SAVE2016

Institute of Software ISCAS

Chinese Academy of Sciences

Pareto Optimal Scheduling for Synchronous Data Flow Graphs on Heterogeneous Multiprocessor

朱雪阳

http://lcs.ios.ac.cn/~zxy/

2016-12-11, 湖南长沙

Yu-Lei Gu, Xue-Yang Zhu, Guangquan Zhang, Yifan He. Pareto Optimal Scheduling for Synchronous Data Flow Graphs on Heterogeneous Multiprocessor. In Proc. Of the 21st International Conference on Engineering of Complex Computer Systems (ICECCS 2016). Dubai, UAE, 6-8 Nov., 2016.

Setting the Context

Code

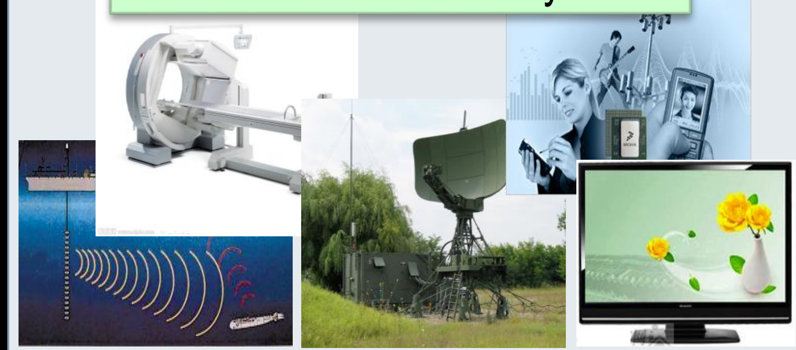
Synthesis

Time and energy optimal

Scheduling

Synchronous Data Flow Graphs

Real-time Embedded Systems



Multi-processor
Real-time requirements
Resource limitations
……

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- Model Description and Problem Formulation
- Basic Ideas of Our Methods
- Static Optimal Scheduling and Mapping
- Experimental Evaluation
- Conclusions and Future Work

Outline

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- Synchronous dataflow graphs (SDFG) are widely used for modeling data-driven applications
  - digital signal processing (DSP) algorithms
  - streaming media programs

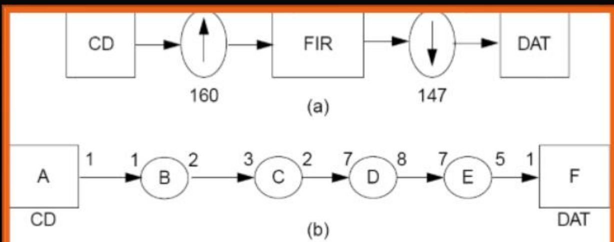
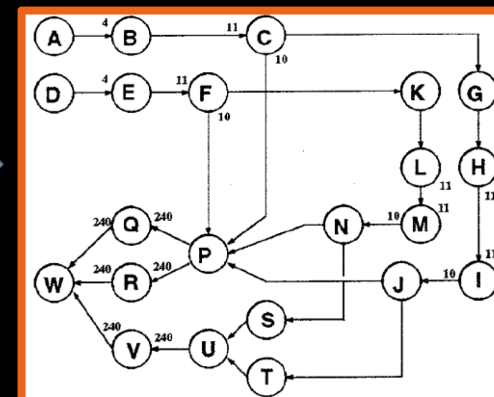


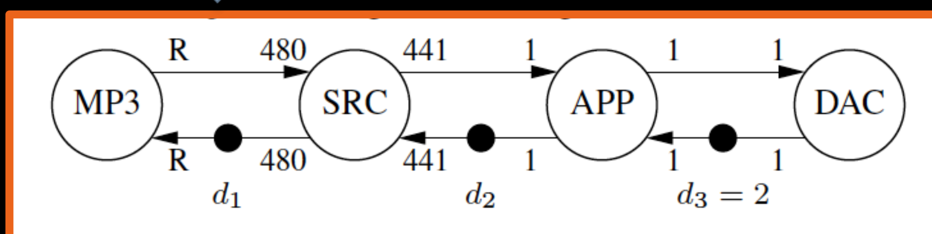
Figure 5. (a). CD to DAT sample rate change system.
(b). Multi-stage implementation of a CD to DAT sample rate system.

A sample rate converter model--compact disk (CD) to digital audio tape (DAT)[Murthy, et al.1997]

A satellite receiver [Ritz, et al. 1995]



An MP3 playback [Wiggers, et al. 2007]



Model Description

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

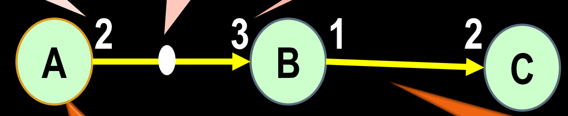
A system model includes an SDFG *G* and its execution platform *P*.

- An execution platform *P* is a set of heterogeneous processors. The energy consumption per unit time for each processor *p* is denoted by ***uEC(P)*** (*p* in use) and ***iEC(p)*** (*p* idle).
- An SDFG is a finite directed graph *G* = <*V*, *E*>

production rate *prd(e)*
e.g. prd(< A,B >)=2
number of tokens produced onto < A,B >by each execution of A.

delay (number of initial tokens) ***d(e)***
e.g. d(< A,B >)=1

consumption rate *cns(e)*
e.g. cns(< A,B >)=3
number of tokens consumed from < A,B > by each execution of B



computation time when v running on processor p *t(v,p)*
e.g. t(A,p1)= 1, t(A,p2)=2

actor *v*
- a computation

edge *e*
- a FIFO channel
- data dependency between actors

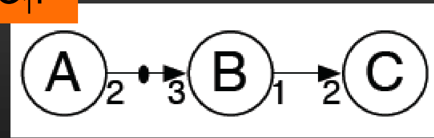
Multi-rate, *e.g. prd*(< A,B >)=2, *cns*(< A,B >)=3
Different number of firings of each actor in one iteration of execution, e.g. 3A,2B,1C in an iteration.

Model Description

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

System Model $M_1=(G_1,P_1)$:

G_1:



P_1:

	Energy		Exec. Time		
	inuse	idle	A	B	C
p1	90	10	1	2	2
p2	30	20	2	4	4
Rep. Vector			3	2	1

repetition vector

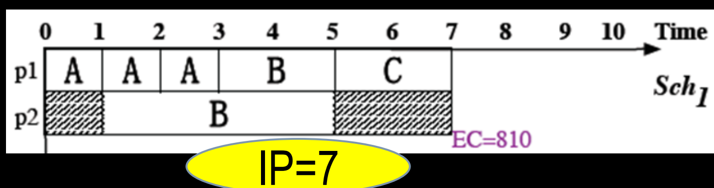
One **iteration** *of G_1*: three firings *of A*, two firings of *B* and one firing of *C, respectively.*

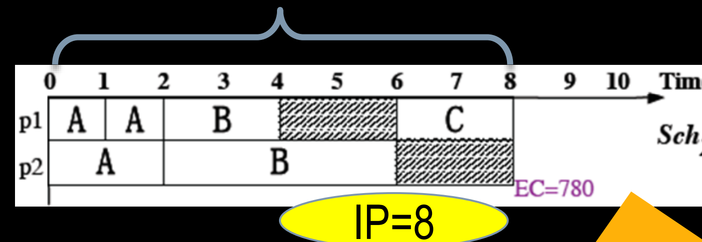
A static schedule arranges actors of an SDFG to be executed repeatedly (periodically).

