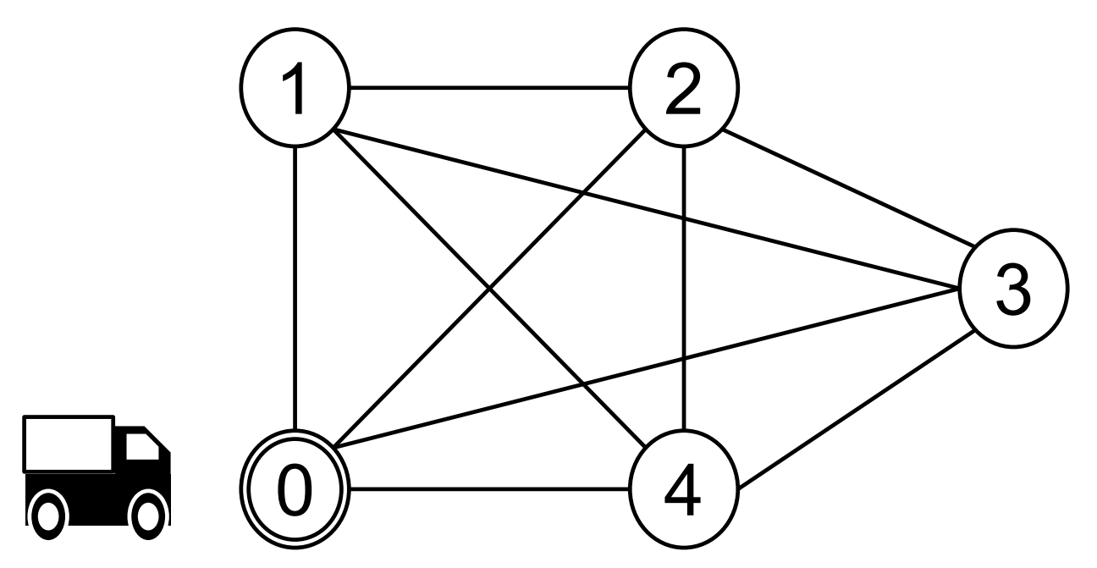
Large Neighborhood Beam Search for Domain-Independent Dynamic Programming

Ryo Kuroiwa and J. Christopher Beck
Department of Mechanical and Industrial Engineering, University of Toronto

Domain-Independent Dynamic Programming (DIDP)

CP-like **model & solve paradigm** having a Python interface (tutorials, examples, and API reference https://didppy.rtfd.io)

Combinatorial Optimization



- TSP-like problem to visit all customers from customer 0 (no return)
- Travel cost from i to j: c_{ij}
- Minimize the total travel cost

Model

Dynamic Programming Model

compute $V(\{1,2,3,4\},0)$ **Original problem**

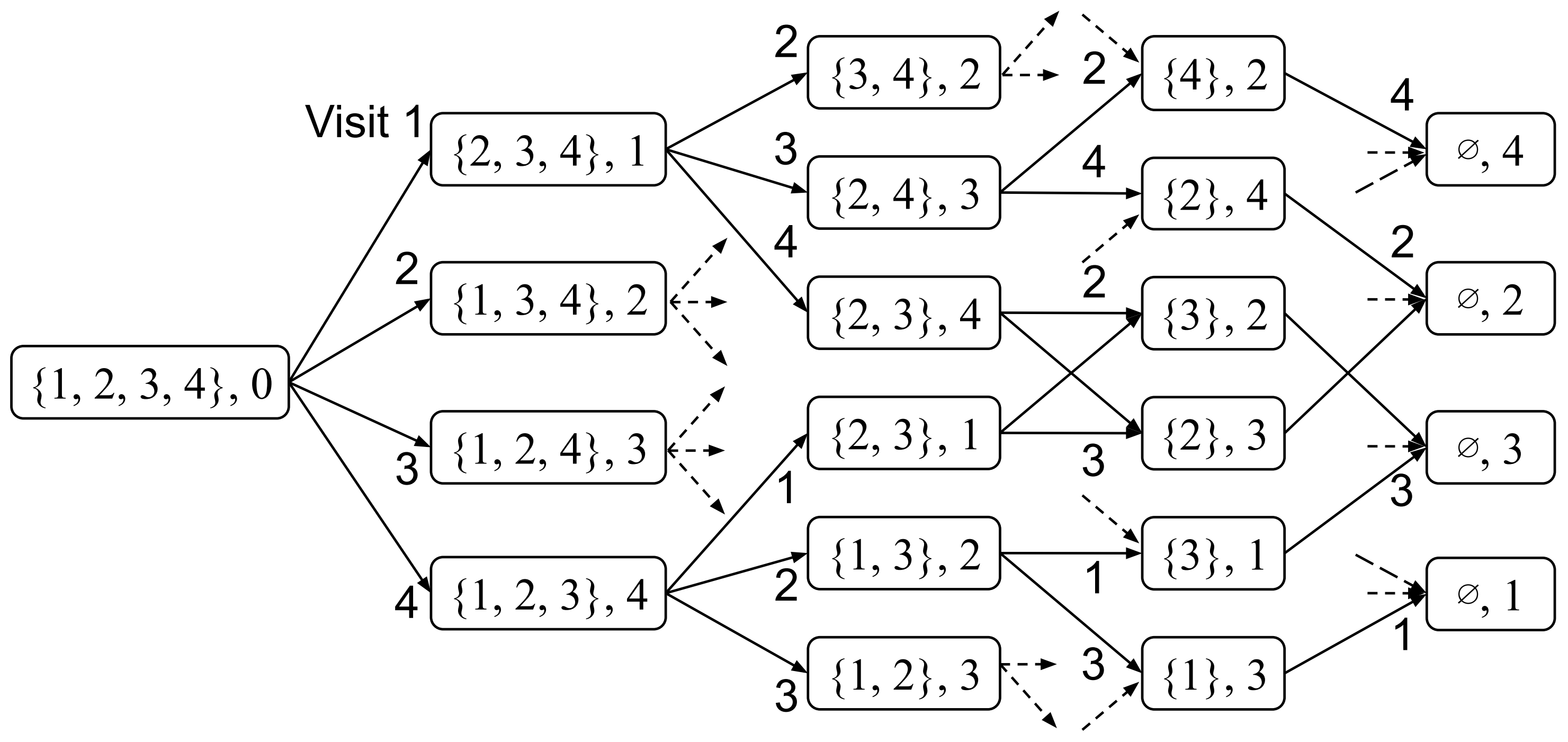
$$V(U,i) = \begin{cases} \min\limits_{j \in U} c_{ij} + V(U \setminus \{j\}, j) & \text{if } U \neq \emptyset \\ 0 & \text{if } U = \emptyset \end{cases}$$

