Computational Protein Design Using AND/OR Branch-and-Bound Search

Yichao Zhou Yuexing Zhou Jianyang Zeng

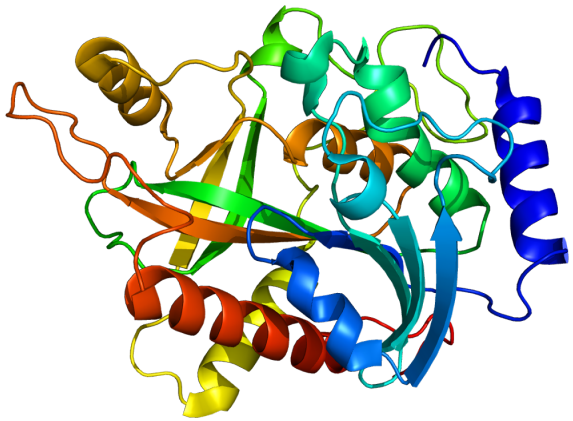
Institute for Interdisciplinary Information Sciences
Tsinghua University

Apr, 2015

What is Structure-Based Protein Design?

Toolbox

- Energy Functions
- Rotamer Library
- Backbone Template



Protein structure template

Design Algorithm

· · ·	Ile	Ser	Ile
Pro	Ala	Ile	Trp
Val	Gly	Ser	Asp
Gly	Gly	Pro	Glu
Ile	Pro	His	· · ·

1D amino acid sequence

Why We Need Protein Design?

Applications

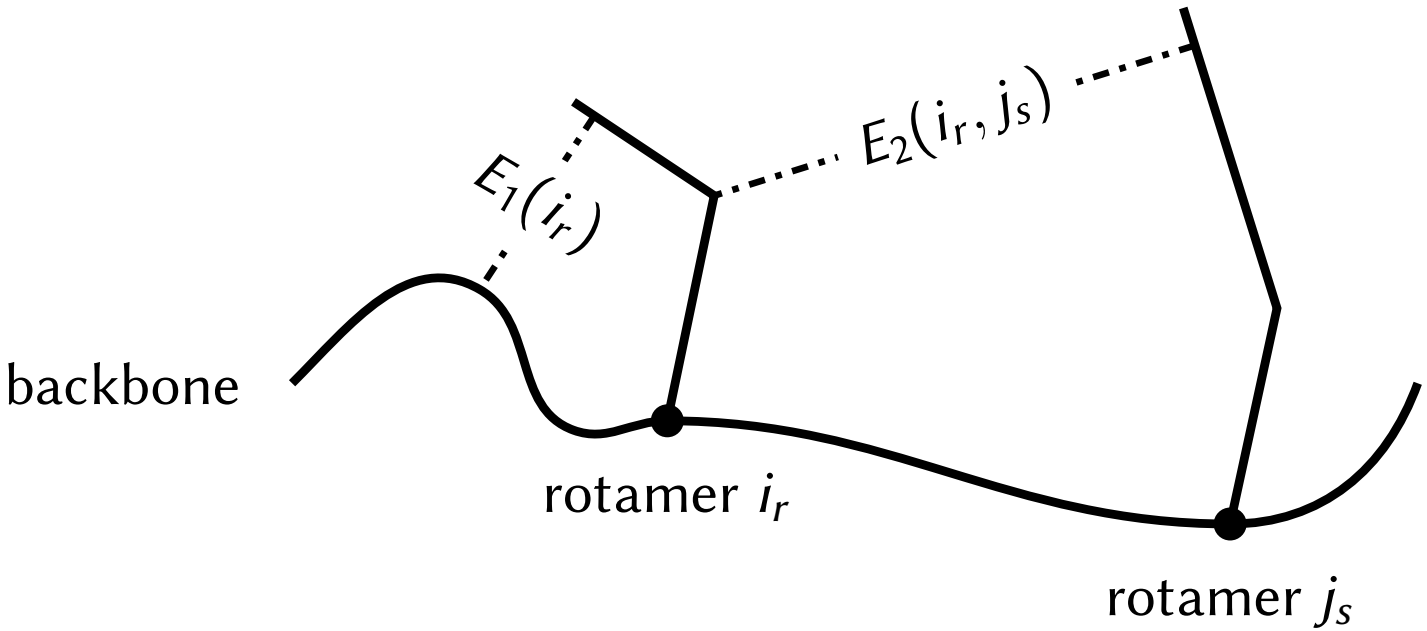
Figure: New Drug Discovery

Figure: Drug Resistance Prediction

Figure: Enzyme Optimization

Figure: New Biosensor Design

Protein Design as an Optimization Problem



Energy Function (Optimization Target)

