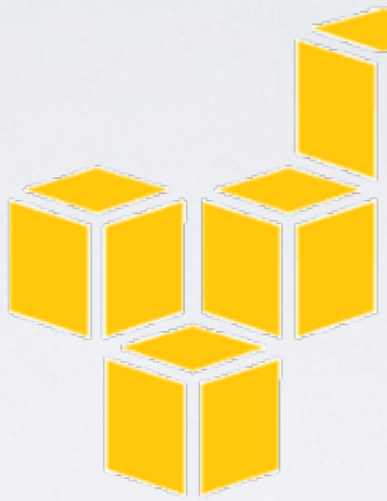
PARTITION WITH SIDE EFFECTS

Krzysztof (Chris) Rządca
Institute of Informatics, University of Warsaw, Poland
joint work with Fanny Pascual
Sorbonne Universités, Université Pierre et Marie Curie, LIP6, Paris

CLOUD COMPUTING: THE FRONT-END OF THE MODERN DATACENTER



amazon web services™

Google Cloud Platform Live

- virtual machines for hire for 0.10-2.00 per hour
- used by many organizations to reduce infrastructure costs

A DATACENTER IS NOT YOUR HPC SUPERCOMPUTER

- **co-location of dozens of tasks on a single physical node (focus of this talk)**
- virtualization&migrations
- Service Level Agreements
- surviving failures
- considering network bandwidth
- ….

CO-ALLOCATING TASKS ONTO MACHINES

bin packing?

task, load 2

task, load 1

machine

multi-dimensional bin packing?

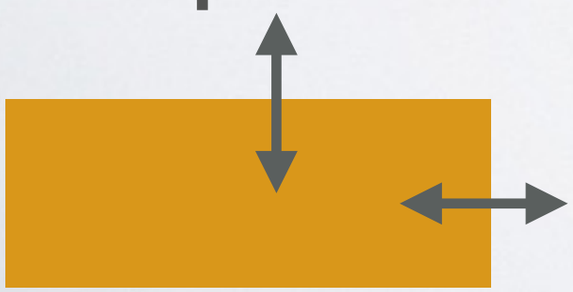
[Stillwell et al 2012]

cpu load

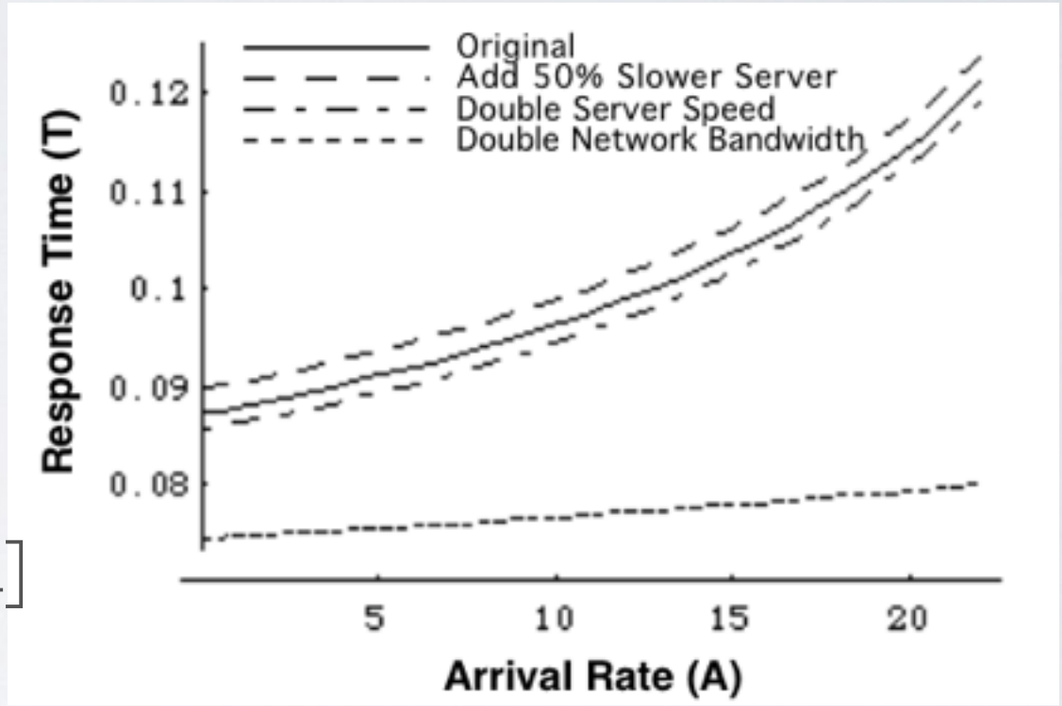
memory load

loads are probabilistic?

[Goel&Indyk99, WMZ11,…]



optimizing user experience?



[Slothouber 95]

CONSTRUCTING THE MODEL: TASKS ARE HETEROGENEOUS AND IMPACT EACH OTHER IN A DIFFERENT WAY!

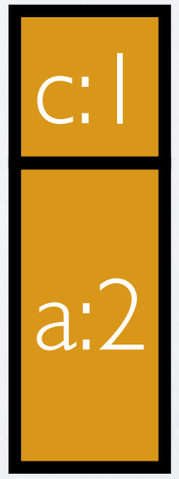
consider a web service with load 2

and a RAM-cached database with load 1

They use different resources
(webservice: mostly network; db: mostly RAM/cache)

colocating a web service with a database

(performance)

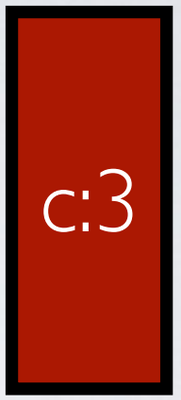


colocating 2 web services

OUR PERFORMANCE MODEL: A TASK HAS **LOAD** AND **TYPE**

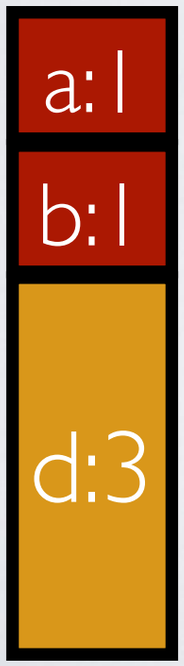
load=2
type=webservice

load=1
type=db



load=3
type=db

task's cost = f_t(sum of loads of type web; sum of loads of type db)
(cost = -performance)