$V(U,i) \geq h(U,i)$ **LB function**

- **Recursive decomposition** to subproblems
- Defined by the **value function** V, which maps a state (subproblem) to the cost
- Decompose by visiting one customer j and terminate when all customers are visited

Solve

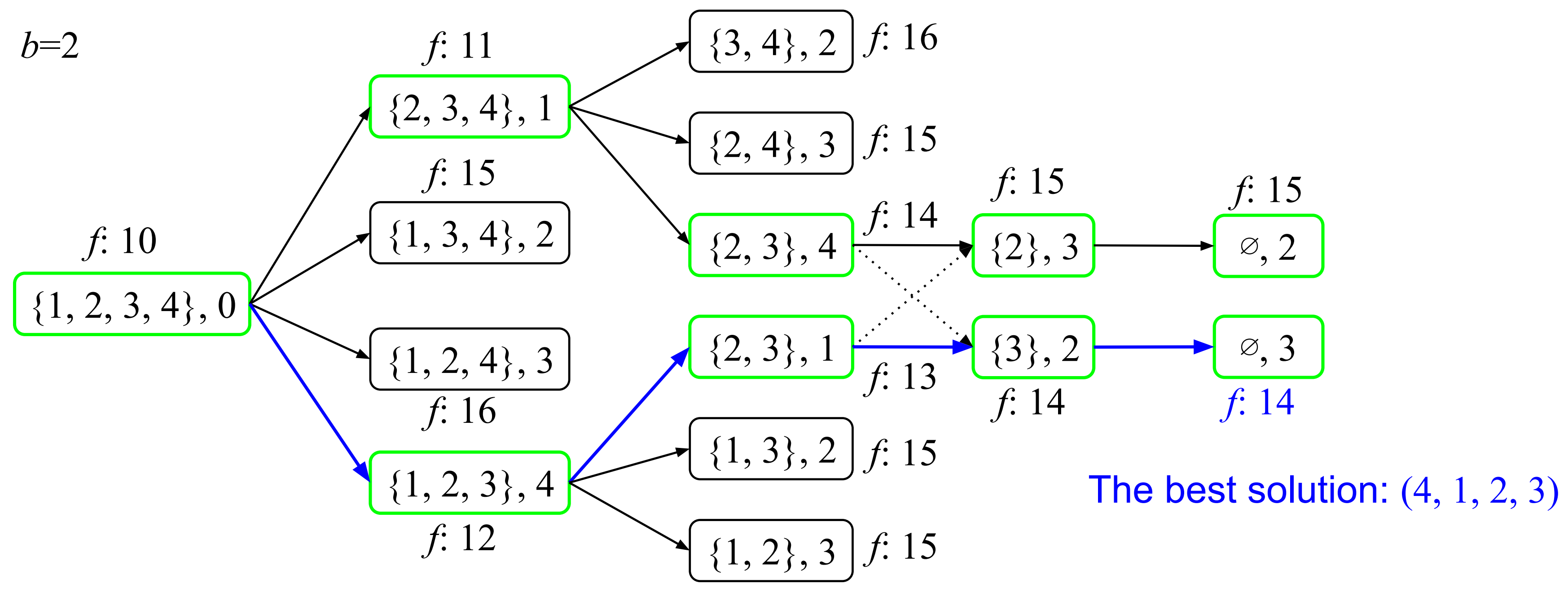
General-Purpose State Space Search Solver



Key difference from tree search: a solution is a path and multiple paths can lead to the same state (the space is a graph)

SOTA Solver: CABS [Kuroiwa and Beck 2023]

- Implementation of Complete Anytime Beam Search [Zhang 1998] for DIDP
- Perform beam search with beam width b=1, 2, 4, 8, …
- Beam search keeps only the b states minimizing the f-values in each layer

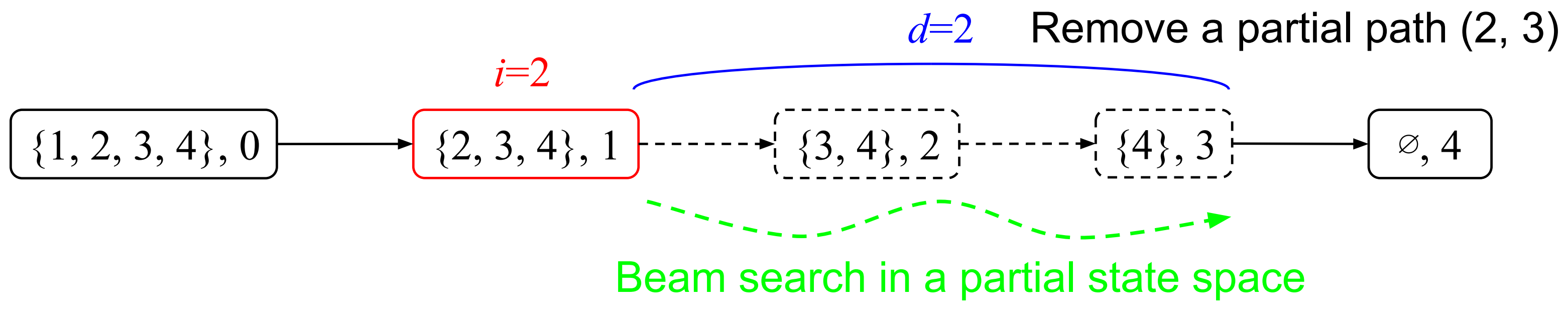


The best solution: (4, 1, 2, 3)