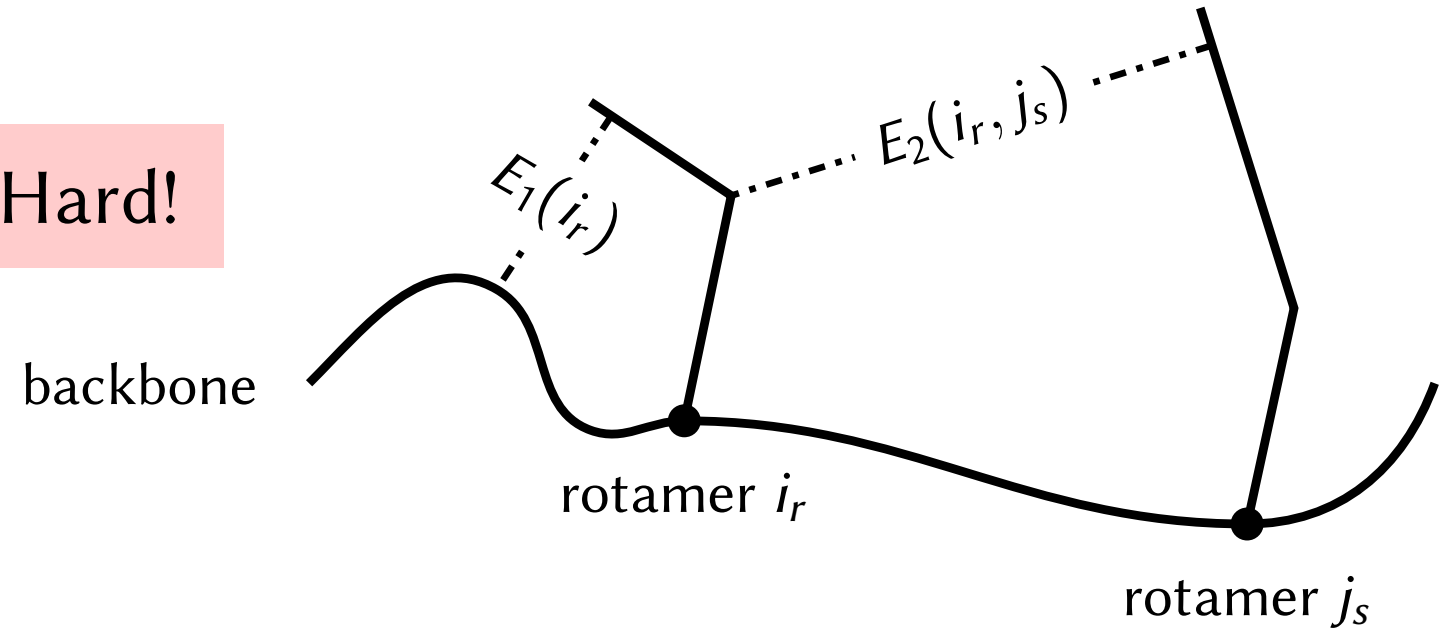
$$E_T = \sum_{i_r} E_1(i_r) + \sum_{i_r} \sum_{j_s, i<j} E_2(i_r, j_s)$$

Self energy of rotamer i_r

Pairwise energy between rotamer i_r and j_s

Protein Design as an Optimization Problem

NP Hard!



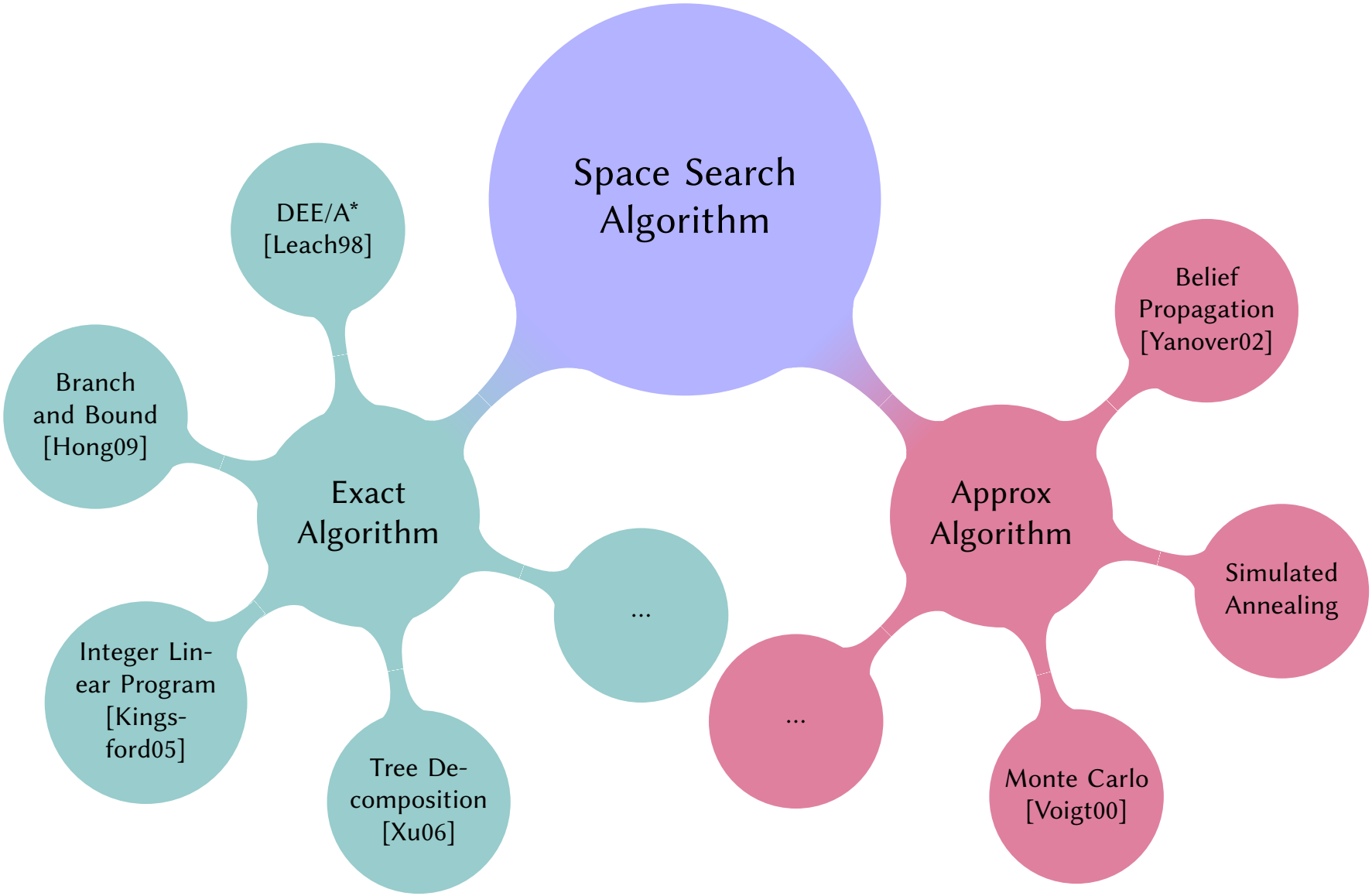
Energy Function (Optimization Target)

