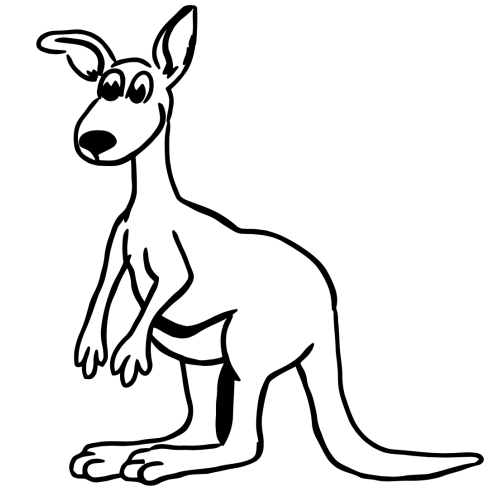
ZUSE INSTITUTE BERLIN

Harnessing over a Million CPU Cores to Solve a Single Hard Mixed Integer Programming Problem on a Supercomputer



http://scip.zib.de

SCIP OPTIMIZATION SUITE

SCIP - SoPlex - ZIMPL - GCG - UG

Yuji Shinano

Zuse Institute Berlin

Outline

ZUSE INSTITUTE BERLIN

- Background and Purpose
  - State-of-the-art Mixed Integer Programming (MIP) solvers
  - Parallelization of MIP solvers
- Ubiquity Generator (UG) framework and ParaSCIP
- Computational results for solving previously unsolved MIP instances on supercomputers
- How to harness over a million CPU cores
- Concluding remarks

Background and Purpose

ZUSE INSTITUTE BERLIN

MIP (Mixed Integer Linear Programming)

- minimizes or maximizes a linear function
- is subject to linear constraints
- has integer and continuous variables

$$\min\{c^\top x : Ax \leq b, l \leq x \leq u, x_j \in \mathbb{Z}, \text{ for all } j \in I\}$$
$$A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c, l, u \in \mathbb{R}^n, I \subseteq \{1, \ldots, n\}$$

The most general form of combinatorial optimization problems

Many applications

Customer Applications

(2012 Gurobi Sales – 200+ new customers)

- Accounting
- Advertising
- Agriculture
- Airlines
- ATM provisioning
- Compilers
- Defense
- Electrical power
- Energy
- Finance
- Food service
- Forestry
- Gas distribution
- Government
- Internet applications
- Logistics/supply chain
- Medical
- Mining
- National research labs
- Online dating
- Portfolio management
- Railways
- Recycling
- Revenue management
- Semiconductor
- Shipping
- Social networking
- Sourcing
- Sports betting
- Sports scheduling
- Statistics
- Steel Manufacturing
- Telecommunications
- Transportation
- Utilities
- Workforce Management

Gurobi Optimization

© 2015 Gurobi Optimization

Background and Purpose

ZUSE INSTITUTE BERLIN

MIP (Mixed Integer Linear Programming)

- minimizes or maximizes a linear function
- is subject to linear constraints
- has integer and continuous variables

$$\min\{c^\top x : Ax \leq b, l \leq x \leq u, x_j \in \mathbb{Z}, \text{ for all } j \in I\}$$
$$A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c, l, u \in \mathbb{R}^n, I \subseteq \{1, \ldots, n\}$$

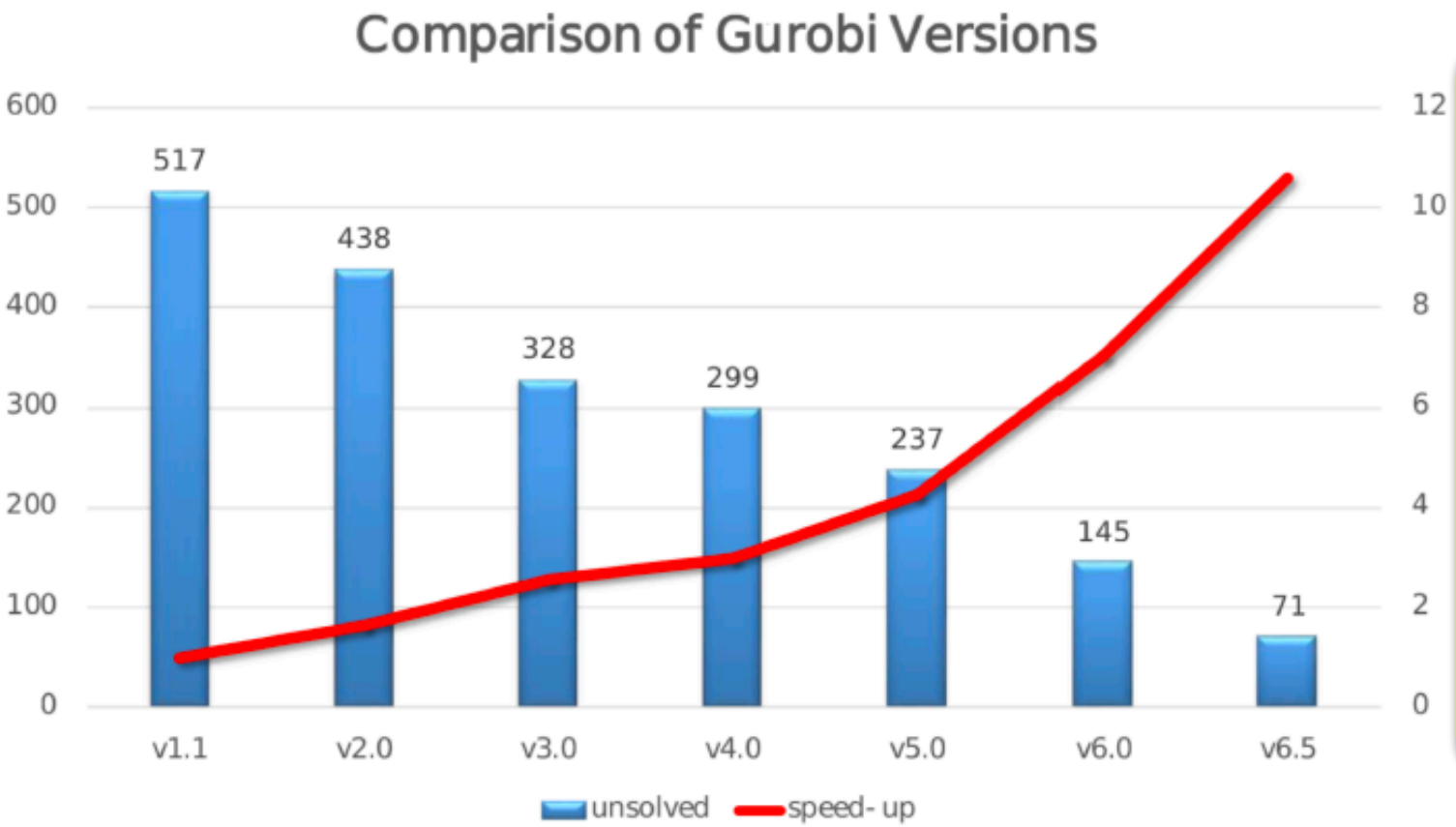
The most general form of combinatorial optimization problems

Many applications

MIP solvability has been improving

Progress in a state-of-the-art MIP solver

Gurobi Keeps Getting Better



Time limit: 10000 sec.
Intel Xeon CPU E3-1240 v3 @ 3.40GHz
4 cores, 8 hyper-threads
32 GB RAM

Test set has 3741 models:
- 235 discarded due to inconsistent answers
- 934 discarded that none of the versions can solve
- speed-up measured on >100s bracket: 1205 models

Time limit:
10000 sec.
Test set:
3741 model
- 235 discarded due to inconsistent answers
- 934 discarded none of the version can solve
- speed-up measured on >100s bracket; 1205 models

© 2016 Gurobi Optimization

GUROBI OPTIMIZATION

Background and Purpose

ZUSE INSTITUTE BERLIN

MIP (Mixed Integer Linear Programming)

- minimizes or maximizes a linear function
- is subject to linear constraints
- has integer and continuous variables

$$\min\{c^\top x : Ax \leq b, l \leq x \leq u, x_j \in \mathbb{Z}, \text{ for all } j \in I\}$$
$$A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, c, l, u \in \mathbb{R}^n, I \subseteq \{1, \ldots, n\}$$

The most general form of combinatorial optimization problems

Many applications

MIP solvability has been improving

Development of a massively parallel MIP solver

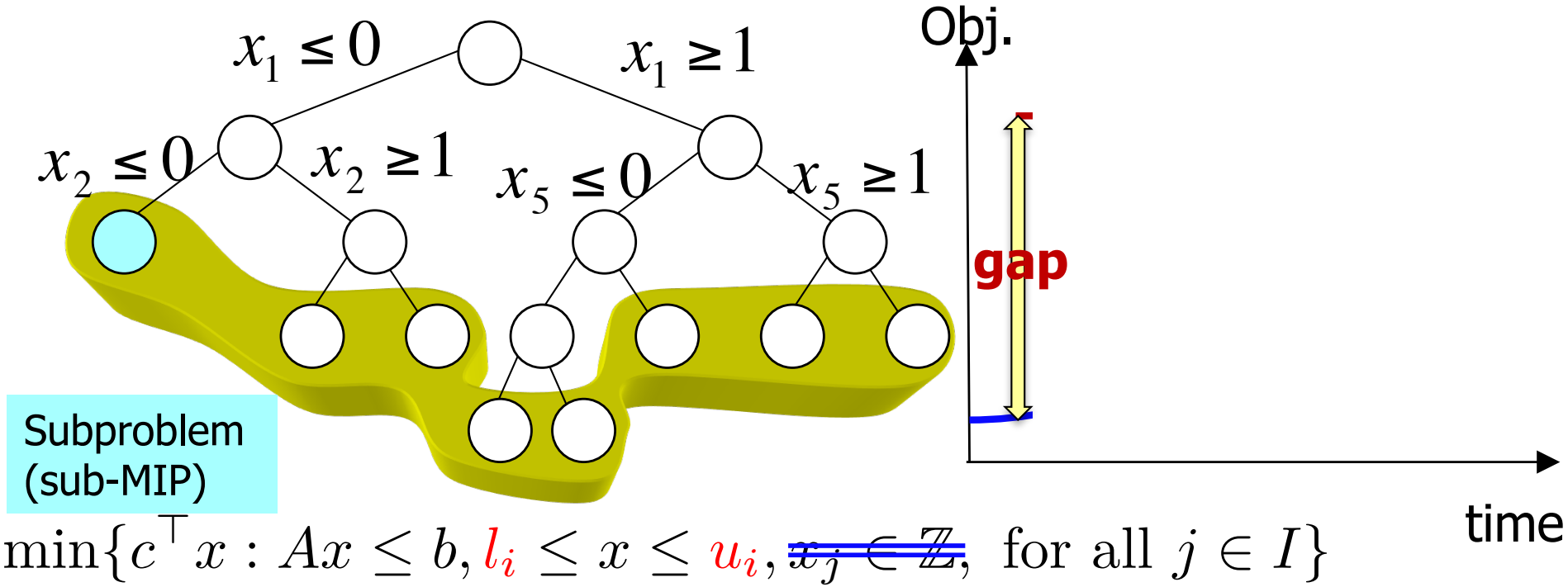
- can solve instances that cannot be solved by state-of-the-art MIP solvers
- keeps catching up performance improvements of state-of-the-art MIP solvers

