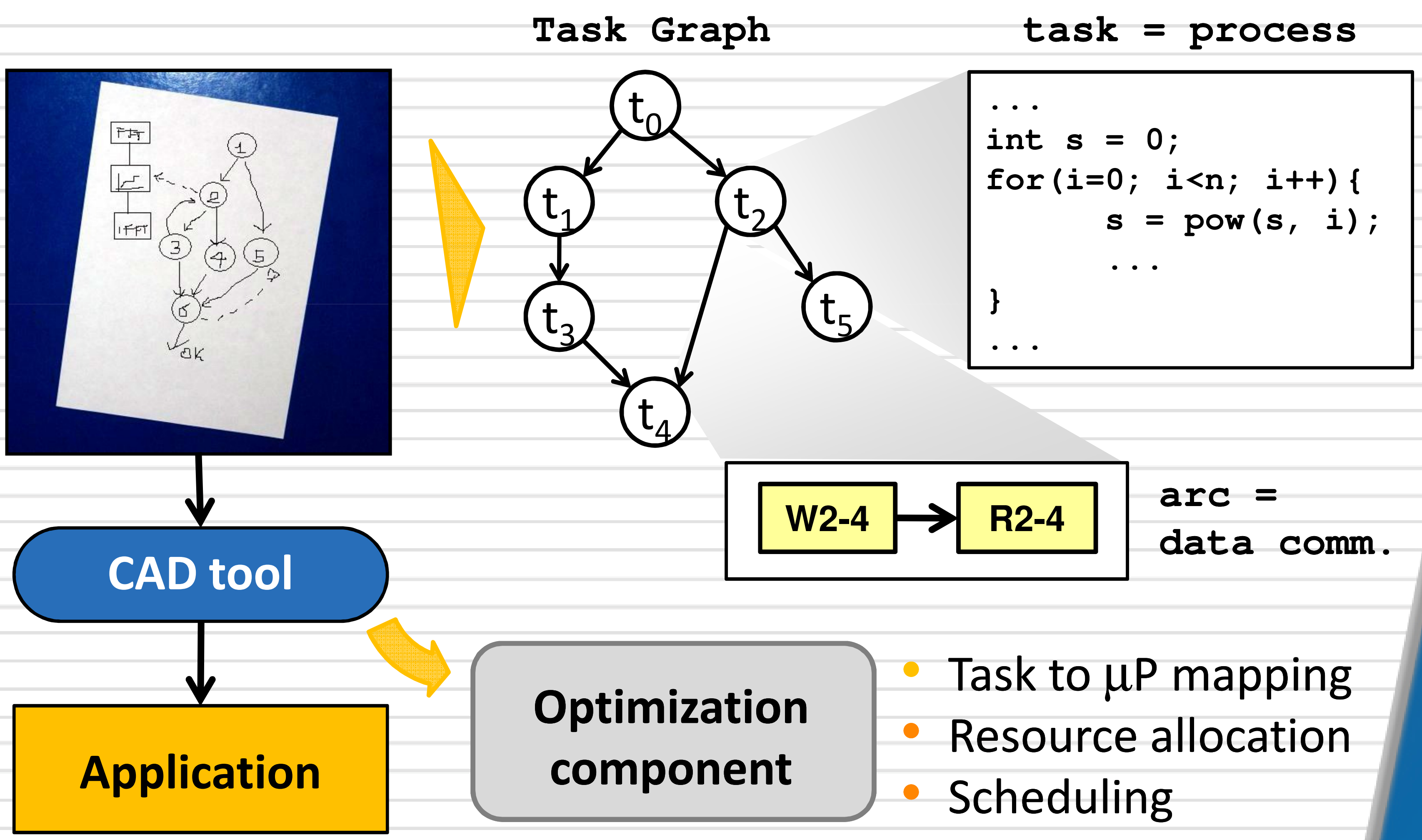
- Often dedicated to a specific application
- Often operate real time tasks
- Parallelism to contain energy consumption

Multicore platforms



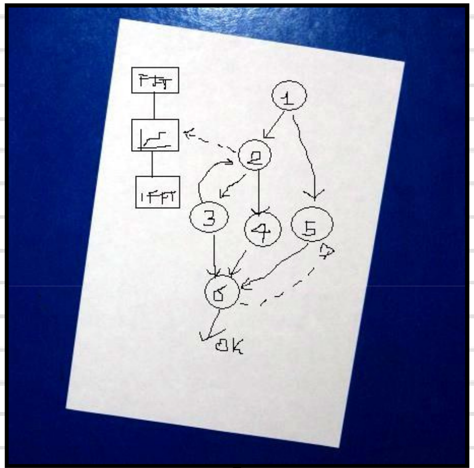
Design flow

It's tricky to exploit parallelism! ⇨ **Need for design tools**

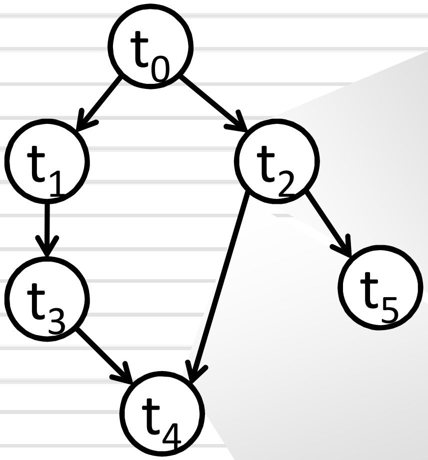


Design flow

It's tricky to exploit parallelism! ⇒ **Need for design tools**



Task Graph



task = process

```
...
int s = 0;
for(i=0; i<n; i++){
      s = pow(s, i);
      ...
}
...
```

W2-4 → **R2-4**

arc = data comm.