$c_a = f_{db}(3; 2)$

M1

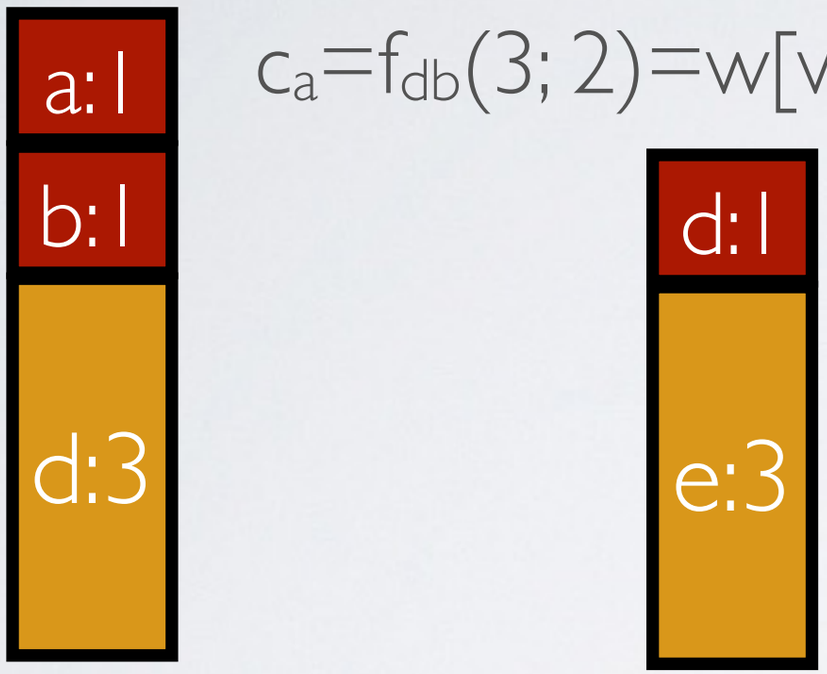
$c_e = f_{web}(3;1)$

M2

goal: min $\sum$ cost

$$\sum cost = 2f_{db}(3;2) + f_{db}(3;1) + + f_{web}(3;2) + f_{web}(3;1)$$

LINEAR PERFORMANCE MODEL: COST = COEFFICIENT*LOAD



M1

$c_a = f_{db}(3; 2) = w[web,db]*3 + w[db,db]*2$

M2

$c_e = f_{web}(3;1) = w[web,web]*3 + w[db,web]*1$

$$\sum cost = 2f_{db}(3;2) + f_{db}(3;1) + f_{web}(3;2) + f_{web}(3;1)$$

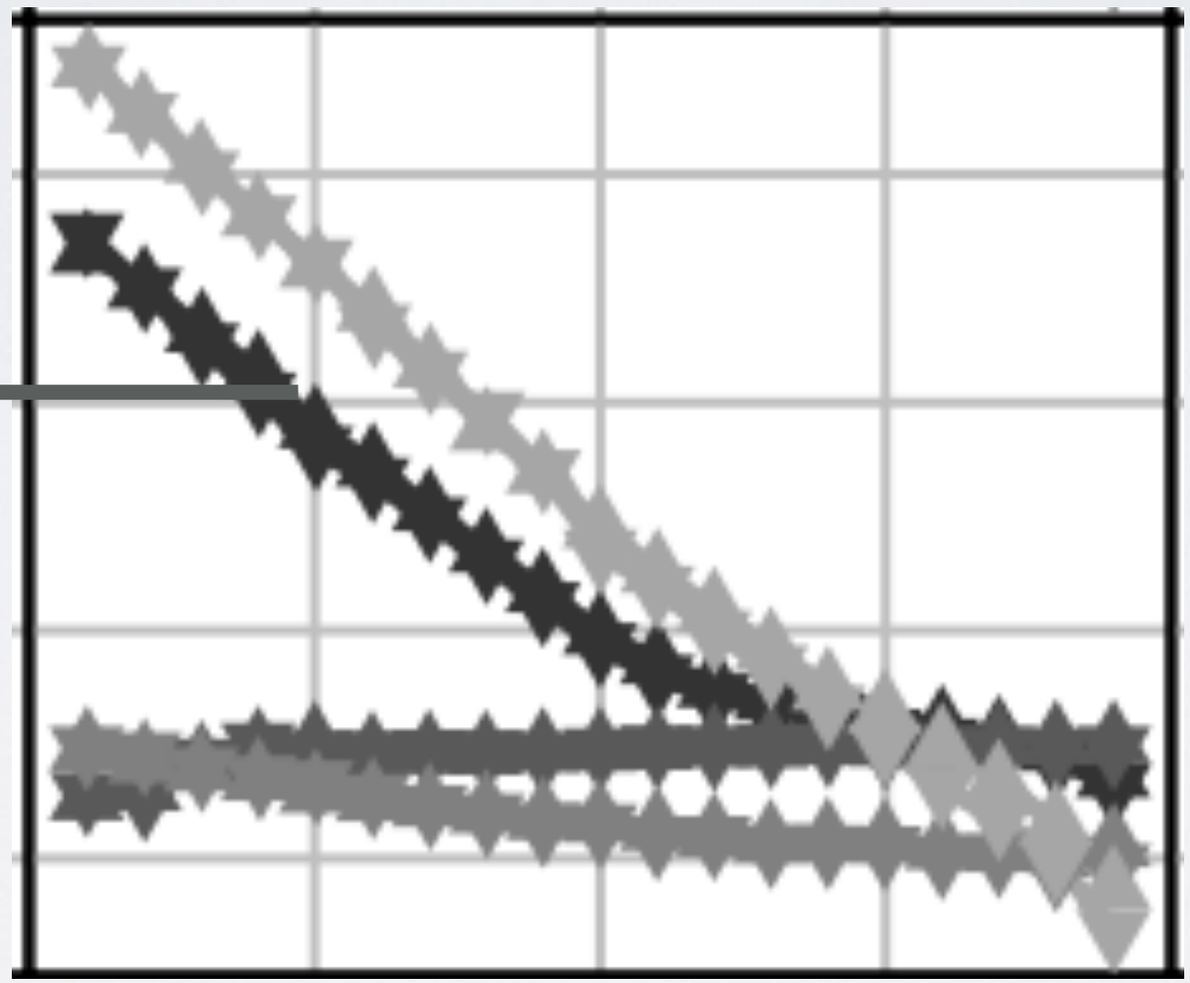
$$\begin{aligned}\sum cost = {} & 2*(w[web,db]*3 + w[db,db]*2) + \\ & + w[web,web]*3 + w[db,web]*2 + \\ & + w[web,db]*3 + w[db,db]*1 + \\ & + w[web,web]*3 + w[db,web]*1\end{aligned}$$