✓ time arrangement and processor allocation

Schedules *of M_1:*

Iteration period (IP): the computation time of an iteration. IP=1/thr.





The number of firings in one iter. is the sum of the elements in the repetition vector *nQ* (=3+2+1)

energy cons. (EC) is the energy cons. of an iteration.

EC=Occupied time *uEC+idle time *iEC

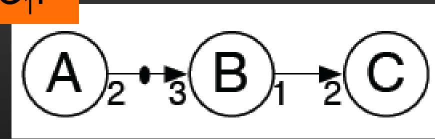
Eg. EC=(6*90+2*10)+(6*30+2*20)=780

In a schedule, an actor can be arranged to fire only when there are enough tokens on its incoming edges.

E.g., in Sch_1: the first firing of B can only fire when the first firing of A is finished (therefore there are 3 tokens on edge <A,B>)

System Model $M_1=(G_1,P_1)$:

G_1:



P_1:

	Energy		Exec. Time		
	inuse	idle	A	B	C
p1	90	10	1	2	2
p2	30	20	2	4	4
Rep. Vector			3	2	1

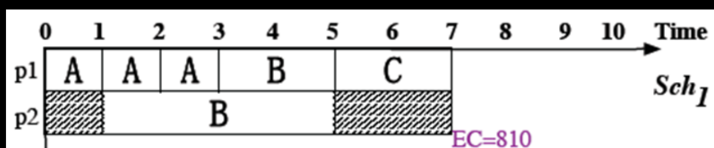
One **iteration** *of* G_1: three firings of *A*, two firings of *B* and one firing of *C, respectively.*

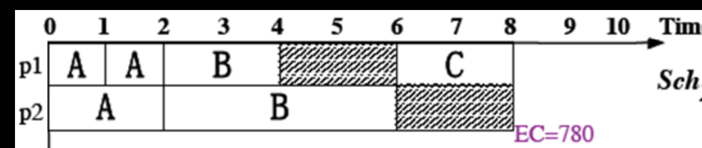
A static schedule arranges actors of an SDFG to be executed repeatedly.

- ✓ time arrangement and processor allocation

Schedules *of* M_1:

S





Many other schedules of M_1……

set of all schedules of M_1

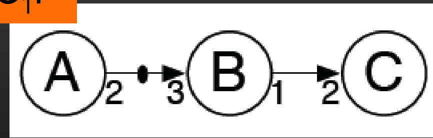
A 'good' schedule: its IP and EC are as small as possible

A Pareto point is a tuple (EC; IP). It is impossible to make one (IP or EC) better off without making the other worse off.

A schedule is a Pareto optimal schedule if its (EC; IP) is a Pareto point.

System Model $M_1=(G_1,P_1)$:

G_1:



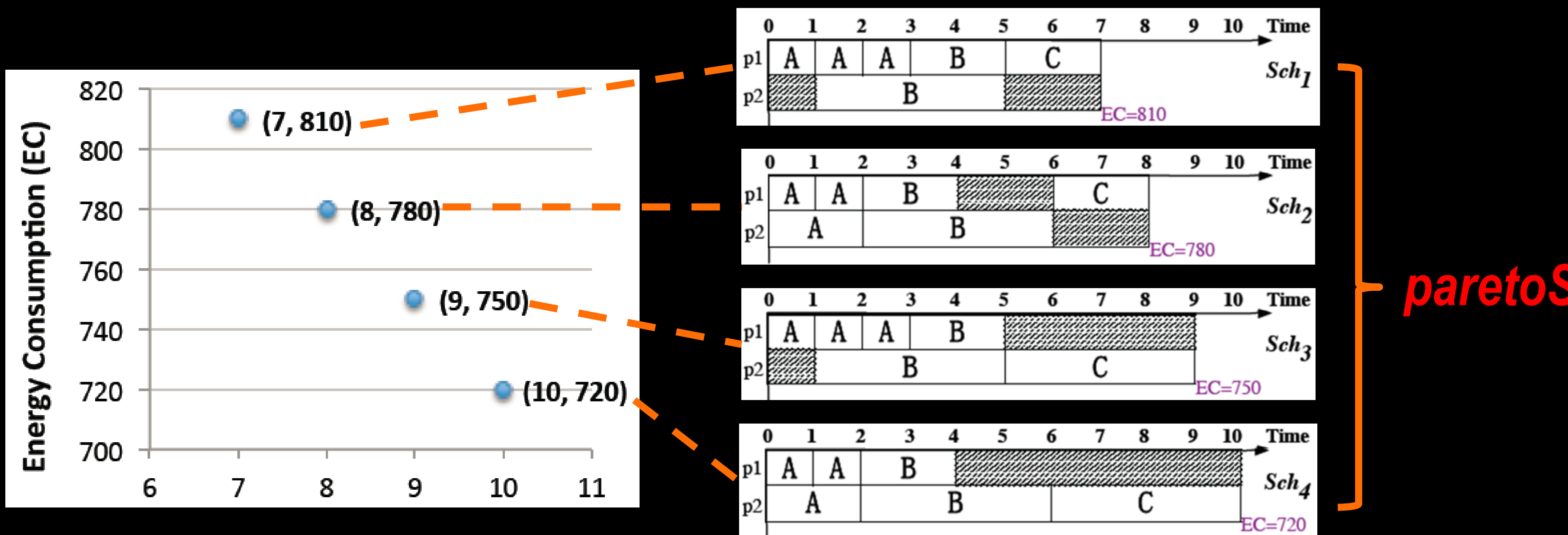
P_1:

	Energy		Exec. Time		
	inuse	idle	A	B	C
p1	90	10	1	2	2
p2	30	20	2	4	4
Rep. Vector			3	2	1

One **iteration** *of G_1*: three firings *of A*, two firings of *B* and one firing of *C, respectively.*

A static schedule arranges actors of an SDFG to be executed repeatedly.

