Solving Domain-Independent Dynamic Programming Problems with Anytime Heuristic Search

🎉 Best Paper Award Runner-up 🎉

Ryo Kuroiwa and J. Christopher Beck

Toronto Intelligent Decision Engineering Laboratory (TIDEL)
Department of Mechanical and Industrial Engineering
University of Toronto

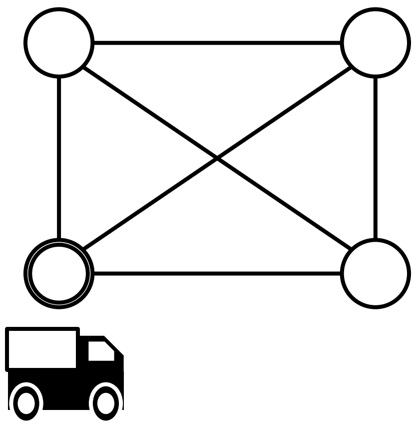
Toronto Intelligent Decision TIDEL Engineering Laboratory

UNIVERSITY OF TORONTO

Recap of DIDP

Novel model-based paradigm for combinatorial optimization

Any combinatorial optimization problem



Model

State-based DP model

compute $V(N \setminus \{0\}, 0)$

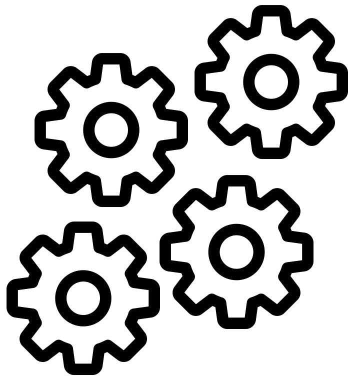
$$V(U, i) = \min_{j \in U} c_{ij} + V(U \setminus \{i\}, j)$$

$$V(\emptyset, i) = c_{i0}$$

$$V(U, i) \geq 0.$$

Solve

DIDP solver



Current solvers are based on **heuristic search**

Recap of DIDP

compute $V(N \setminus \{0\}, 0, 0)$ **Target state**

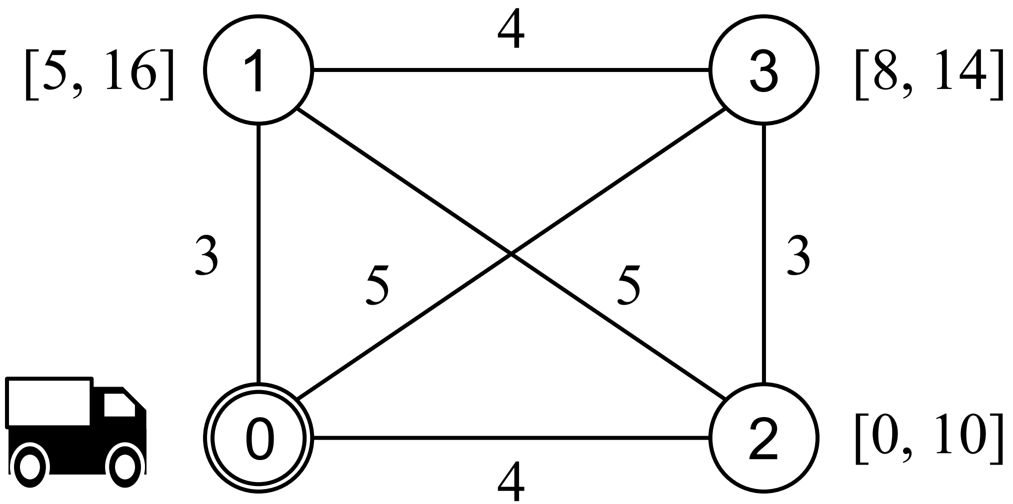
$$V(U, i, t) = \begin{cases} \min\limits_{j \in U : t + c_{ij} \leq b_j} c_{ij} + V(U \setminus \{i\}, j, \max\{t + c_{ij}, a_j\}) & \text{if } U \neq \emptyset \\ c_{ij} + V(\emptyset, 0, t + c_{i0}) & \text{else if } i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Transitions

Base case

$V(U, i, t) \leq V(U, i, t')$ if $t \leq t'$ **Dominance**

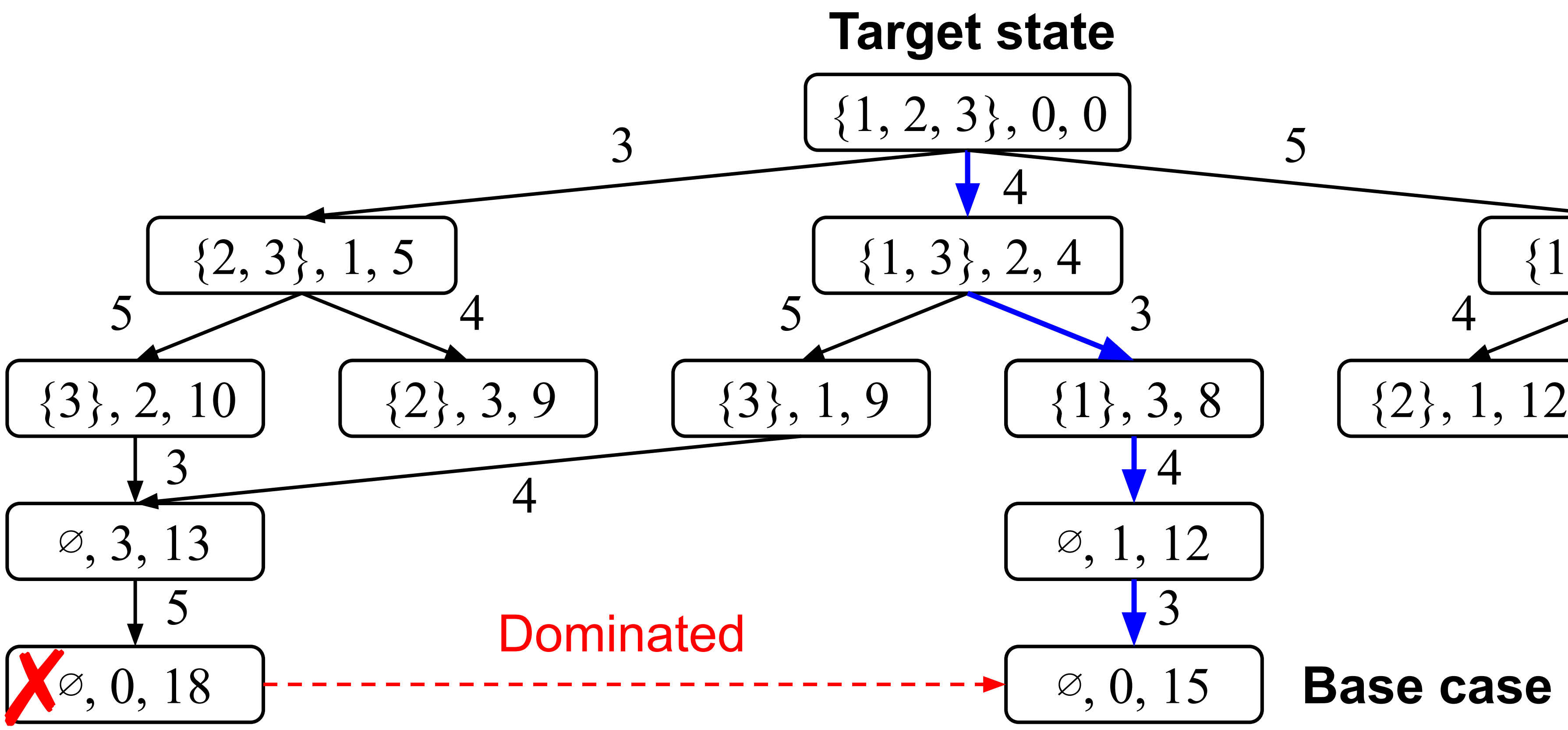
$V(U, i, t) \geq 0.$ **Dual bound**



TSPTW

Prototype Solver: CAASDy

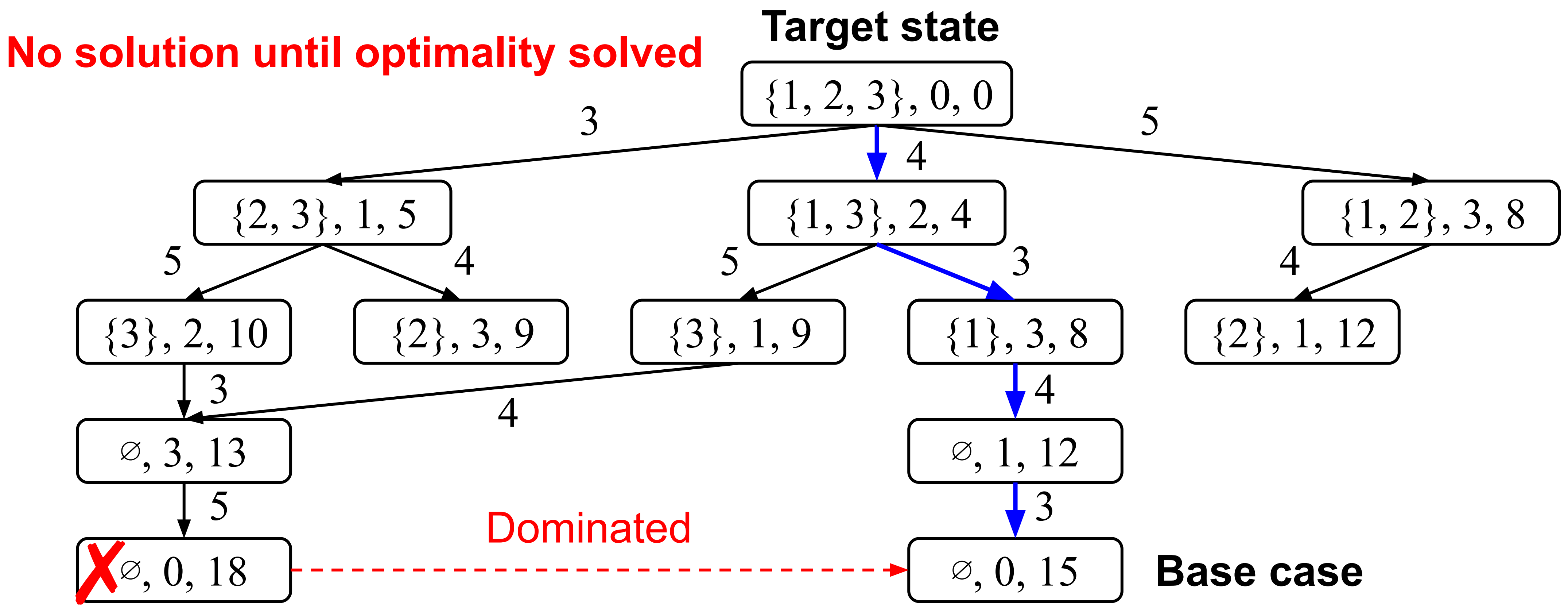
- Solve **DP as a shortest path** in the state space using A*
- **Heuristic: dual bound** defined in a DP model



