Technische Universiteit
Eindhoven
University of Technology

Embedded Systems
INSTITUTE

Reliable Dynamic Embedded Data Processing Systems

Twan Basten

Joint work with
Marc Geilen, AmirHossein Ghamarian,
Hamid Shojaei, Sander Stuijk, Bart Theelen, Yang Yang
and others

Funding:
NWO PROMES
EC FP6 Betsy, FP7 MNEMEE
SenterNovem Octopus

'Knowing is not understanding.'
Charles Kettering

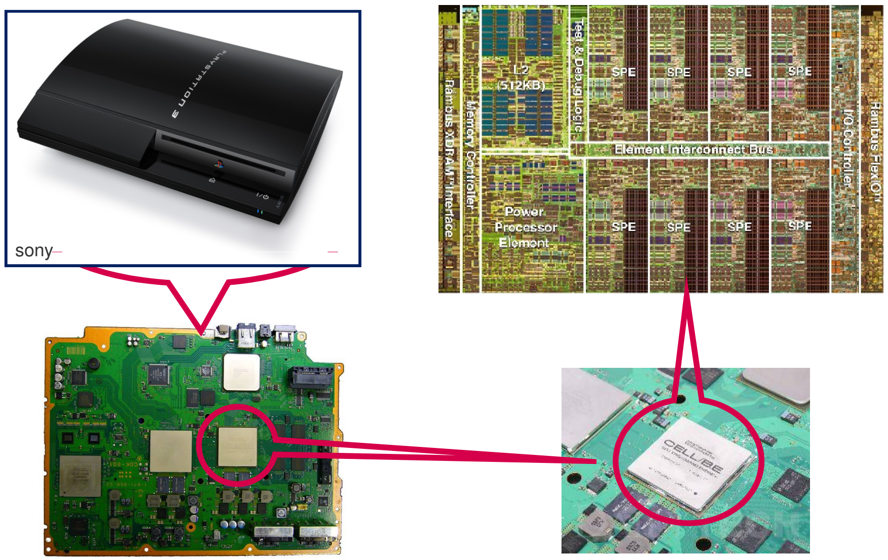
---

2 Embedded Data Processing Systems

---

3 Embedded Data Processing Systems

Embedded Systems INSTITUTE



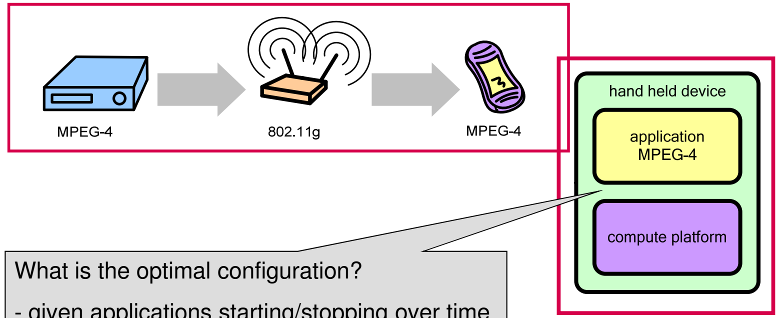
---

4 Embedded Data Processing Systems

Embedded Systems INSTITUTE

Trend: Dynamic Behavior

- Multiprocessor systems / networking / concurrency
- Application convergence / application evolution



What is the optimal configuration?

- given applications starting/stopping over time
- given data-dependent execution times
- given varying communication bandwidth
- given battery status
- …