- ✓ time arrangement and processor allocation

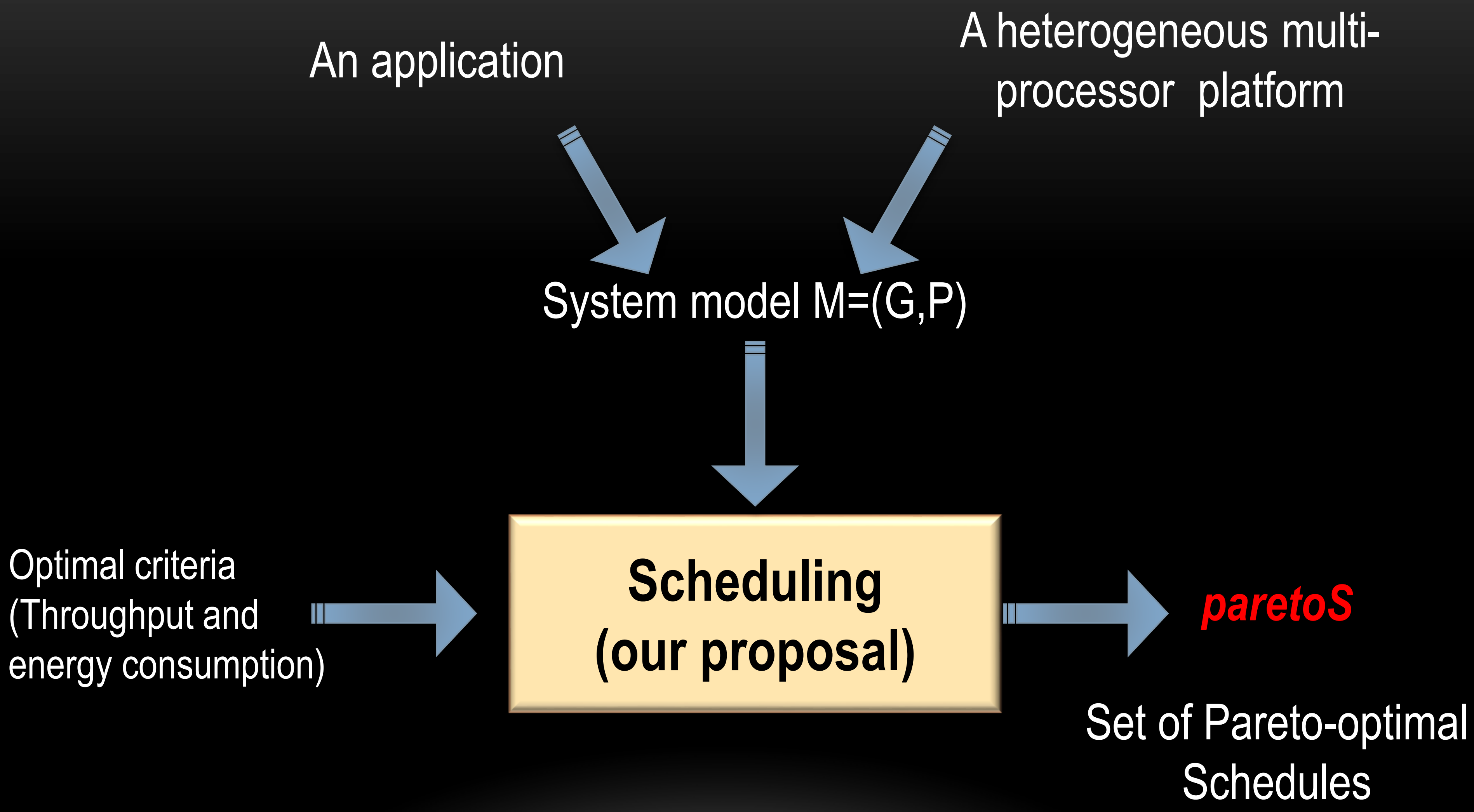


Pareto space of M_1

Pareto optimal schedules of M_1

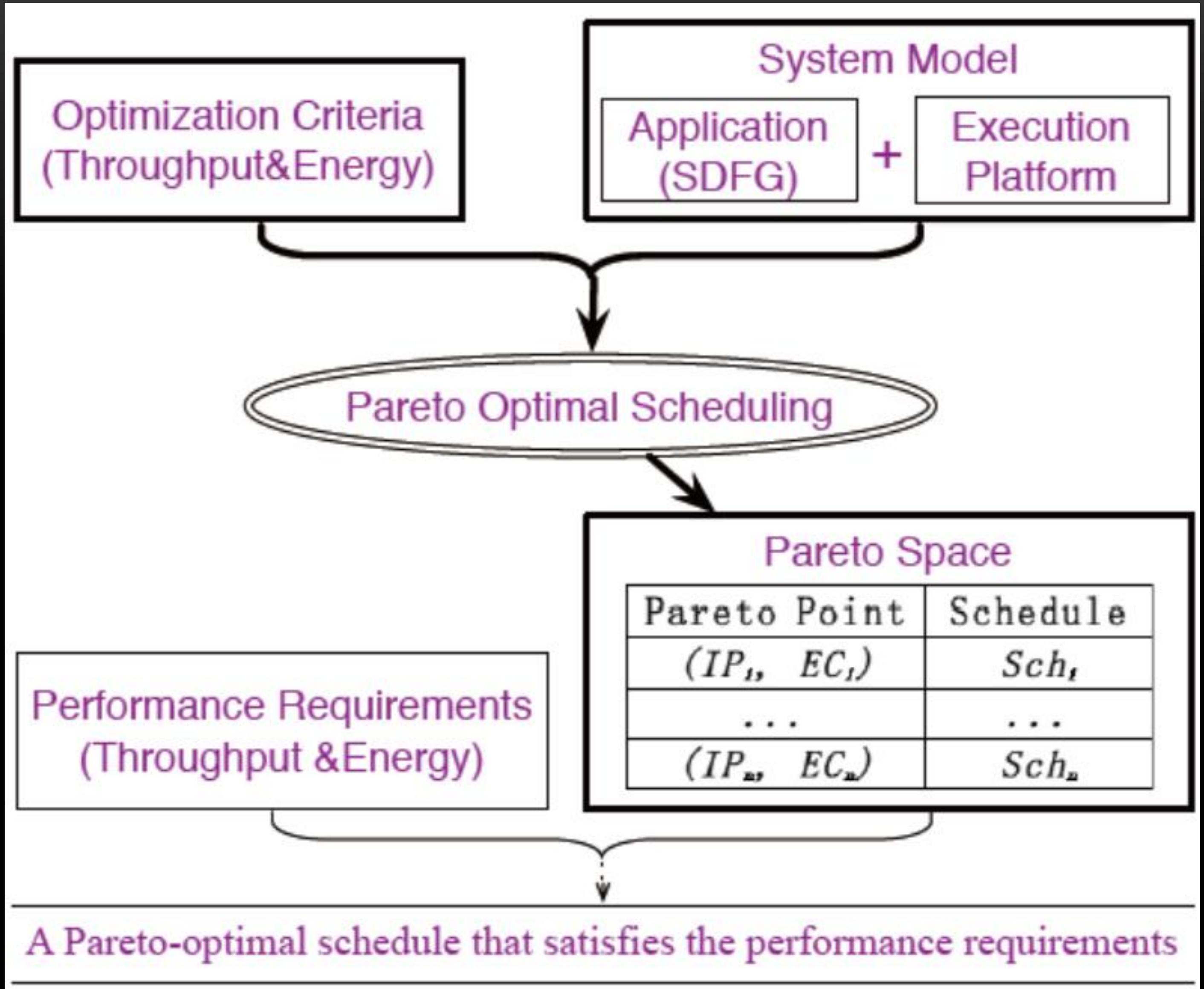
A Pareto point is a tuple (EC; IP). It is impossible to make one (IP or EC) better off without making the other worse off.

A schedule is a Pareto optimal schedule if its (EC; IP) is a Pareto point.