$$E_T = \sum_{i_r} E_1(i_r) + \sum_{i_r} \sum_{j_s, i<j} E_2(i_r, j_s)$$

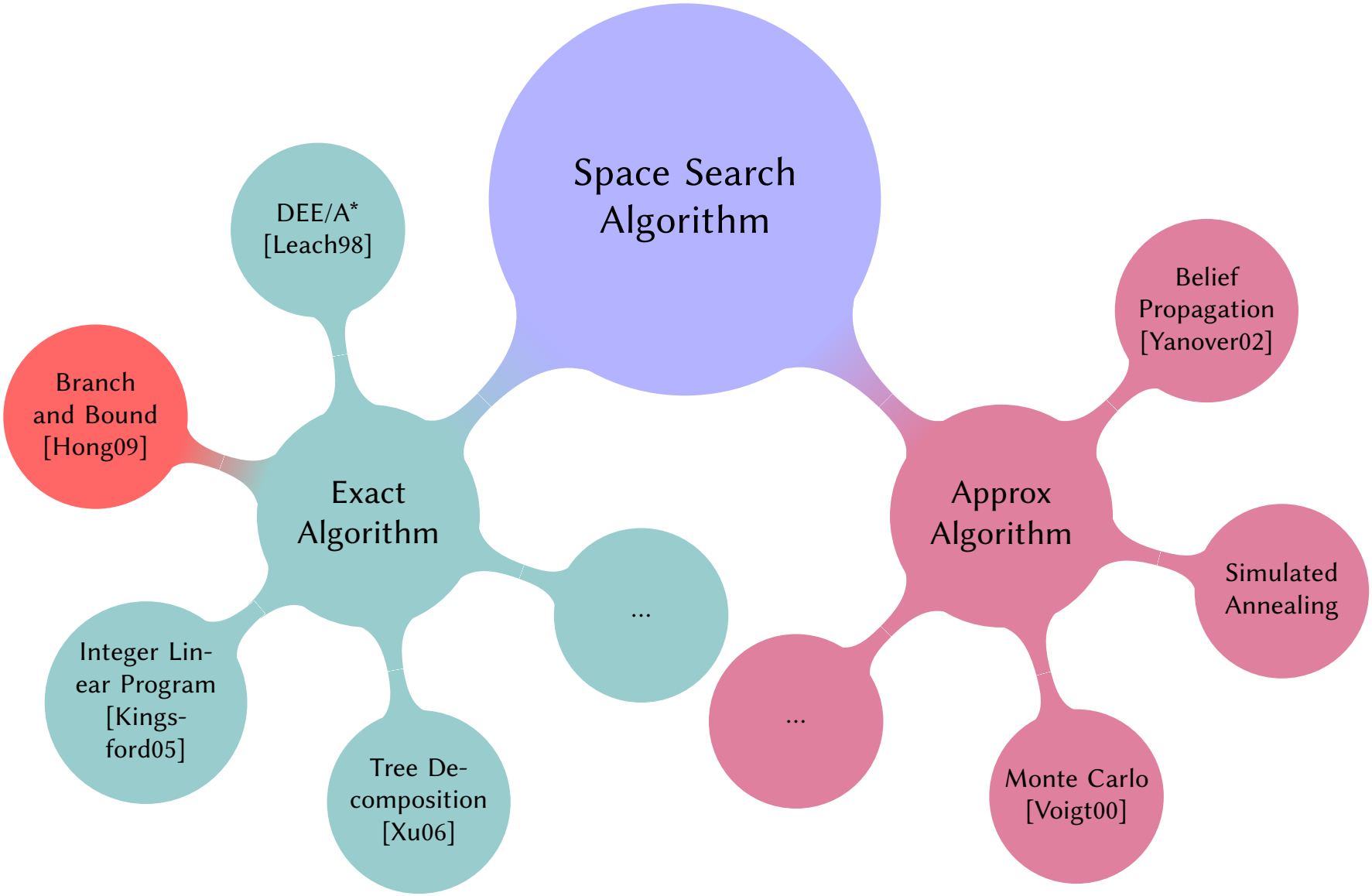
Self energy of rotamer i_r

Pairwise energy between rotamer i_r and j_s

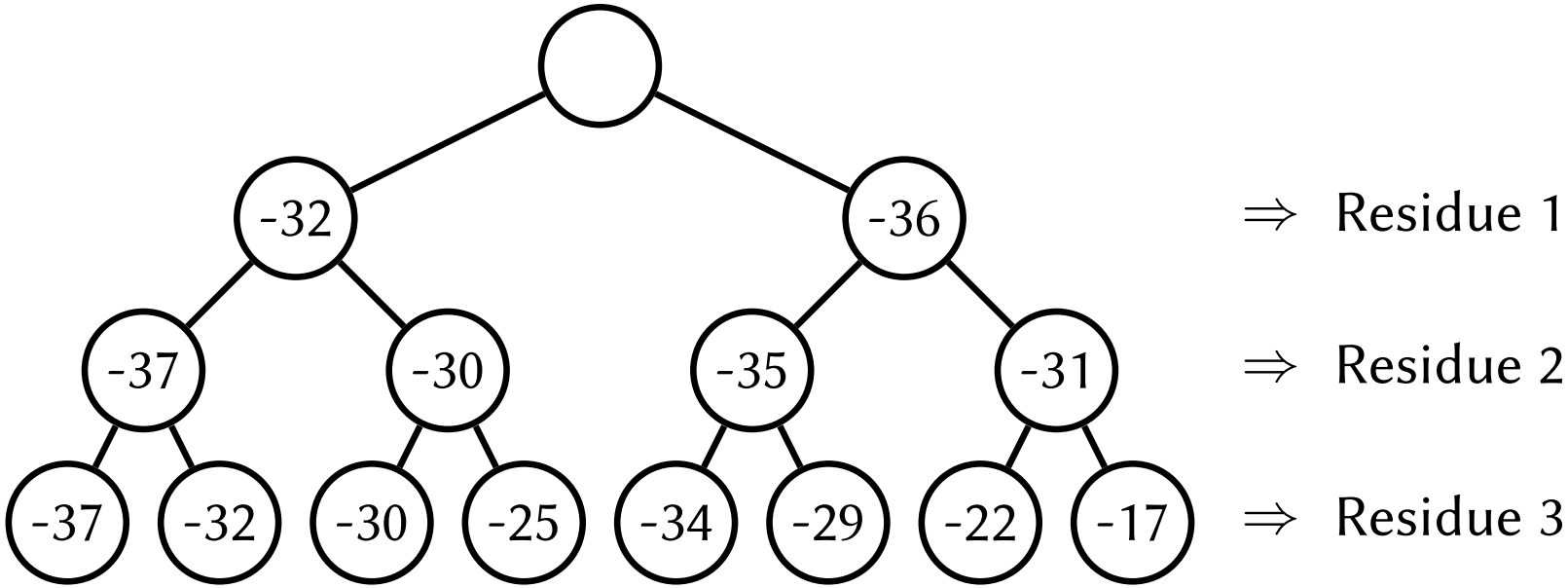