Implemented in https://github.com/domain-independent-dp/didp-rs

Prototype Solver: CAASDy

- Solve **DP as a shortest path** in the state space using A*
- **Heuristic: dual bound** defined in a DP model

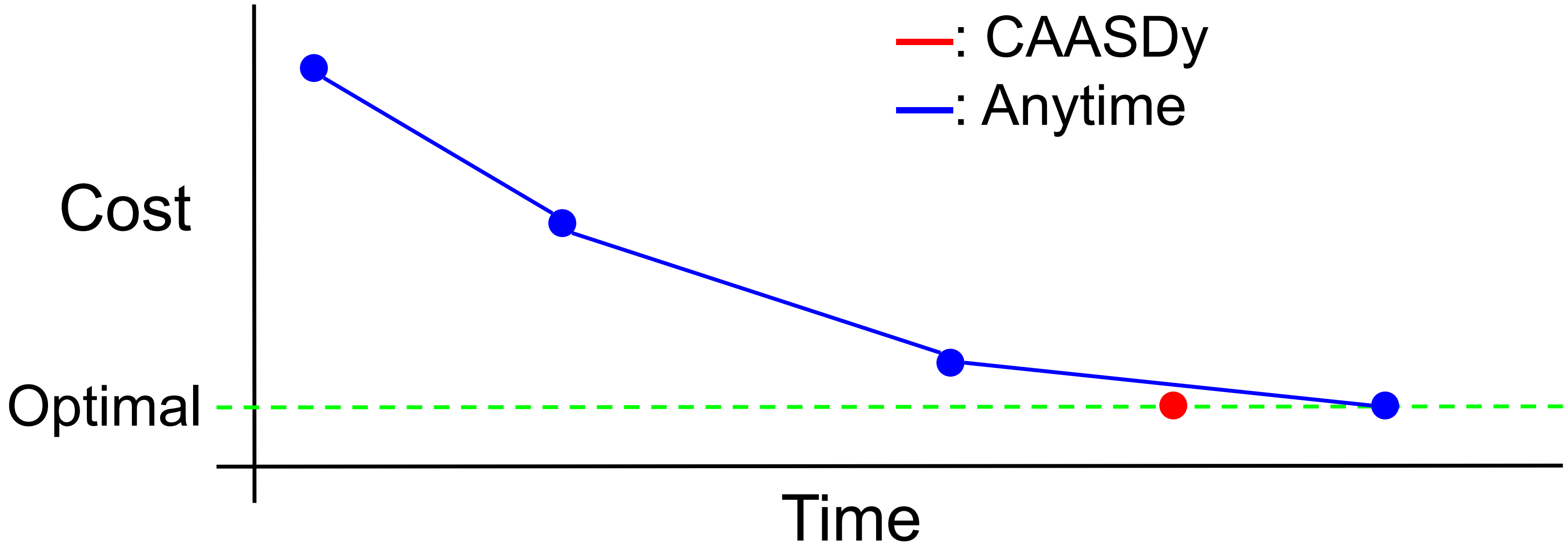


Implemented in https://github.com/domain-independent-dp/didp-rs

Anytime Solvers

- Quickly find a solution and continuously improve it
- Standard in OR (e.g., MIP and CP)

Can we develop **anytime solvers for DIDP?**



Anytime Heuristic Search Algorithms

Algorithm	Description	Reference
Depth First Branch-and-Bound (DFBnB)	DFS	
Cyclic Best-First Search (CBFS)	Hybrid of DFS and best-first search	Kao et al. 2009
Anytime Column Progressive Search (ACPS)	Hybrid of DFS and beam search	Vadlamudi et al. 2012
Anytime Pack Progressive Search (APPS)	Hybrid of DFS and beam search	Vadlamudi et al. 2016
Discrepancy-Bounded DFS (DBDFS)	Discrepancy-based	Beck and Perron 2000
Complete Anytime Beam Search (CABS)	Iterative beam search	Zhang 1998

Implemented in https://github.com/domain-independent-dp/didp-rs

Anytime Heuristic Search Algorithms

Algorithm	Description	Reference
Depth First Branch-and-Bound (DFBnB)	DFS	
Cyclic Best-First Search (CBFS)	Hybrid of DFS and best-first search	Kao et al. 2009
Anytime Column Progressive Search (ACPS)	Hybrid of DFS and beam search	Vadlamudi et al. 2012
Anytime Pack Progressive Search (APPS)	Hybrid of DFS and beam search	Vadlamudi et al. 2016
Discrepancy-Bounded DFS (DBDFS)	Discrepancy-based	Beck and Perron 2000
Complete Anytime Beam Search (CABS)	Iterative beam search	Zhang 1998

Implemented in https://github.com/domain-independent-dp/didp-rs

Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality

$k = 2$

$\{1, 2, 3\}, 0, 0$

f: 0

Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality

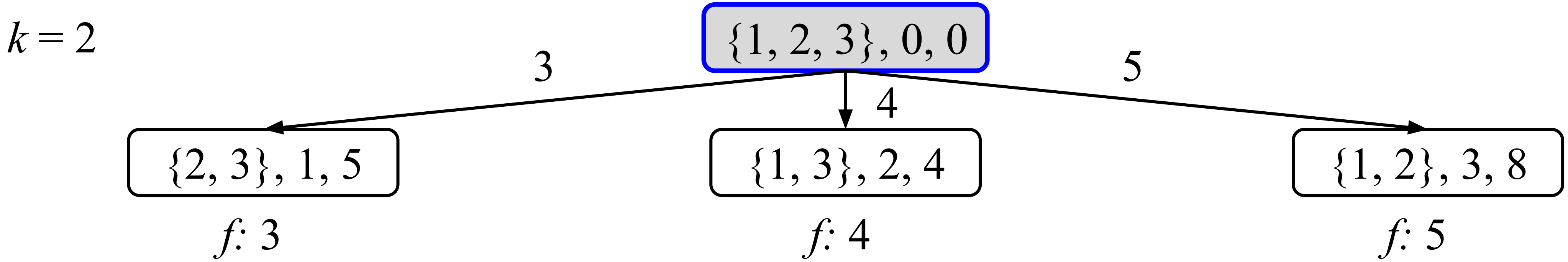
$k = 2$

$\{1, 2, 3\}, 0, 0$

f: 0

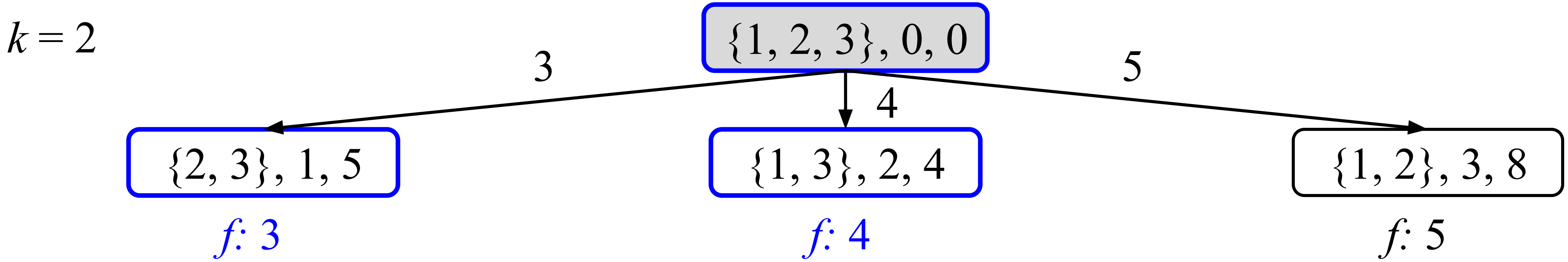
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



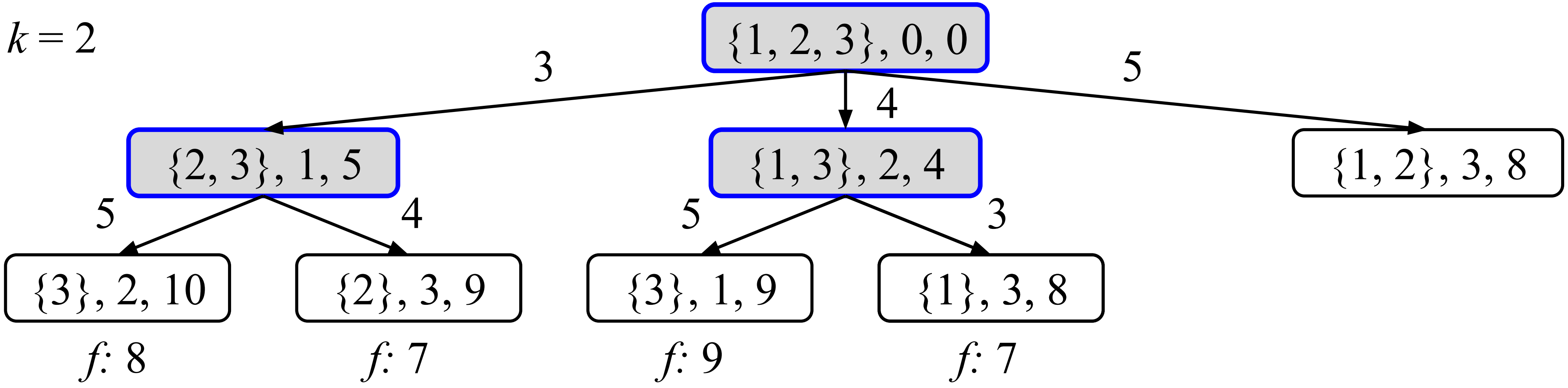
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



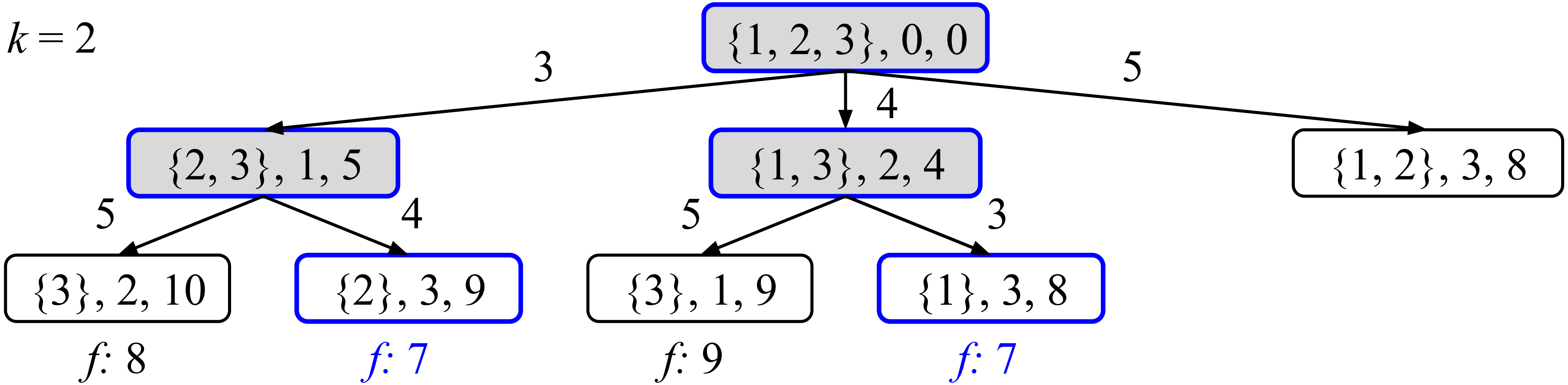
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