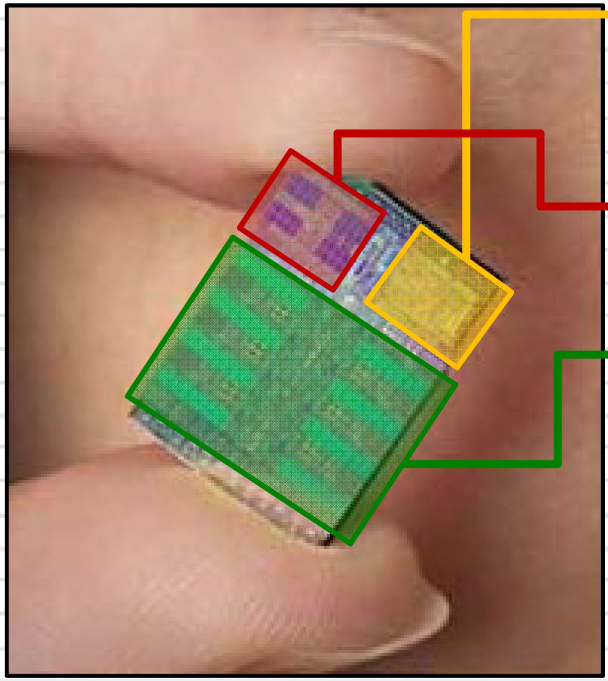
CellFlow → **Application**

Optimization component

- Task to µP mapping
- Resource allocation
- Scheduling

Cell BE processor

Cell Broadband Engine: High performance multicore µP

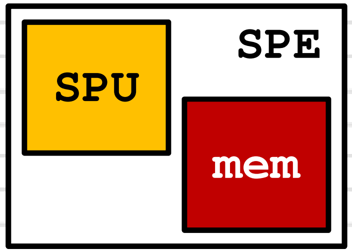


General purpose Power Processing Element (PPE)

On Chip memory (DRAM)

8 Synergistic Processing Elements (SPE)

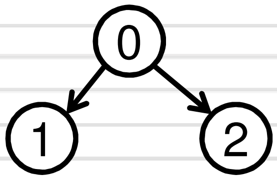
- GP Processors
- Primed by PPE
- Small memory



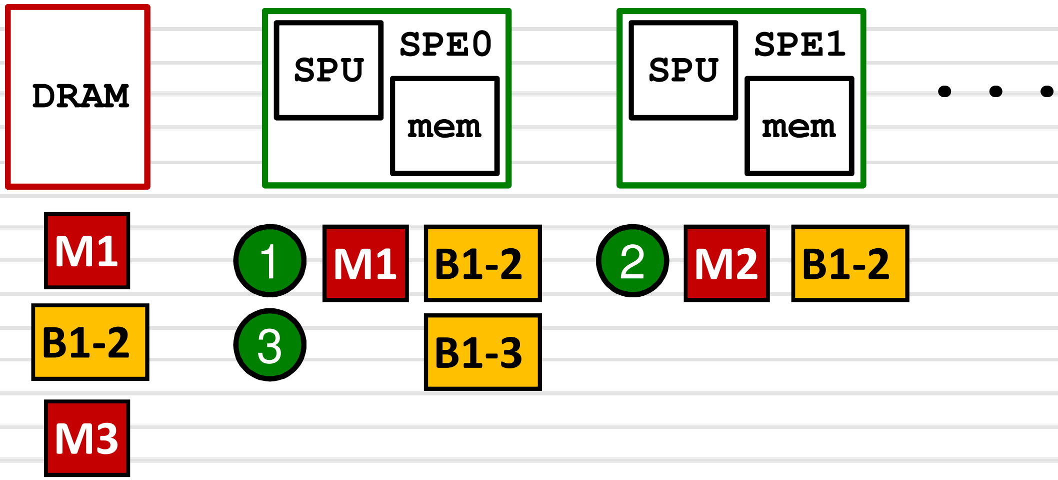
Exploiting the full power of Cell is very tricky