WHY?
MOTIVATING THE (GENERAL) MODEL…

[PODZIMEK15]: COLOCATING CPU-INTENSTIVE BENCHMARKS HAS SEVERE PERFORMANCE IMPACT

throughput of Scalac benchmark

linux kernel allocates workloads to cores

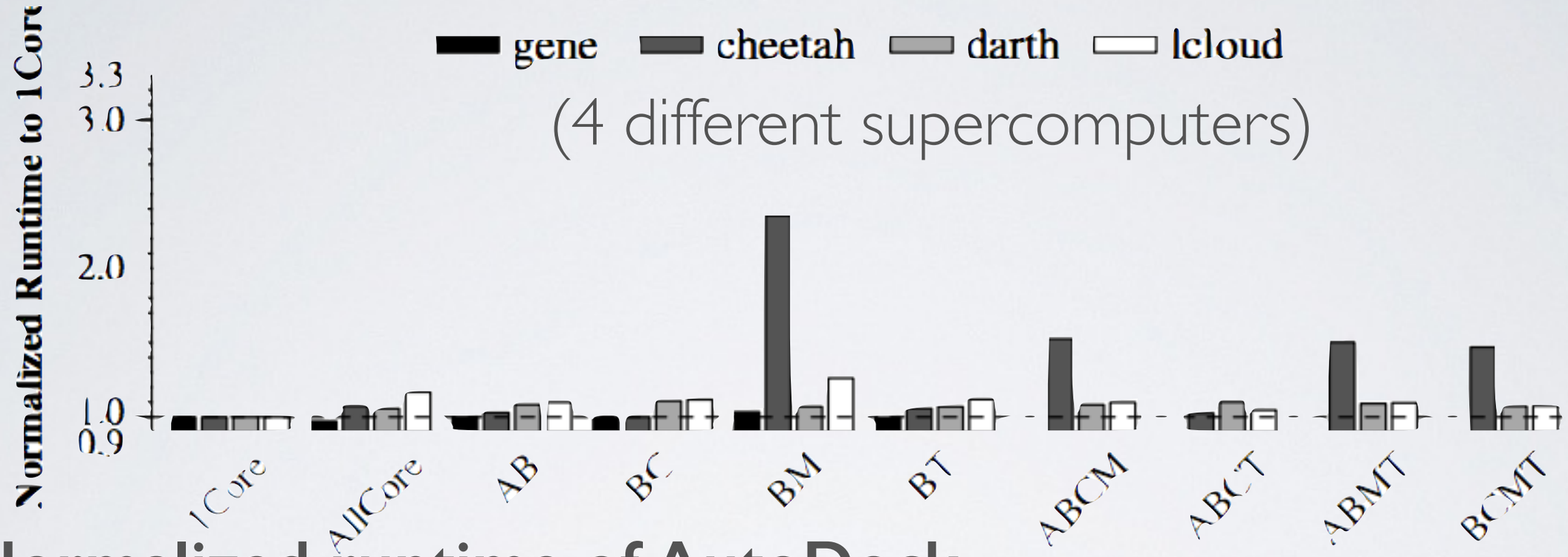


25% 50% 75% 100%

CPU usage of specs benchmark (run in the background)

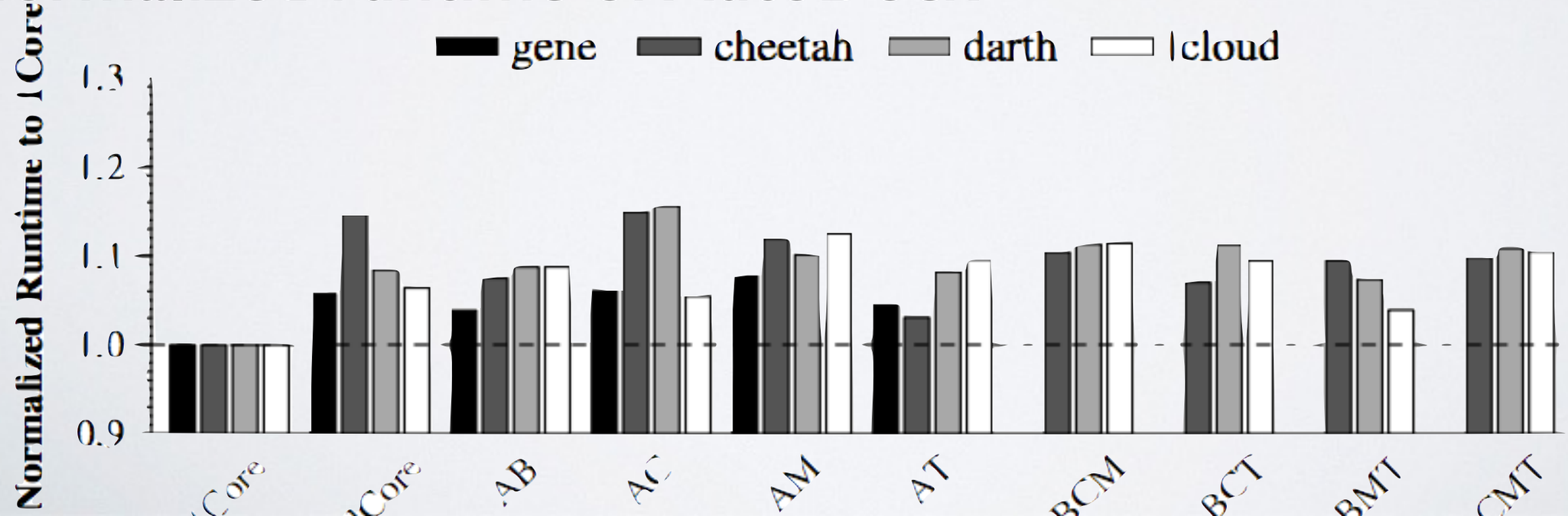
[KIM15]:PERFORMANCE IMPACT OF COLOCATION DEPENDS ON WORKLOAD

Normalized runtime of Blast when colocated with AutoDock (A), CacheBench (C), Montage (M) and ThreeKaonOmega (T)



(4 different supercomputers)