Search Algorithm



Search Algorithm



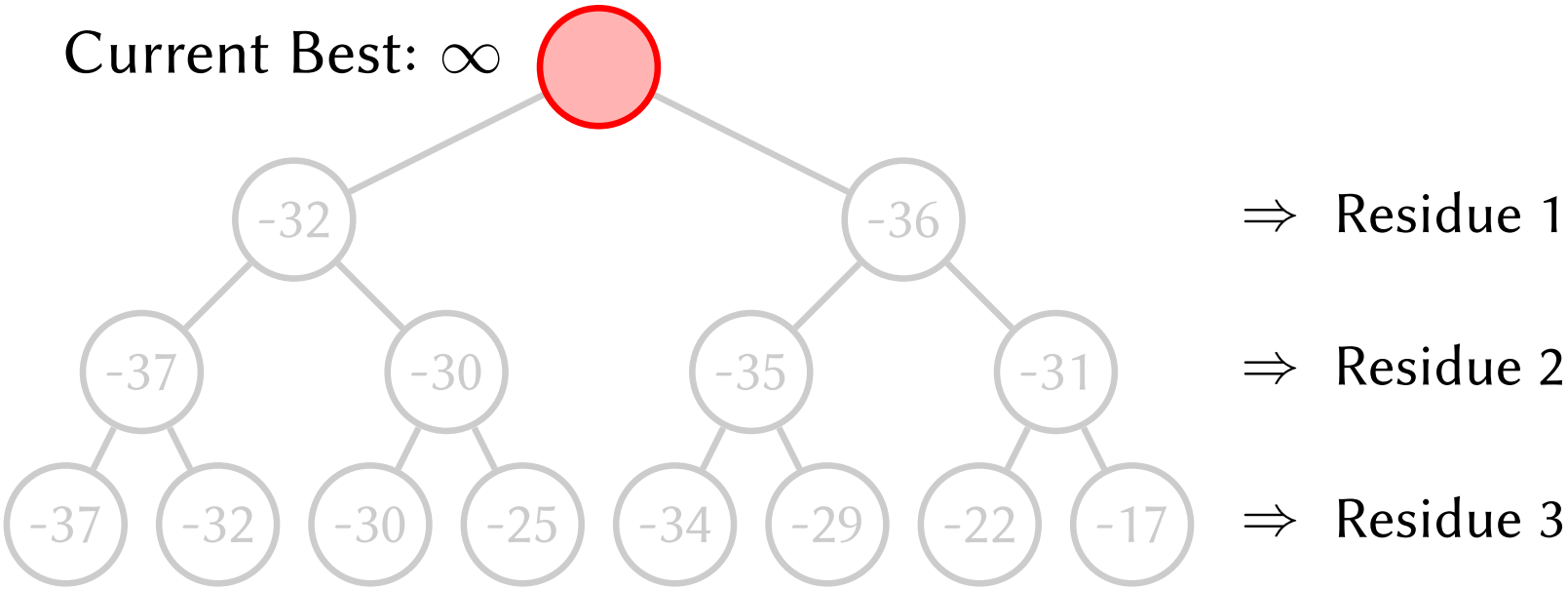
Traditional Branch-and-bound Search Space



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree rooted at x.