- **Intra**-application dynamism
- **Inter**-application dynamism

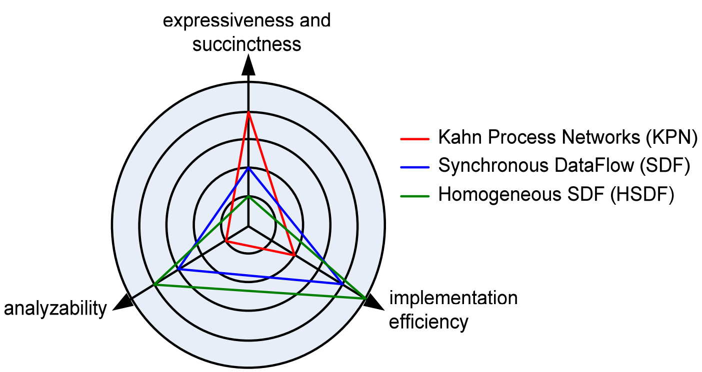
Intra-application dynamism

Dataflow analysis

Dataflow Analysis: Challenges

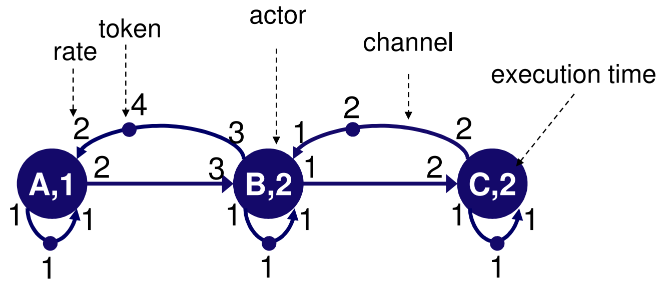
predictability and efficiency

… are contradictory



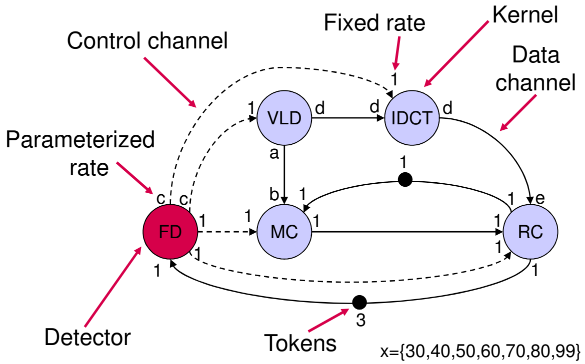
Synchronous Data Flow Graphs

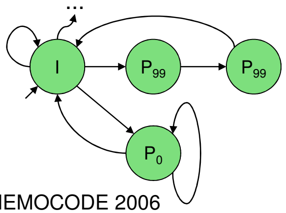
(SDFGs [Lee 1986])



9 Scenario-aware MPEG-4 SP Decoder Model

(FSM-based Scenario-Aware DataFlow (SADF))





MEMOCODE 2006

x={30,40,50,60,70,80,99}

Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

Execution time		
VLD	P_0	0
	others	40
IDCT	P_0	0
	others	17
MC	I, P_0	0
	P_{30}	90
	P_{40}	145
	P_{50}	190
	P_{60}	235
	P_{70}	265
	P_{80}	310
	P_{99}	390
RC	I	350
	P_0	0
	P_{30}, P_{40}, P_{50}	250
	P_{60}	300
	P_{70}, P_{80}, P_{99}	320
FD	All	0

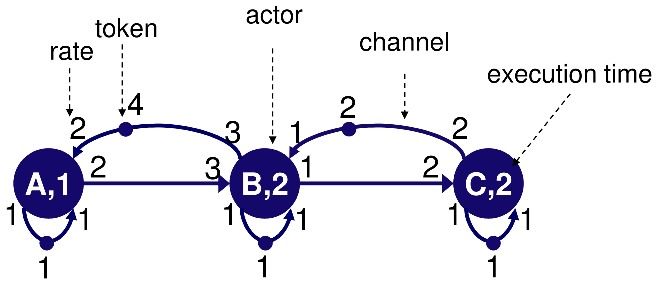
10

Throughput analysis for SDF

ACSD 2006

11 Synchronous Data Flow Graphs

(SDFGs [Lee 1986])

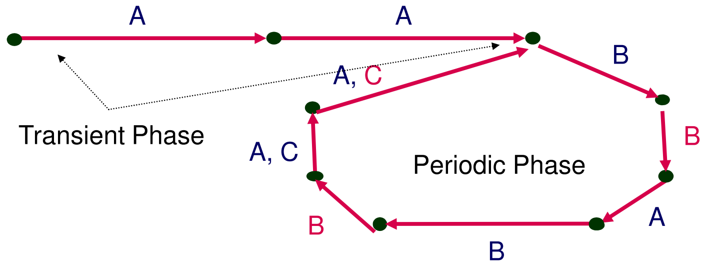


Throughput: average number of actor firings over time

Iteration: smallest non-empty set of actor firings that does not change the token distribution (example: A:3, B:2, C:1)