- $g(S)$: the actual path cost to state S from the original problem
- $h(S)$: the estimated path cost given by the LB function defined in the DP model
- $f(S) = g(S) + h(S)$: the priority in search

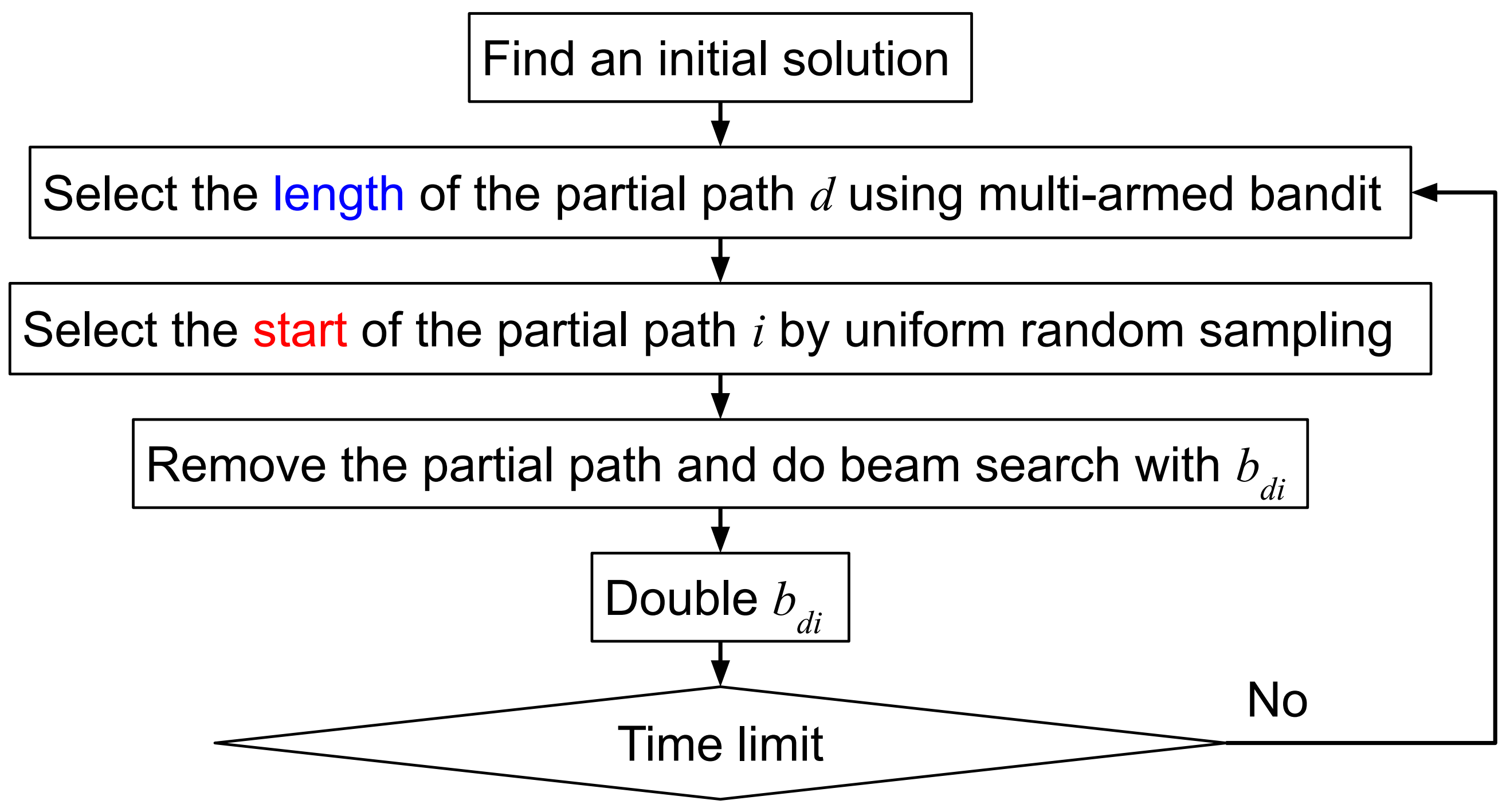
Large Neighborhood Beam Search

Remove a partial path from a solution and search for a better one



Beam search in a partial state space

Overview of Large Neighborhood Beam Search (LNBS)