Normalized runtime of AutoDock

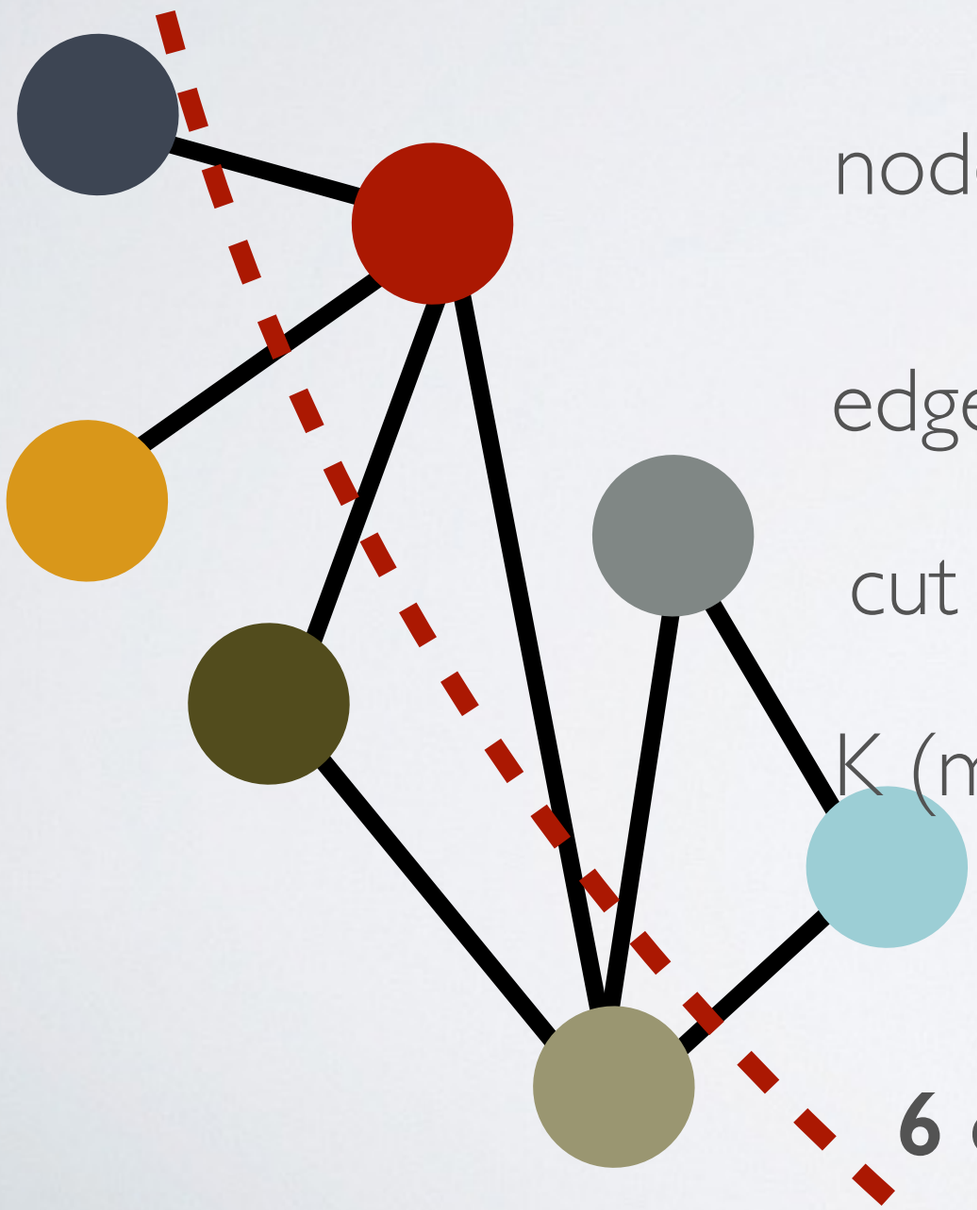


OUR RESULTS:
LINEAR COST MODEL

$$COST = \sum_{T:TYPE} COEFF_T * LOAD_T$$

THE TOTAL COST IS NP-HARD IF THERE ARE MANY TYPES

reduction from Simple Max Cut: cut a graph in two so that at least K edges cut



node — task (unit weight)

each task is of different type

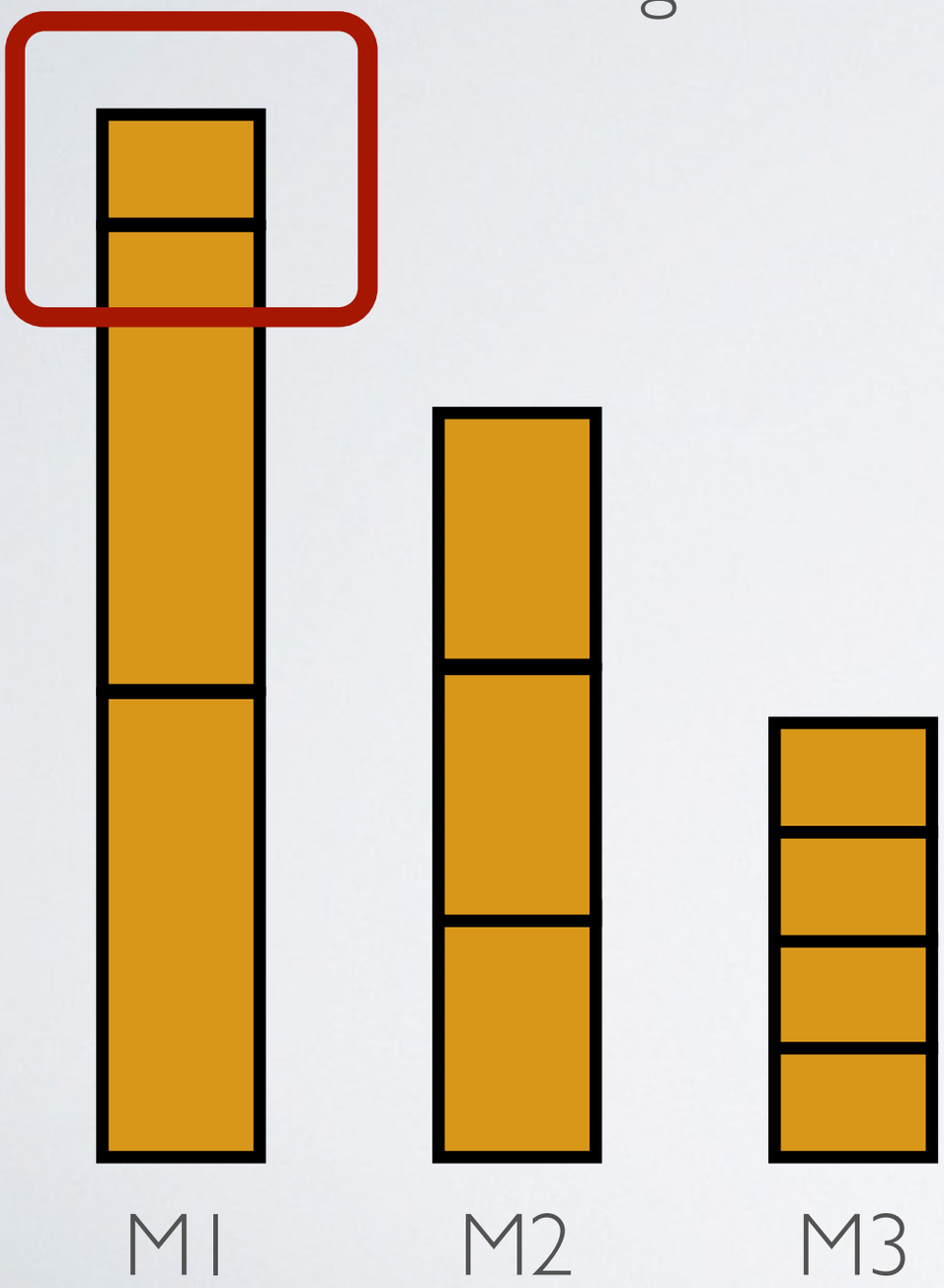
edge (i,j) — w[i,j]=1/2; no edge — w[k,i]=0

cut — partition into 2 machines

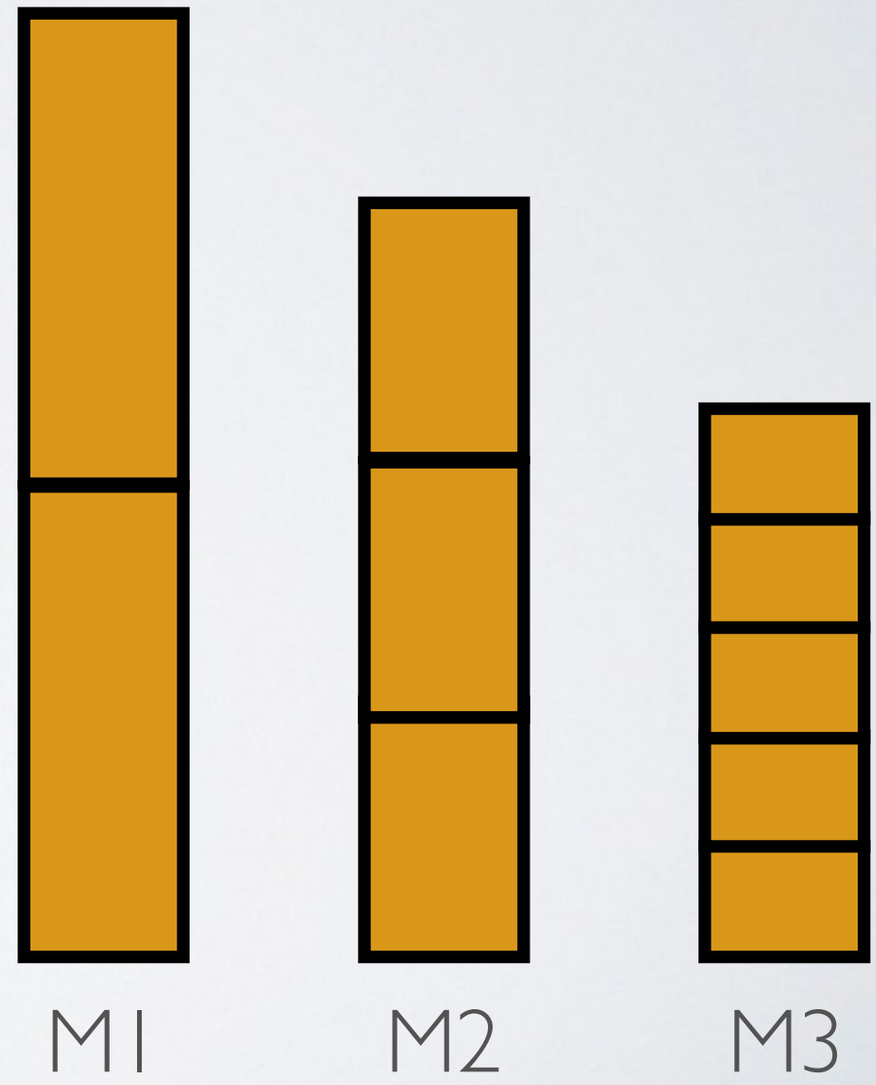
K (min # of edges cut) — max cost = |E|-K