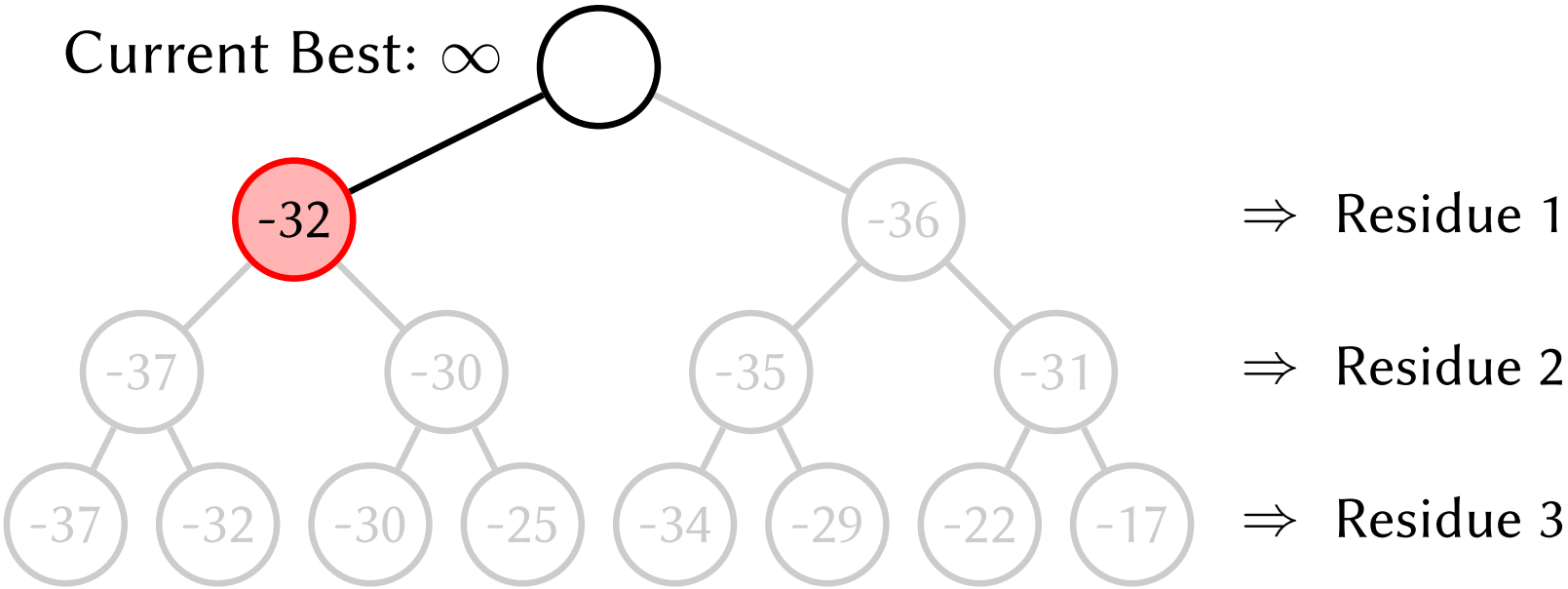
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

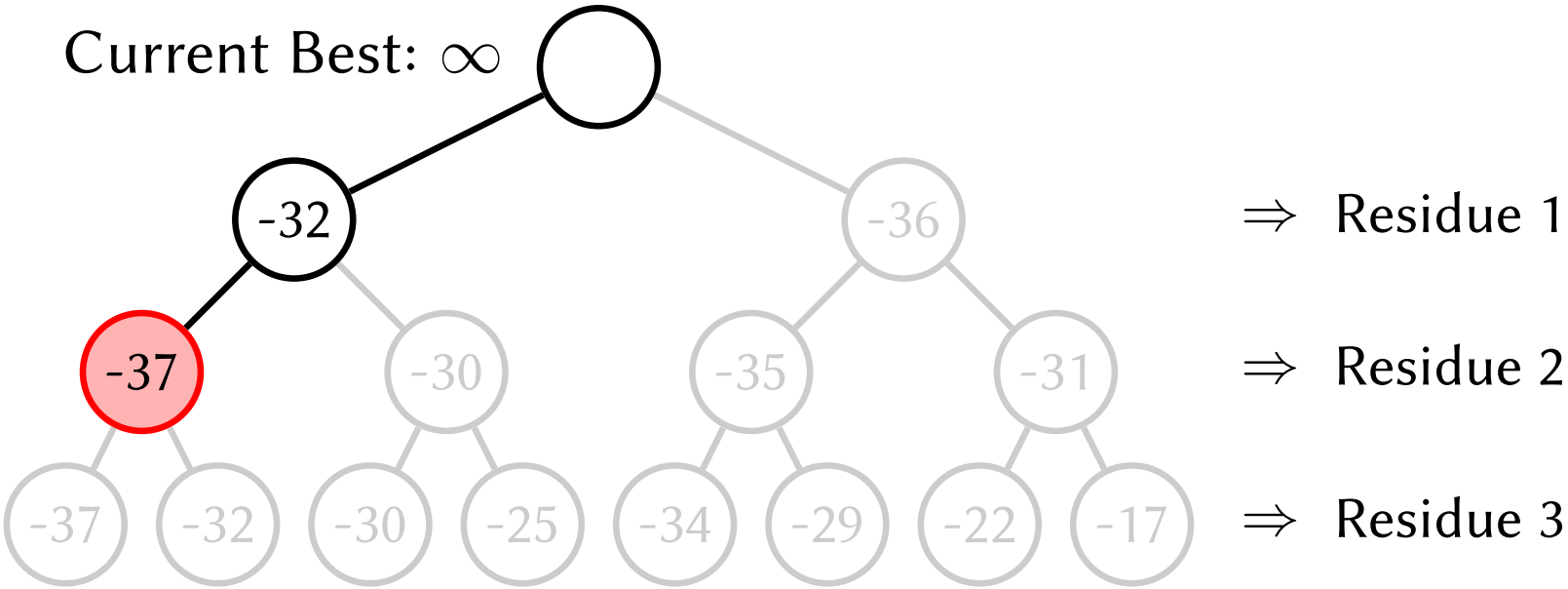
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