Parallelization of MIP solvers

ZUSE INSTITUTE BERLIN

Branch-and-bound looks suitable for parallelization

- MIP solvers: LP based Branch-and-cut algorithm



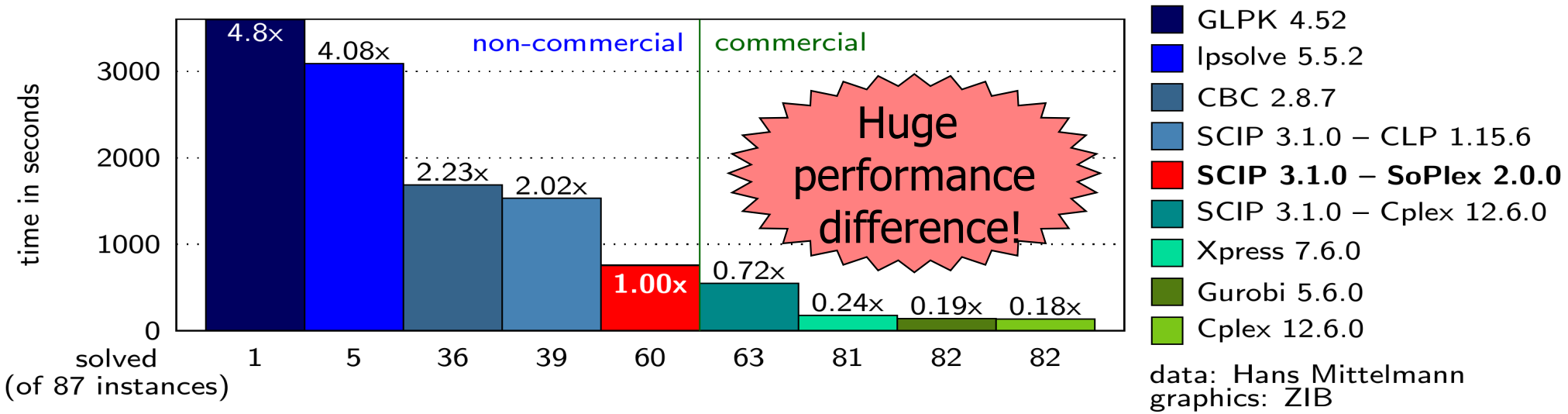
$$\min\{c^\top x : Ax \le b, l_i \le x \le u_i, \cancel{x_j \in \mathbb{Z}}, \text{ for all } j \in I\}$$

Subproblems (sub-MIPs) can be processed independently

Utilize the large number of processors
for solving extremely hard MIP instances
(**previously unsolved problem instances from MIPLIB**)

Performance of state-of-the-art MIP Solvers

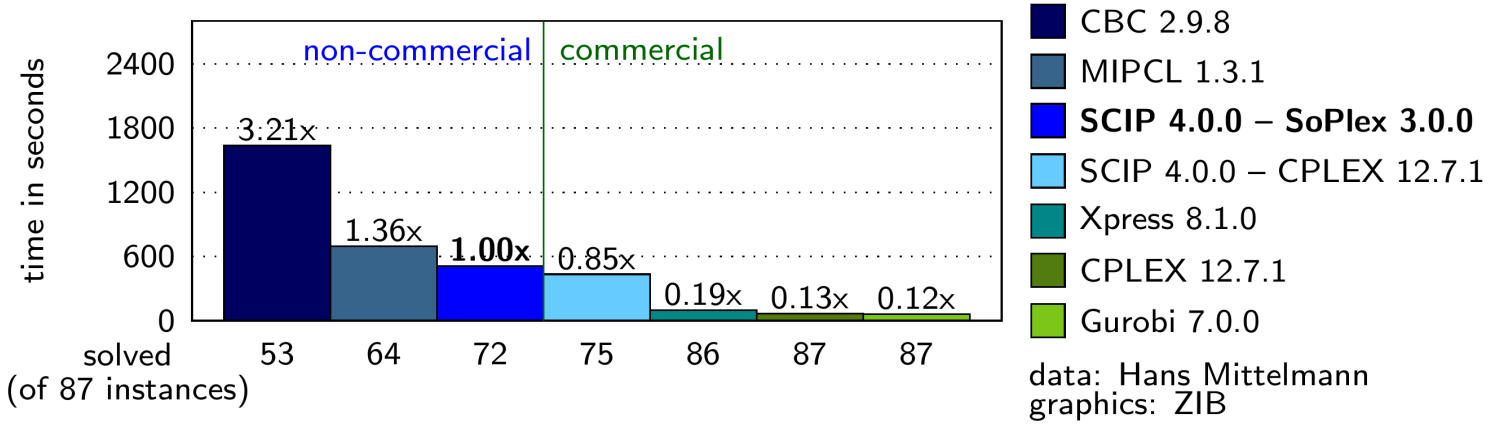
ZUSE INSTITUTE BERLIN



MIP solver benchmark (1 thread): Shifted geometric mean of results taken from the homepage of Hans Mittelmann (23/Mar/2014).

Unsolved or failed instances are accounted for with the time limit of 1 hour.

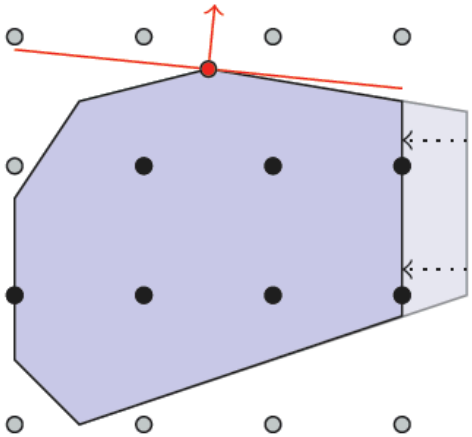
As of 14/April/2017



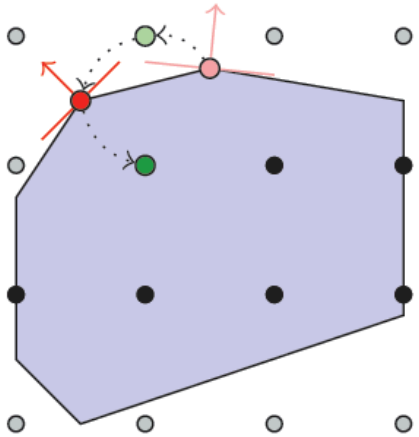
Solving techniques involved in SCIP

ZUSE INSTITUTE BERLIN

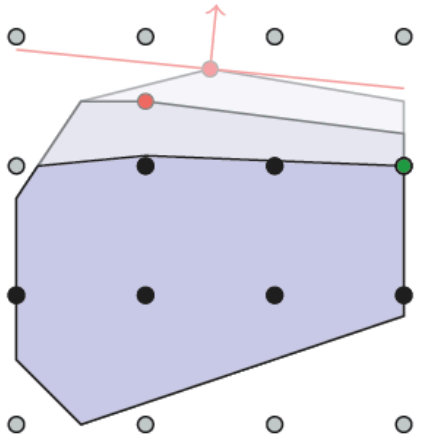
Presolving



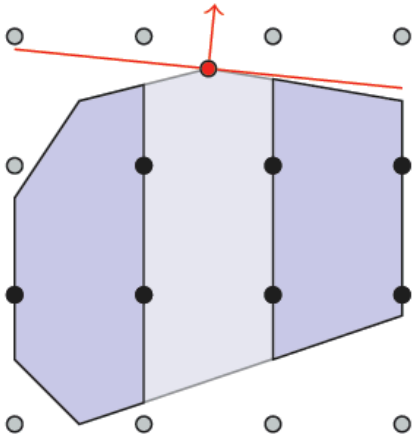
Primal heuristics



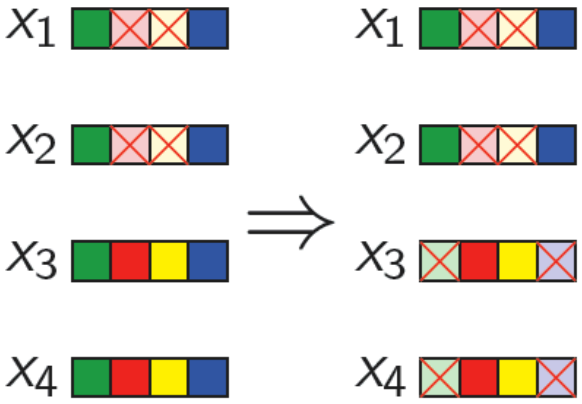
Cutting planes



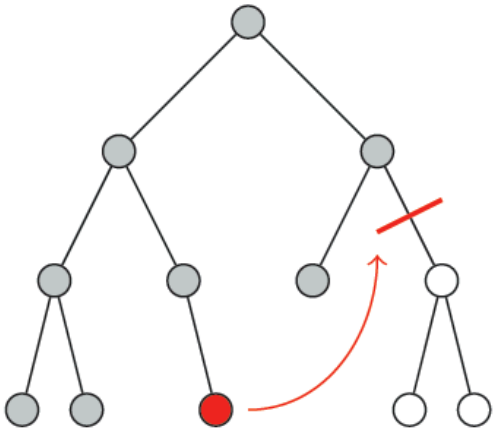
Branch & Bound



Domain propagation



Conflict analysis



UG Ubiquity Generator Framework

ZUSE INSTITUTE BERLIN

LoadCorrdinator **UG framework**

Loads are coordinated by a special process or thread

Base solver
I/O , presolve

Base solver
Using API to control solving algorithms
Using **MPI** or **pthreads** for communications