12 SDF Throughput Analysis

Self-timed execution:



Throughput can be calculated from the periodic phase

Efficient implementation
Consider one designated firing per iteration only to detect a recurrent state

13 Throughput Analysis: Our Result

	State Space	Dasdan Gupta	Howard	Young Tarjan Orlin
		Traditional methods		
MP3 dec.	1. 10^{-3}	1. 10^{-3}	1. 10^{-3}	1. 10^{-3}
Modem	1.10^{-3}	82. 10^{-3}	81. 10^{-3}	81. 10^{-3}
Sample Rate	2. 10^{-3}	>1800	>1800	>1800
Satellite	56. 10^{-3}	>1800	>1800	>1800
H.263 decoder	10. 10^{-3}	>1800	>1800	>1800

Runtimes of various throughput analysis methods (in seconds)

14

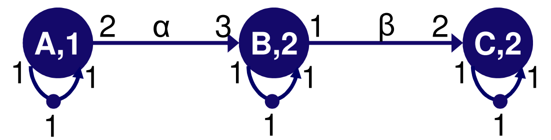
Storage-throughput trade-offs

IEEE T Comp 2008

15 Lessons to be Learned

- Repeated application of throughput analysis
- Causal dependencies potentially reducing performance

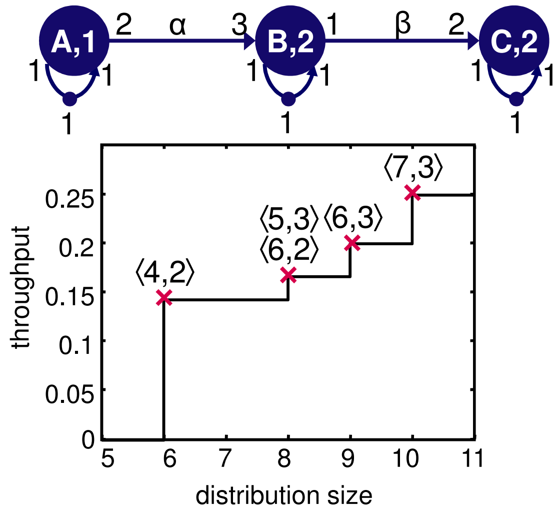
16 Trade-off: Storage vs. Throughput



Tokens on channels must be stored in memory.

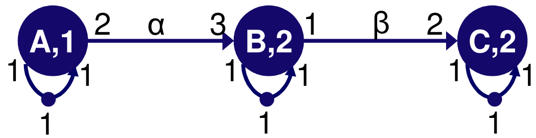
- Separate memory for each channel.
- Storage distribution, for example, ⟨α, β⟩ → ⟨4,2⟩

17 Trade-off: Storage vs. Throughput



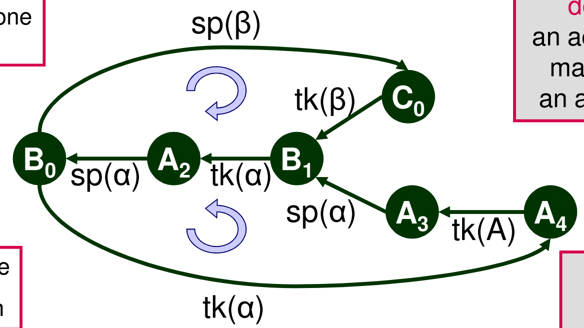
Find all minimal storage distributions for any possible throughput