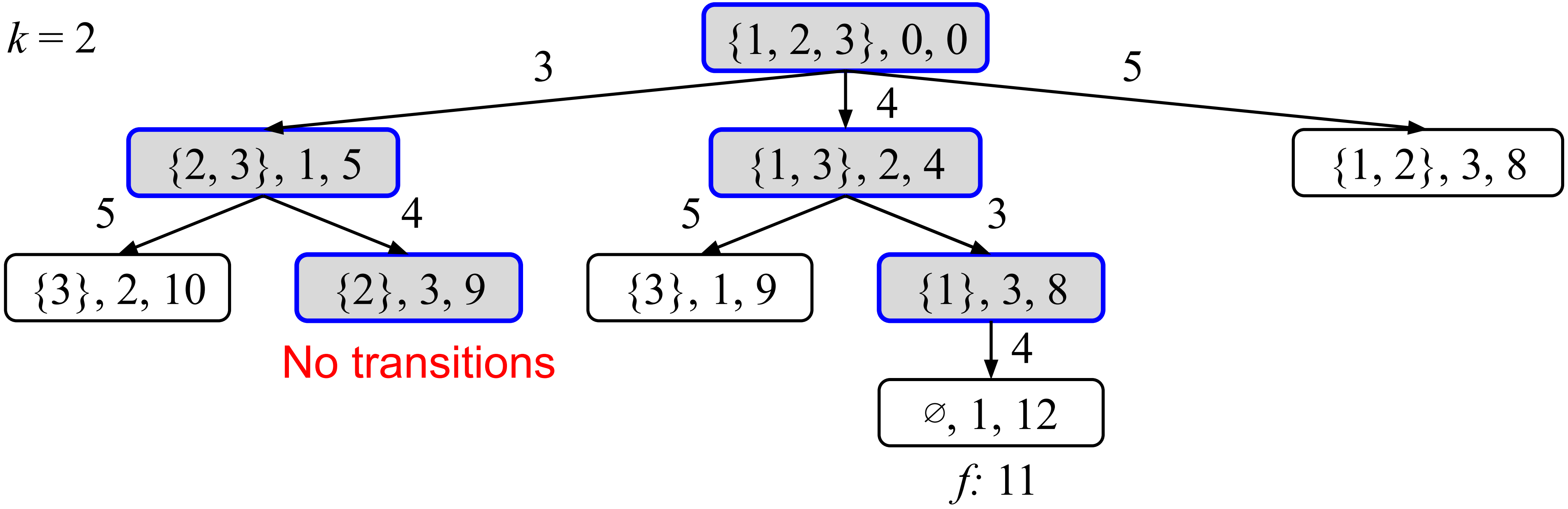
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



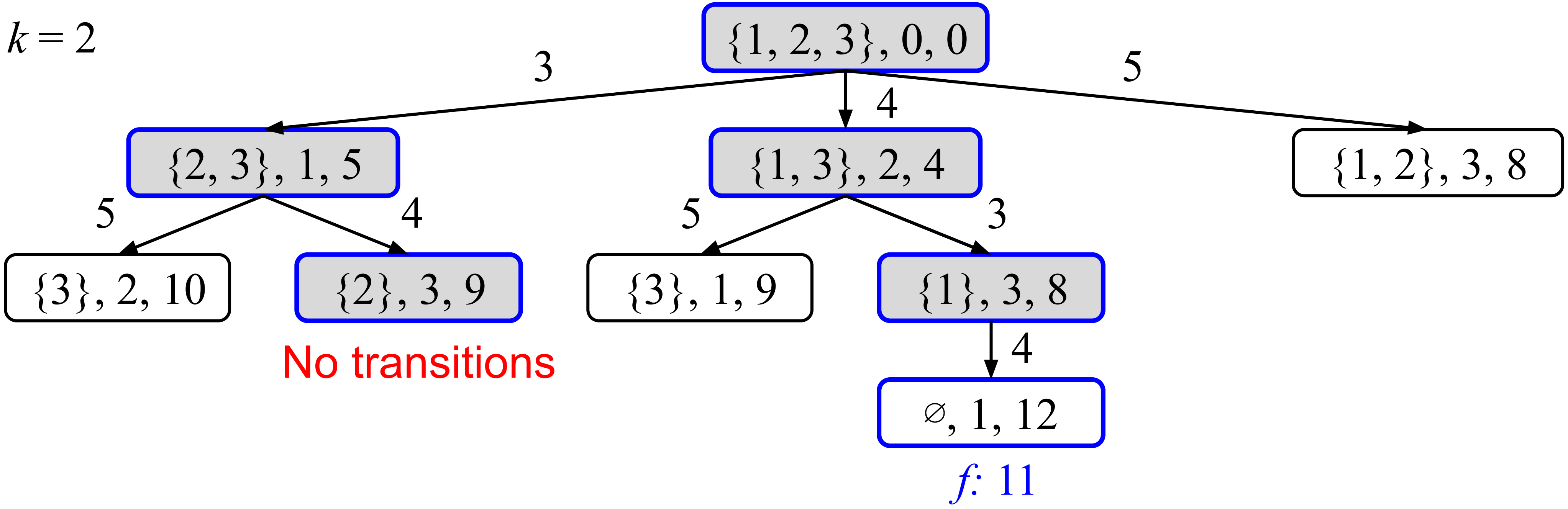
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



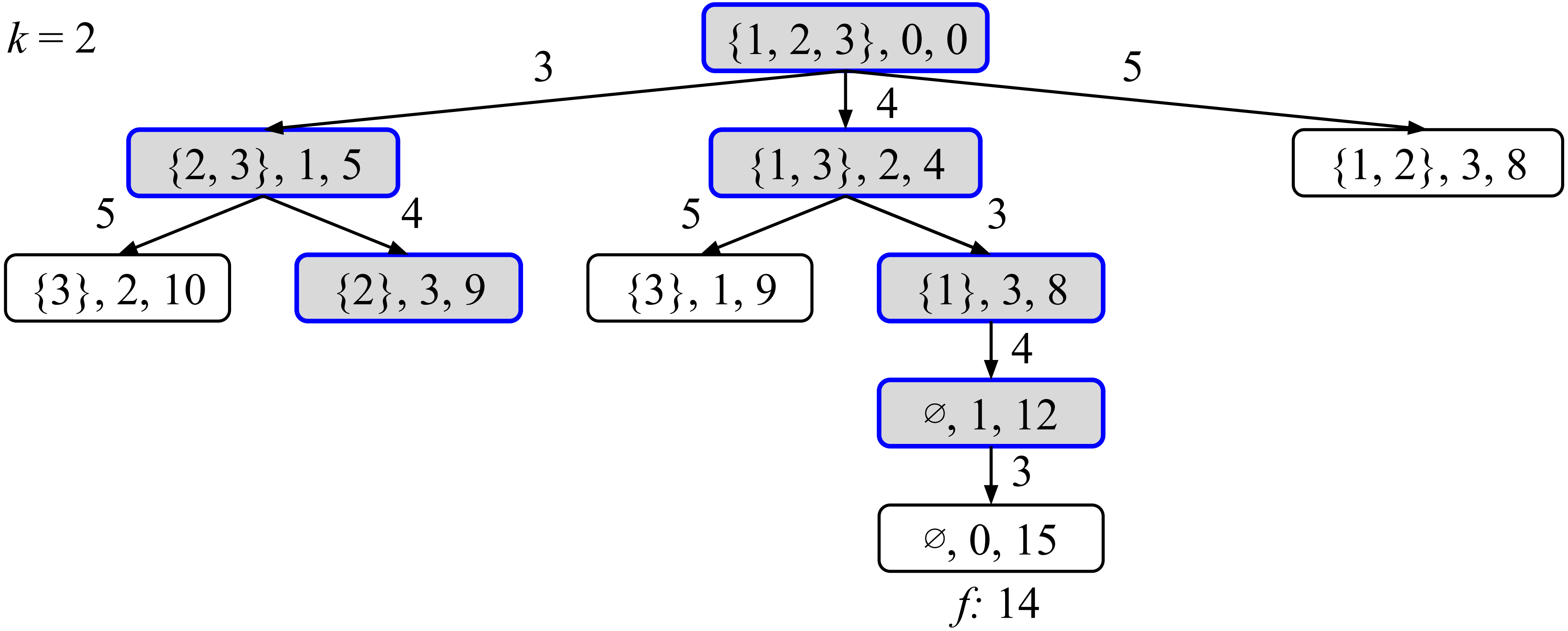
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



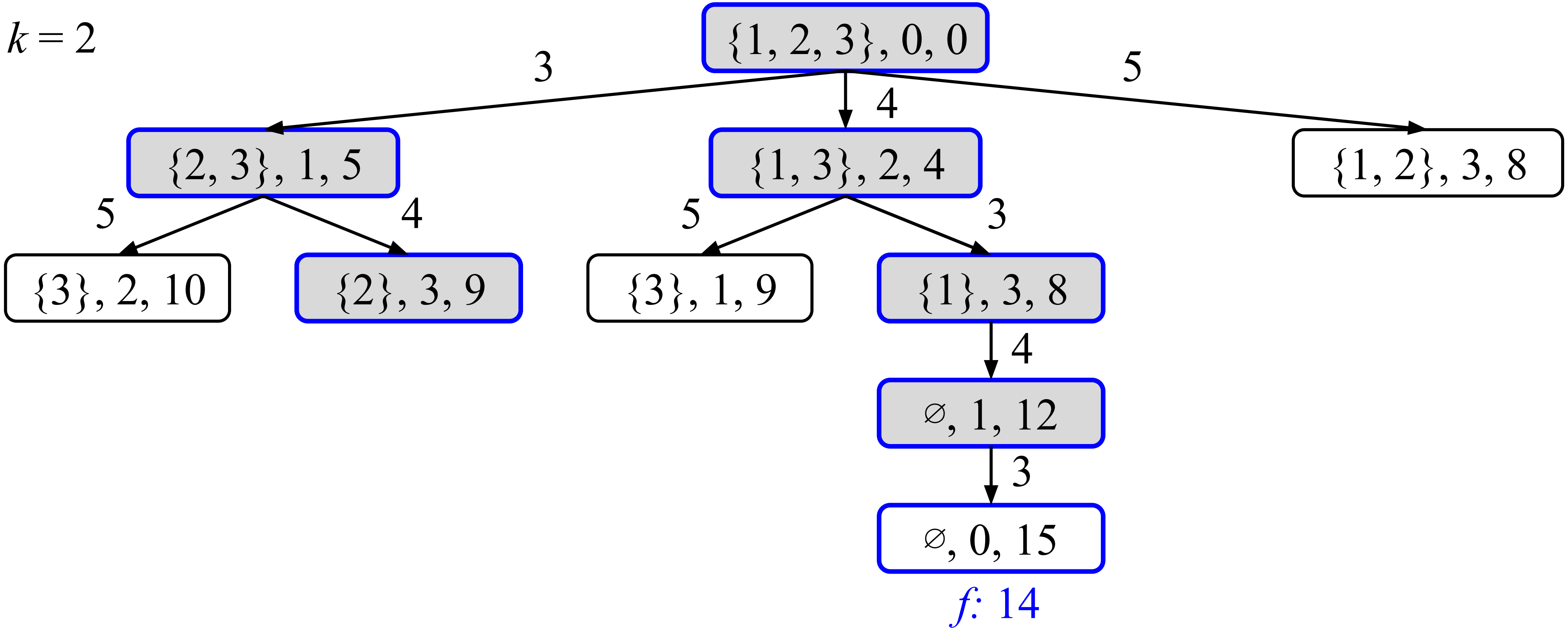
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