Problem Formulation

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences



The goal of our work

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- Model Description and Problem Formulation
- Basic Ideas of Our Methods
- Static Optimal Scheduling and Mapping
- Experimental Evaluation
- Conclusions and Future Work

Outline

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

System model $M=(G,P)$

S: set of all schedules of *M*

paretoS: set of pareto optimal schedules of *M*

POS - Pareto Optimal Scheduling

$paretoS \subseteq S$, $paretoS \subseteq subS$, $subS \subseteq S$

Extension

Pruning

POS

Parallelization

PPOS

Approximation

$PPOS_a$

Exact method

Approximate method

Basic Ideas

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- Model Description and Problem Formulation
- Basic Ideas of Our Methods
- Static Optimal Scheduling and Mapping
- Experimental Evaluation
- Conclusions and Future Work

Outline

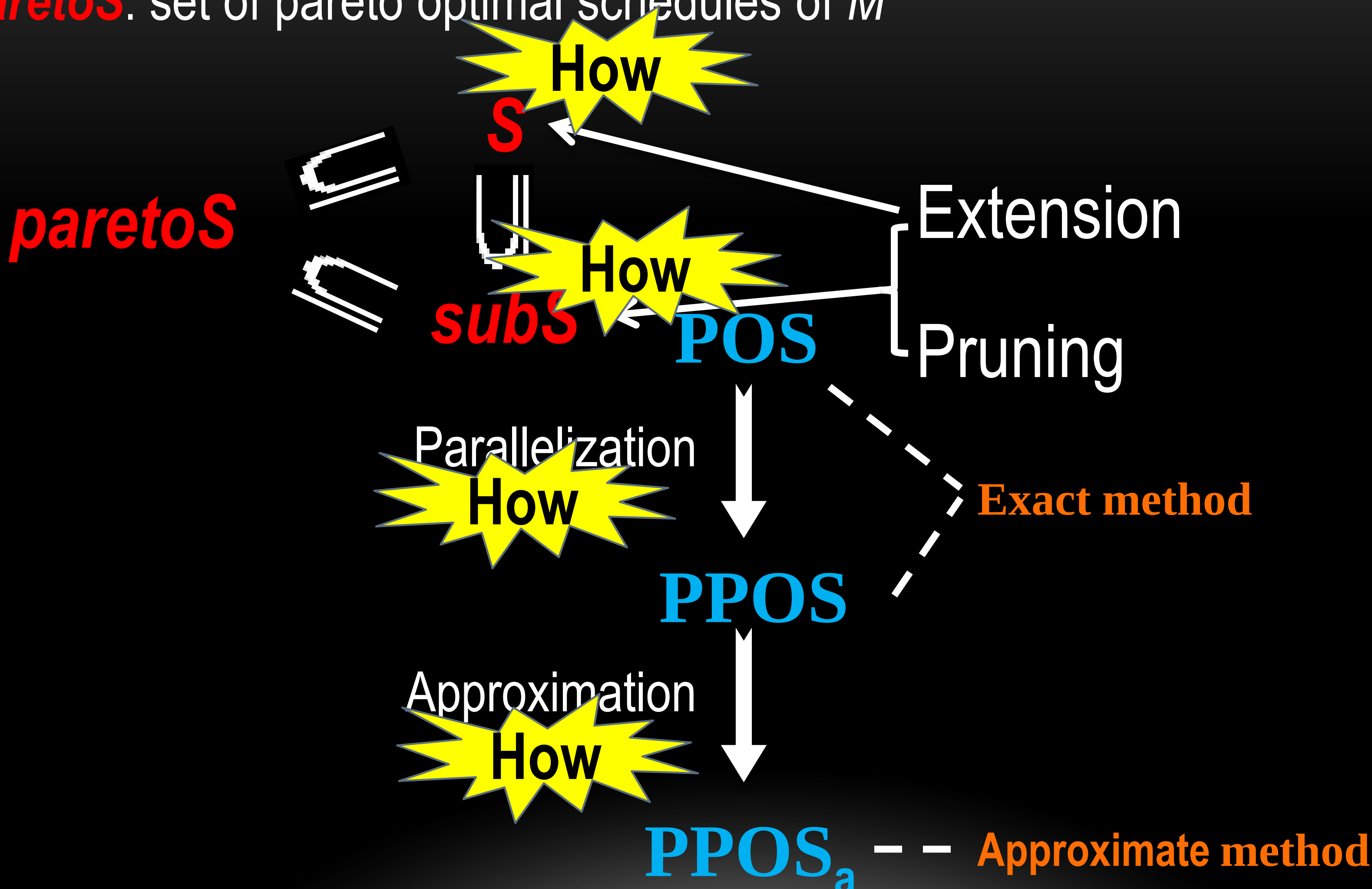
Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

POS - Pareto Optimal Scheduling

System model $M=(G,P)$

S: set of all schedules of M

paretoS: set of pareto optimal schedules of M



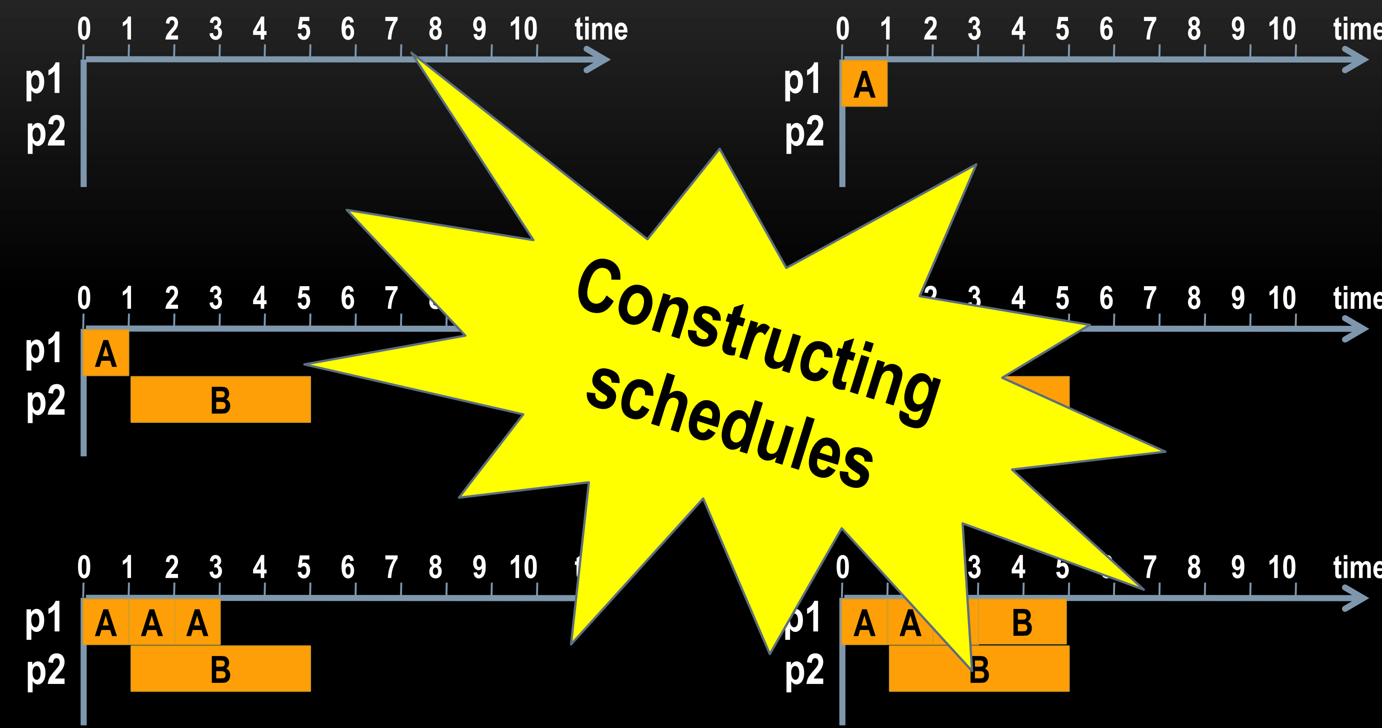
Static Optimal Scheduling and Mapping

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

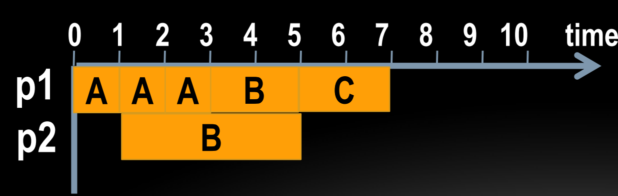
How to construct ***S*** with Extension?