IN AN OPTIMAL PARTITION, TASKS OF EACH TYPE ARE ORDERED BY LOAD (SPT)

non-SPT assignment

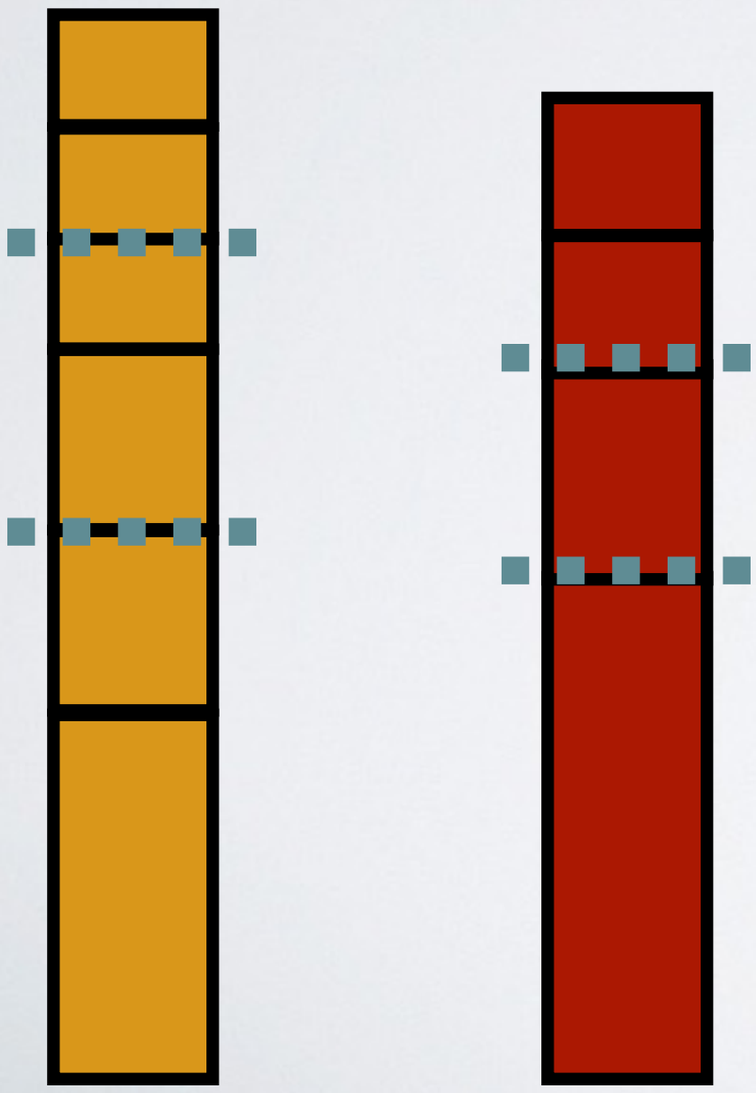


SPT assignment

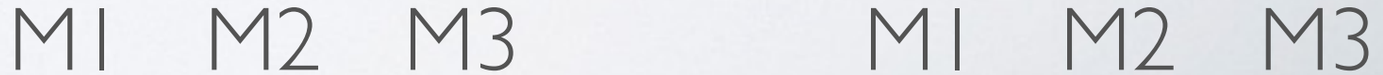


CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)



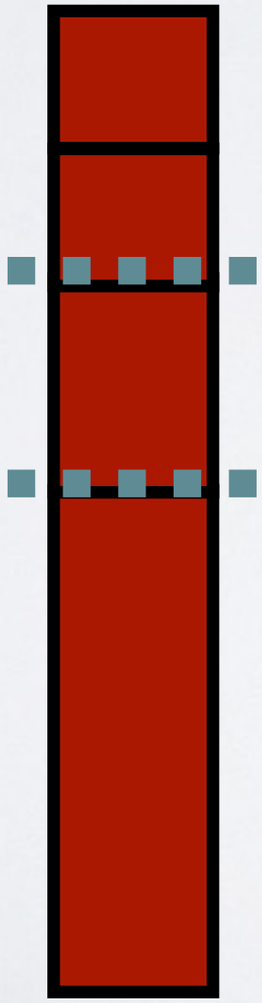
2. **juxtapose**: find the optimal combination (test all possibilities)

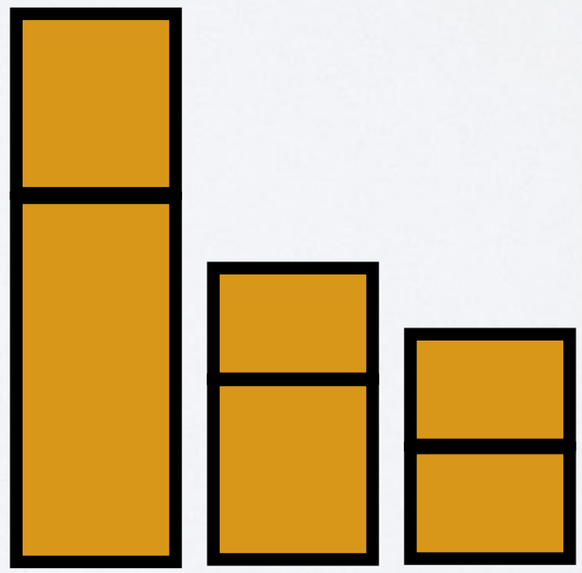


CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)

2. **juxtapose**: find the optimal combination (test all possibilities)