Base solver
Using API to control solving algorithms
Using **MPI** or **pthreads** for communications

Base solver
Using API to control solving algorithms
Using **MPI** or **pthreads** for communications

(UG) Solver

shared memory
ug[**SCIP**, **pthreads**]
(FiberSCIP)

Parallel Solver Instantiation

distributed memory
ug[**SCIP**, **MPI**]
(ParaSCIP)

External parallelization

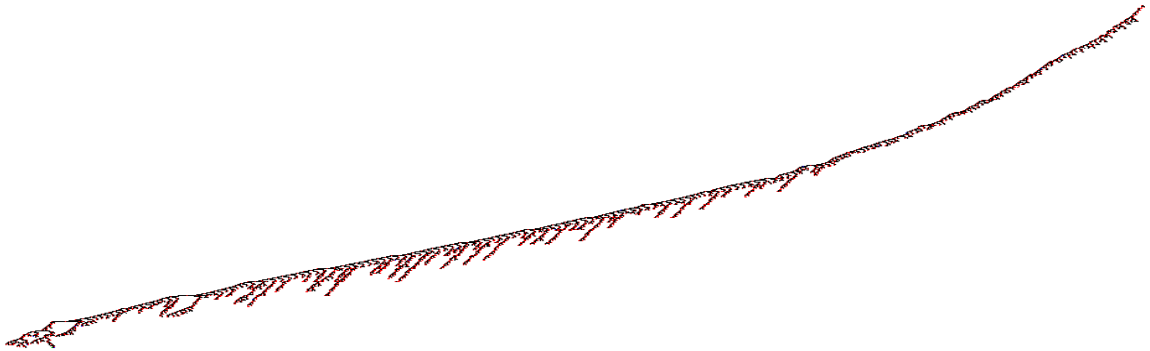
Run on PC

Run on PC clusters and supercomputers

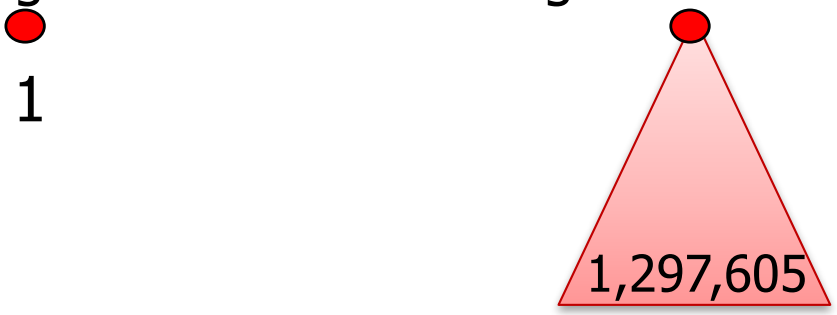
Dynamic load balancing is needed

ZUSE INSTITUTE BERLIN

- Highly unbalanced tree is generated

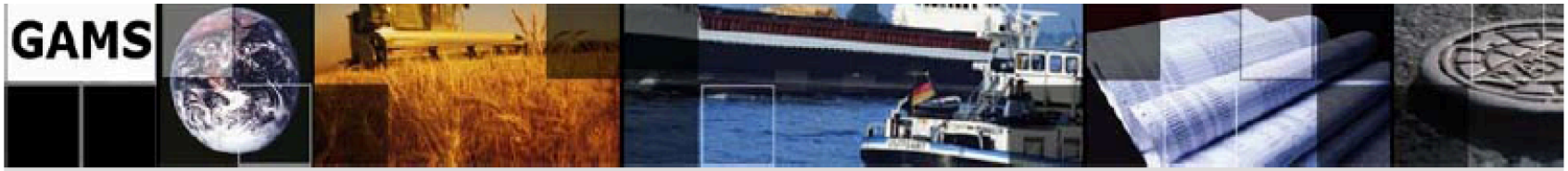


- Two types of irregularity can be handled well
  - Irregular # of nodes are generated by a sub-MIP



  - Irregular computing time for a node solving

Real observation
for solving ds in parallel
with 4095 solvers



Problems with a-priori Partitioning

- 99% of sub-problems very easy to solve
- 1% (almost) as difficult as the original problem

- How can we find *n* sub-problems with similar (but reduced) level of difficulty?
  - B&C Code keeps a list of *open/unexplored* nodes
  - Problem-bounds of these open nodes represent partitioning of the original problem

Node	Nodes Left	Objective	IInf	Best Integer	Best Node	Cuts/ ItCnt	Gap
0	0	29.6862	64		29.6862	165	
100	37	17.0000	14		25.0000	2230	
200	70	21.8429	22		24.0000	4022	

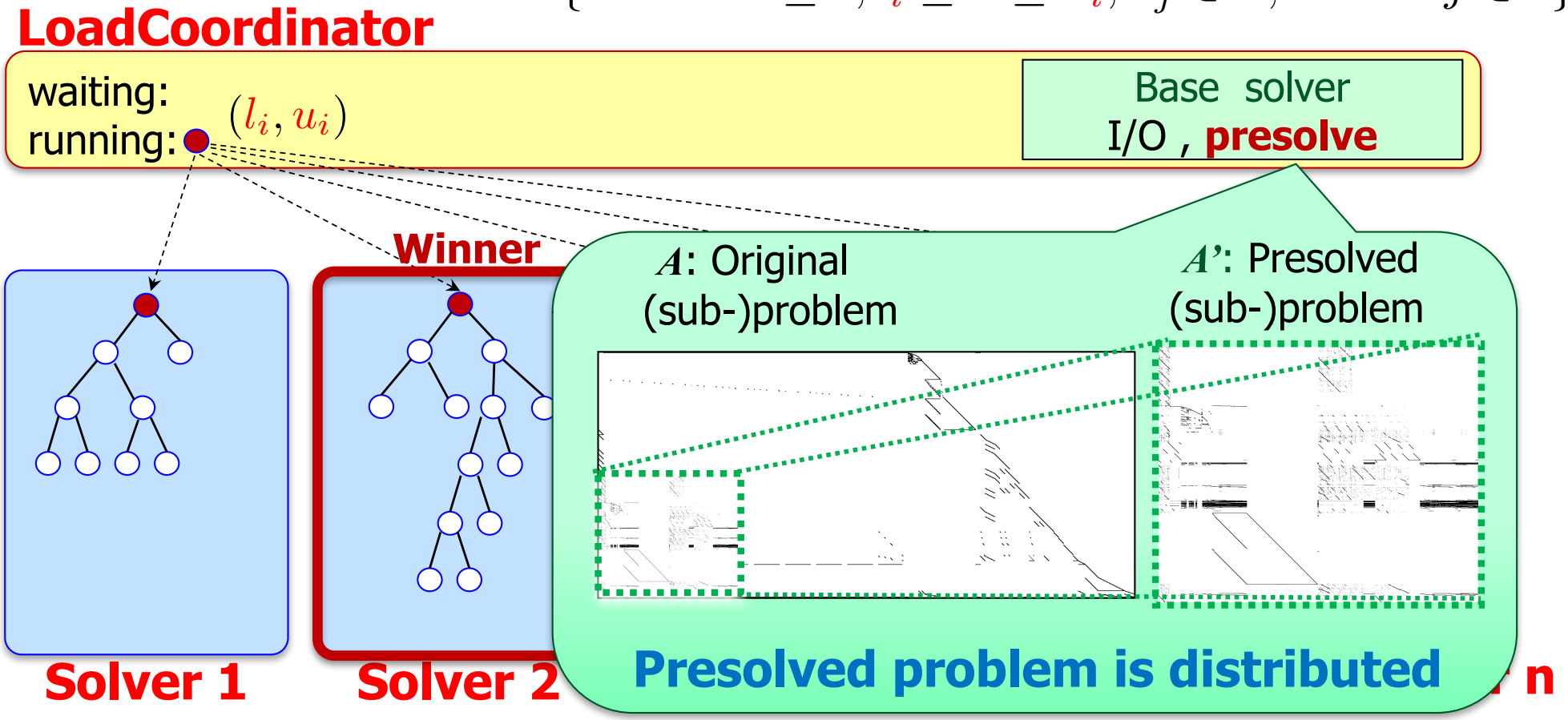
- GAMS/CPLEX Option **`dumptree n`** creates *n* bound files

How UG do parallel tree search

ZUSE INSTITUTE BERLIN

[Ramp-up(Racing)]

$$\min\{c^\top x : Ax \leq b, l_i \leq x \leq u_i, x_j \in \mathbb{Z}, \text{ for all } j \in I\}$$