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- A Constructing schedule (CS) is a partial schedule



A schedule:

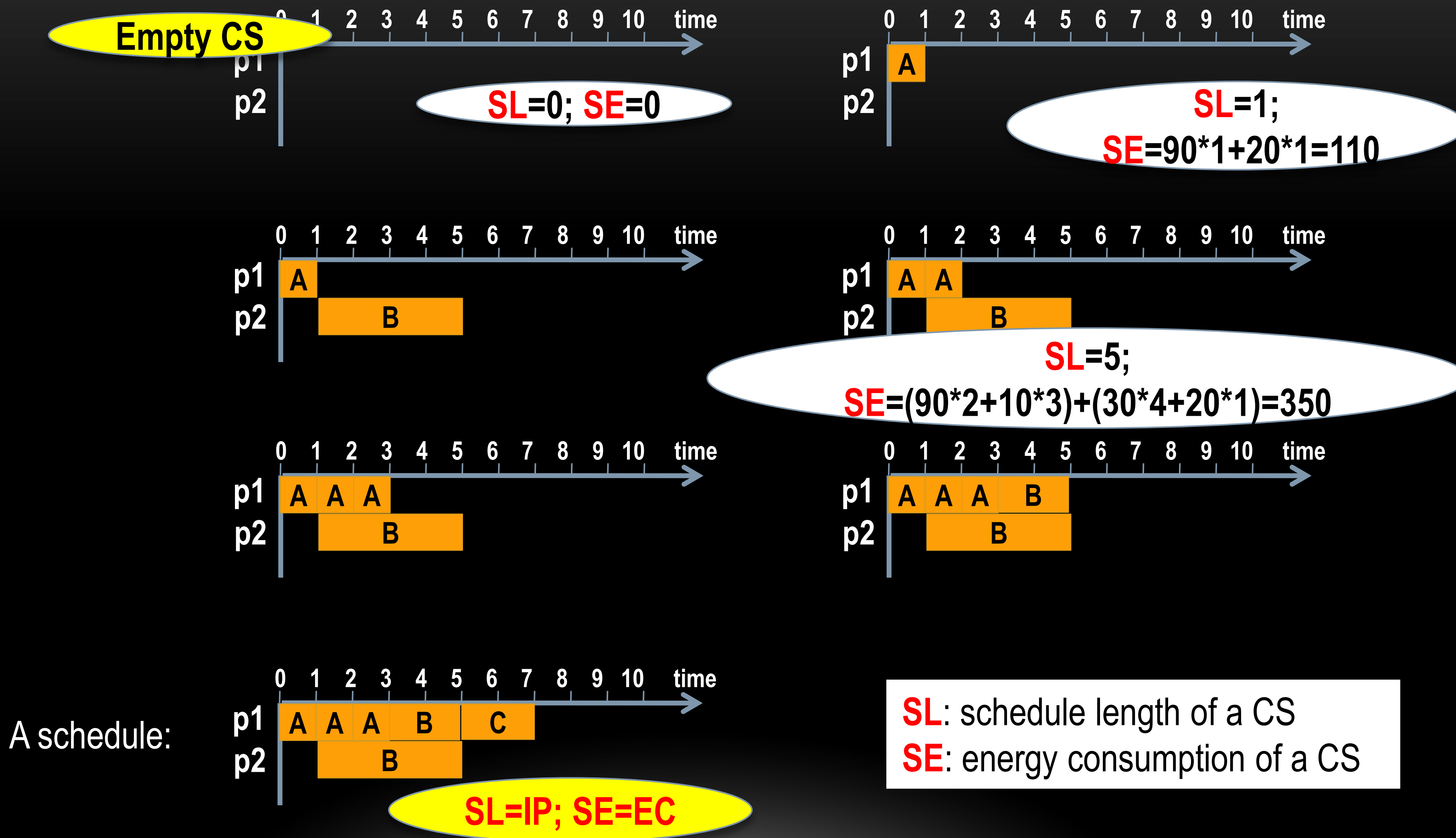


One iteration: 3A, 2B, 1C

Constructing schedules

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

A Constructing schedule (CS) is a partial schedule



SL: schedule length of a CS
SE: energy consumption of a CS

Constructing schedules

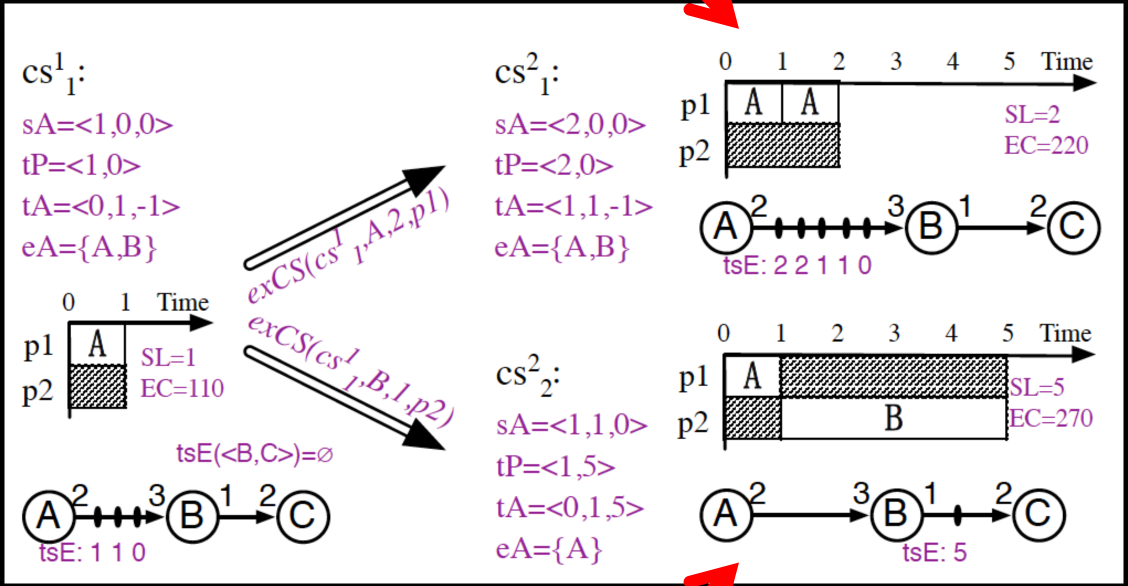
Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

Effect of Extension: a new firing is arranged on 'right' time point and available processor and the state of the SDFG is changed accordingly

cs^2_1 is an Extension of cs^1_1 on the 2nd firing of actor *A* and processor p_1

cs^1_1 :
the first firing of actor *A* has been scheduled

A is enabled
B is enabled



A is enabled
B is enabled

A is enabled

cs^2_2 is an Extension of cs^1_1 on the first firing of actor *B* and processor p_2