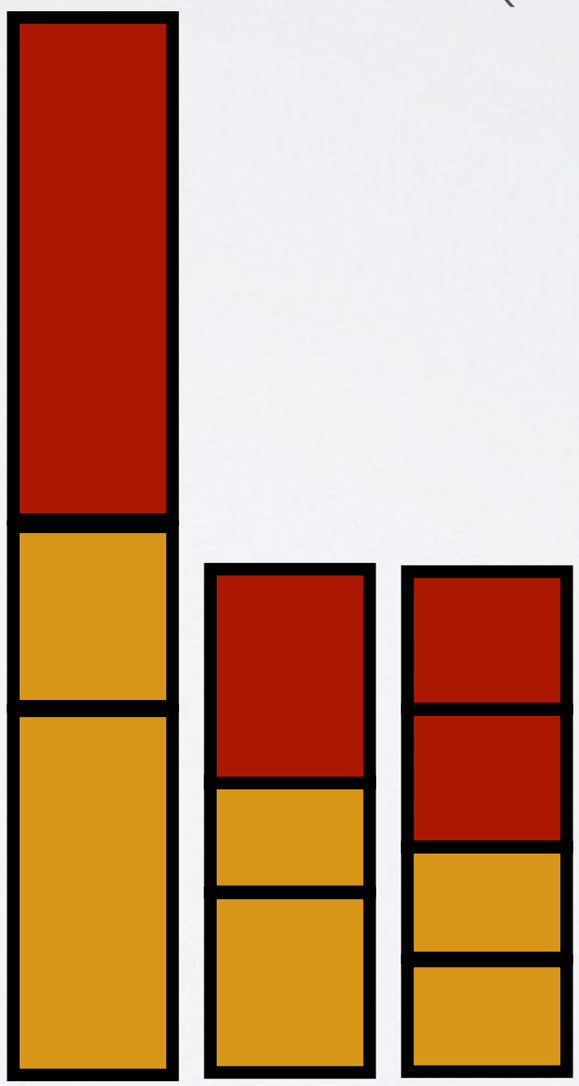


M1 M2 M3

CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)

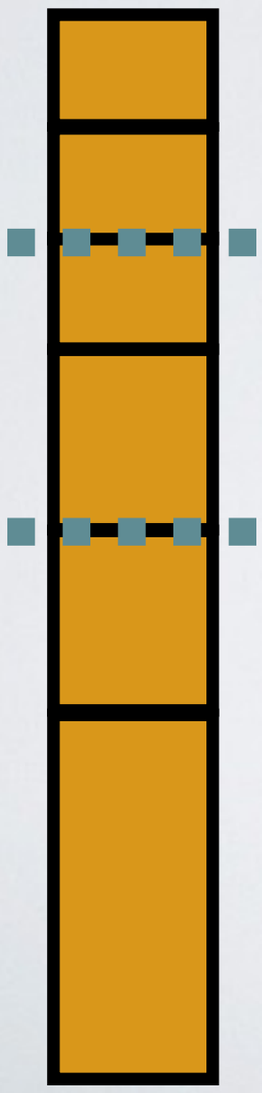
2. **juxtapose**: find the optimal combination (test all possibilities)

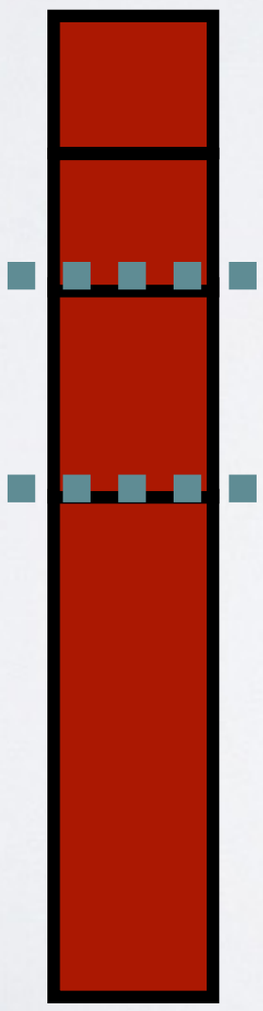


M1 M2 M3

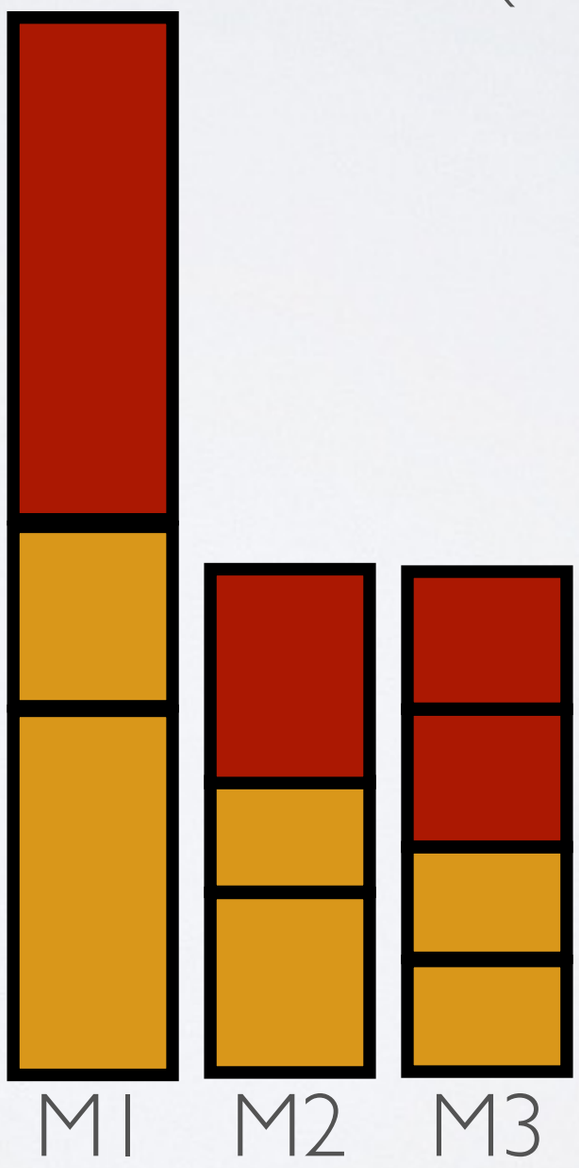
CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)



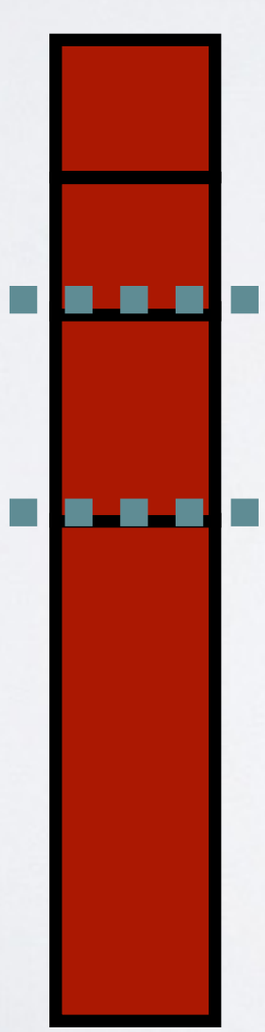


2. **juxtapose**: find the optimal combination (test all possibilities)