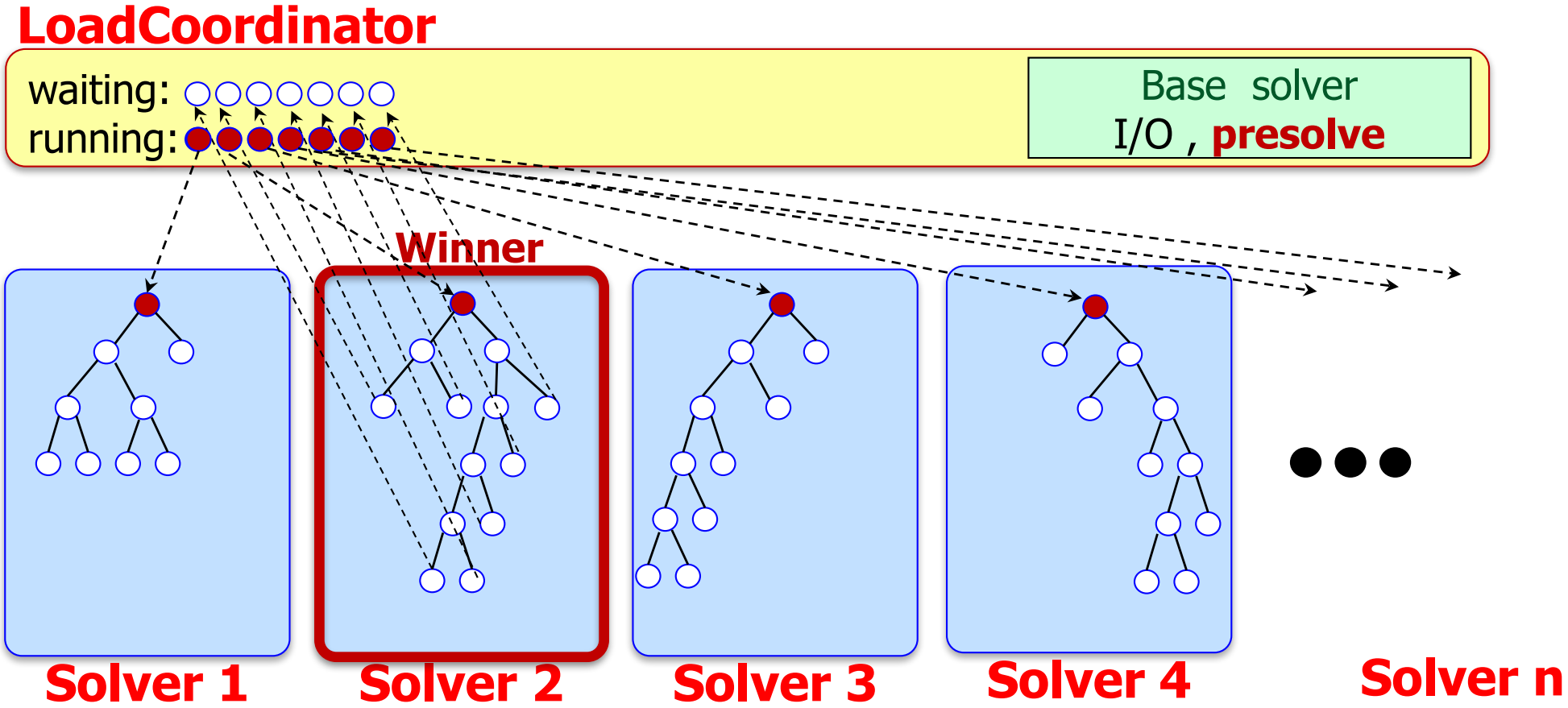


All Solvers start solving immediately, trying to generate different search trees

How UG do parallel tree search

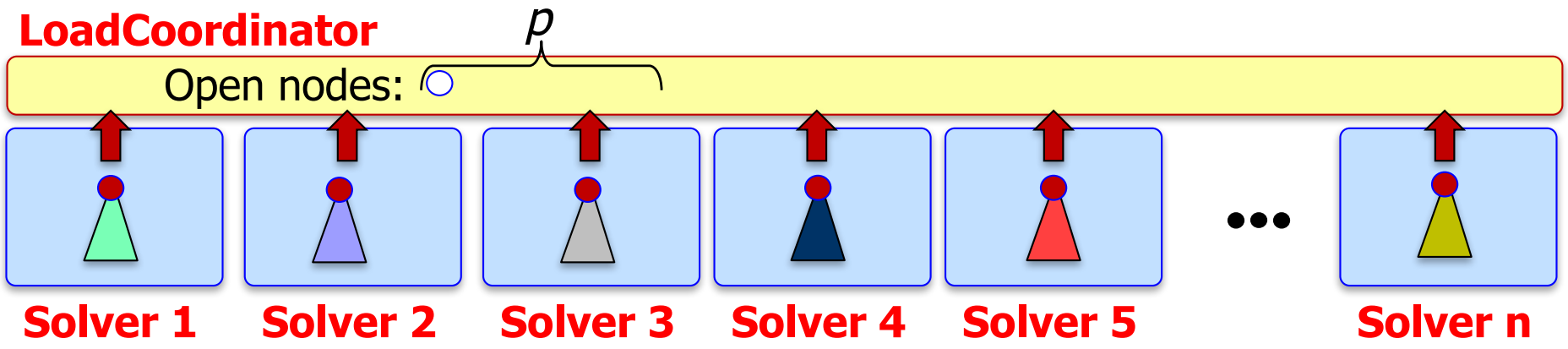
ZUSE INSTITUTE BERLIN

[Ramp-up(Racing)]



Winner is selected by taking into account dual bound, # nodes, etc.

Dynamic load balancing

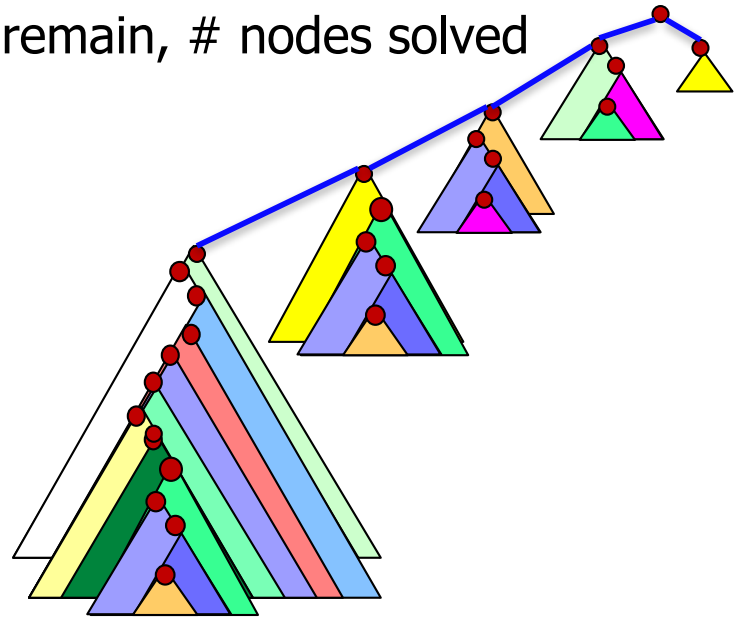


Try to keep *p* open nodes in LoadCoordinator

Notification message: best dual bound, # nodes remain, # nodes solved
- Send periodically and asynchronously
- Interval is specified by a parameter

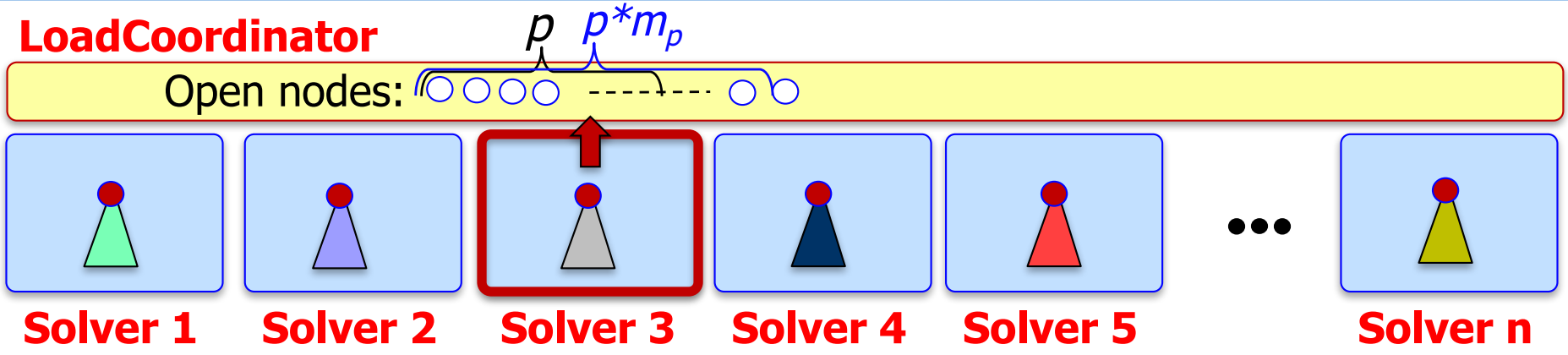
LoadCoordinator makes
selected Solvers in **collecting mode**

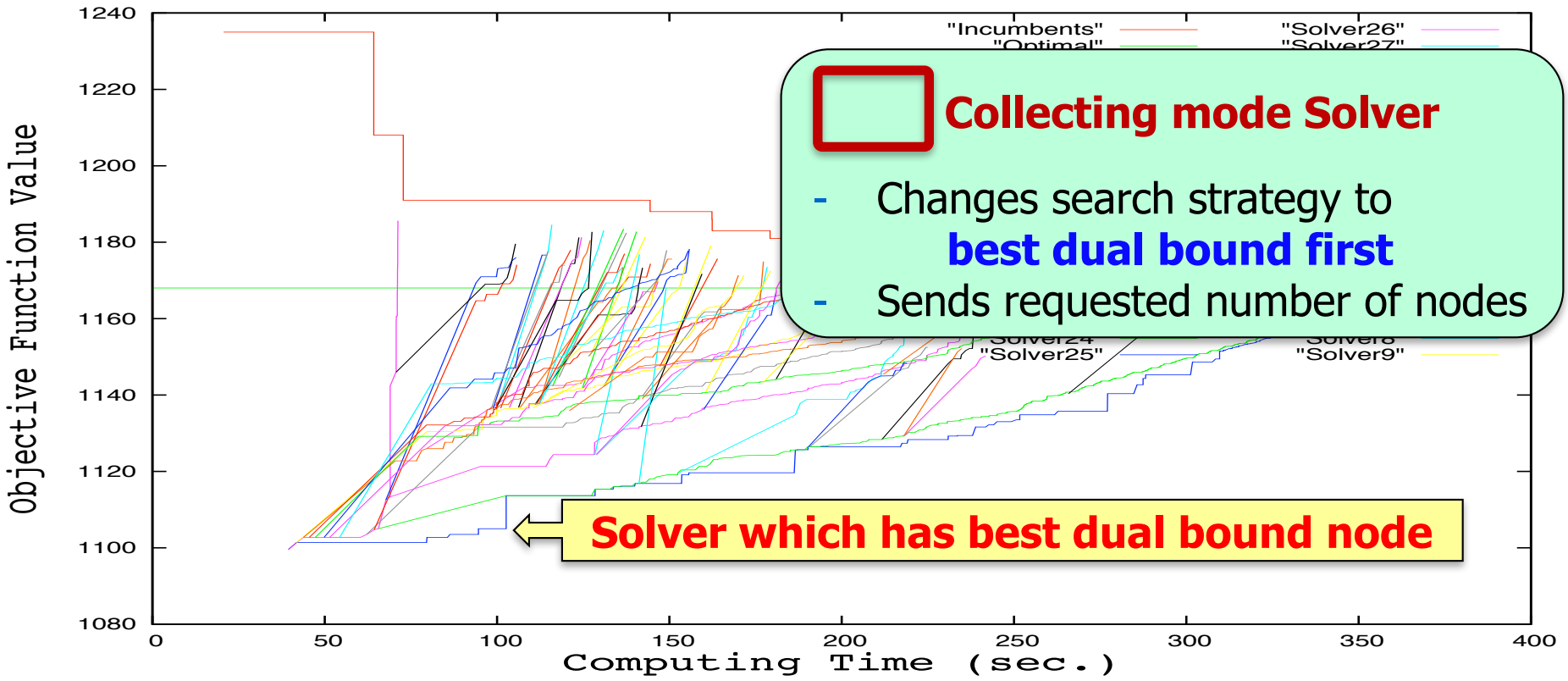
Expected to have
heavy nodes:
large subtree underneath