Find an initial solution

Select the length of the partial path d using multi-armed bandit

Select the start of the partial path i by uniform random sampling

Remove the partial path and do beam search with b_{di}

Double b_{di}

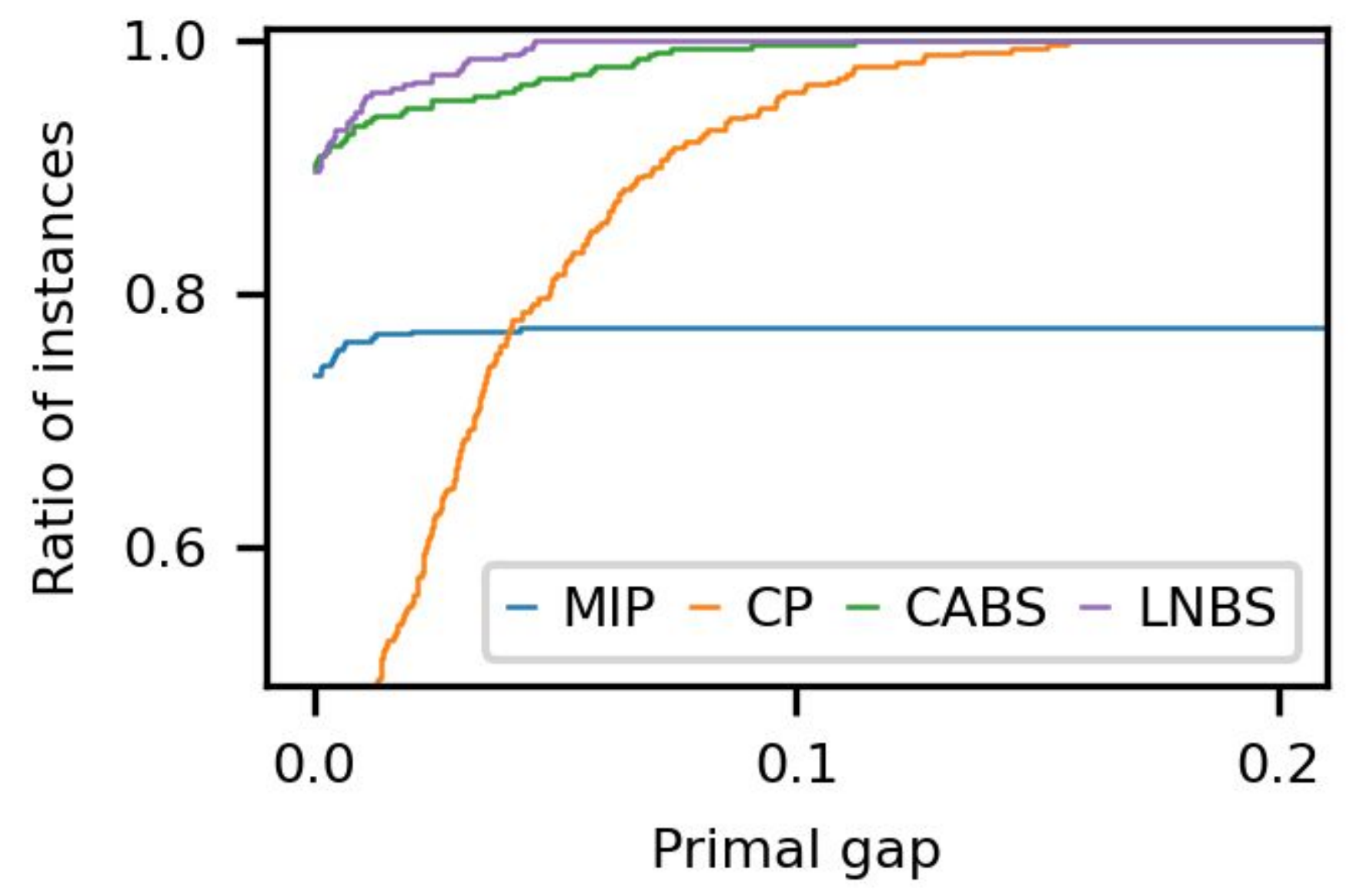
Time limit

No

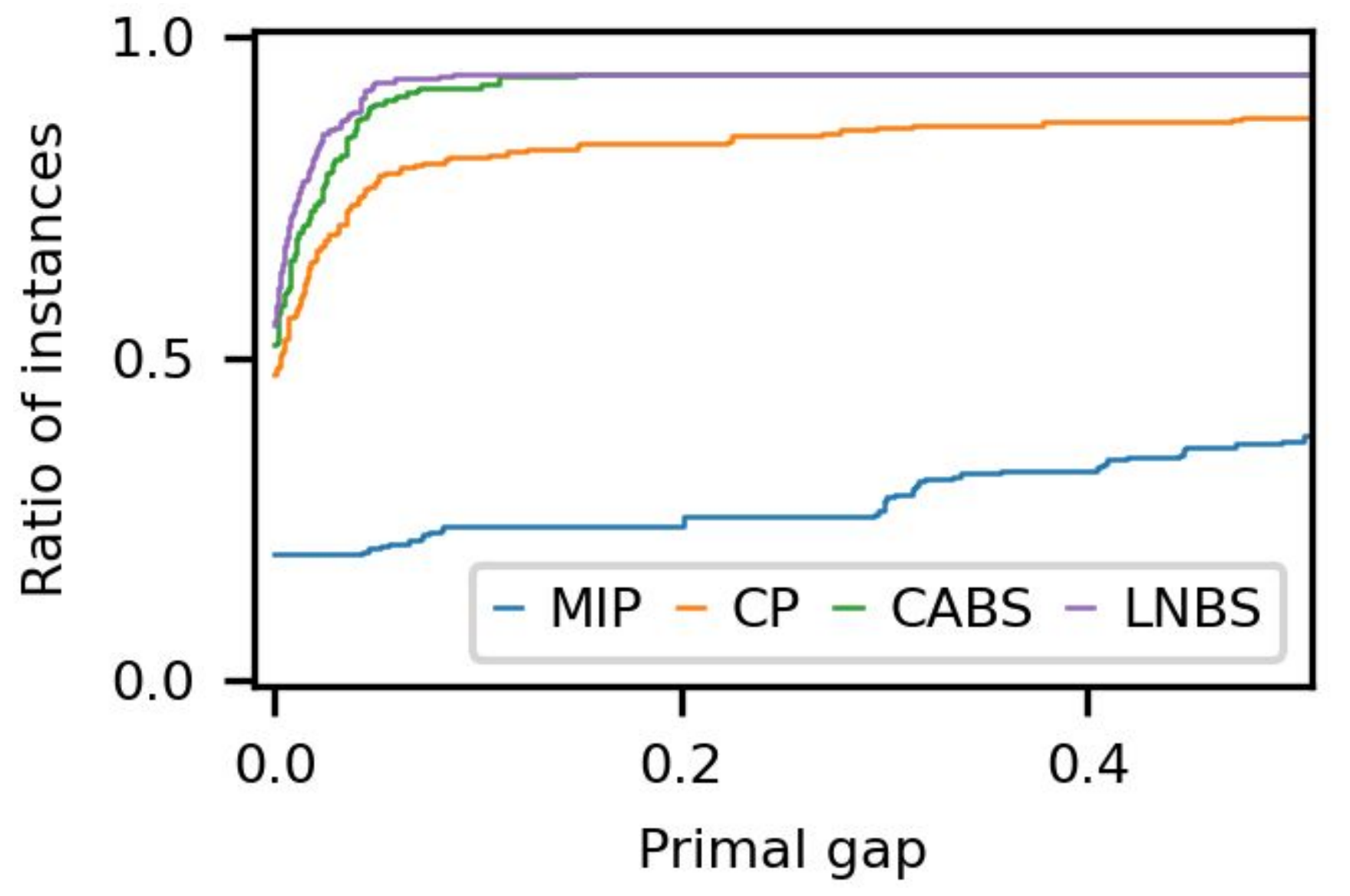
Experimental Evaluation

- x: gap to the best known solution cost scaled from 0 to 1 (with 30 min)
- y: ratio of instances with the primal gap ≤ x
- Upper left is better