18 Dependency Graph



Storage distribution ⟨α, β⟩ → ⟨4,2⟩

firings in one period



dependency: an actor firing start may depend on an actor firing end

sp - space

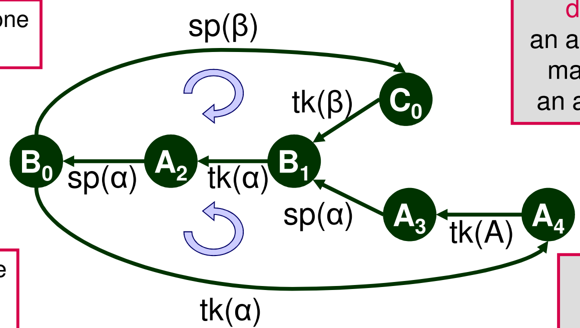
tk - token

cycles denote storage dependencies

19 Dependency Graph

resolving storage dependencies may increase throughput

firings in one period



dependency: an actor firing start may depend on an actor firing end

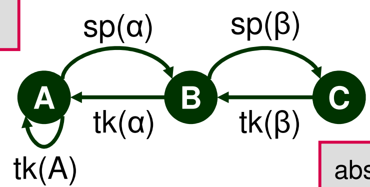
sp - space

tk - token

cycles denote storage dependencies

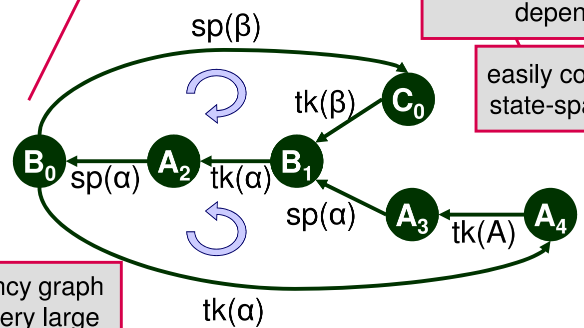
20 Abstract Dependency Graph

dependencies between actors



abstract dependency graph contains all storage dependencies

easily constructed from state-space exploration



dependency graph may be very large

21 Experimental Results

	Example	Satellite	MP3	H.263
#actors	3	22	13	4
#channels	2	26	12	3
min throughput > 0	1/7	0.18	1/137	1/21876
Distr. size	6	1542	12	4753
Max through				1/10000
Distr. size				8006
#Pareto points	4	2	3	3254
#Distributions	5	2	3	3254
Exec. time	1ms	7ms	2ms	53min

Approximation

increase buffers with n times
the minimal step size

22 Trade-off Analysis

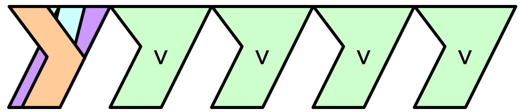
- Start small, recursively resolve bottlenecks
- Fast in general, approximation possible
- Generalizable to other resources, computational models

23

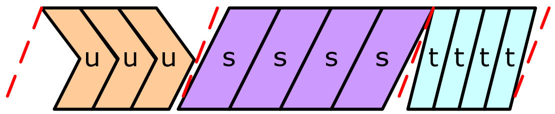
Scenarios

24 Scenario-aware Throughput Analysis

- SDF throughput analysis considers worst-case actor execution times



- Scenario-aware throughput analysis considers worst-case execution times per scenario, but it must consider scenario transitions

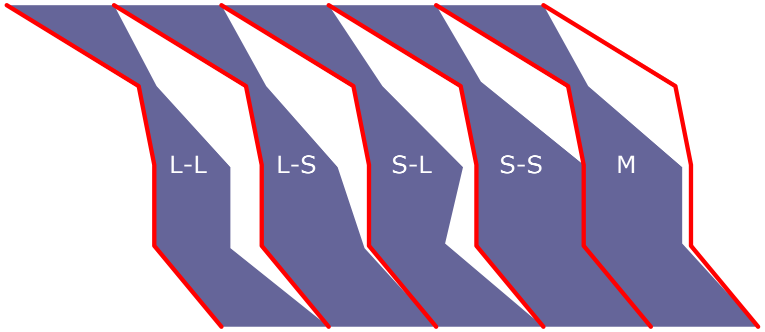


- Analyzing all scenario transitions separately can be avoided
- Separate scenario transitions with an invariant reference schedule

ACM TECS 2010

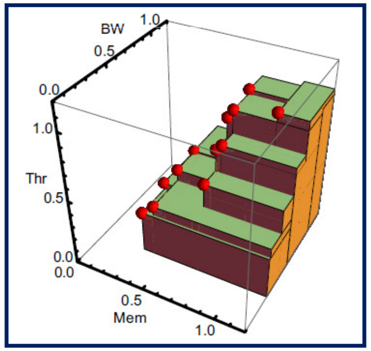
MP3 Result

- SDF model: 9.28 Mcycles per frame
- SADF model: 5.83 Mcycles per frame



26 Pointers

- Yang Yang: shared resources, trade-off analysis
- Sander Stuijk: SDF3 toolset, http://www.es.ele.tue.nl/sdf3
- Future: shared resources + scenarios + trade-off analysis