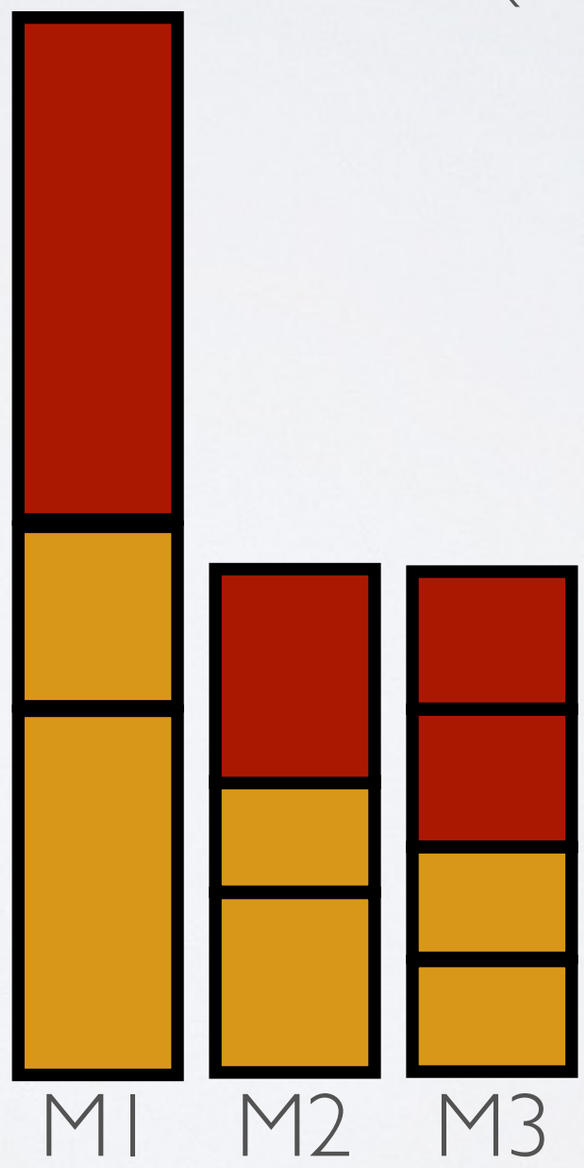


CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)



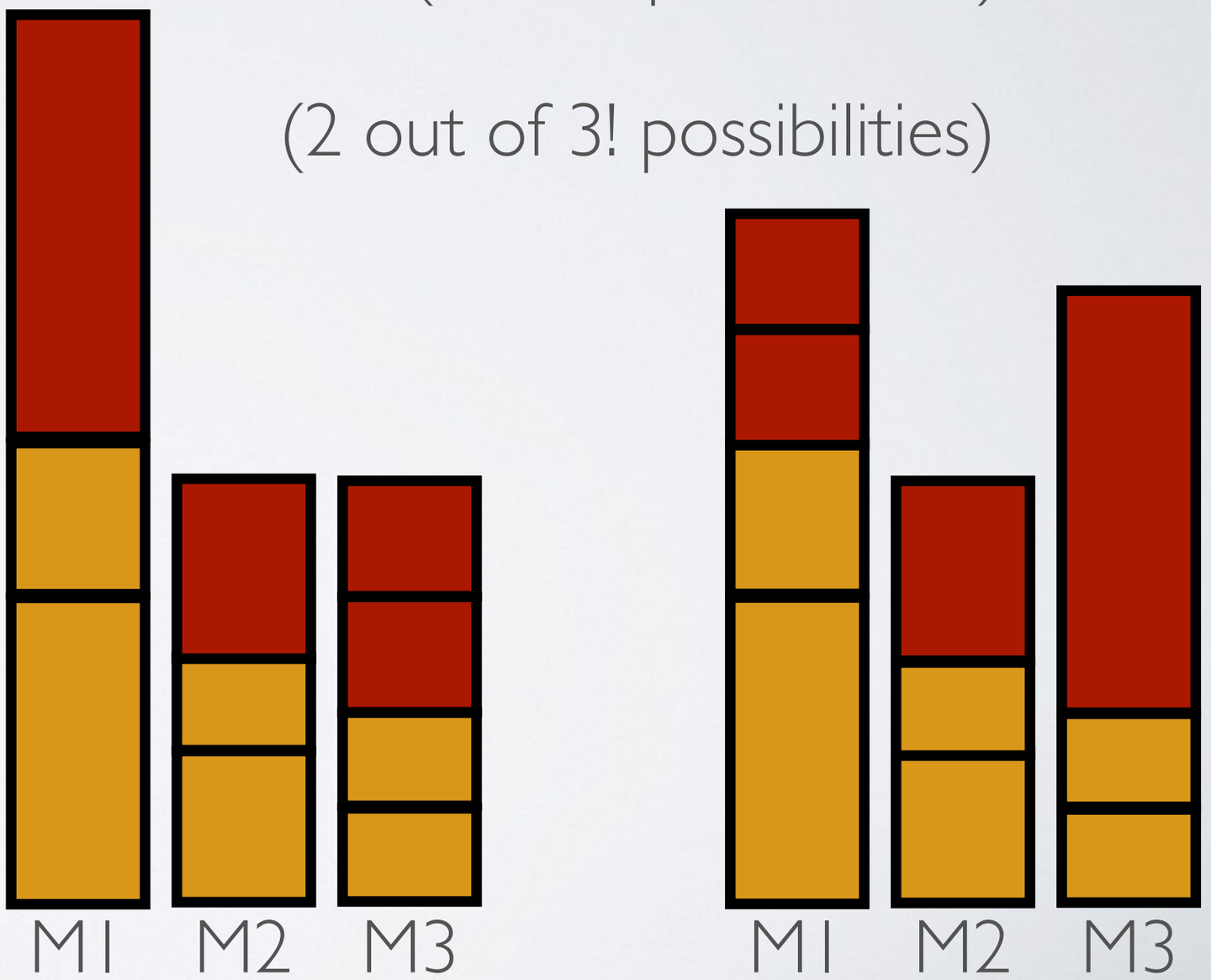
2. **juxtapose**: find the optimal combination (test all possibilities)



CUT-JUXTAPOSE: POLY IN #TASKS EXPONENTIAL IN #MACHINES AND #TYPES

1. Pick (m-1) **cut** points (each type independently)

2. **juxtapose**: find the optimal combination (test all possibilities)