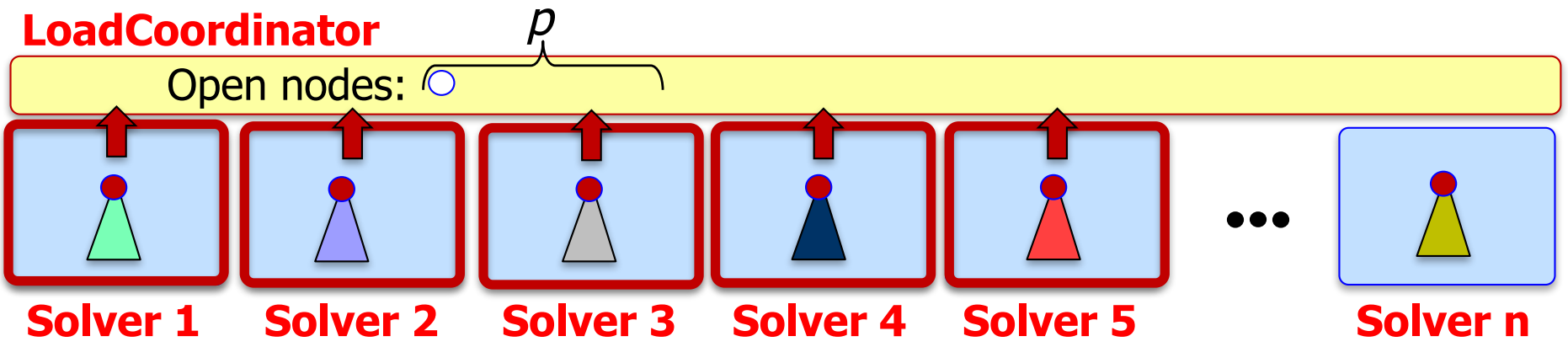
Global view of tree search

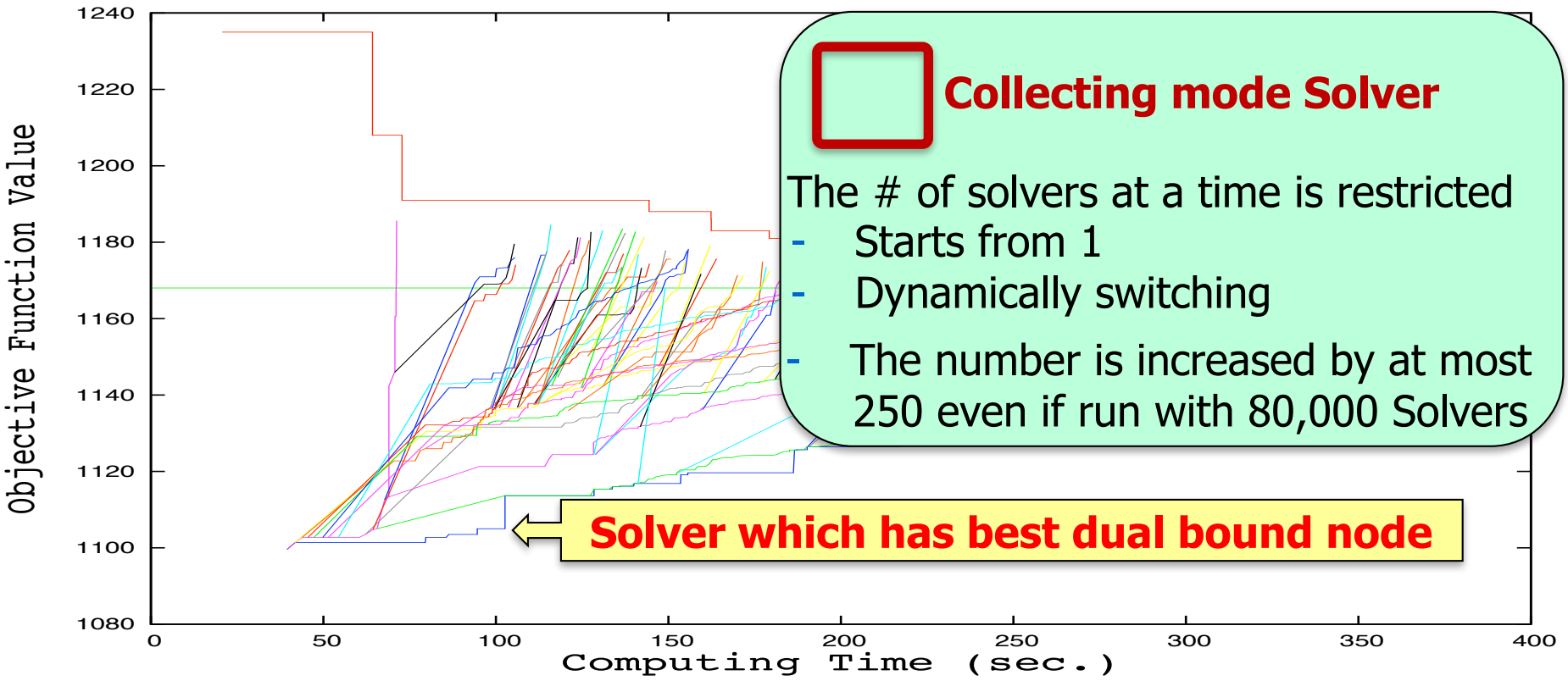
Dynamic load balancing





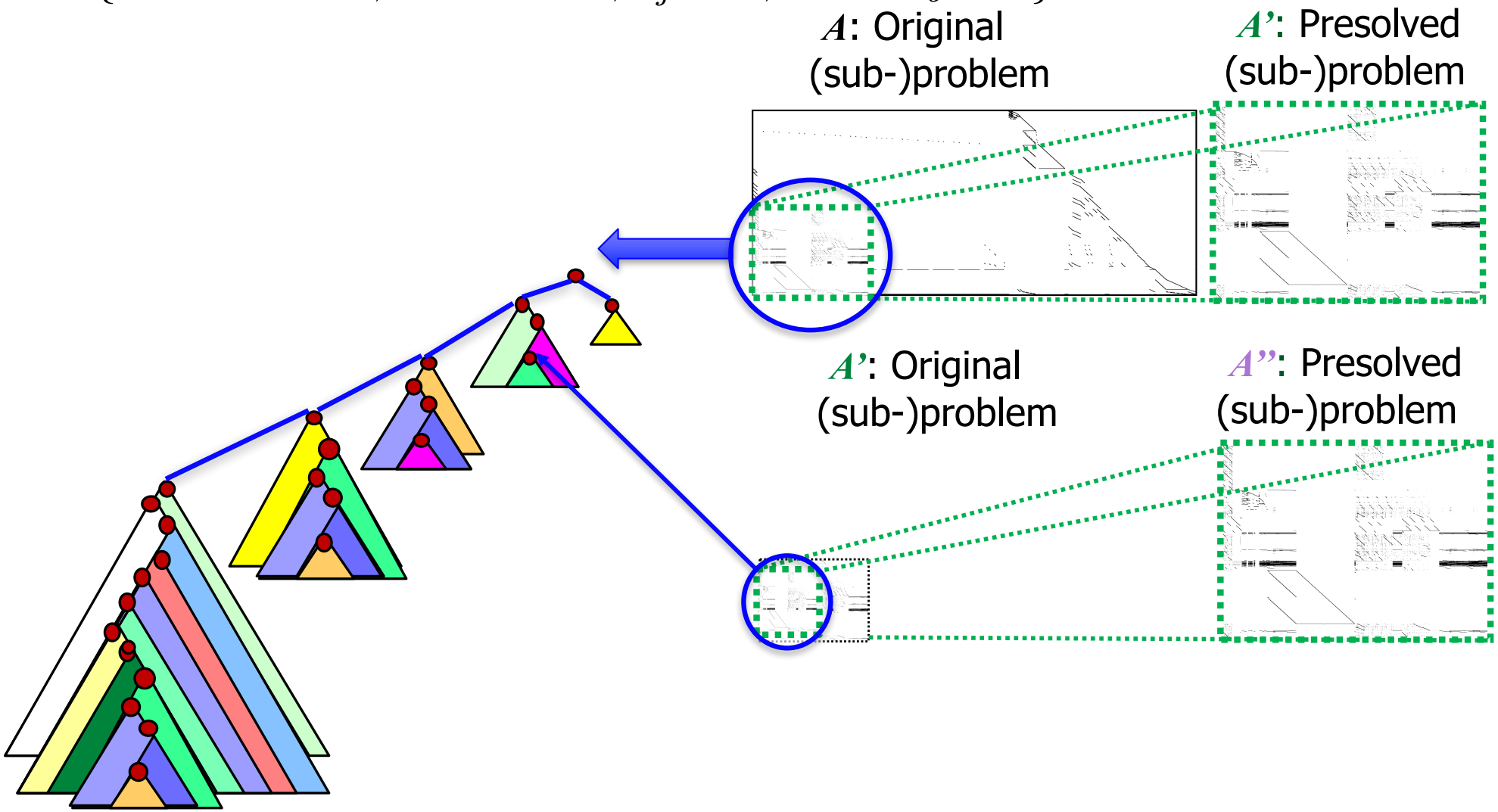
Why can it handle large scale?