Extension

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

Constructing Schedules with Extension

Algorithm 1 Extend(CSs)

Require: A set of constructing schedules CSs of system model $\mathcal{M} = (G, P)$

Ensure: A set of constructing schedules CSs', including extensions of all constructing schedules in CSs

```
CSs' = ∅
for all cs in CSs do
    for all α ∈ cs.eA do
        for all p ∈ P do
            CSs' ← exCS(cs, α, cs.sA(α) + 1, p)
        end for
    end for
end for
return CSs'
```

Each step considers all possible Extensions

check all CS in the set CSs

check all enabled actors

check all processors

Extend a CS on each enabled actor and each processor

Schedules of a system model are constructed step by step, beginning with a set including an empty CS. At each step, we get a set of Extensions of current set of CSs.

$$CSs_0 \rightarrow CSs_1 \rightarrow CSs_2 \rightarrow \ldots \rightarrow CSs_i \rightarrow CSs_{i+1} \rightarrow \ldots \rightarrow CSs_{nQ} = S$$

CSs_0={empty CS}

CSs_{i+1}=**Extend**(CSs_i), i=1~nQ

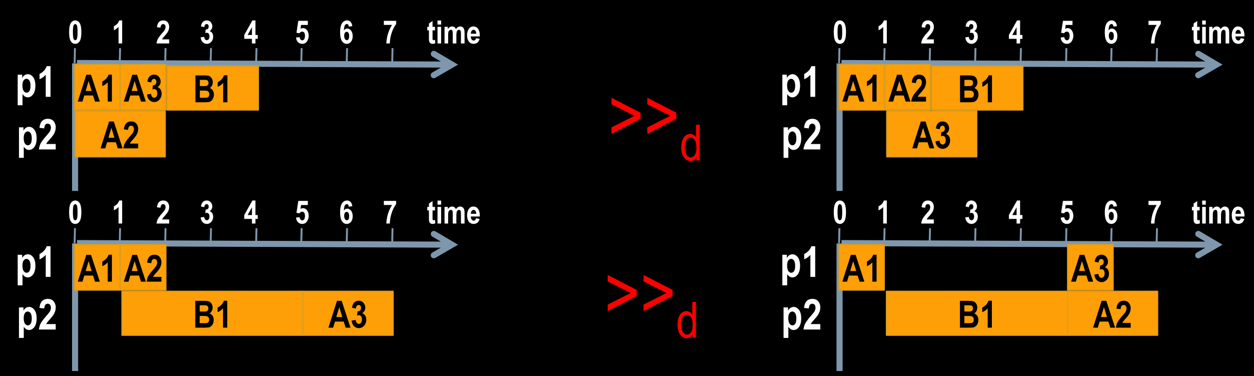
S is constructed after *nQ* steps!

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

How to construct ***subS*** with extension and pruning (**POS**)?

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- cs_1 dominate cs_2, denoted by $cs_1 >>_d cs_2$
  - ✓ two CSs are compared only when they have arranged the same number of firings of each actor;
  - ✓ occupied time of each processor; (the less the better)
  - ✓ finish time on each processor; (the less the better)
  - ✓ the possible start time of the next firing of each actor. (the less the better)
- ✓ Set CSs is a Partial Order Set under domination relation.



Property 1 (Theorem 8). SL and SE of the dominated CS are never better than SL and SE of the dominator.

Property 2 (Theorem 9). The domination relation of two constructing schedules is preserved by Extension.

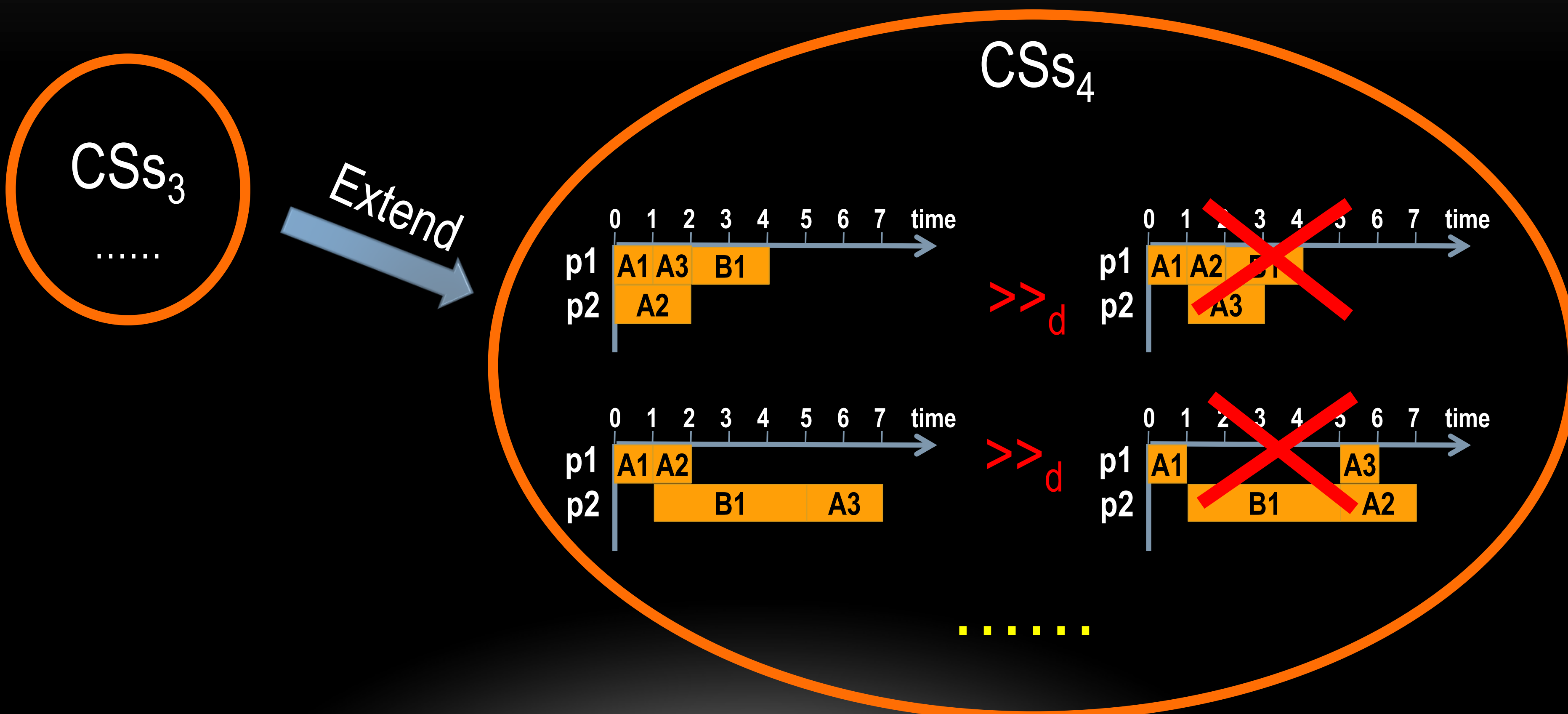
Domination Relation of CSs

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

Any extension of a dominated CS will be dominated by an extension of its dominator