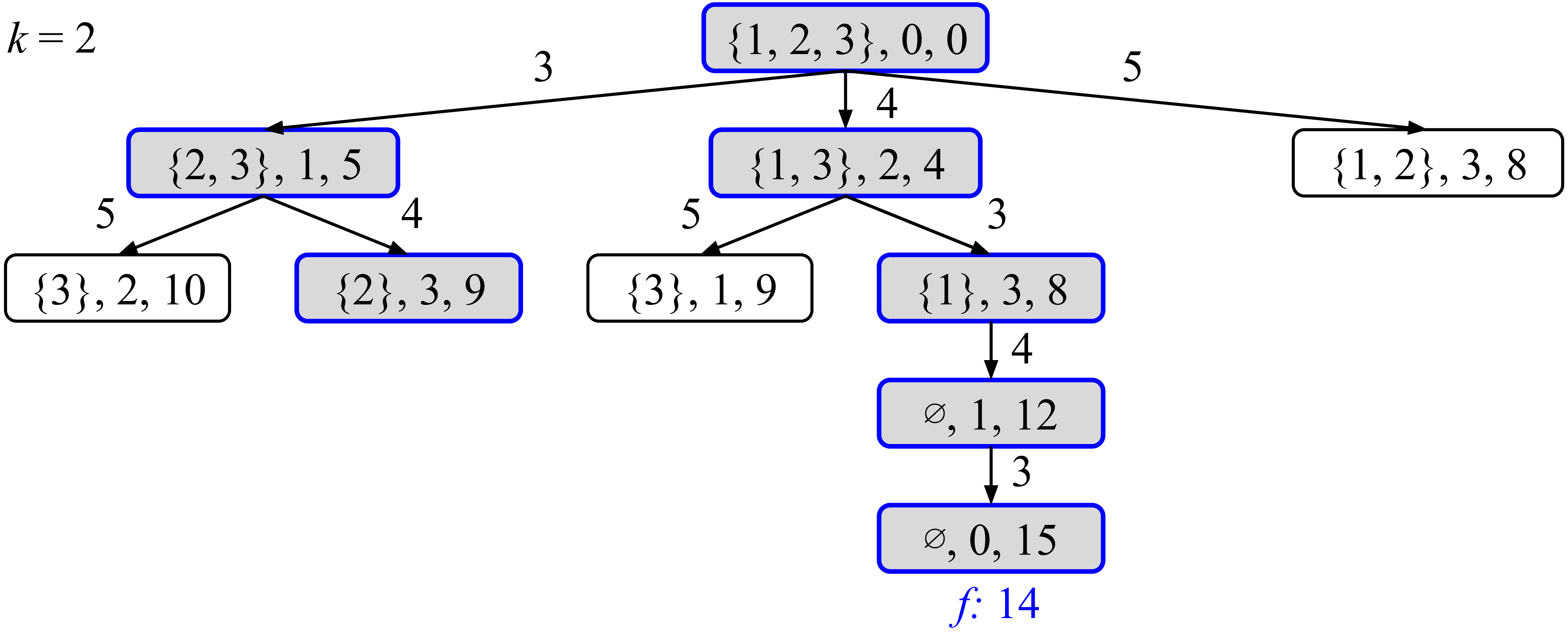
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



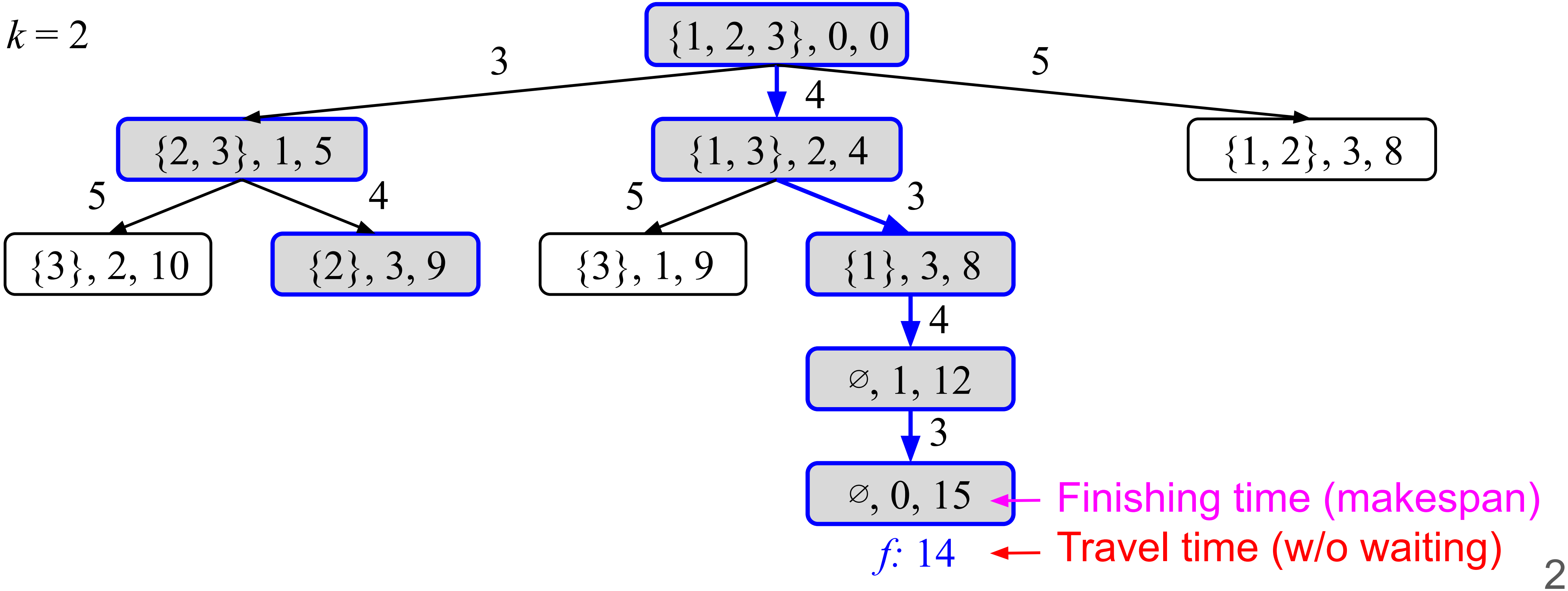
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



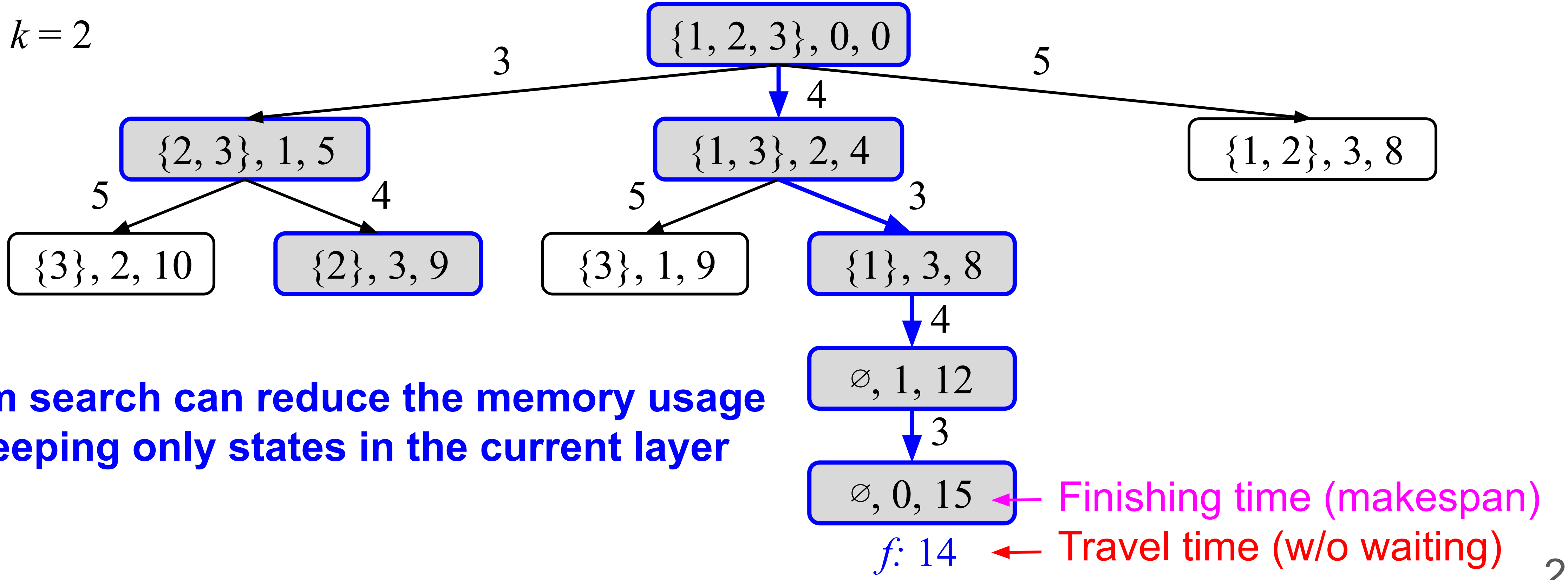
Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



Beam Search

- Keep k best states according to the f-values at each layer
- No guarantee of completeness nor optimality



Beam search can reduce the memory usage by keeping only states in the current layer

Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost

$k = 8$, incumbent: 14

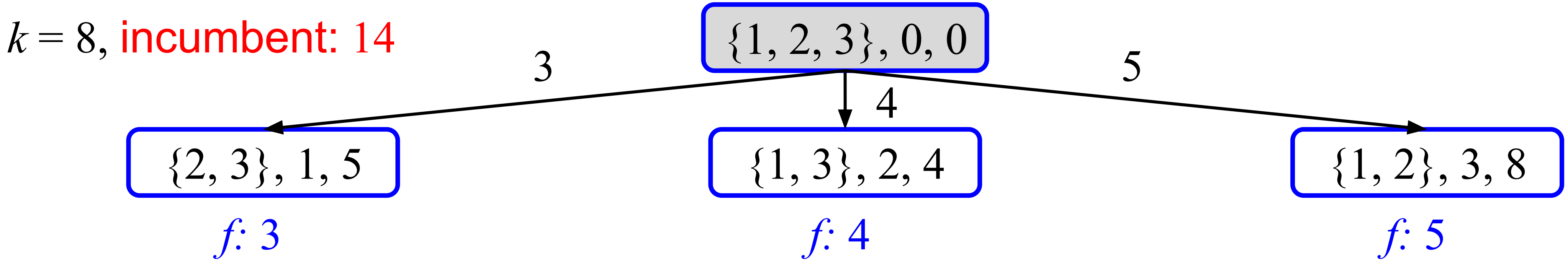
$\{1, 2, 3\}, 0, 0$

f: 0

Zhang 1998

Complete Anytime Beam Search (CABS)

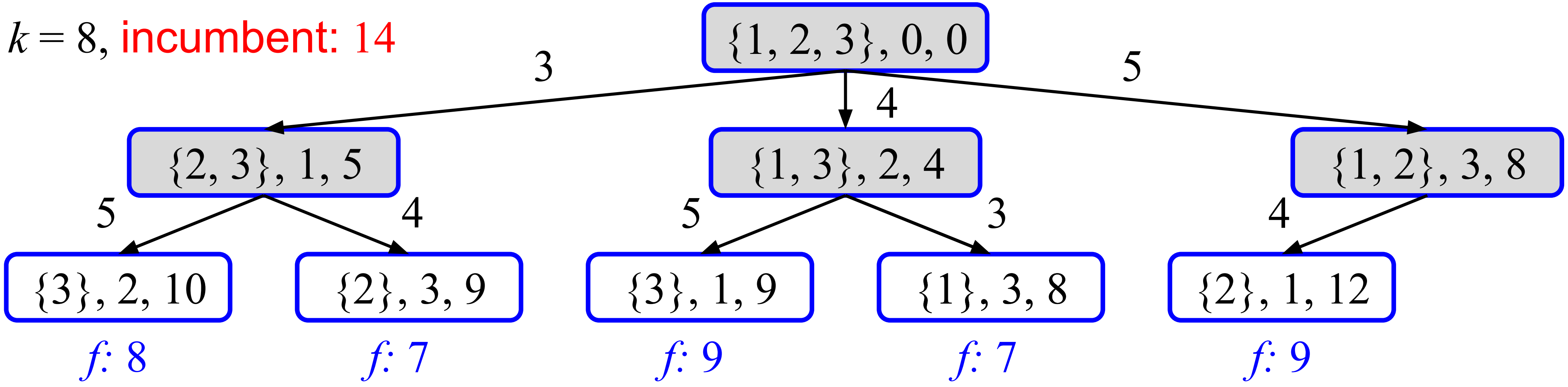
- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost



Zhang 1998

Complete Anytime Beam Search (CABS)

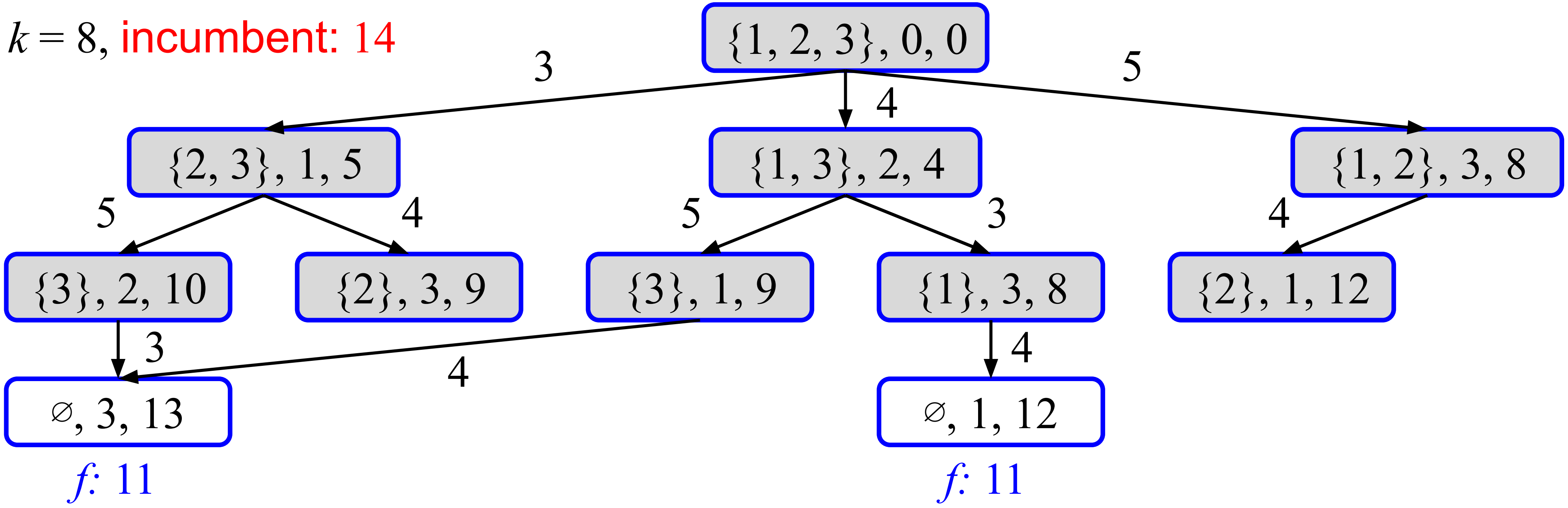
- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost



Zhang 1998

Complete Anytime Beam Search (CABS)

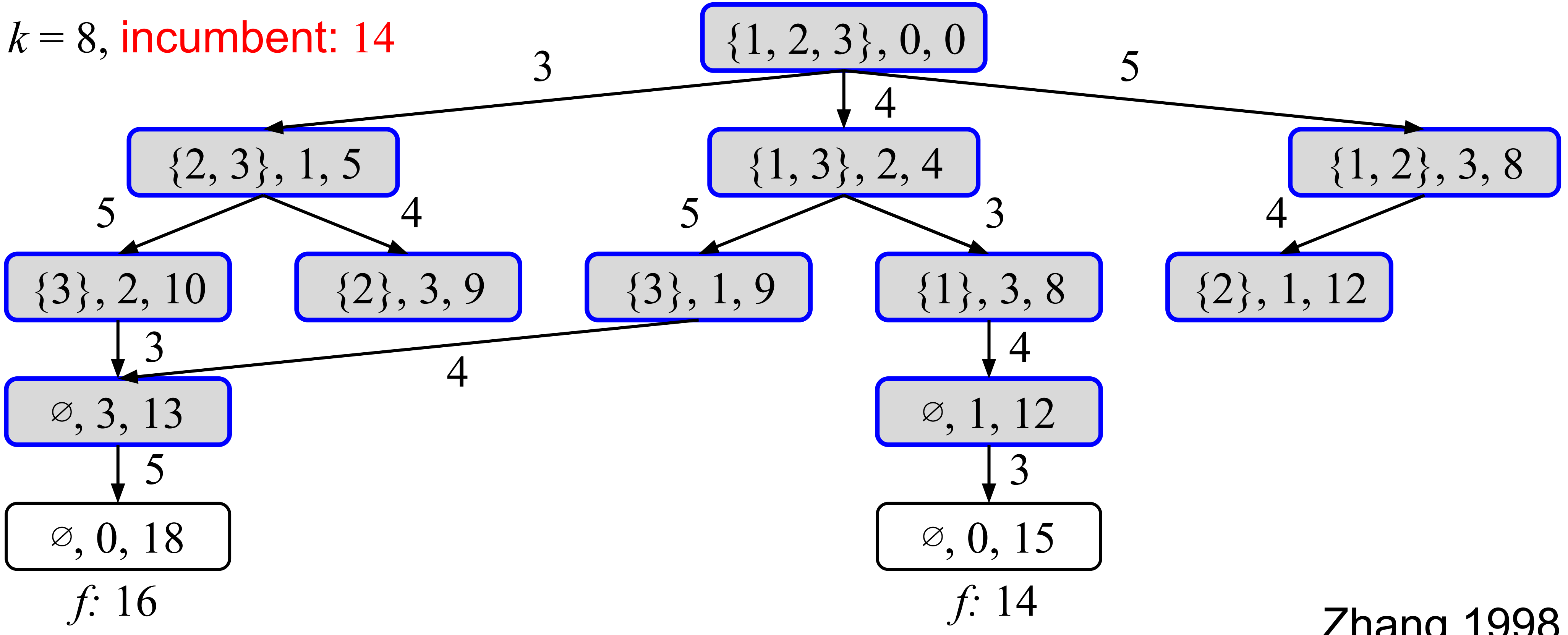
- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost



Zhang 1998

Complete Anytime Beam Search (CABS)

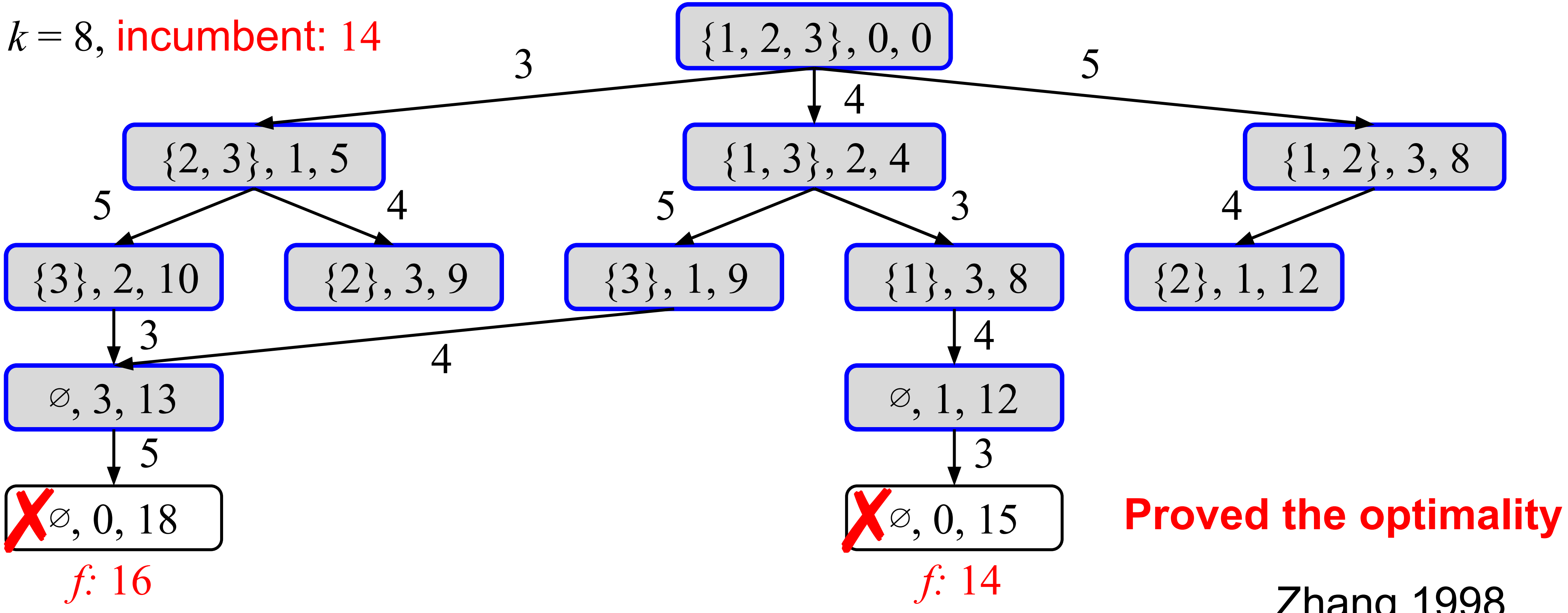
- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost



Zhang 1998

Complete Anytime Beam Search (CABS)

- Beam search with $k = 1, 2, 4, 8, 16, \ldots$ until states are exhausted
- Prune a state s if $f(s) \geq$ the incumbent solution cost

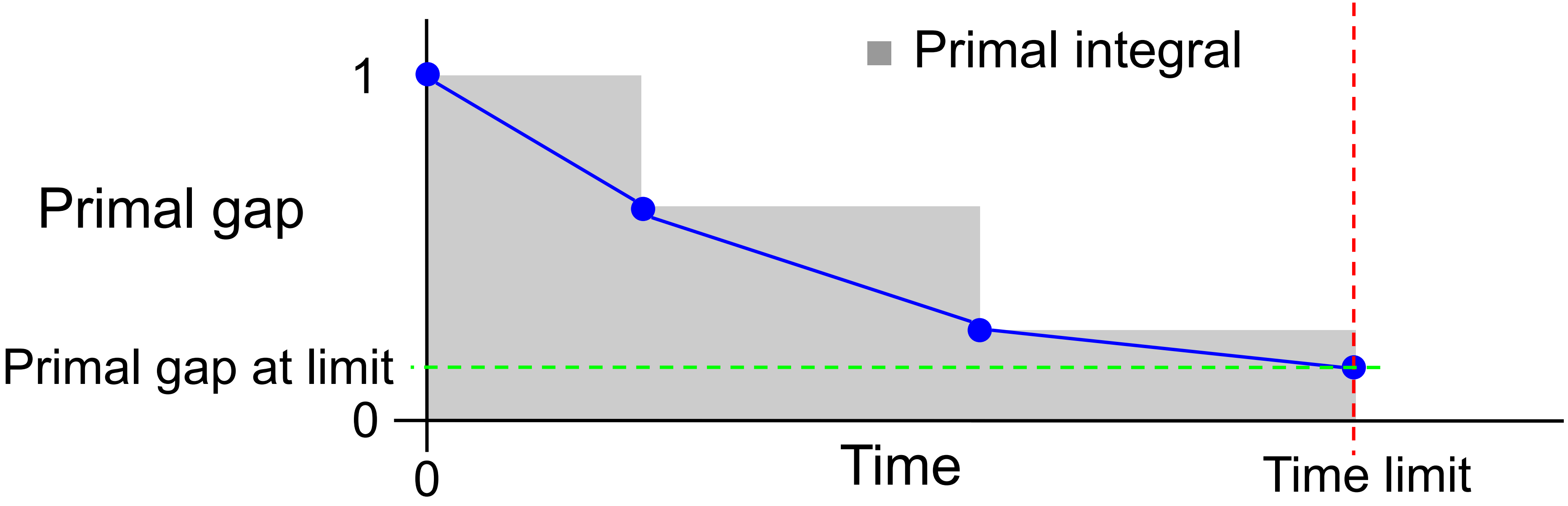


Zhang 1998

Experimental Evaluation

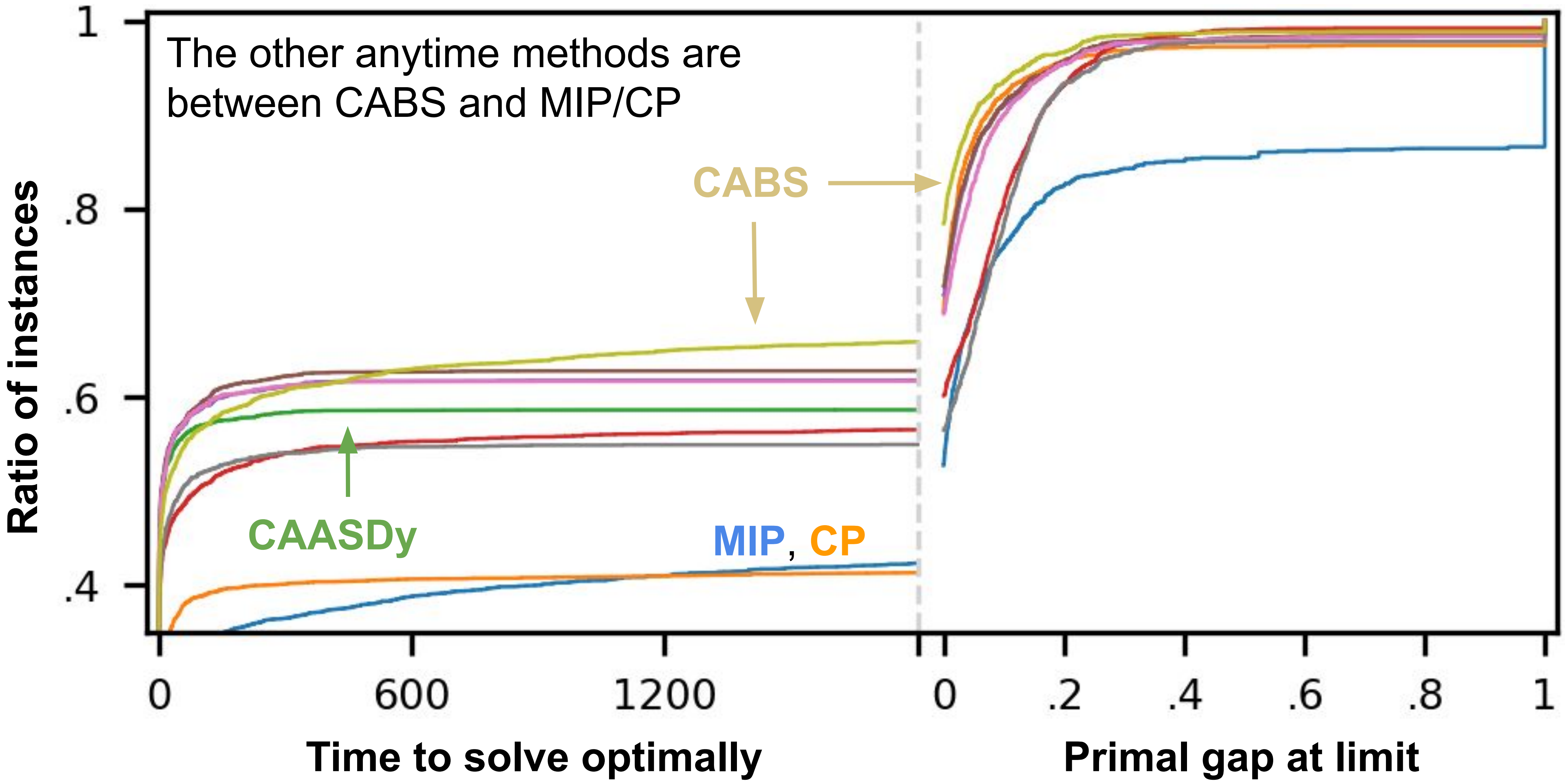
Primal Integral

Primal gap: $\dfrac{\text{solution cost} - \text{best known cost}}{\text{solution cost}}$ (1 if no solution found)