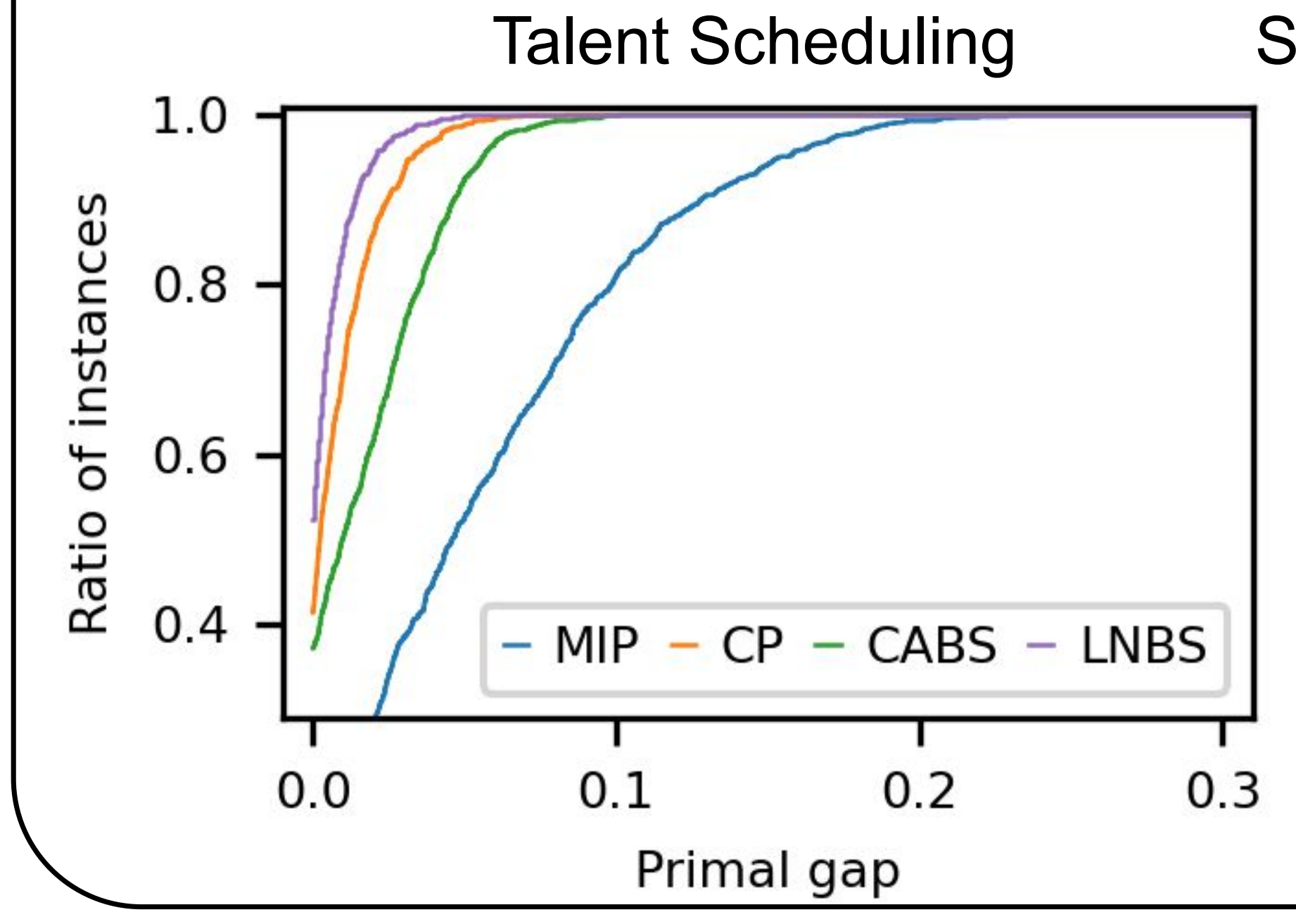
TSP with Time Windows (TSPTW)