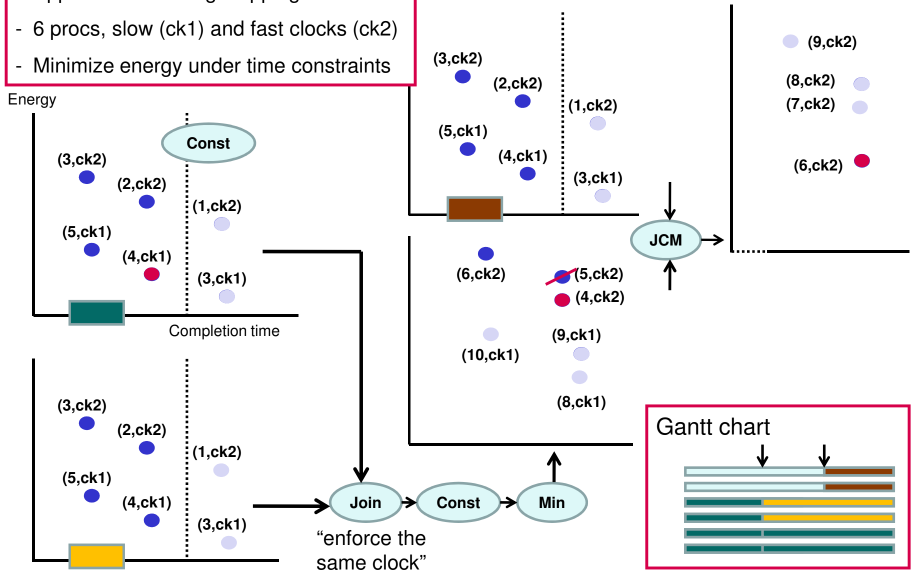


Inter-application dynamism

Run-time adaptation

28 Run-time Application Management

- Applications starting/stopping over time
- 6 procs, slow (ck1) and fast clocks (ck2)
- Minimize energy under time constraints



29 Pareto Algebra

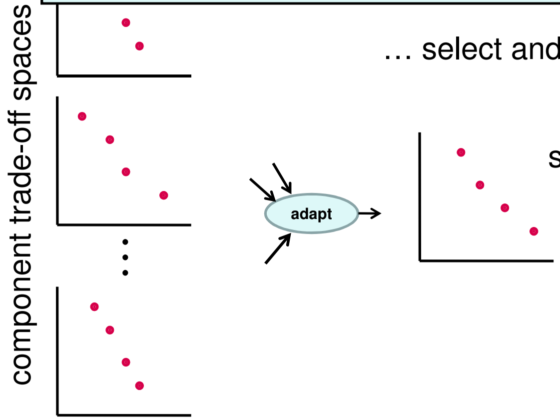
Goal: Compositional computation of trade-offs

30 Pareto-Algebraic Characterization

… of run-time adaptation

product - derive | constrain | abstract - minimize

… select and configure



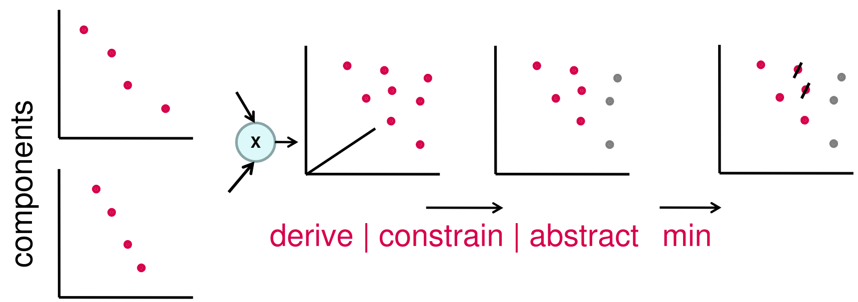
system trade-off space
(at the time of a change)