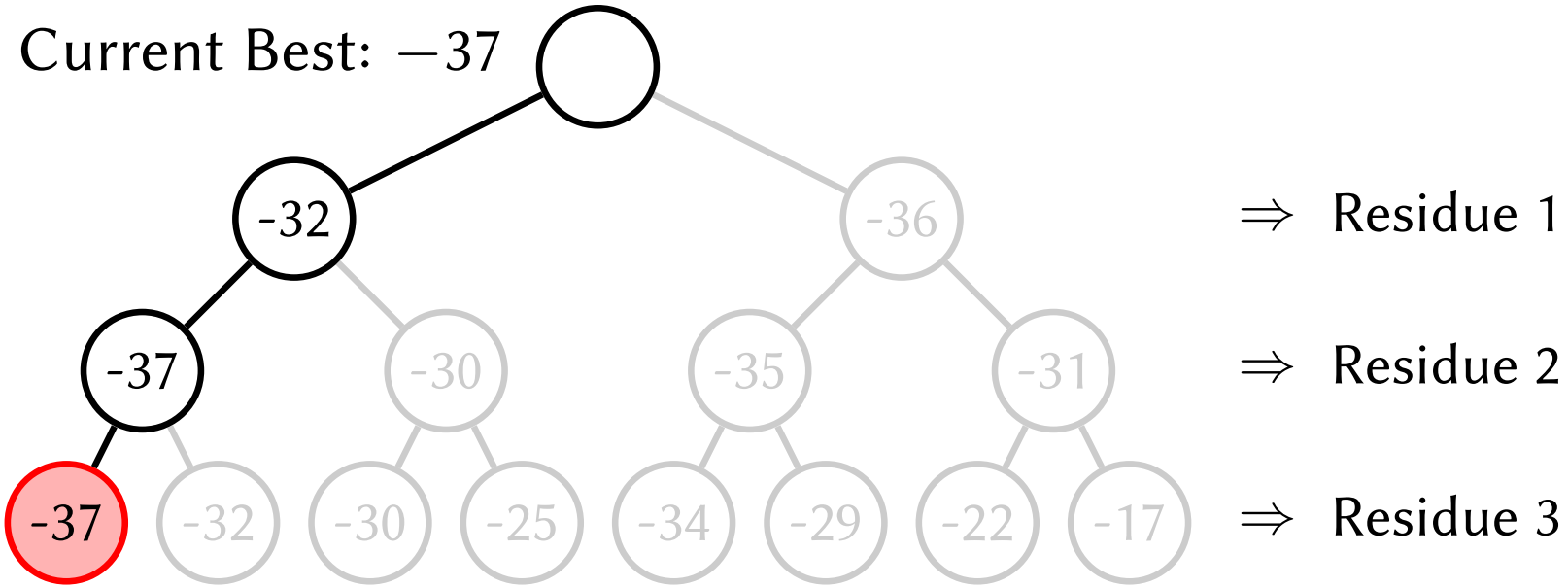
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