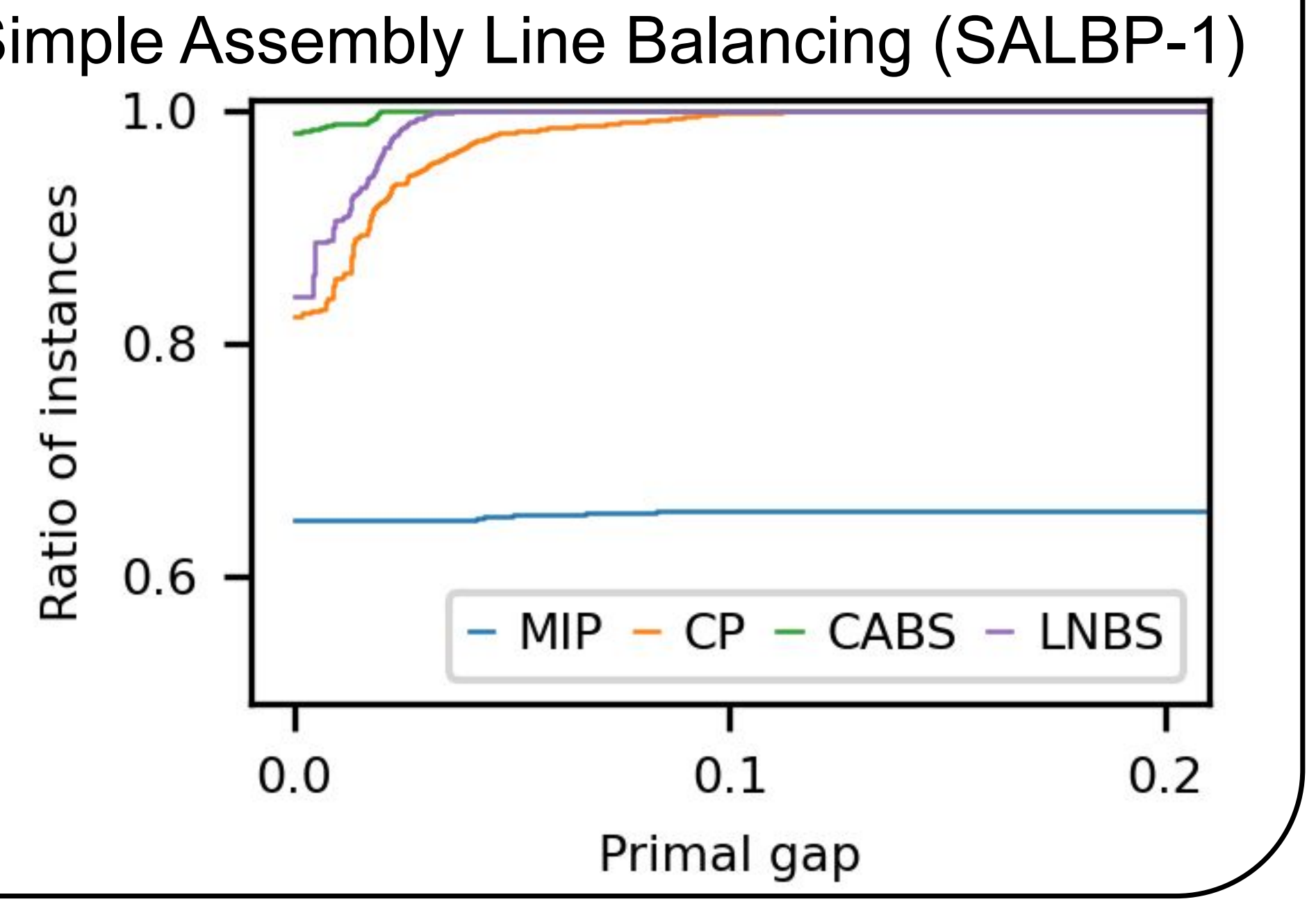
Pick-and-Delivery TSP (m-PDTSP)



Talent Scheduling



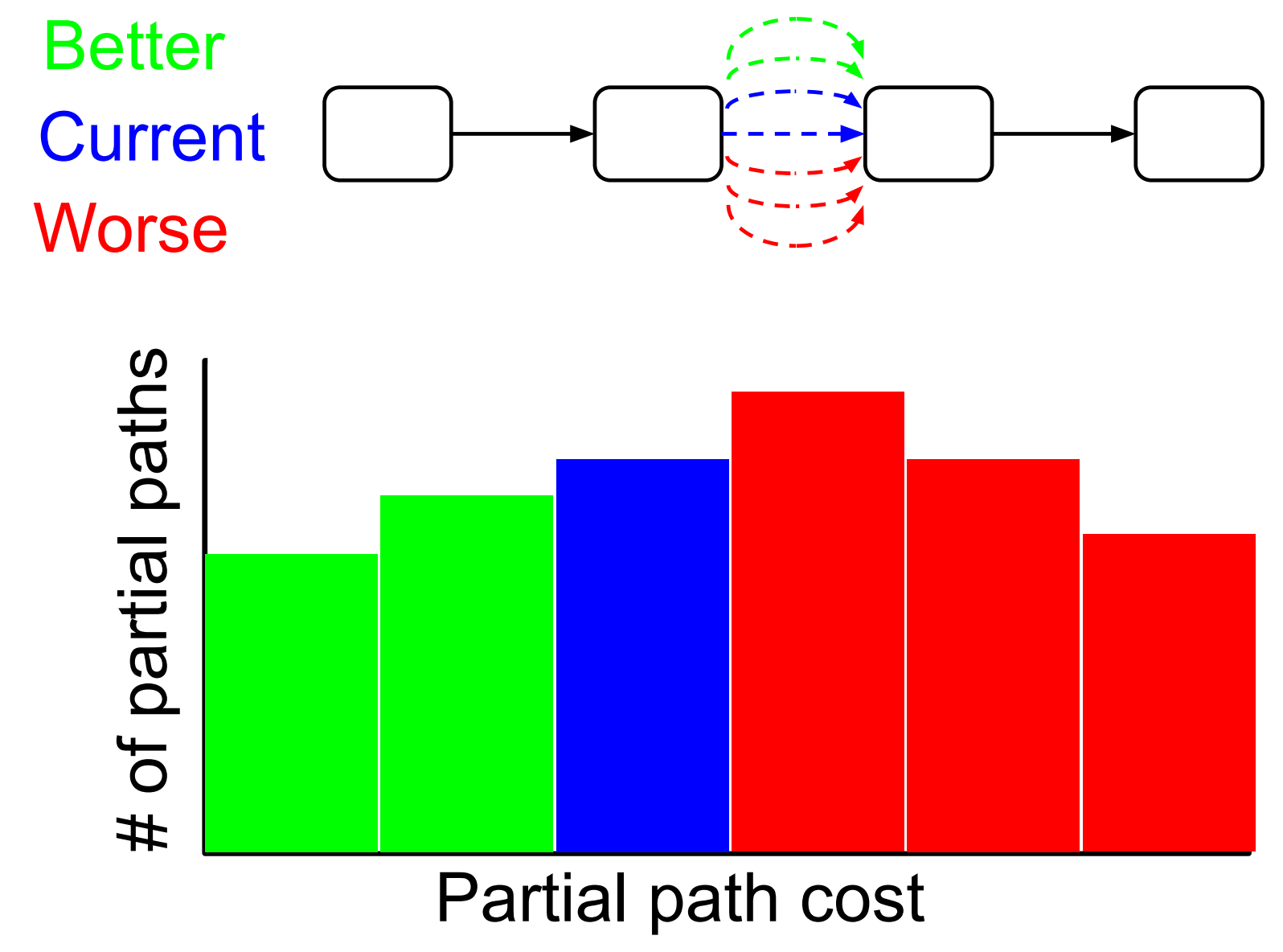
Simple Assembly Line Balancing (SALBP-1)