Problem specification

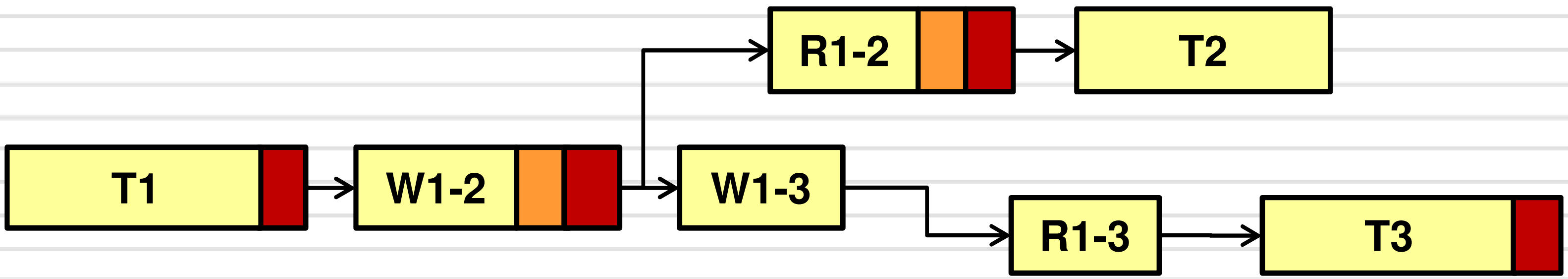
A task graph



The CELL platform

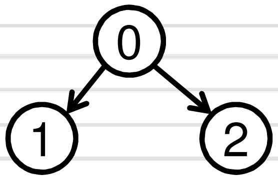


Time behavior

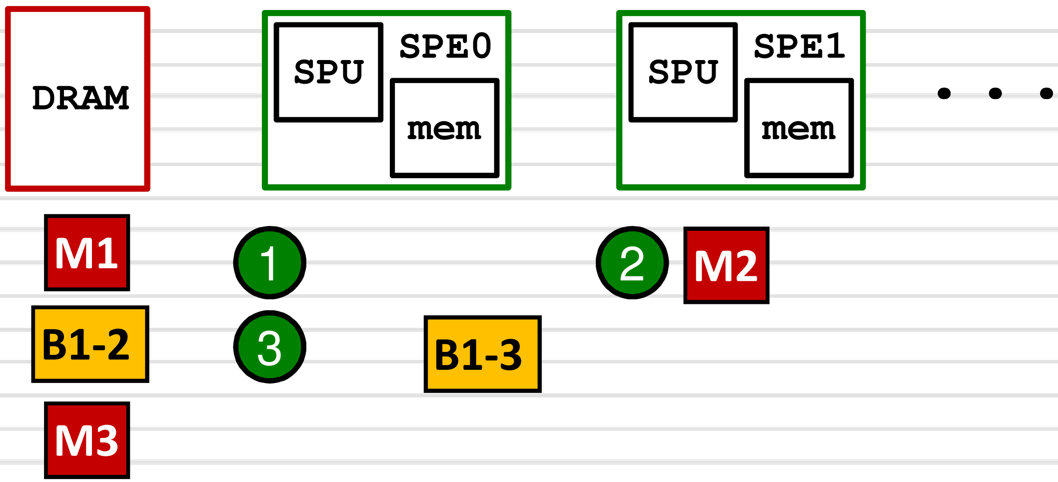


Problem specification

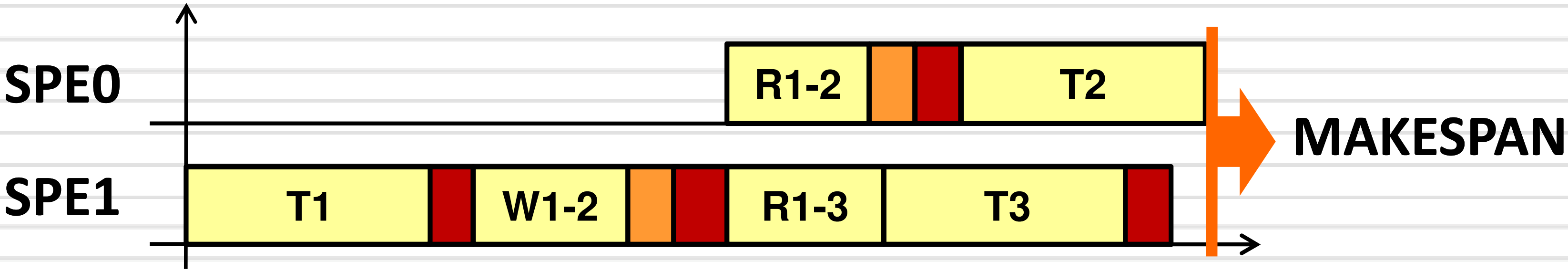
A task graph



The CELL platform



OUTPUT: spe & mem allocation + schedule



Problem specification

A task graph

```
@TASK_GRAPH 0 {
          PERIOD 1000000

          TASK t0_0 TYPE 0
          TASK t0_1 TYPE 1
          TASK t0_2 TYPE 2

          ARC a0_0 FROM t0_0 TO t0_1 TYPE
0
          ARC a0_1 FROM t0_0 TO t0_2 TYPE
1
}

@TRANS 0 {
#type comm rd_ls rd_lr rd_rr wr_ls wr_lr
wr_rr
 0 8033 171 212 248 113 152 190
 1 3468 135 163 218 105 153 195
}

@PE 0 {
#type version dur ext_dur comp_mem
 0 0 1708 1827 1213
 1 0 1837 1947 2143
 2 0 1240 1351 4807
}
```

The CELL platform

```
@PLATFORM {
  # clock  bus_bandwidth
  1        1000
  #id                capacity
  0                  100000
  1                  100000
  2                  100000
  3                  100000
  4                  100000
  5                  100000
}
```

2. Solution “ingredients”

Solution ingredients: MILP

- It's a declarative programming paradigm
- Problem =

Linear objective function

$$\begin{aligned} \min \quad & z = c^T x \\ \text{s.t.} \quad & Ax \geq b \\ & x_i \geq 0 \\ & x_i \text{ integer} \quad \forall i \in I \end{aligned}$$

Linear constraints

Efficient algorithms are available

- Simplex, interior point
- Branch & bound, branch & cut
- ...

Introduction to Constraint Programming

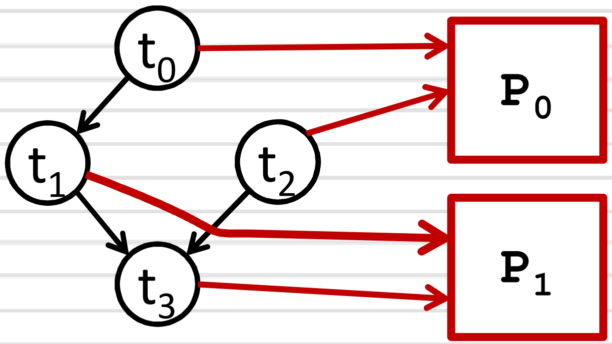
- It's another declarative programming paradigm
- Mainly targets Constraint Satisfaction Problems (NP-hard)

CSP = <X, D, C>

- **X** is a set of variables
- **D** is the set of their domains
- **C** is a set of constraints (any!)

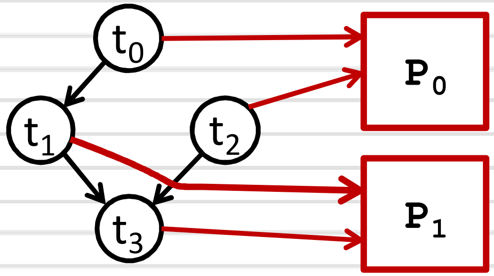
All constraints embed a filtering algorithm to remove inconsistencies

A sample problem: scheduling a TG



- No communications
- Each task has a fixed duration dur_i
- Each task uses a processor
- Deadline **dl**

Modeling with CP



CSP = <X, D, C>

For each task: dur_i

Deadline: dl

Variables & domains:

$START_i \in \{0, ..., dl\}$

$END_i \in \{0, ..., dl\}$

Constraints:

$\forall i \; END_i = START_i + dur_i$

$END_0 \leq START_1, \; END_1 \leq START_3$

$END_2 \leq START_3$

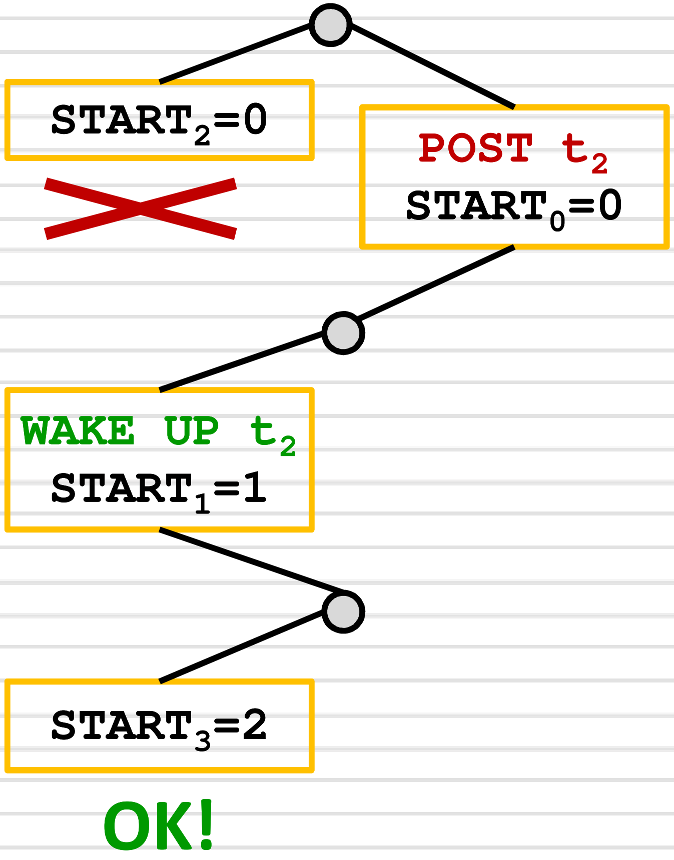
$cumulative([START_0, START_2], [dur_0, dur_2], [1,1], 1)$

$cumulative([START_1, START_3], [dur_1, dur_3], [1,1], 1)$

Global Constraints

Solving a CP problem

Depth First Search



Optimality via a sequence of feasibility problems

Constraints:

$\forall i\ END_i = START_i + dur_i$

$END_0 \leq START_1,\ END_1 \leq START_3$

$END_2 \leq START_3$

cumulative([$START_0$, $START_2$]...)

cumulative([$START_1$, $START_3$]...)

Data:

$dur_0=1,\ dur_1=1,\ dur_2=2,$

$dur_3 = 1,\ dl = 4$

Domains:

$START_0 \in \{0, 1, 2, 3, 4\}$

$END_0 \in \{0, 1, 2, 3, 4\}$

$START_1 \in \{0, 1, 2, 3, 4\}$

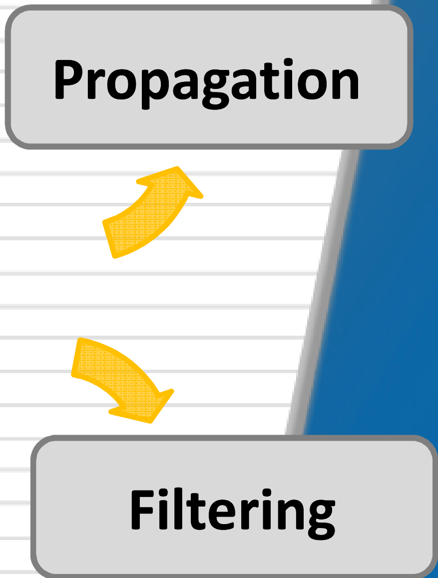
$END_1 \in \{0, 1, 2, 3, 4\}$

$START_2 \in \{0, 1, 2, 3, 4\}$

$END_2 \in \{0, 1, 2, 3, 4\}$

$START_3 \in \{0, 1, 2, 3, 4\}$

$END_3 \in \{0, 1, 2, 3, 4\}$



3. Solution methods

#1: Logic Based Benders' Decomposition

Problem features

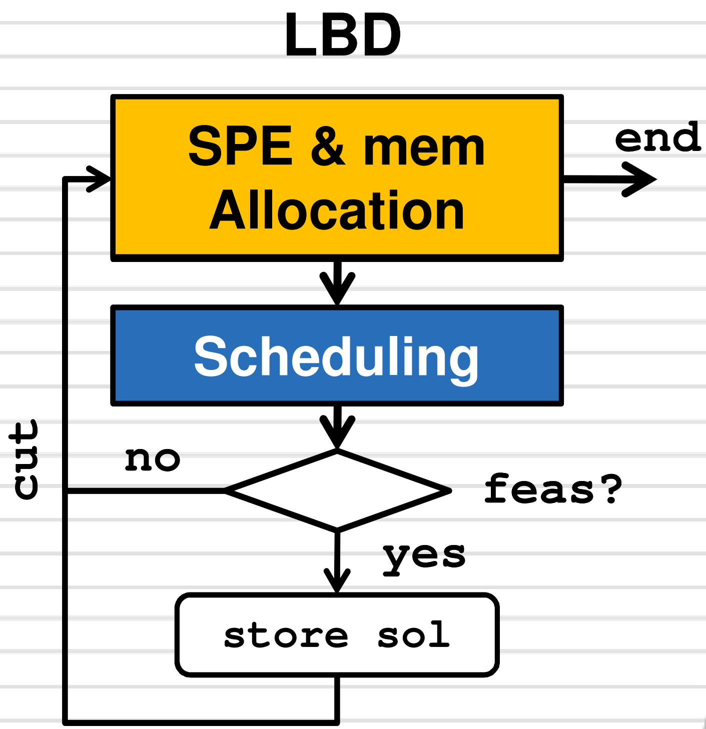
- Large allocation decision space → $\approx n^8 \times 2^n \times 2^{\frac{n}{2}} \times 3^{\frac{n}{2}}$
- Makespan objective → **Good back-propagation**

Tackle with OR techniques

Tackle with CP

Use LBD

- Hooker ('95)
- Allows to mix solvers
- SubP relaxations
- CUTS



#1: Logic Based Benders' Decomposition

Problem features

- Large allocation decision space
- Makespan objective

$T(A) > 10^4 T(S)$!