31 Pareto-Algebraic Characterization

… of run-time adaptation

product - derive | constrain | abstract - minimize

… select and configure

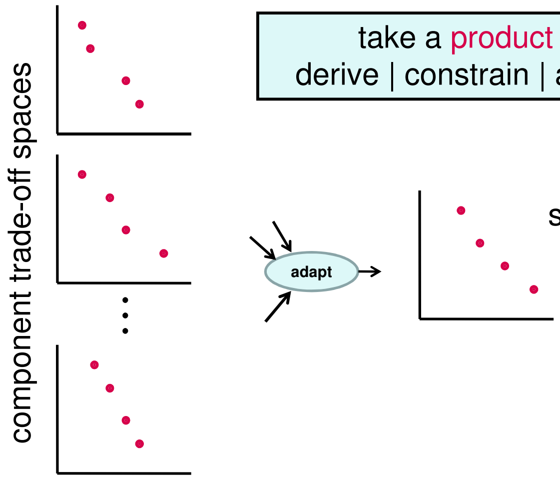


32 Complexity

n components with p Pareto points each

take a product and on-the-fly
derive | constrain | abstract - minimize

$O(p^{2n})$



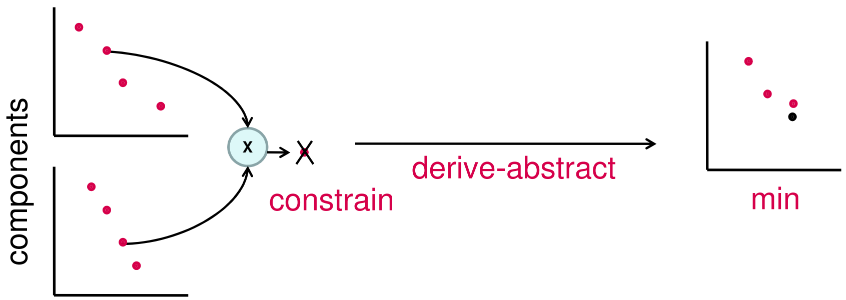
system trade-off space
(at the time of a change)

33 Complexity

n components with p Pareto points each

take a product and on-the-fly
derive | constrain | abstract - minimize

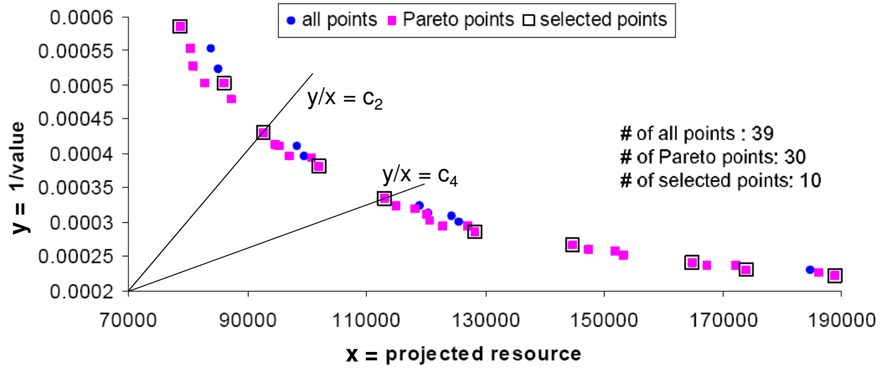
$O(p^{2n})$



34 Complexity Control

keep n and p small

- Compositional reasoning, among others
- Approximate Pareto sets



35 Multi-dimensional Multiple-choice Knapsack Problem

MMKP

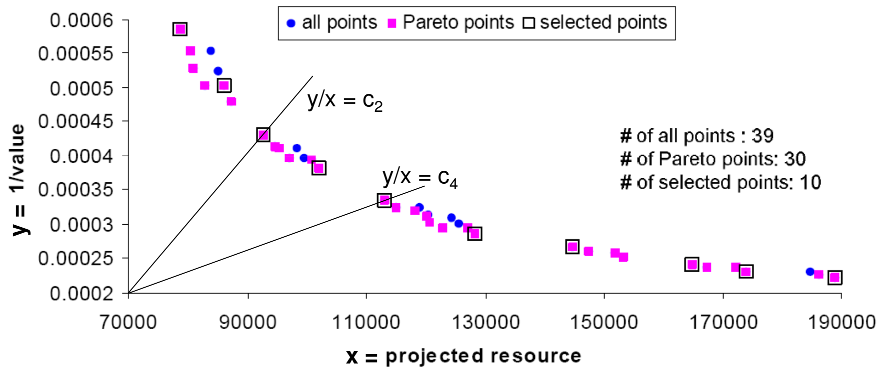
- One optimization objective (value)
- Multiple resource dimensions with capacity constraints
- Multiple independent applications
- Multiple independent configurations per application