Layered presolving

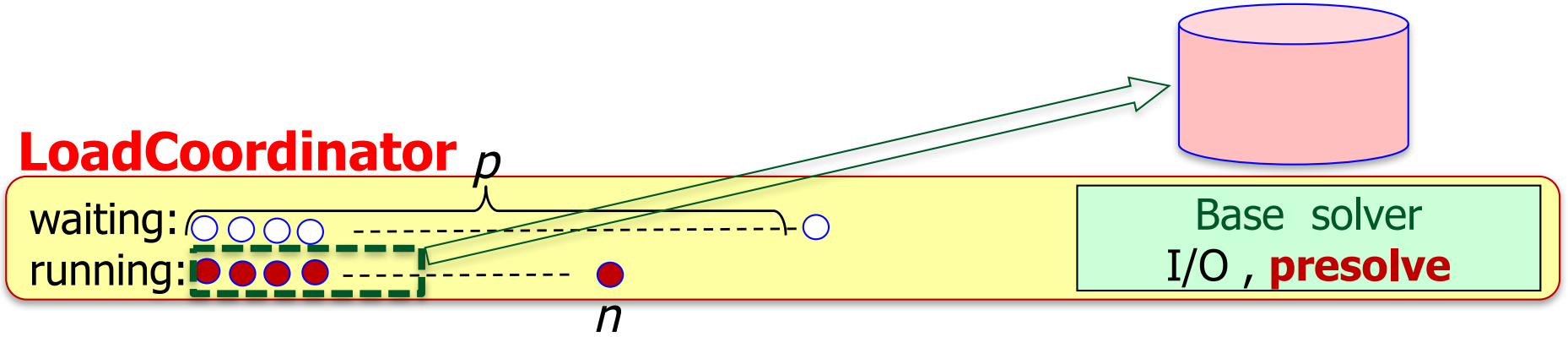
$$\min\{c^\top x : Ax \leq b, l_i \leq x \leq u_i, x_j \in \mathbb{Z}, \text{ for all } j \in I\}$$

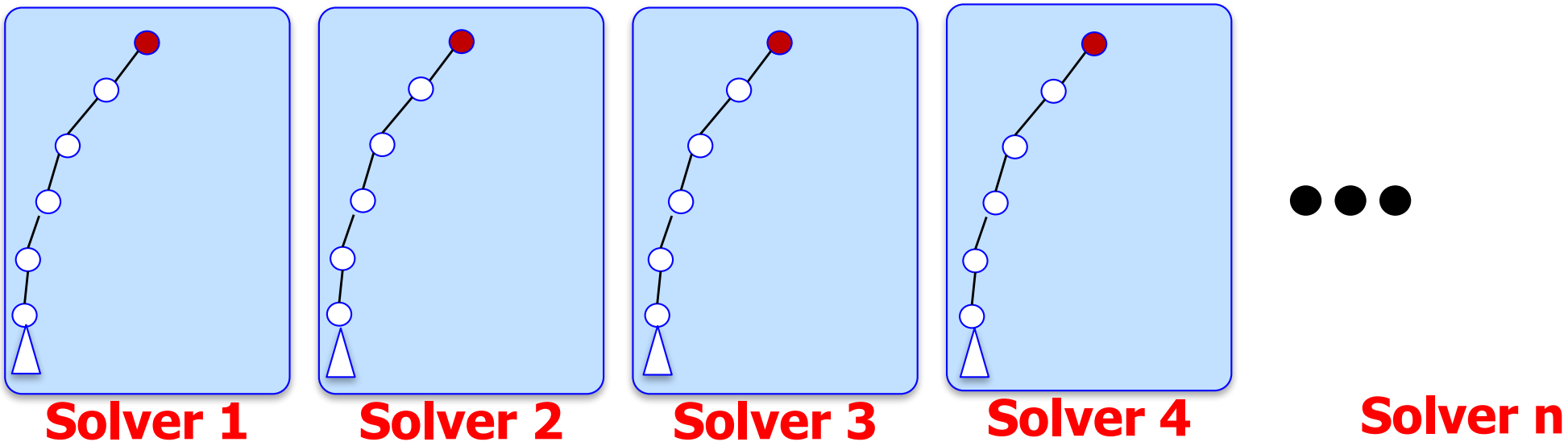


Global view of tree search

Check pointing of UG

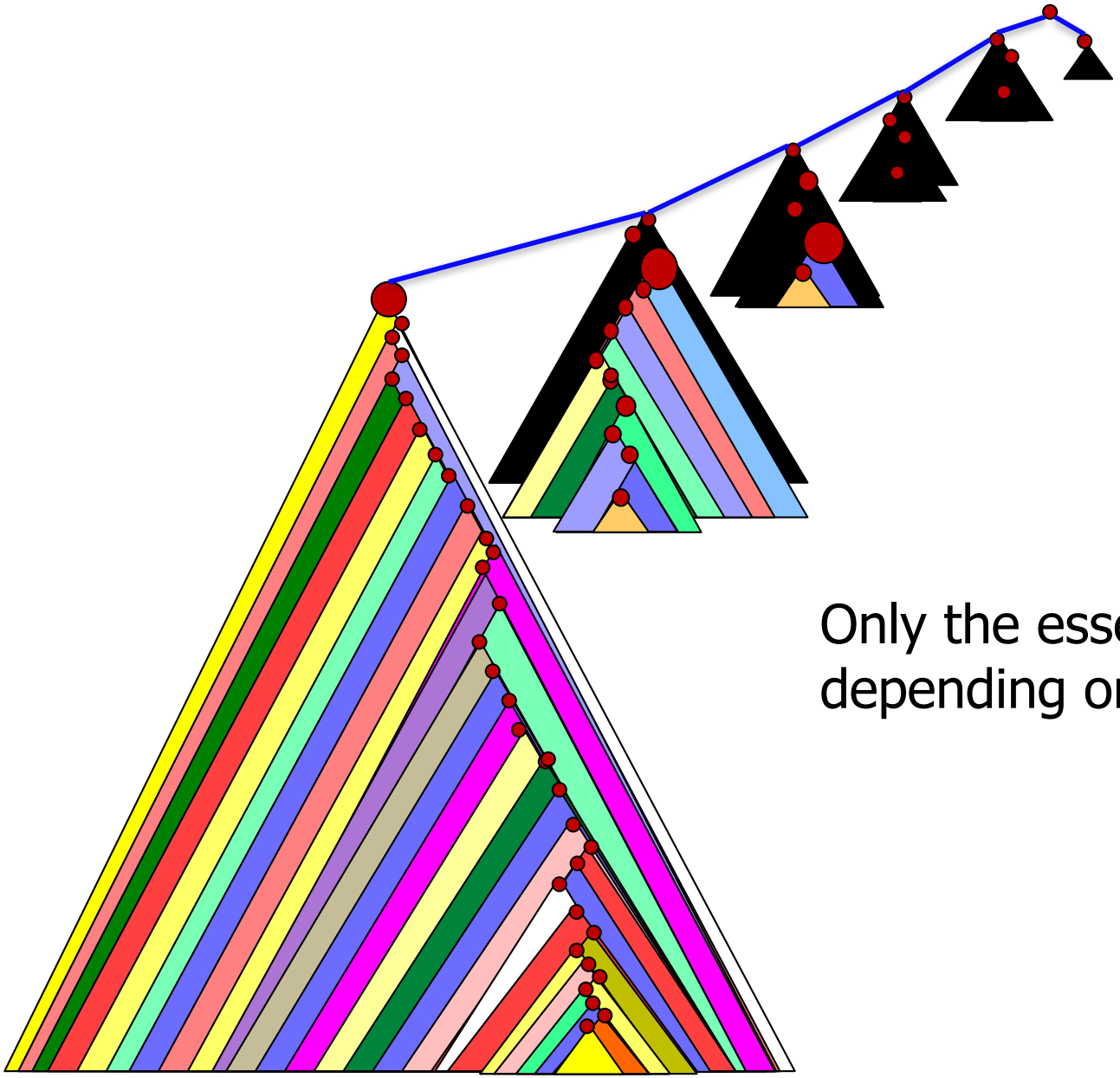
ZUSE INSTITUTE BERLIN





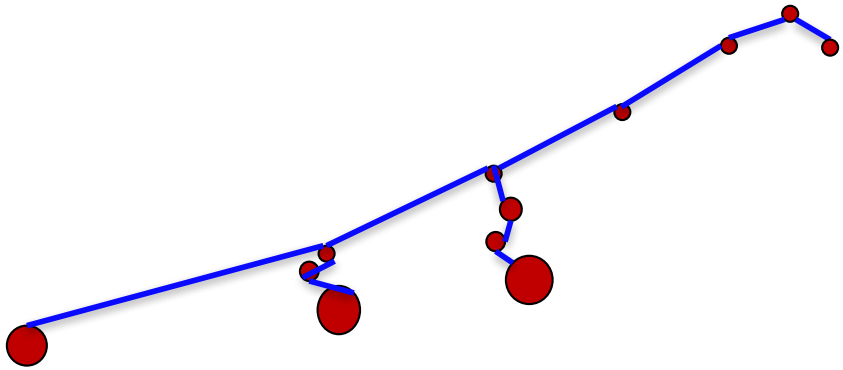
Only essential root nodes of subproblems are saved
If a sub-tree has been solved, checkpoint file contains comp. statistics

Check pointing



Only the essential nodes are saved depending on run-time situation

Restarting



Only the essential nodes are saved depending on run-time situation

Huge trees might be thrown away, but the saved nodes' dual bound values are calculated more precisely.

Main results for MIP solving by ParaSCIP

ZUSE INSTITUTE BERLIN

OPEN INSTANCES FROM MIPLIB2010 SOLVED BY PARASCIP

Date	Name	Rows	Cols	Int	Bin	Con	SCIP	CPLEX	Computer	Runs	Cores	Time(h.)	Optimal value
March 2011	rmatr200-p20	29406	29605		200	29405	2.0.1	12.2	Alibaba	1	160	2	837
March 2011	50v-10	233	2013	183	1464	366	2.0.1	12.2	HLRN II	1	1024	5	3313.18
March 2011	probportfolio	302	320		300	20	2.0.1	12.2	HLRN II HLRN II	1 2	1024 20		16.7342
March 2011	reblock354	19906	3540		3540		2.0.1	12.2	HLRN II HLR				521.2281657
Jun 2012	dg012142	6310	2080		640	1440	2.1.1	12.4					2300867
July 2012	dc1c	1649	10039		8380	1659	2.1.1					700	1767903.6501
August 2012	germany50-DBM	2526	8189	88							256	590	473840
March 2013	dolom1*	1803	11612						III	2	12288	16	6609253
January 2015	set3-10	3747							HLRN III HLRN III	3 2	6144 3072	33 24	185179.043049708
January 2015	set3-20						3.1.1	12.6	HLRN III HLRN III	1 3	6144 3072	12 36	159462.572721458

Keep solving open instances!