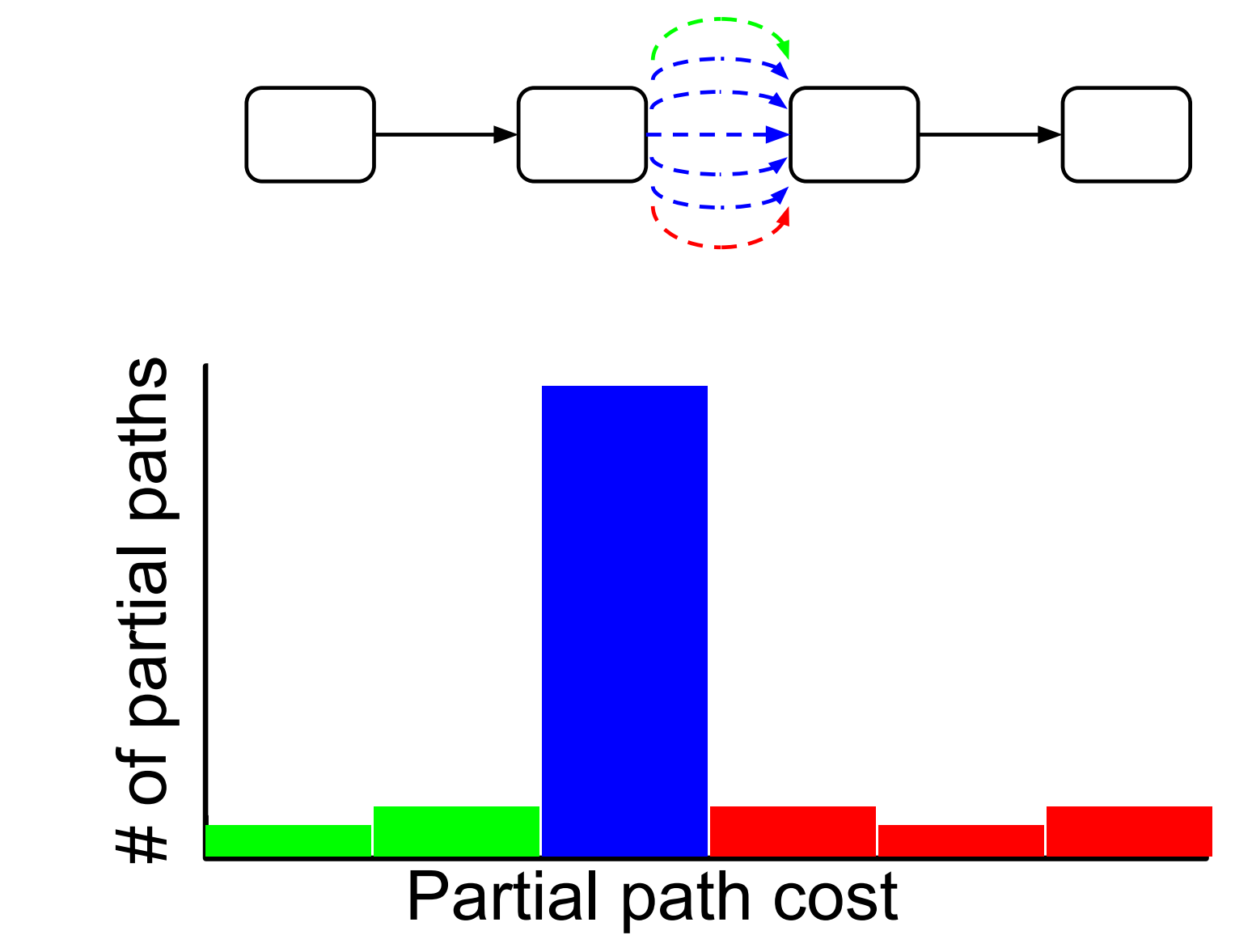
Why is LNBS Worse in SALBP-1?

- Hypothesis: when partial path costs are not diverse (low entropy), finding a better solution in a partial state space is difficult
- No much difference in easy problems (the solution length is small)