Integer linear model

T_{ij} = 1 if t_i on SPE_j

M_{ij} = 1 if comp. data on SPE_j

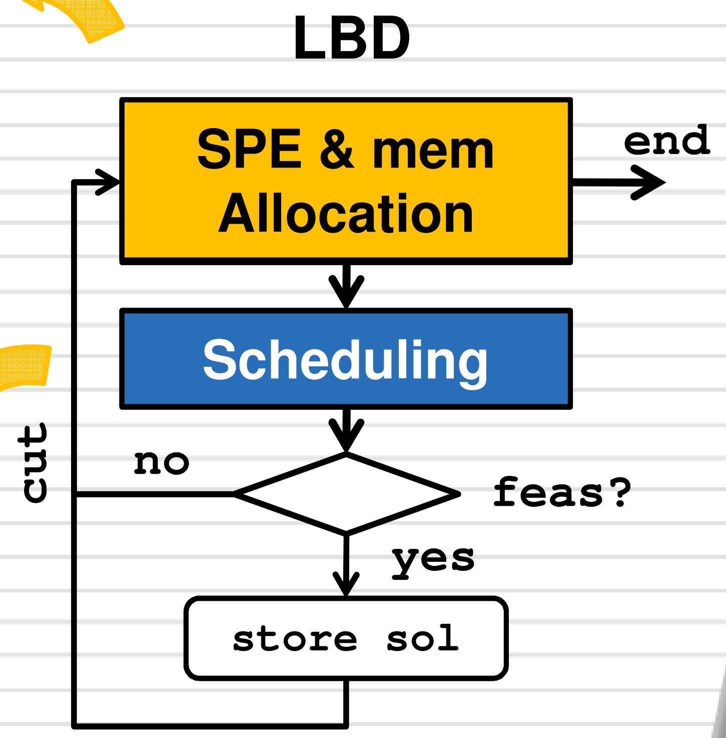
R_{rj} = 1 buffer local to reader

W_{rj} = 1 buffer local to writer

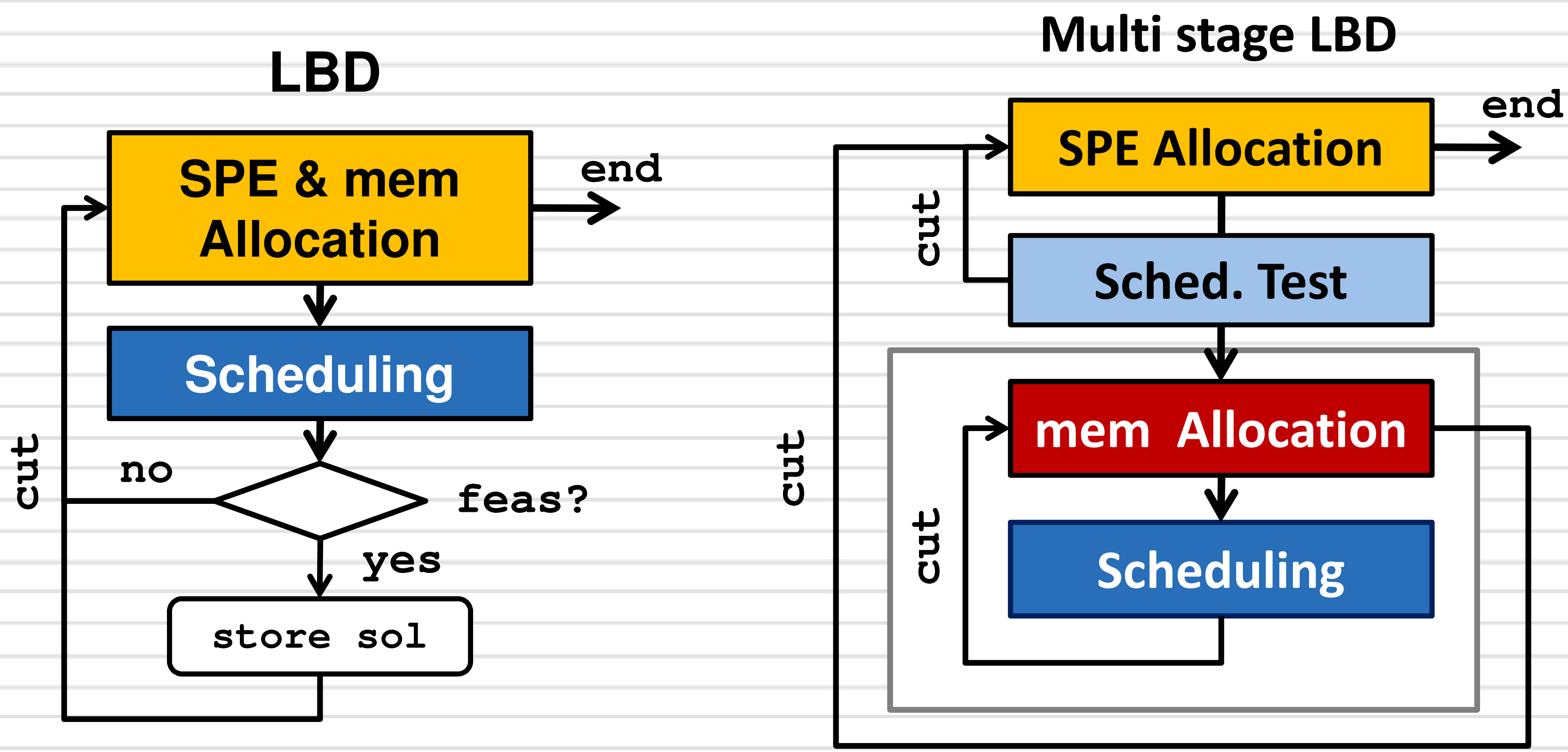
CP model

START, END vars

Cumulative constraints



Multistage LBD



- **PRO:** - much simpler and easier subproblems
- **CON:** - hard to design good high level objective functions
  - loosely couple subproblems

Cuts for complex problems

Most basic cut: nogood

σ: $T_{00}=1$ | $T_{01}=0$ | $T_{10}=1$ | . . . | $M_{00}=1$ | $M_{01}=0$ | . . . | $R_{00}=1$ | . . . | $W_{00}=1$

$$\sum_{\sigma(T_{ij})=1}(1-T_{ij}) + \sum_{\sigma(M_{ij})=1}(1-M_{ij}) + \sum_{\sigma(M_{ij})=0} M_{ij} + \ldots > 0$$

TOO WEAK ▶ strenghten via a refinment procedure

- Select a subset of decisions (set of vars)
- Re
- If s

$T_M = \alpha\ T_S \Rightarrow$ **SP can be safely solved α times**

- Iterative methods: de Siqueira and Puget, Junker
- **Requires to solve *O(n*log(n))* relaxed NP-hard problems**

#2: pure CP approach

As for the CONS?