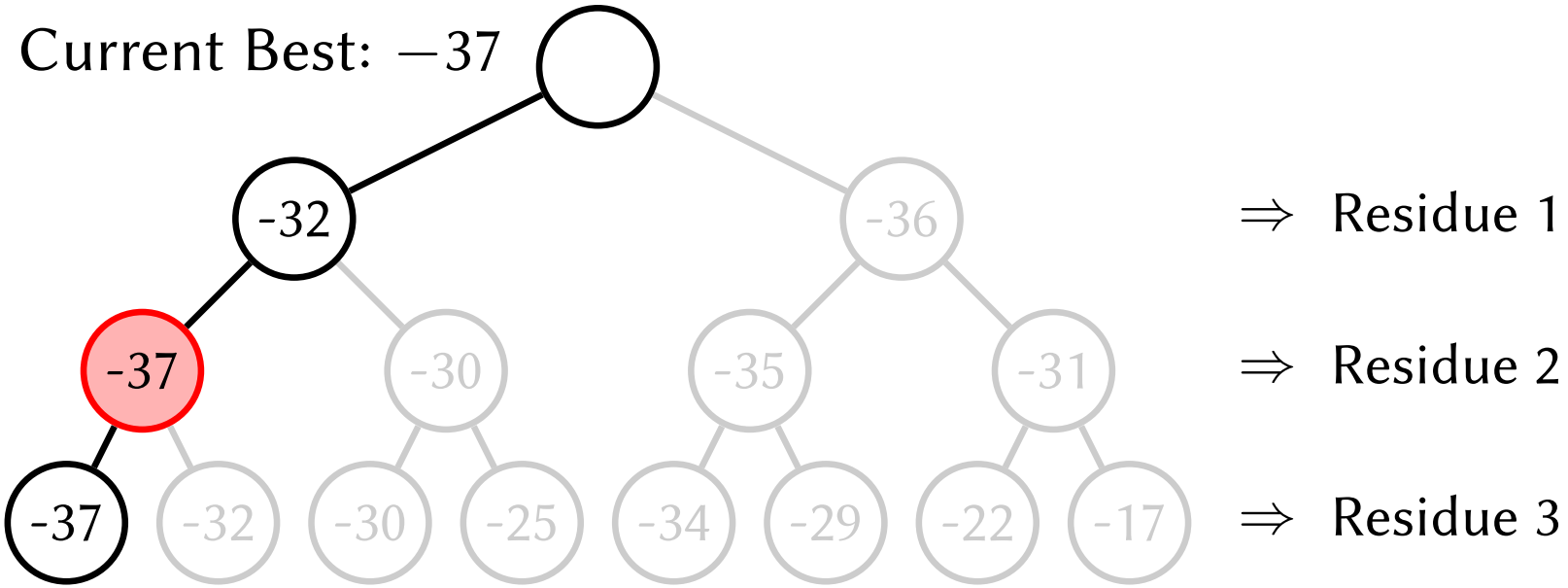
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

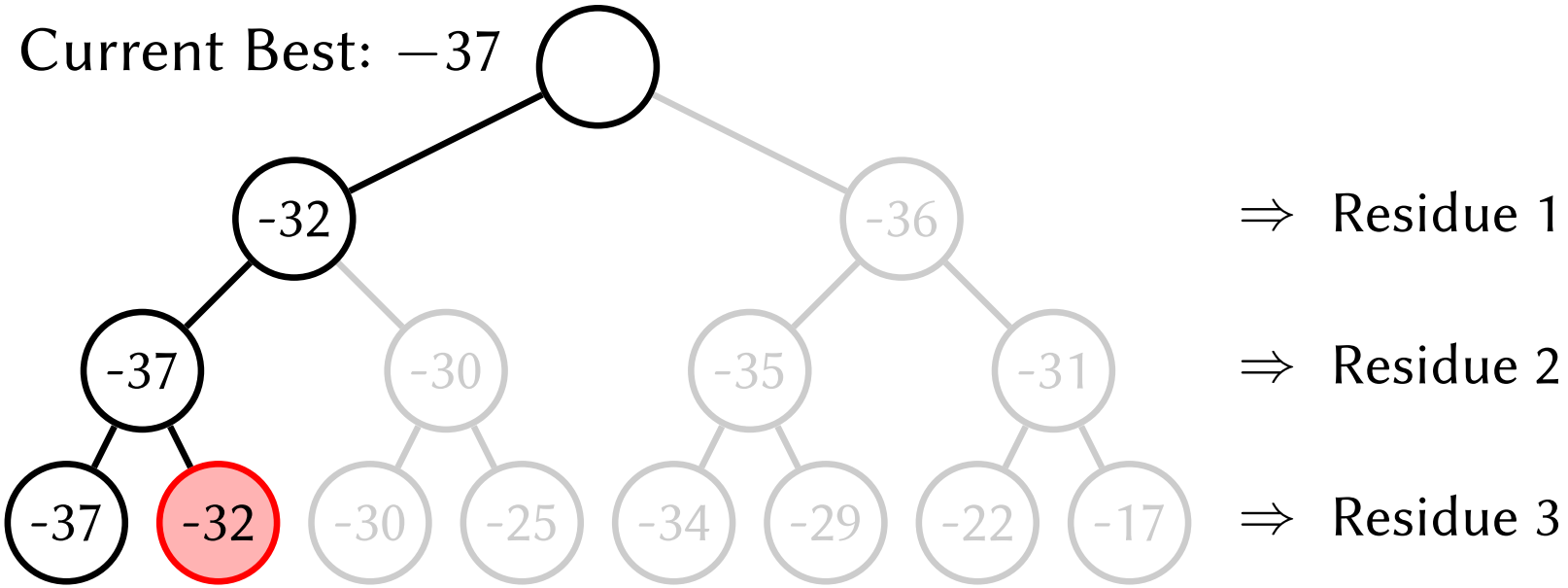
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