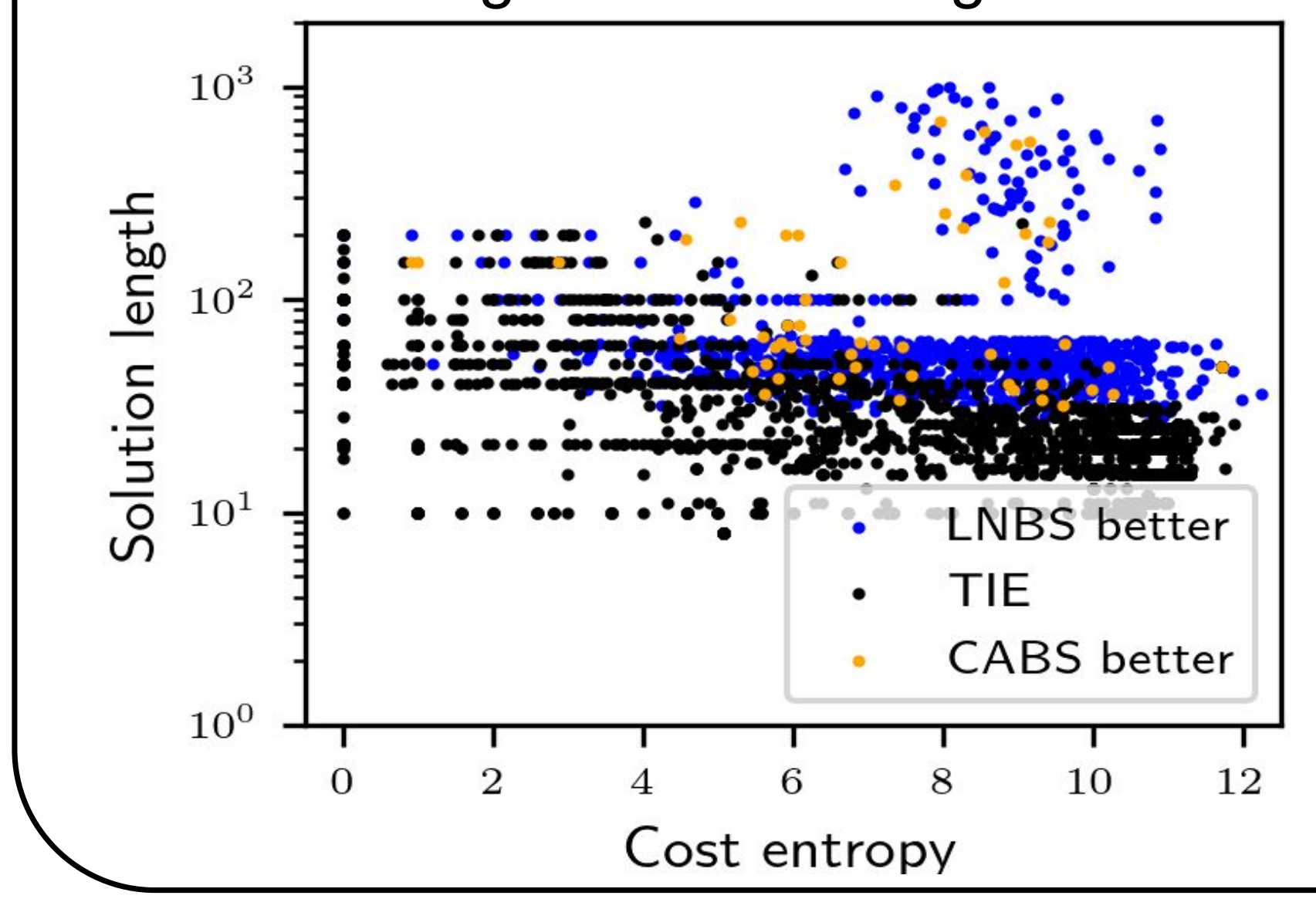
Diverse => easy to find a better one?



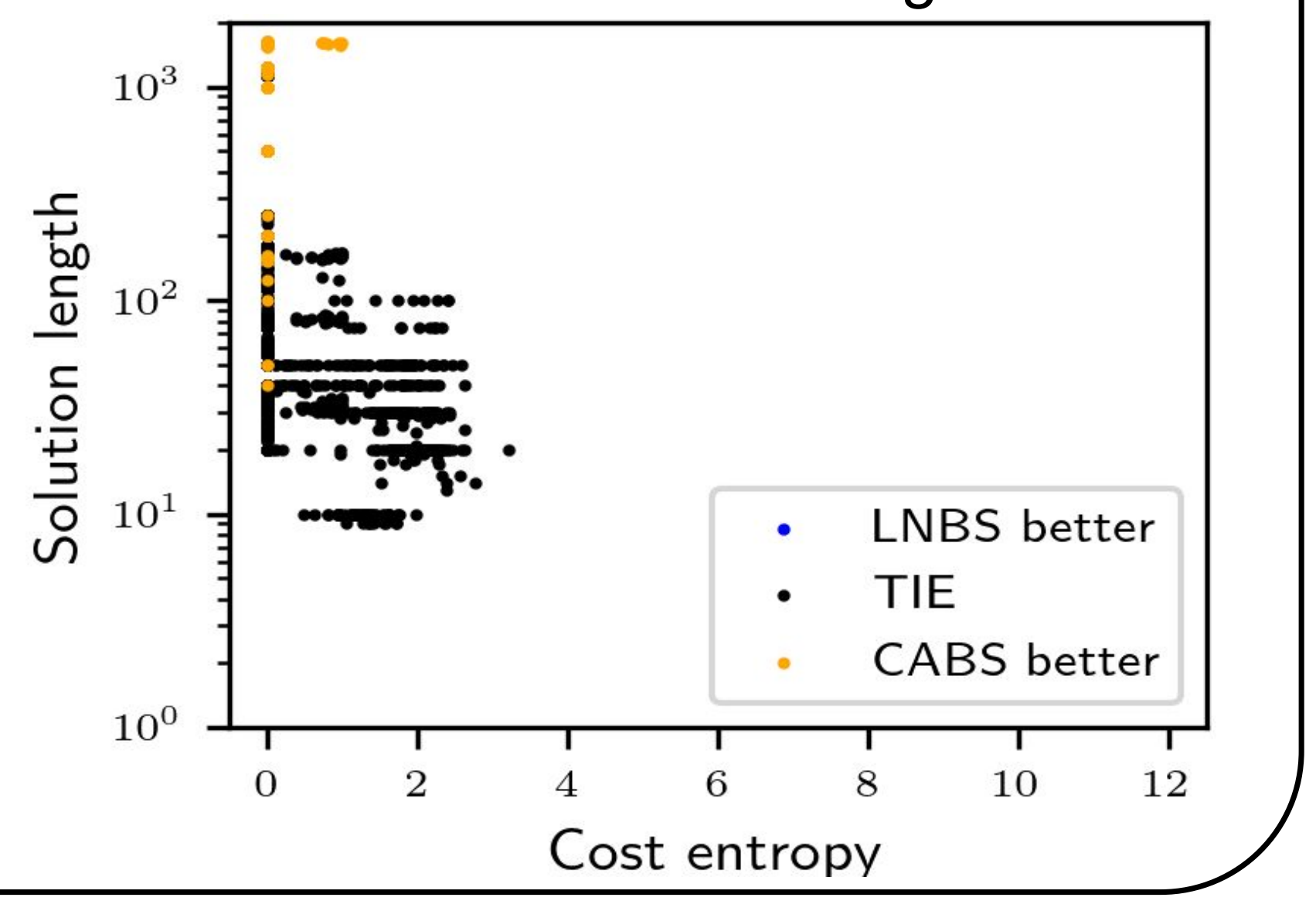
Not diverse => difficult?



Routing and Scheduling Problems



Other Problems Including SALBP-1


Solution length
10^3
10^2
10^1
10^0
Cost entropy
LNBS better
TIE
CABS better