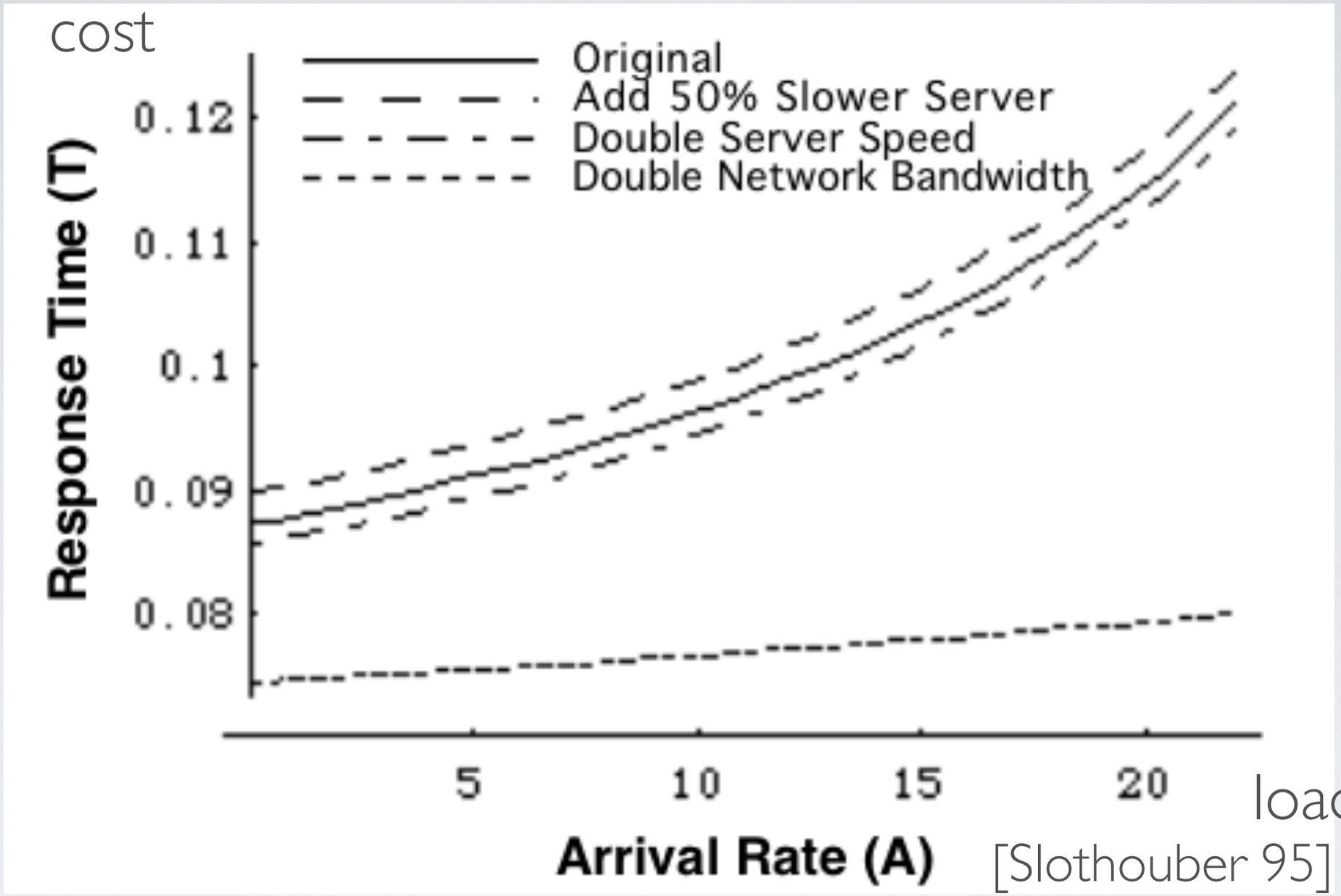
overall complexity:

$$O(n^{(m-1)T}(m!)^{T-1})$$

OTHER RESULTS IN THE LINEAR COST MODEL

- dynamic programming algorithm when the number of lengths of jobs is constant $O(mn^2 \sum_t l_t)$

- dynamic programming algorithm for a single type $O(mn^2)$



[Slothouber 95]

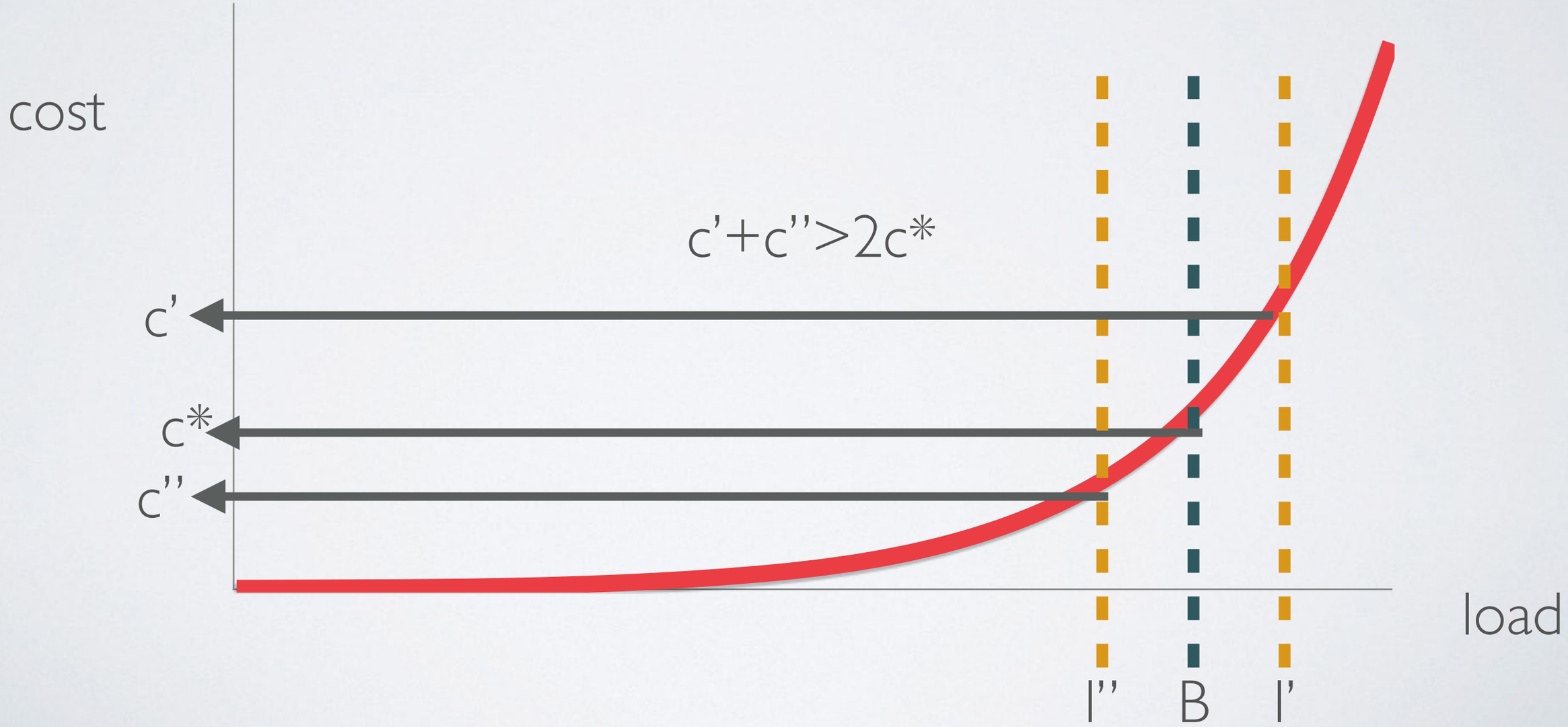
OUR RESULTS: CONVEX COST FUNCTIONS

THE TOTAL COST IS NP-HARD IF COST FUNCTION IS STRICTLY CONVEX (EVEN FOR A SINGLE TYPE)

reduction from 3-Partition

sketch: if loads are not equal to B,

the cost of exceeding B is larger than what we save elsewhere



SUMMARY: TASKS' **TYPES** ARE USEFUL FOR ALTERNATIVE MODELING OF DATACENTER RESOURCE MANAGEMENT

- datacenters are not supercomputers! (co-allocation, more regular load, no/limited queue, ….)

- theoretically-sound results are rare (compared to, e.g., standard high-performance computing)

- tasks' types model tasks' heterogeneity (a webserver, a database; a computational job) and their mutual performance impact

- we have early results on complexity (few types, few machines -> poly; many types -> NP-hard); but no approx algorithms (yet?)

ACKNOWLEDGEMENTS

- Joint work with Fanny Pascual (LIP6, Paris-Sorbonne)

- Sponsored by a Google Faculty Research Award and a grant from Polish National Research Center

- Inspired by talks with Jarek Kuśmierek (Google), Piotr Skowron (University of Warsaw → Google → Oxford)

- …and early results from services' performance measurements done by Andrzej Skrodzki (University of Warsaw)

- HiPC reviewers pointed out 2 problems in proofs!

PARTITION WITH SIDE EFFECTS

THANKS!

Krzysztof Rządca
krz@mimuw.edu.pl