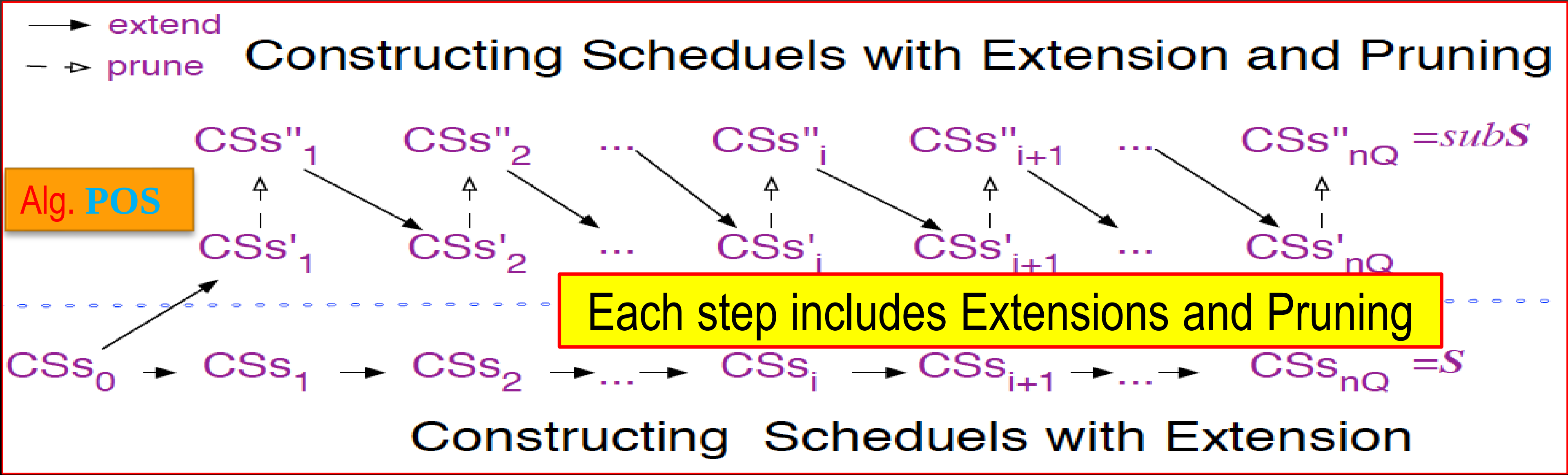
No Pareto optimal schedule is extended by a dominated CS (in any step)

Therefore the dominated CSs can be removed safely



pruning

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences



subS **is constructed after *nQ* steps!**

paretoS $\subseteq$ ***subS*** $\subseteq$ ***S***

The number of constructing schedules of M_1 at each step

i	1	2	3	4	5	6
$\lvert CS_i \rvert$	2	8	24	48	96	192
$\lvert CS_i'' \rvert$	2	7	13	14	9	13

With Extension only

With Extension and Pruning

Constructing Schedules with Extension and Pruning

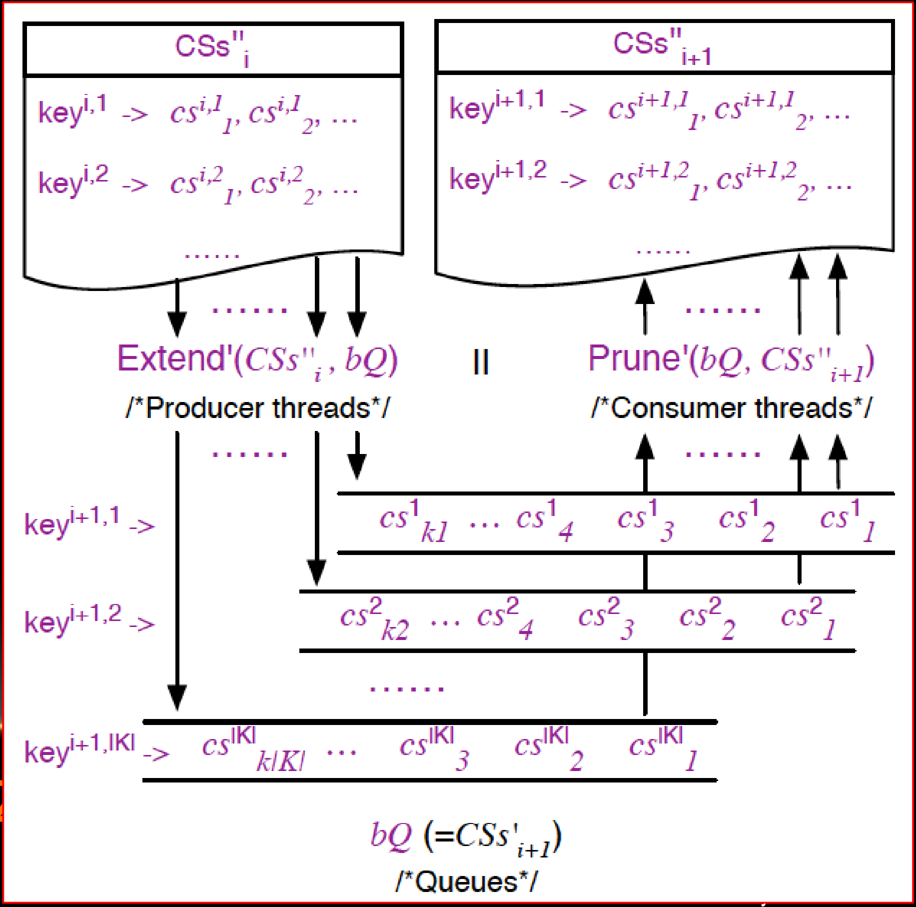
Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

How to parallelize the construction procedure (PPOS)?

At each step of POS
- ✓ each Extension does not affect each other
- ✓ Pruning operations are possible only for those CSs, in which, for each actor, the number of the scheduled firings ($sA(V)$) are same.

Alg. PPOS

At each step
- ✓ Divide sets CSs'_i and CSs''_i into some subsets according to sA
- ✓ extension and pruning are conducted on those subsets in parallel (without data race).



Paralleliz

How to approximate the procedure ($\mathbf{PPOS_a}$)?

$\mathbf{PPOS_a}$ is obtained by pruning the space according to a **quasi-domination** relation

- ✓ compares only the schedule length SL and energy consumption SE of two constructing schedules.

Approximation

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

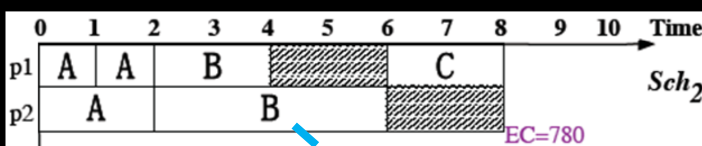
POS

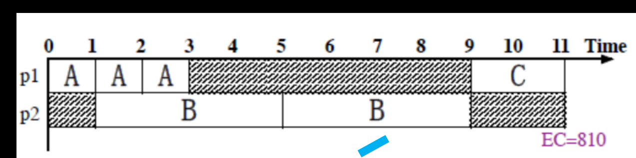
Parallelization ↓

PPOS - - Exact method

Approximation ↓

$PPOS_a$ - - Approximate method





Consider both **firing mapping** and **actor mapping** (all firings of an actor are arranged on the same processor).