- Best solutions are found late
- Bad makespan estimantion at higher levels

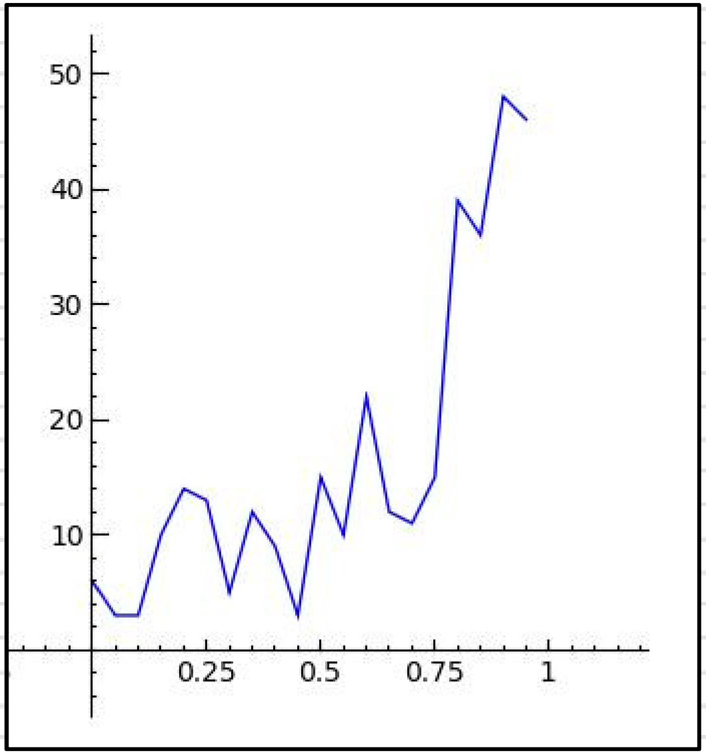
Pure CP approach

- No decomposition
- Exploit propagation to compute makespan bounds

In particular

- Quite sraightforward model
- <u>Carefully designed search strategy</u>

Last impr. it./number of it.



CP Model

- SPE allocation:

  T_i = j iff t_i on SPE_j

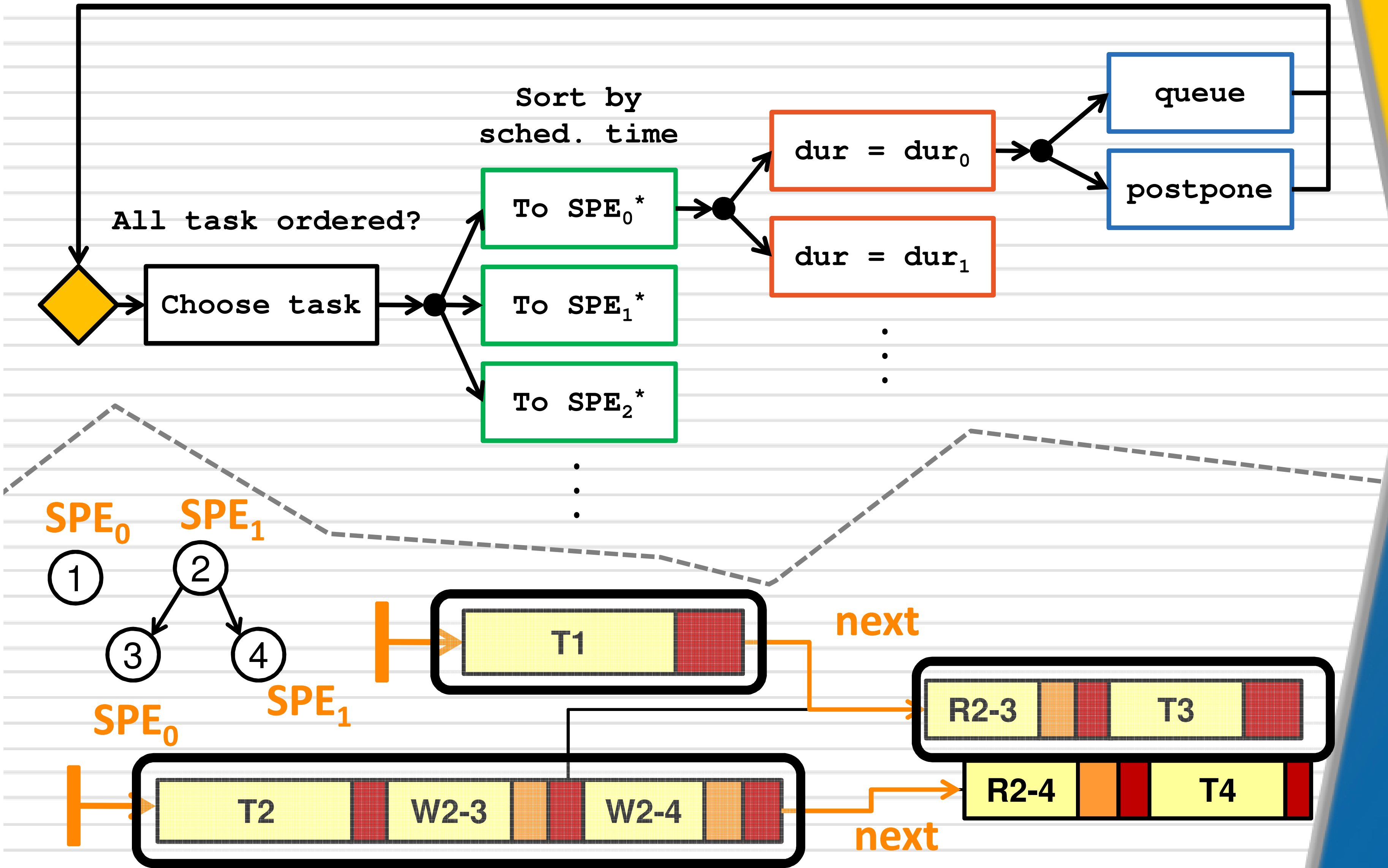
- MEM allocation:

  M_i = 1 iff t_i locally allocates comp. Data
  R_r = 1 iff buffer local to reader
  W_r = 1 iff buffer local to writer
  buffer allocation constraints
  memory capacity constraints

- Scheduling:

  START, END vars for each operation (rd, ex, wr)
  cumulative constraints for SPEs

CP Search strategy

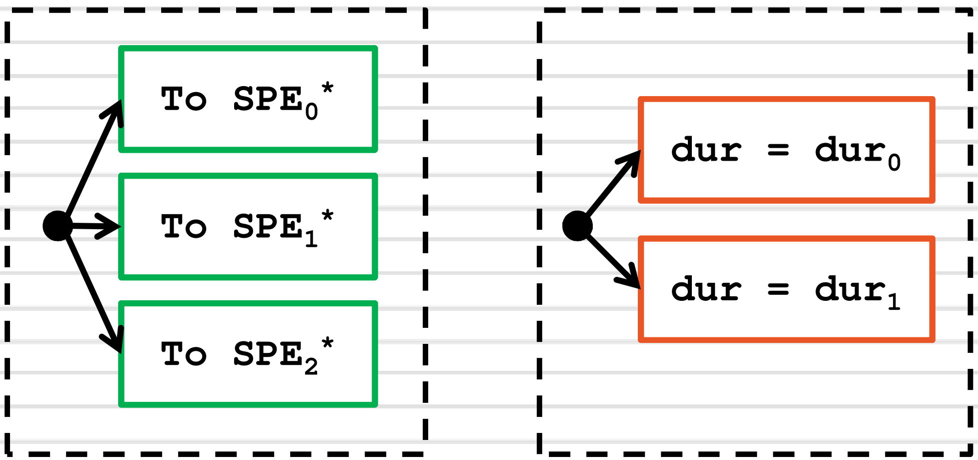


CP Search Extras

Thrashing...

An early bad decision sunks the performance

- randomization
- Frequent restarts



CP cuts

- During CP search cuts are generated and refined
- A basic cut is an infeasible SPE & MEM allocation

Experimental setup

Three groups of instances

1. Real (90) synth benchmark – memory impact negligible)
2. Real-like, communication intensive (100)
3. Real-like, computation intensive (100)

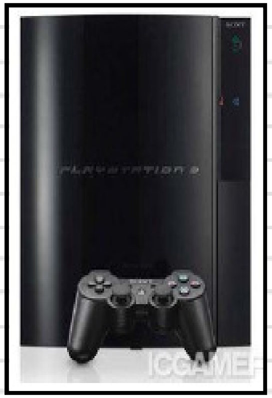
same graph structure, artificial durations and requirements

Negligible communication durations

Very poor impact of memory allocation...

One platform

1. CELL BE
2. 6 SPE available



Graph generator

- Random to realistic struc.
- Attribute dependencies
- TGFF file format

Results for group 1

Memory allocation has no impact on durations

		MS-LBD		CP	
tasks	**arcs**	**> TL**	**time**	**> TL**	**time**
15	9-14	0	0.42	0	0.01
15	14-27	0	0.57	0	0.01
25	30-56	1	80.88	0	0.12
25	56-66	2	274.39	0	0.07
30	47-72	5	354.81	1	0.90
30	73-83	7	280.02	0	0.41

Results for group 2

Memory allocation has strong impact on durations

		MS-LBD		CP	
tasks	arcs	> TL	time	> TL	time
10-11	6-14	0	4.01	4	4.68
12-13	8-15	0	6.32	5	0.06
14-15	12-19	0	5.54	5	180.16
16-17	15-22	0	28.35	6	226.66
18-19	17-24	0	105.50	7	10.74
20-21	21-29	1	210.89	10	---
22-23	21-30	2	388.00	9	250.00
24-25	24-35	3	268.57	9	85.00
26-27	27-39	4	375.00	8	160.49
28-29	32-43	5	528.00	6	432.25

Results for group 3

Buffer allocation has no impact on durations

		MS-LBD		CP	
tasks	arcs	> TL	time	> TL	time
10-11	6-14	0	0.21	0	0.01
12-13	8-15	0	1.16	0	0.02
14-15	12-19	0	1.00	0	0.03
16-17	15-22	0	10.89	0	1.53
18-19	17-24	0	48.92	0	0.07
20-21	21-29	1	116.10	1	2.70
22-23	21-30	1	69.16	0	64.05
24-25	24-35	3	269.57	0	78.46
26-27	27-39	7	88.67	3	66.50
28-29	32-43	8	310.00	4	425.50