Novembe solved with 864 cores in 9.5 hours

December 10 was solved with 48 restarted jobs

Titan: dy XK7, Opteron 6274 16C 2.2GHz, Cray Gemini interconnect

ISM: Fujitsu PRIMERGY RX200S5

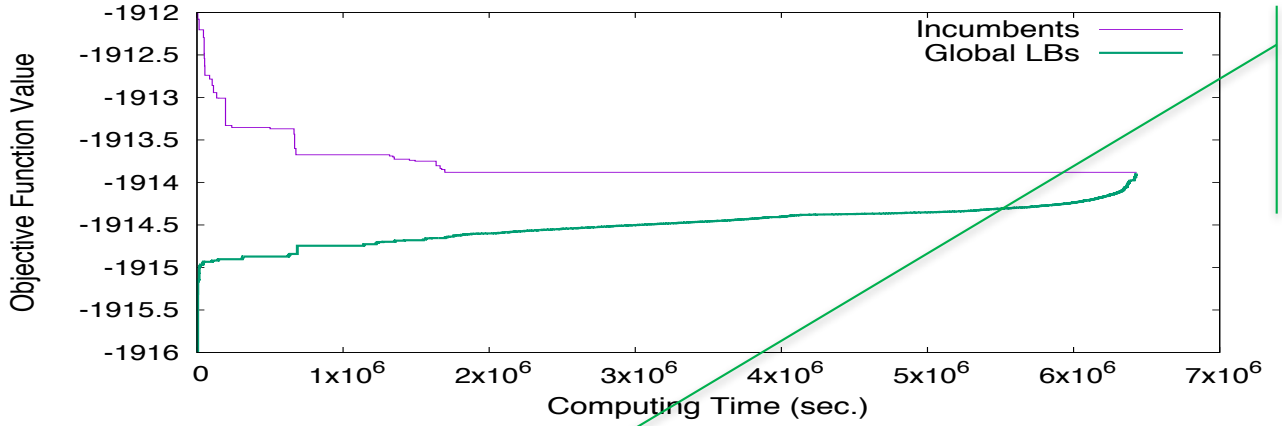
HLRN II: SGI Altix ICE 8200EX (Xeon QC E5472 3.0 GHz/X5570 2.93 GHz)

HLRN III: Cray XC30 (Intel Xeon E5-2695v2 12C 2.400GHz, Aries interconnect)

The biggest and the longest computation

ZUSE INSTITUTE BERLIN

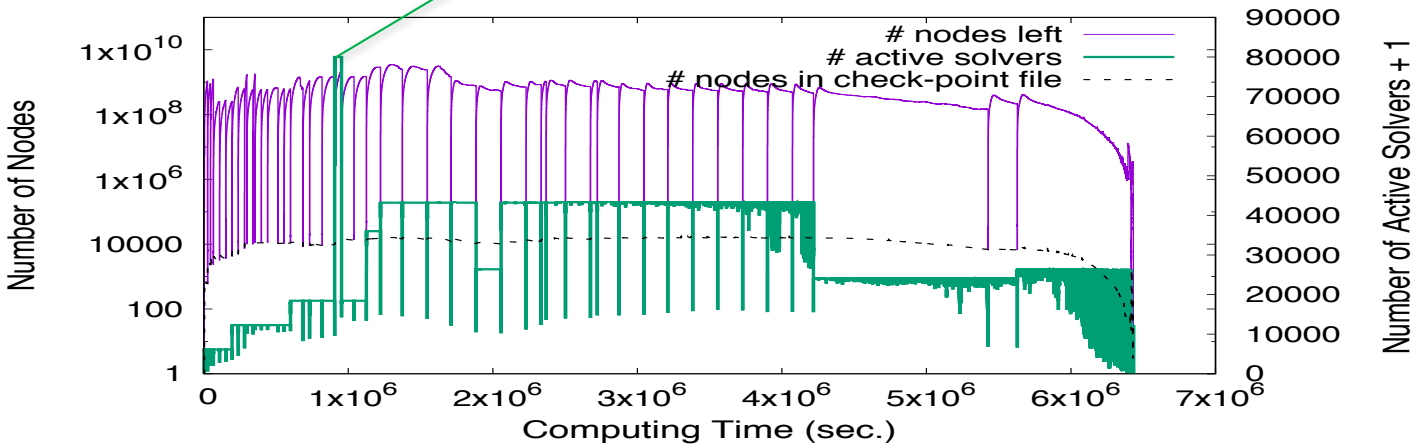
Solving rmine10: 48 restarted runs with 6,144 to 80,000 cores



How upper and lower bounds evolved

Titan with 80,000 cores
The others: HLRN III

It took about 75 days and 6,405 years of CPU core hours!



How open nodes and active solvers evolved

UG can handle up to 80,000 MPI process

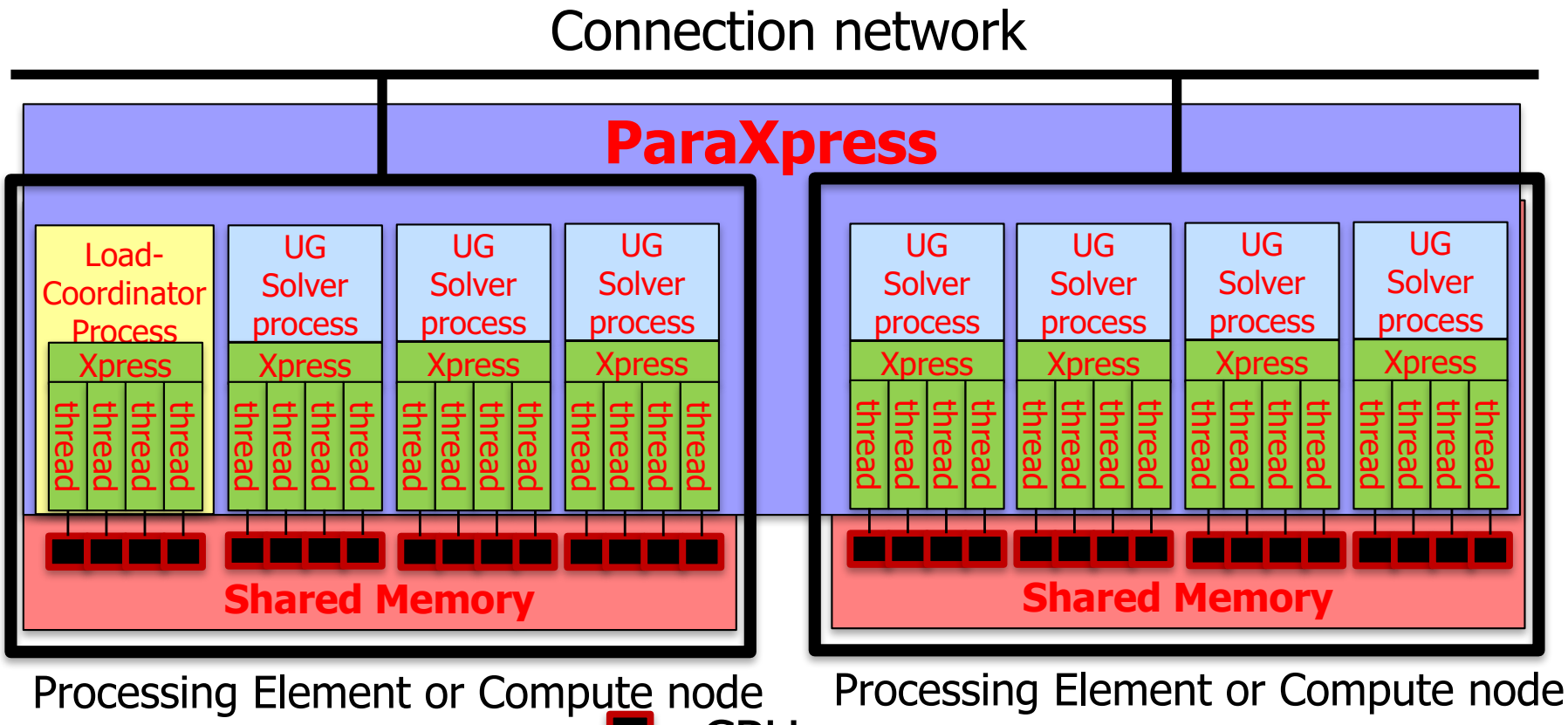
Combining with internal parallelizaion

ZUSE INSTITUTE BERLIN

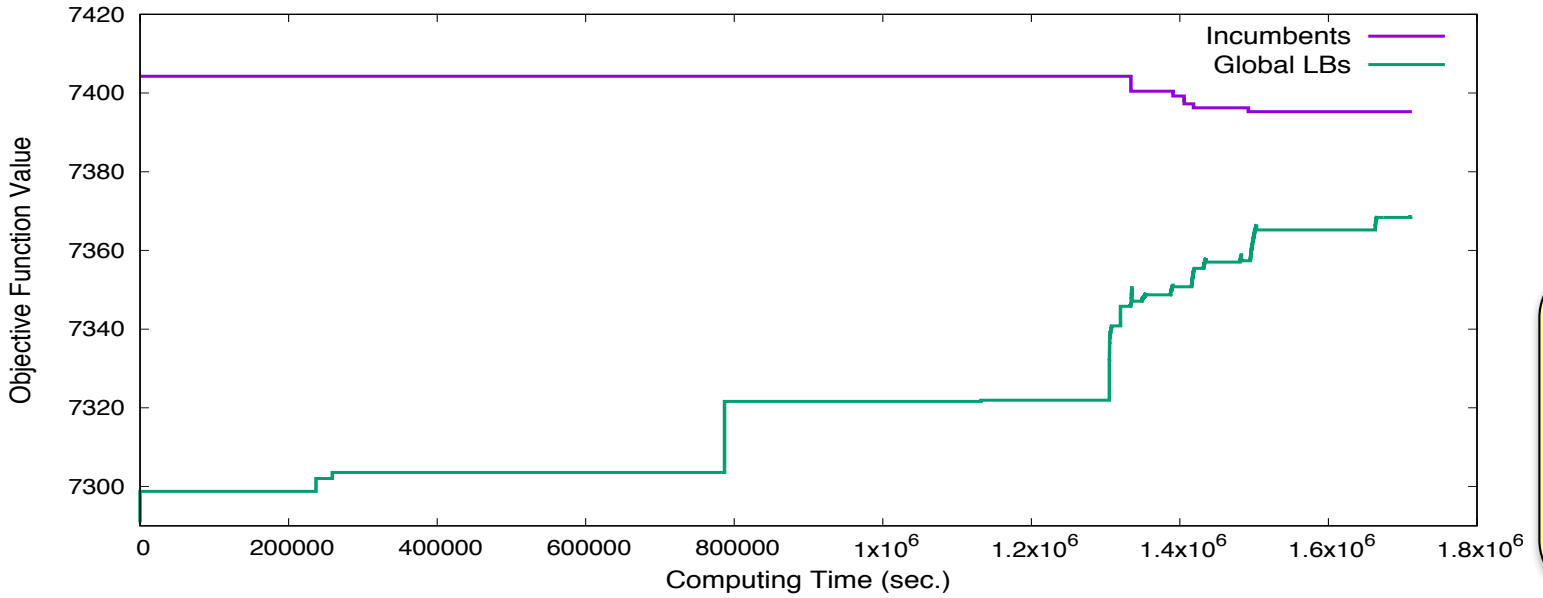
ug[Xpress,MPI] : ParaXpress

FICO™

- A powerful massively parallel MIP solver
  - Can handle, hopefully efficiently, up to 80,000 (MPI processes) x 24 (threads) = 1,920,000 (cores)