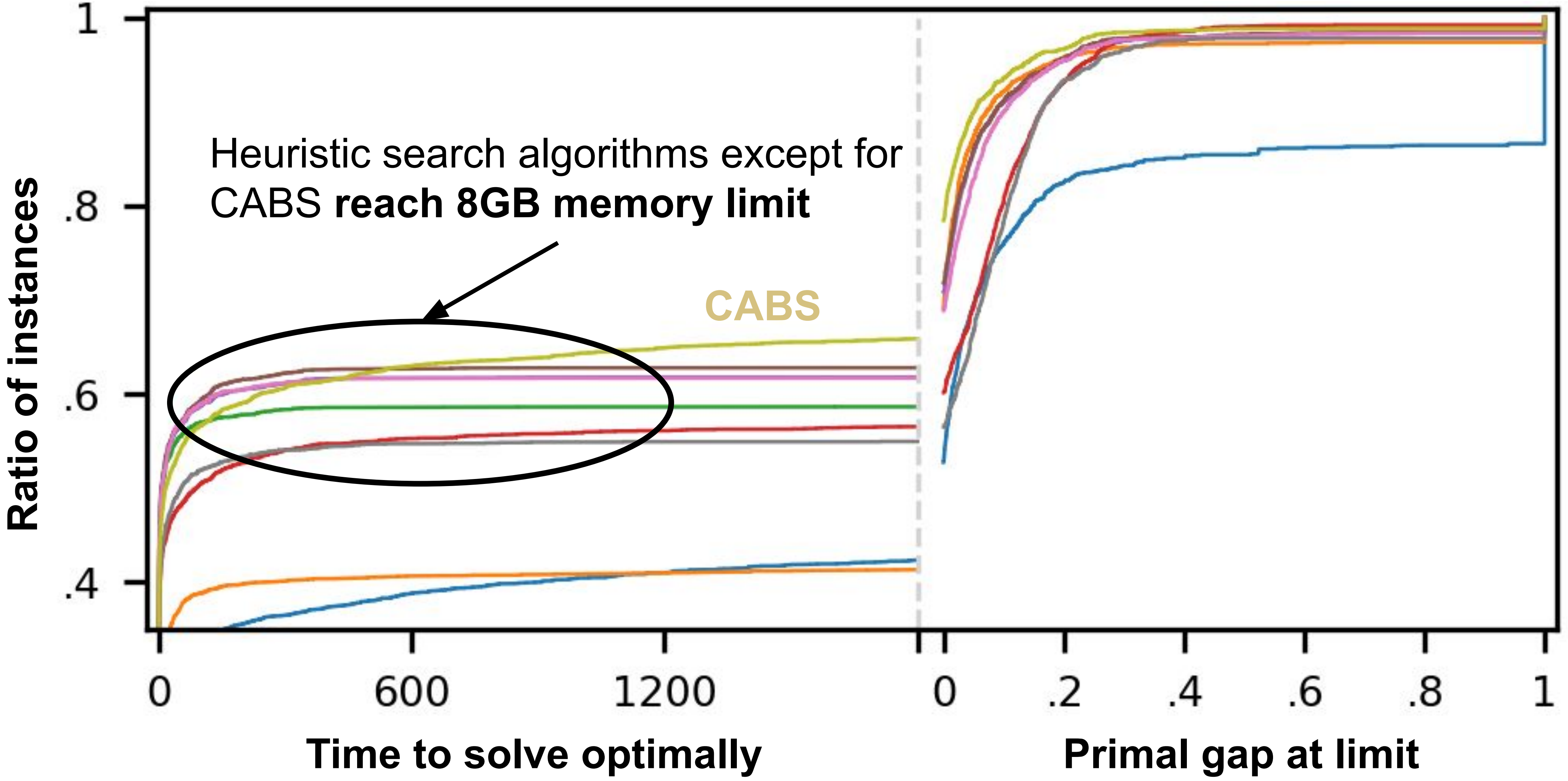


Berthold 2013

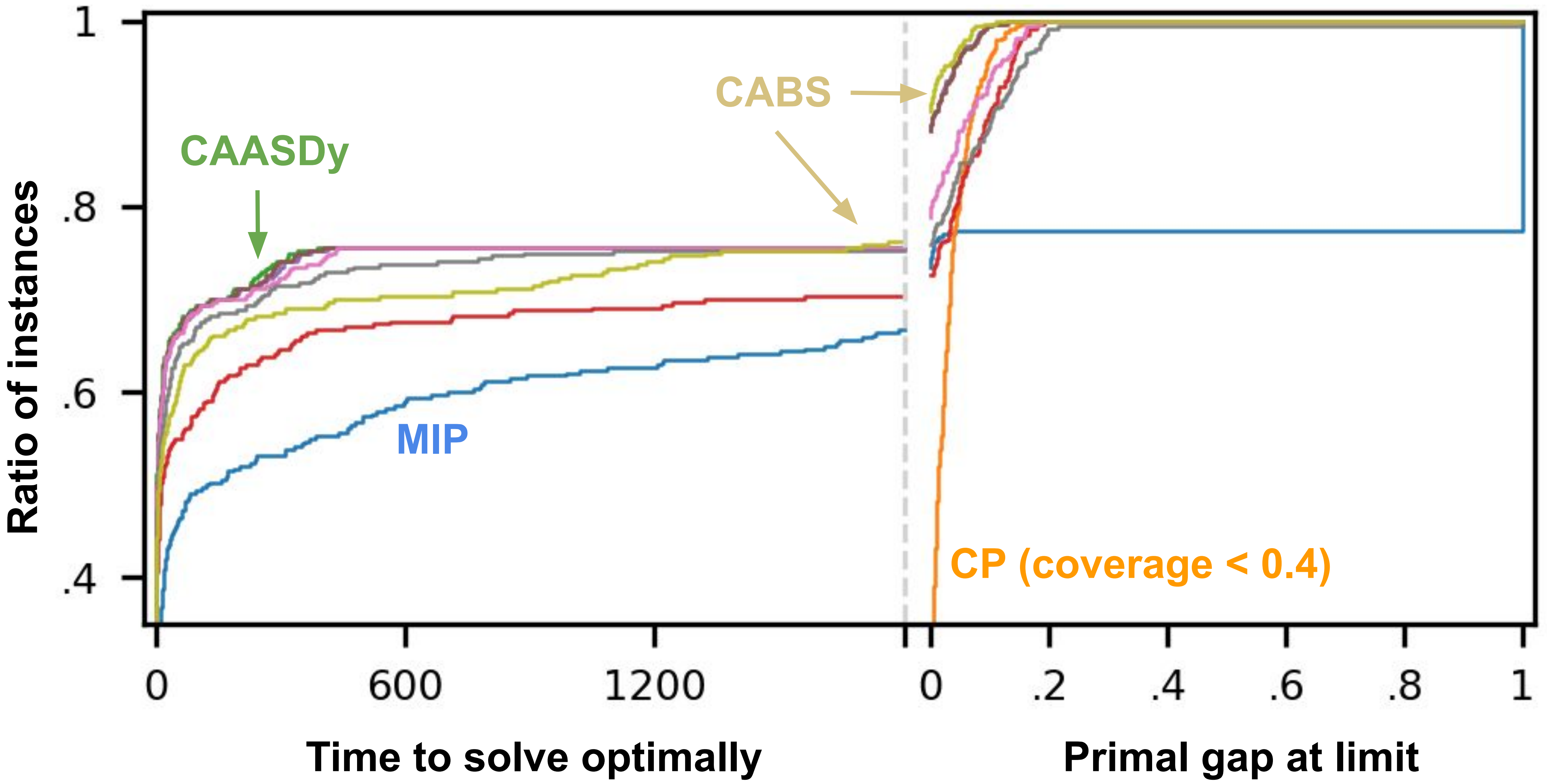
Coverage and Gap (Mean over All Problems)



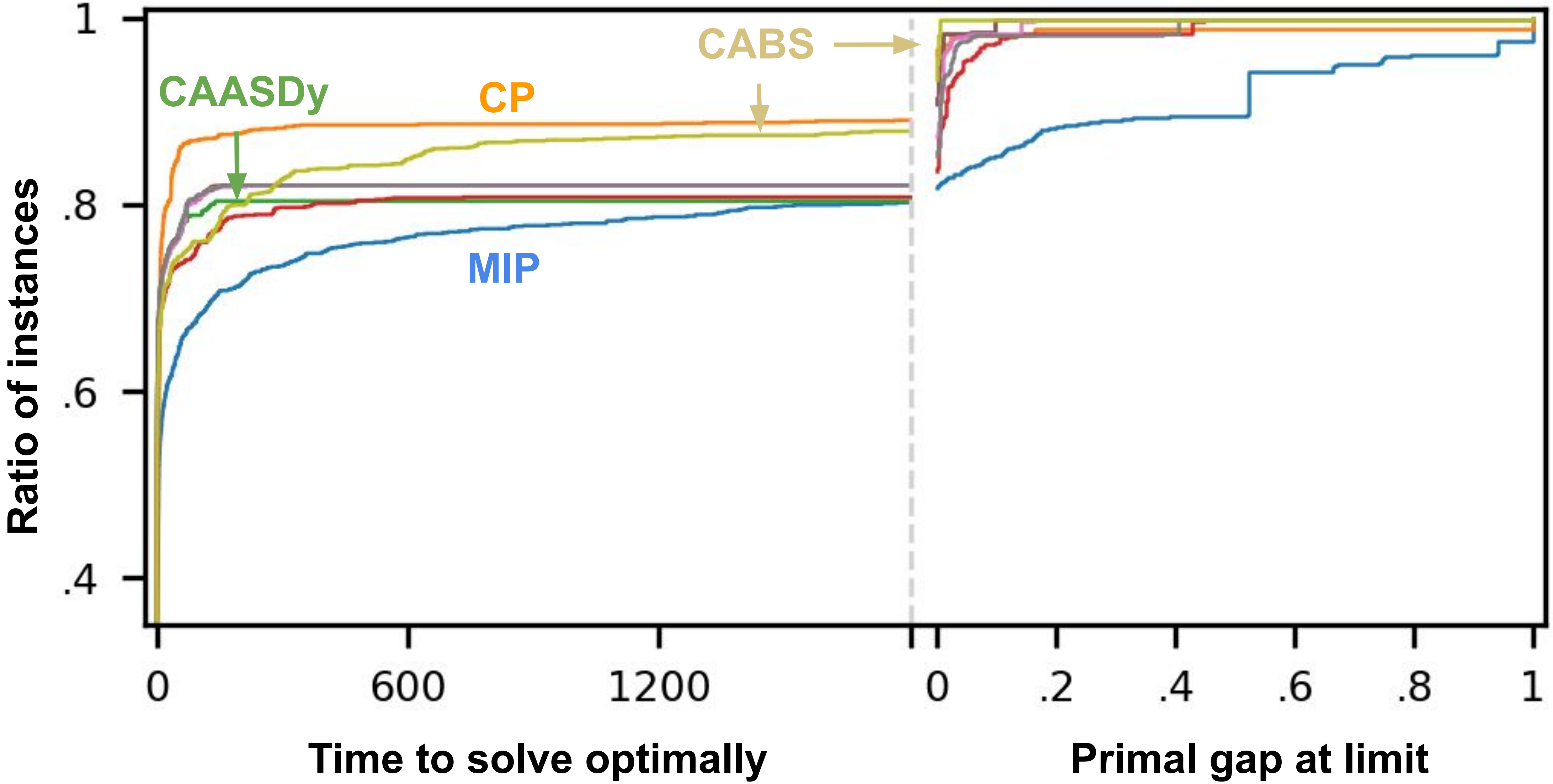
Coverage and Gap (Mean over All Problems)



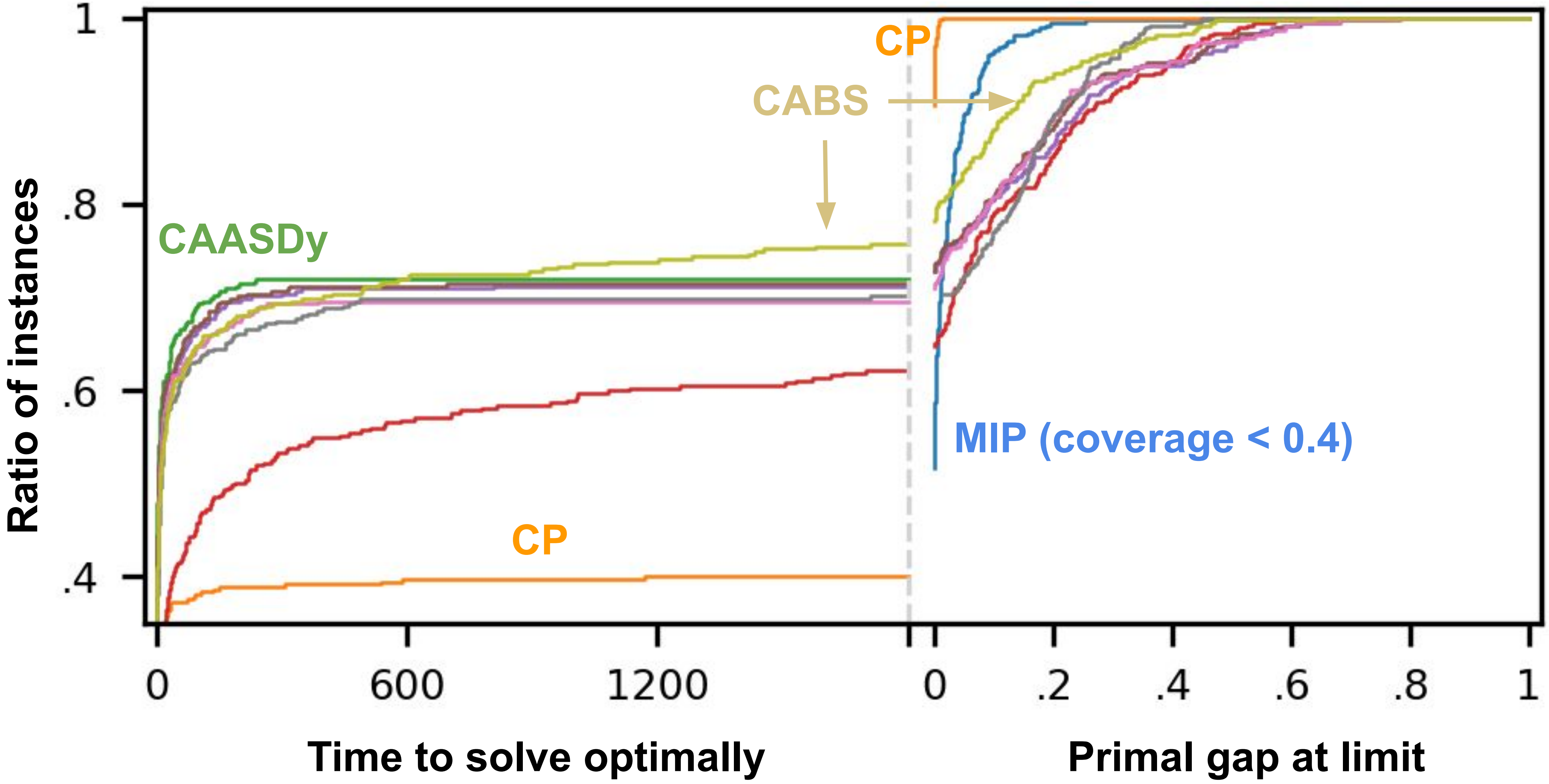
Coverage and Gap (TSPTW)