#3: Hybrid solver

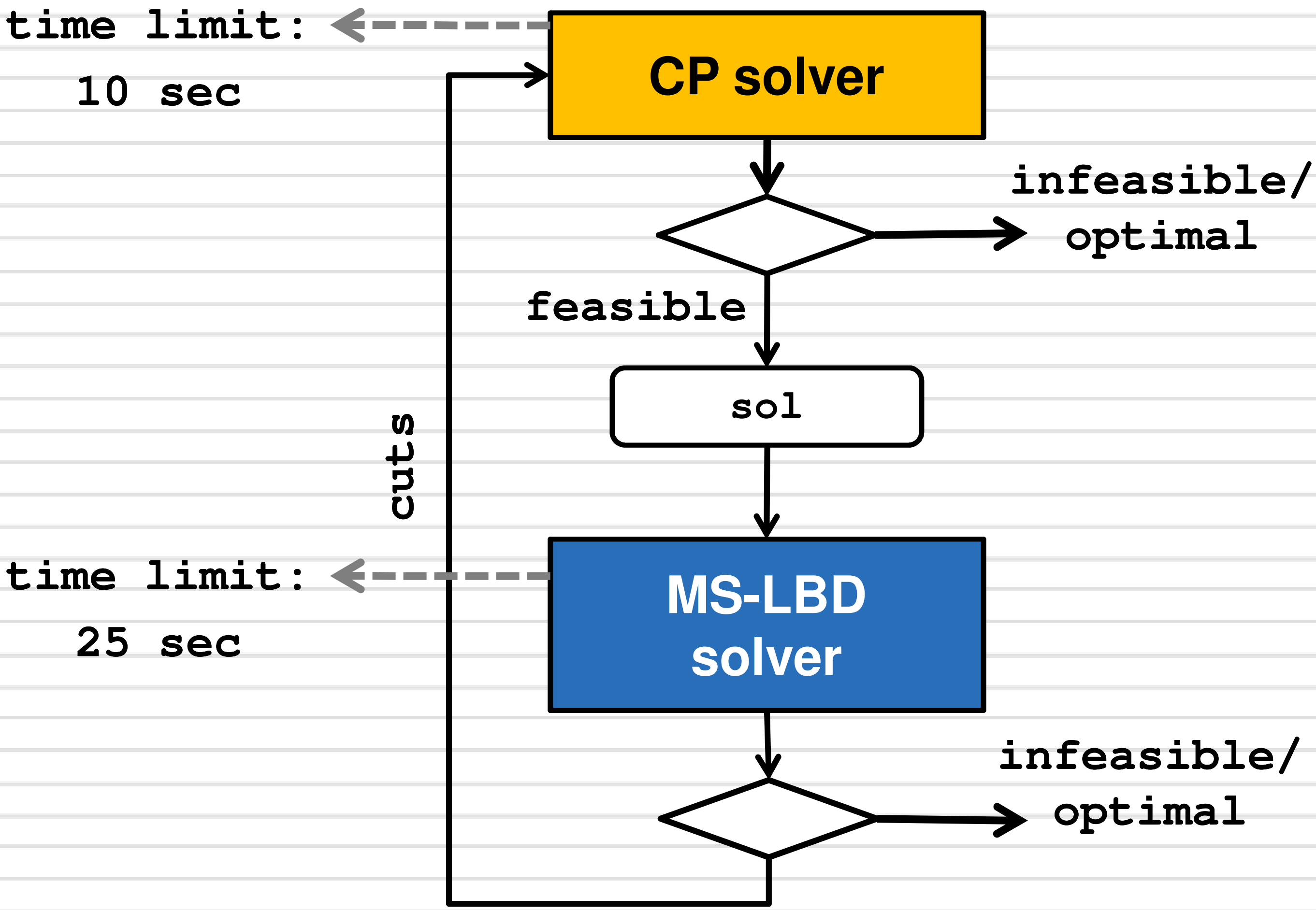
Some considerations

- MS-LBD solver is robust due to the use of strong cuts...
- ...but it cannot effectively exploit makespan bounds

- CP solver can exploit and compute makespan bounds...
- ...it seems more capable to quickly find good solutions...
- ...but it cannot deal with buffer allocation

We could build another hybrid solver!

Structure of the new hybrid (LBD like)



- Both solvers can improve solution & prove optimality
- Cuts greatly help the CP solver

Results for group 1

Memory allocation has no impact on durations

		MS-LBD		CP		HYB	
tasks	arcs	> TL	time	> TL	time	> TL	time
15	9-14	0	0.42	0	0.01	0	0.01
15	14-27	0	0.57	0	0.01	0	0.01
25	30-56	1	80.88	0	0.12	0	0.13
25	56-66	2	274.39	0	0.07	0	0.08
30	47-72	5	354.81	1	0.90	0	34.46
30	73-83	7	280.02	0	0.41	0	0.42

Results for group 2

Memory allocation has strong impact on durations

		MS-LBD		CP		HYB	
tasks	arcs	> TL	time	> TL	time	> TL	time
10-11	6-14	0	4.01	4	4.68	0	8.03
12-13	8-15	0	6.32	5	0.06	0	6.38
14-15	12-19	0	5.54	5	180.16	0	6.79
16-17	15-22	0	28.35	6	226.66	0	9.22
18-19	17-24	0	105.50	7	10.74	0	54.12
20-21	21-29	1	210.89	10	---	3	52.08
22-23	21-30	2	388.00	9	250.00	5	63.79
24-25	24-35	3	268.57	9	85.00	3	42.71
26-27	27-39	4	375.00	8	160.49	3	168.64
28-29	32-43	5	528.00	6	432.25	5	56.98

Results for group 3

Buffer allocation has no impact on durations

		MS-LBD		CP		HYB	
tasks	arcs	> TL	time	> TL	time	> TL	time
10-11	6-14	0	0.21	0	0.01	0	0.01
12-13	8-15	0	1.16	0	0.02	0	0.02
14-15	12-19	0	1.00	0	0.03	0	0.03
16-17	15-22	0	10.89	0	1.53	0	0.95
18-19	17-24	0	48.92	0	0.07	0	0.08
20-21	21-29	1	116.10	1	2.70	0	36.66
22-23	21-30	1	69.16	0	64.05	0	3.69
24-25	24-35	3	269.57	0	78.46	0	49.65
26-27	27-39	7	88.67	3	66.50	2	133.62
28-29	32-43	8	310.00	4	425.50	3	255.01

Conclusion & future works

In conclusion…

1. A number of approaches to solve an important allocation & scheduling problem arising in CAD tools for MPSoCs
2. Nice CP features: integration & side constraints easily added
3. Hybridization & decomposition can be used to combine the strenghts and minimize the weaknesses of different solvers

Some possible developments…

1. The CELL flowmodel has changed: adapt the best approaches
2. Yet improve the solver (in particular the CP one)
3. Different instances are best tackled with different solvers => machine learning
4. …any suggestion?

Thank you!

Questions?

Decomposition based Methods for Allocation and Scheduling Problems arising in System Design

Luca Benini
*Michele Lombardi
Michela Milano
Martino Ruggiero

DEIS, University of Bologna