Pick one configuration per application optimizing value within constraints

NP hard

36 A Parameterized Compositional Heuristic

- Project all resource dimensions into one dimension
- Approximate Pareto set in 2-dimensional space



product-constrain-derive-abstract-minimize

37 A Parameterized Compositional Heuristic

- Project all resource dimensions into one dimension
- Approximate Pareto set in 2-dimensional space

parameter **p**: maintain (at most) **p** Pareto points

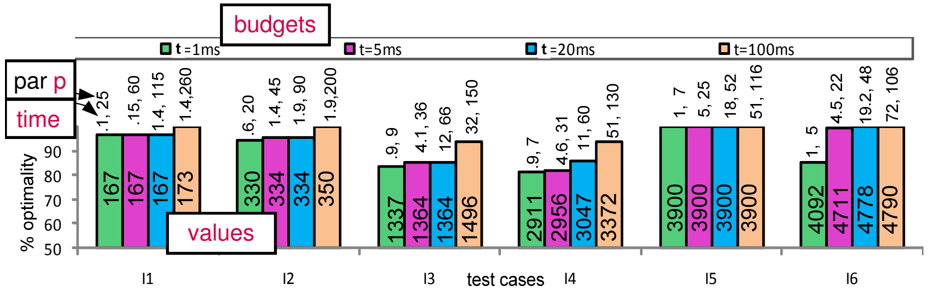
Allows to budget analysis time

analysis time $\leq$ **c · n · p**^{2}

(n number of applications; c platform constant)

product-constrain-derive-abstract-minimize

38 Controlling Time Budgets



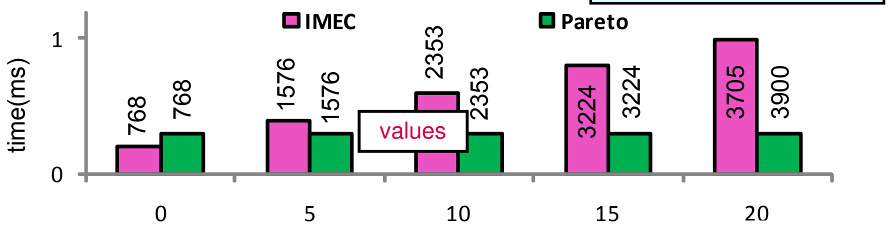
- MMKP benchmark: www.laria.u-picardie.fr/hifi/OR-Benchmark/MMKP
- Always within budget
- Good values
- Similar results to the IMEC, SOC 2006, non-compositional state-of-the-art heuristic

SimIt-ARM simulator
206 Mhz StrongARM

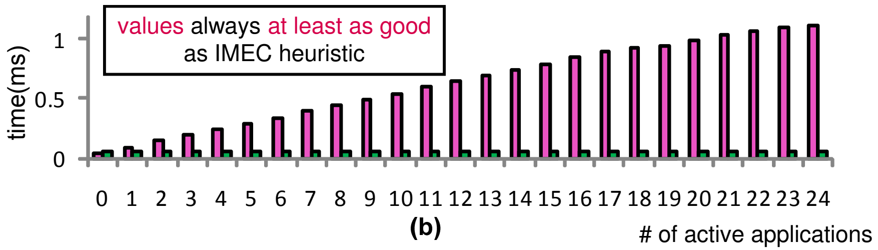
39 Compositionality

- Applications start at various times
- 5 at a time (a), 1 at a time (b)

what about applications stopping ?



(a)



(b)

40 Conclusions

- Goal: reliable, resource efficient data processing systems
- Dynamic behavior complicates analysis
- Intra-application dynamism:
  scenario-aware dataflow analysis and design-space exploration
- Inter-application dynamism:
  Pareto-algebra-based run-time adaptation

Thank you !

Questions ?

More info: www.es.ele.tue.nl/~tbasten/
SDF3 toolkit: www.es.ele.tue.nl/sdf3/
Pareto calculator: www.es.ele.tue.nl/pareto/

> ‘An understanding of the natural world and what's in it
> is a source of not only a great curiosity but great fulfillment.’
> David Attenborough