Coverage and Gap (m-PDTSP)



Coverage and Gap $(1||\sum w_i T_i)$

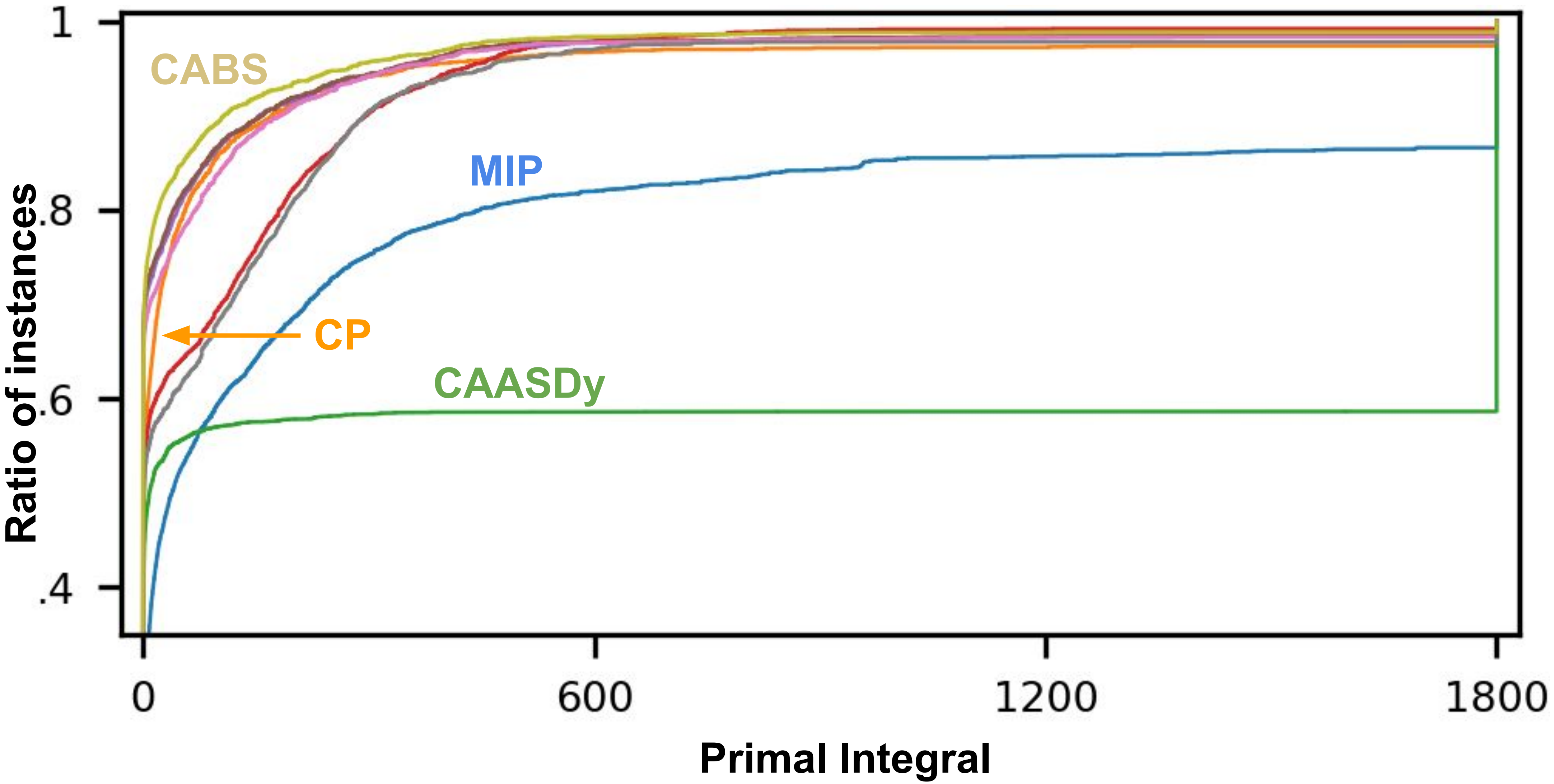


Coverage in Each Problem

	Description	MIP	CP	CAASDy	CABS
TSPTW (340)	TSP with time	227	47	257	**259**
CVRP (207)	vehicle routing	**26**	0	5	6
SALBP-1 (2100)	assembly line	1357	1584	1653	**1801**
Bin Packing (1615)	bin packing	1157	**1234**	922	1163
MOSP (570)	manufacturing	225	437	483	**527**
Graph-Clear (135)	building security	24	4	76	**103**
Talent Scheduling (1000)	scheduling actors	6	7	224	**253**
m-PDTSP (1117)	pick up & delivery	945	**1049**	947	1035
$1\|\|\sum w_i T_i$ (375)	job scheduling	109	150	270	**285**

# of optimality solved instances with 8GB and 30-min

Primal Integral (Mean over All Problems)



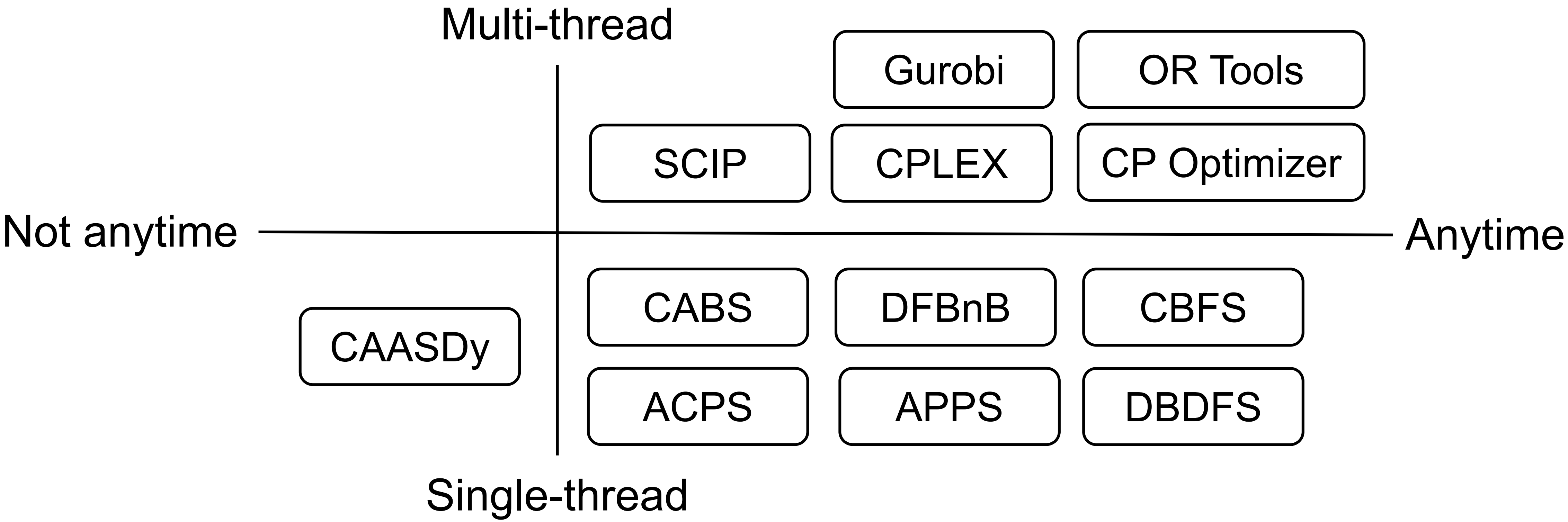
Mean Primal Gap and Primal Integral

	Description	MIP	CP	CABS
TSPTW (340)	TSP with time	0.227/484.05	0.026/48.97	**0.003/8.97**
CVRP (207)	vehicle routing	0.585/1157.43	0.317/601.15	**0.185/351.21**
SALBP-1 (2100)	assembly line	0.345/634.64	0.005/28.48	**0.000/1.92**
Bin Packing (1615)	bin packing	0.039/86.19	**0.002**/8.04	**0.002/5.26**
MOSP (570)	manufacturing	0.039/100.41	0.004/13.01	**0.000/0.36**
Graph-Clear (135)	building security	0.110/311.83	0.015/44.27	**0.000/0.49**
Talent Scheduling (1000)	scheduling actors	0.051/142.69	**0.002/18.14**	0.011/26.36
m-PDTSP (1178)	pick up & delivery	0.078/180.00	0.013/26.04	**0.002/5.33**
$1\|\|\sum w_i T_i$ (375)	job scheduling	0.018/74.56	**0.000/2.26**	0.034/73.60

Mean primal gap at limit / primal integral

Conclusion

- Anytime DIDP solvers are promising!
- Trade-off between time and memory
- Future work: parallelization?



Please Use DIDP!

We need your ideas to advance DIDP!

- Visit our website: https://didp.ai
- Start DIDP with Python: `pip install didppy`
  Tutorials and API Reference: https://didppy.rtfd.io
- Start DIDP with YAML: `cargo install didp-yaml`
- Clone the repository:
  `git clone https://github.com/domain-independent-dp/didp-rs`
  Everything in Rust 

Why Not Anytime Weighted A*?

- A user may provide 0 dual bound (heuristic)
- Finding a satisficing solution is usually much easier in combinatorial optimization than in AI planning