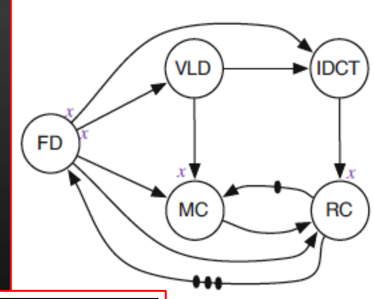
- fmPPOS, amPPOS; $fmPPOS_a$, $amPPOS_a$

- Model Description and Problem Formulation
- Basic Ideas of Our Methods
- Static Optimal Scheduling and Mapping
- Experimental Evaluation
- Conclusions and Future Work

Outline

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

System model:



	Energy		Exec. Time				
	inuse	idle	FD	VLD	IDCT	MC	RC
p1	90	10	0	1	1	1	1
p2	30	20	0	2	2	2	2

Sce.	x	Rep. Vector					nQ
		FD	VLD	IDCT	MC	RC	
P5	5	1	5	5	1	1	13
P10	10	1	10	10	1	1	23
P30	30	1	30	30	1	1	63

Number of processors

Number of firings in an iter.

Models with different nQ

Compare to model checking method (MC) to evaluate the exactness of the proposed methods

Results for Firing Mapping			
Pareto Space (EC,IP)			
$\#Pro$	P5	P10	P30
2	(990,9)(960,10)	(1790,15)(1760,16)	(5020,42)(4990,43),(4960,44)
4	(1020,5)	(1820,9)	(5080,22)(5020,23)
Execution Time (s) MC [19]/fmPPOS/fmPPOS$_a$			
2	0.1/0.1/0.1	0.2/0.2/0.2	6.4/1.2/1.0
4	13.5/0.3/0.1	19m/11.4/0.2	N/N/0.7
Results for Actor Mapping			
Pareto Space (EC,IP)			
2	(1230,11)(1020,12)(990,13)(960,14)	(2330,21)1820,22)((1790,23)(1760,24)	(6730,61)(5020,62)(4990,63),(4960,64)
4	(1380,7)(1320,12)(1080,13)	(2480,12)(2420,22)(1880,23)	(6880,32)(6820,62)(5080,63)
Execution Time (s) amPPOS/amPPOS$_a$			
2	0.1/0.1	0.1/0.6	1.4/0.4
4	0.1/0.1	0.2/0.4	0.4/0.4

N - Timeout.

Experimental results

with different parameters

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

– For all scenarios and parameters of MPEG-4 decoder that MC method can finish, our methods return exact Pareto spaces as MC does.

– Our methods outperform MC method more when the models grow larger.

Results for Firing Mapping

Pareto Space (EC,IP)			
$\#Pro$	P5	P10	P30
2	(990,9)(960,10)	(1790,15)(1760,16)	(5020,42)(4990,43), (4960,44)
4	(1020,5)	(1820,9)	(5080,22)(5020,23)
Execution Time (s) MC [19]/fmPPOS/fmPPOS$_a$			
2	0.1/0.1/0.1	0.2/0.2/0.2	6.4/1.2/1.0
4	13.5/0.3/0.1	19m/11.4/0.2	N/N/0.7

Results for Actor Mapping

Pareto Space (EC,IP)			
2	(1230,11)(1020,12) (990,13)(960,14)	(2330,21)1820,22) ((1790,23)(1760,24)	(6730,61)(5020,62) (4990,63),(4960,64)
4	(1380,7)(1320,12) (1080,13)	(2480,12)(2420,22) (1880,23)	(6880,32)(6820,62) (5080,63)
Execution Time (s) amPPOS/amPPOS$_a$			
2	0.1/0.1	0.1/0.6	1.4/0.4
4	0.1/0.1	0.2/0.4	0.4/0.4

N - Timeout.

Experimental results

with different parameters

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

The number of actors in an SDFG (nA) and the sum of the elements in the repetition vector (nQ) have significant impact on the performance of various methods.

We generate six groups of SDFGs with different values of *nA* and *nQ*.
Each group includes 30 graphs with the same values of *nA* and *nQ*.
– E.g. a group named *a5q20* includes 30 SDFGs with *nA* = 5 and *nQ* = 20

	fmPPOS Exe. Time (AVG/MAX/MIN)	fmPPOS_a Exe. Time (AVG/MAX/MIN)	NADRS of fmPPOSa	amPPOS Exe. Time (AVG/MAX/MIN)	amPPOS_a Exe. Time (AVG/MAX/MIN)
$a5q20$	0.2/1.1/0.1	0.1/0.3/0.1	1.0%	0.1/0.4/0.1	0.1/0.4/0.0
$a5q50$	1.9/11.3/0.1	0.3/0.7/0.1	0.6%	0.4/1.4/0.1	0.5/1.7/0.1
$a10q50$	37.2/8.2m/0.3[b]	0.7/7.3/0.1	0.8%	2.6/30.0/0.1[a]	0.7/2.5/0.1
$a10q100$	4.6m/23.7m/1.1[b]	7.7/2.6m/0.2	0.5%	12.7/3.2m/0.3[a]	1.7/16.9/0.2

[a] Timeout on 1 case.
[b] Timeout on 7 cases.

The relative small cases
– used to evaluated the performance of exact methods and measure the accuracy of the appr. methods.

The large cases
– used to evaluate the performance of proposed approximate methods.

	$\#Pro = 4$		$\#Pro = 8$	
	fmPPOS_a AVG/MAX/MIN	amPPOS_a AVG/MAX/MIN	fmPPOS_a AVG/MAX/MIN	amPPOS_a AVG/MAX/MIN
$a10q50$	0.7/5.5/0.1	0.8/3.4/0.1	0.7/4.3/0.1	0.7/2.3/0.1
$a10q100$	5.6/1.8m/0.2	1.7/11.5/0.4	5.3/1.4m/0.1	1.5/12.5/0.4
$a20q100$	44.2/14.6m/0.5[a]	1.1m/29.2m/0.4	1.5m/29.4m/0.5	12.2/2.2m/0.4[a]
$a10q1000$	4.2m/21.4m/19.9[b]	40.6/4.6m/0.9[a]	3.1m/15m/4.2[b]	70.8/12.2m/1.5[a]

[a] Timeout on 1 to 2 cases.
[b] Timeout on 9 to 13 cases.

They run on a 2.4GHz server with 160 cores, 32MB Cache and 256GB RAM.
At each step, we allocate 90 percent of cores to pruning threads (Consumers) and 10 percent to extension threads (Producers).

- Model Description and Problem Formulation
- Basic Ideas of Our Methods
- Static Optimal Scheduling and Mapping
- Experimental Evaluation
- Conclusions and Future Work

Outline

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- A Parallelized Pareto Optimal Scheduling method (PPOS) for scheduling SDFGs on heterogeneous multiprocessor platforms.
  - ✓ The optimization criteria are throughput and energy consumption
  - ✓ PPOS is an exact method that can find all exact Pareto optimal schedules
  - ✓ with firing mapping or with actor mapping
- An approximation variant of PPOS, PPOSa, has been presented to deal with larger system models.
- The exactness and efficiency of the exact methods are further confirmed by the experimental results. The approximate methods return results close to the exact ones.

Conclusions

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

- The design of complex embedded systems usually needs to take into account various resource constraints. Besides processors and energy constraints, we will extend our methods to deal with more resource constraints in the future work.

Future Work

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences

Thanks!

Xue-Yang Zhu
Institute of Software
Chinese Academy of Sciences