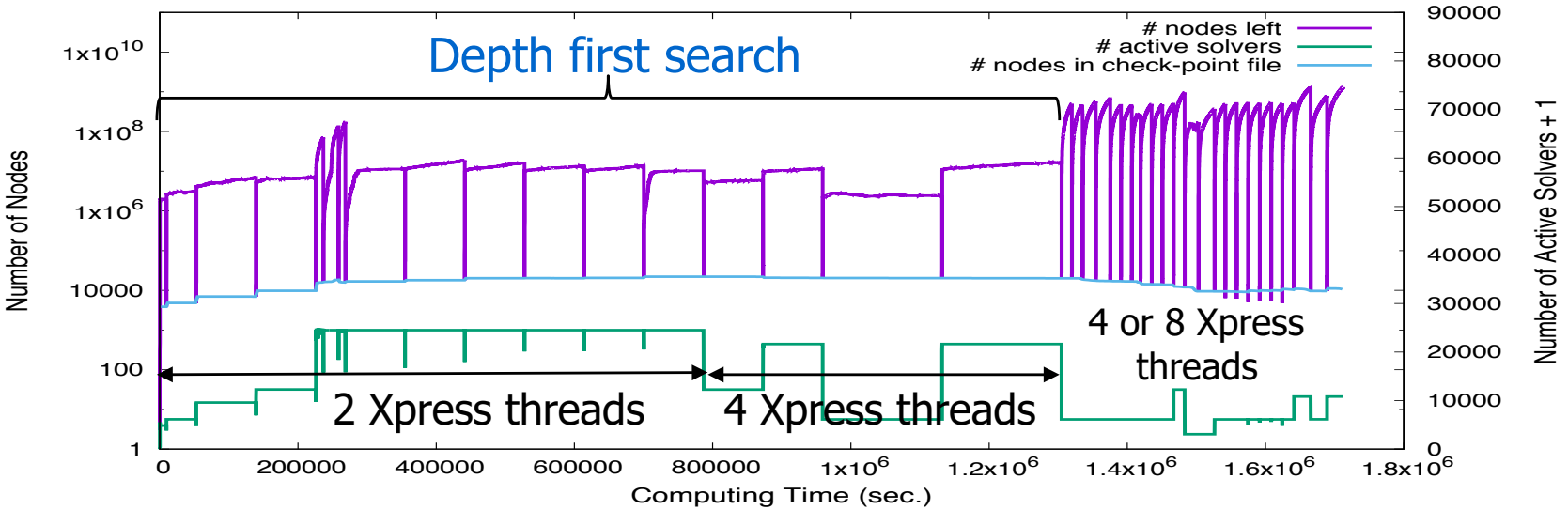


Solving open instances (ger50_17_trans)



Rows: 499
Cols: 22414
Ints:18062

Using up to 43,344 on HLRN III (Cray XC30)

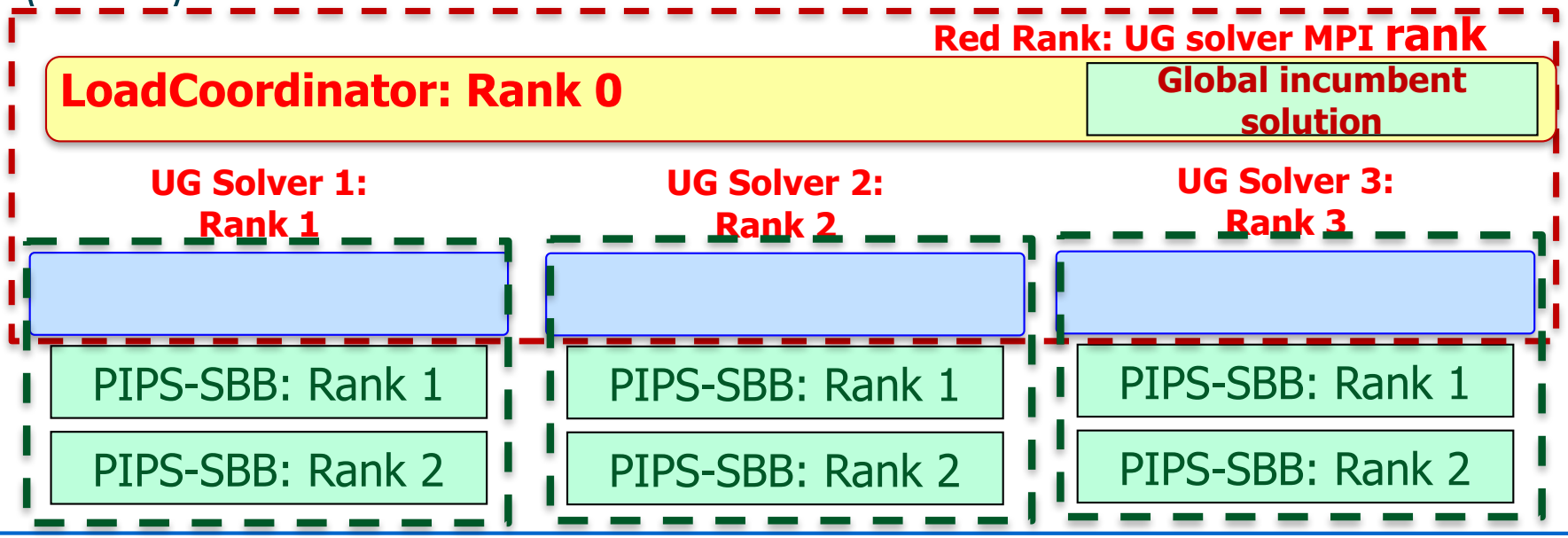


Combining with distributed MIP solver

ZUSE INSTITUTE BERLIN

ug[PIPS-SBB,MPI]

- PIPS-SBB: a specialized solver for two-stage Stochastic MIPs that uses Branch & Bound to achieve finite convergence to optimality
  - Use PIPS-S: Backbone LP solver: PIPS-S (M. Lubin, et al. Parallel distributed-memory simplex for large-scale stochastic LP problems, Computational Optimization and Applications, 2013.)
    - One branch node is processed in parallel with distributed data structure
- 80,000 (MPI processes) x 100 (PIPS-SBB MPI processes) = 8,000,000 (cores)

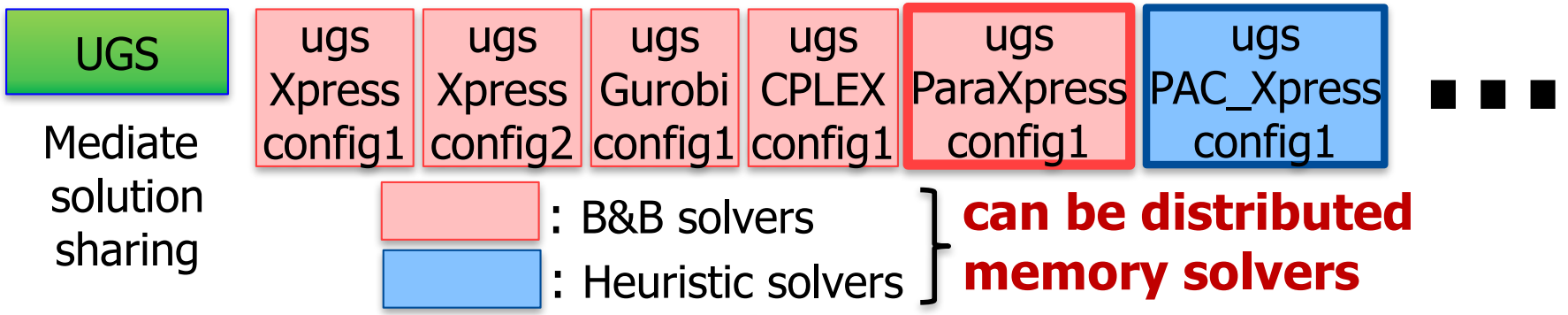


Run different solvers with different configurations in parallel

ZUSE INSTITUTE BERLIN

UGS: UG synthesizer

- Runs many different solvers in parallel
  as MPMD(Multiple Program, Multiple Data) model MPI program



The first job:
Multiple ParaXpresses run with different configurations
with completely different solver implementations
sharing incumbent solutions

The following jobs:
Restarted from the most promising check pointing file

Concluding remarks

ZUSE INSTITUTE BERLIN

- **UG** is a general framework to parallelize any kind of state-of-the-art branch-and-bound based solvers
  - **ug[SCIP,*]**
    - Tool to develop parallel general branch-and-cut solvers. Customized SCIP solver can be parallelized with the least effort
      - **ug[SCIP-Jack, *]:** Solver for Steiner Tree Problems and its variants
        - only the solver which can run on a distributed memory computing environment (solved three open benchmark instances)
  - **ug[Xpress, MPI](= ParaXpress), ug[PIPS-SBB, MPI]**
    - **Reay to run on over a million CPU cores!**
- **UGS** is another general framework to configure a parallel solver that can realize any combination of algorithm portfolio and racing

Thank you very much for your attention!