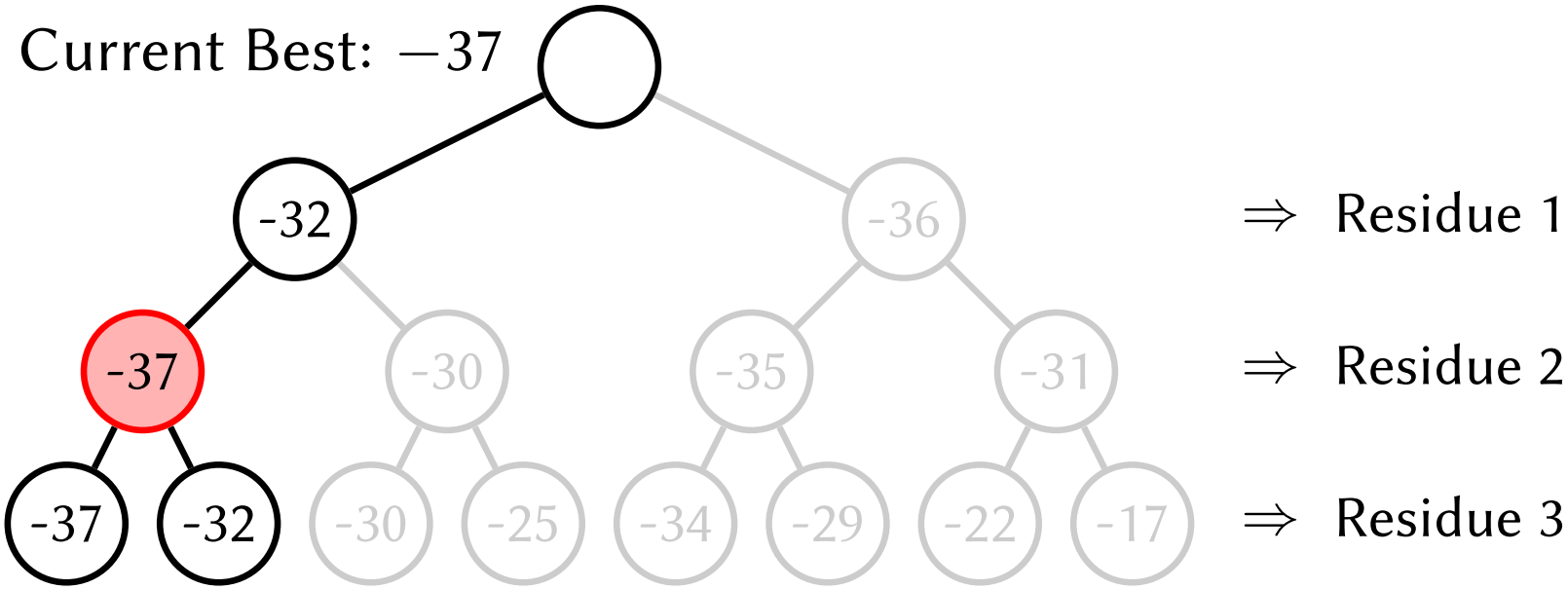
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

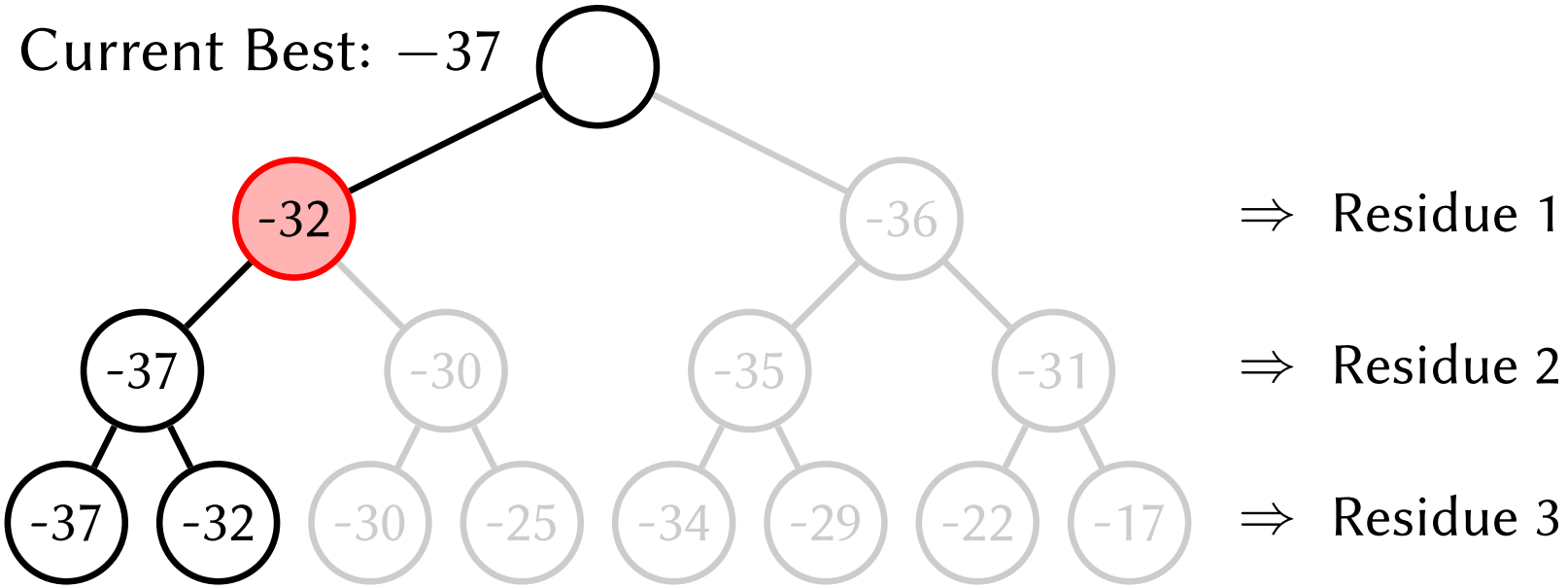
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

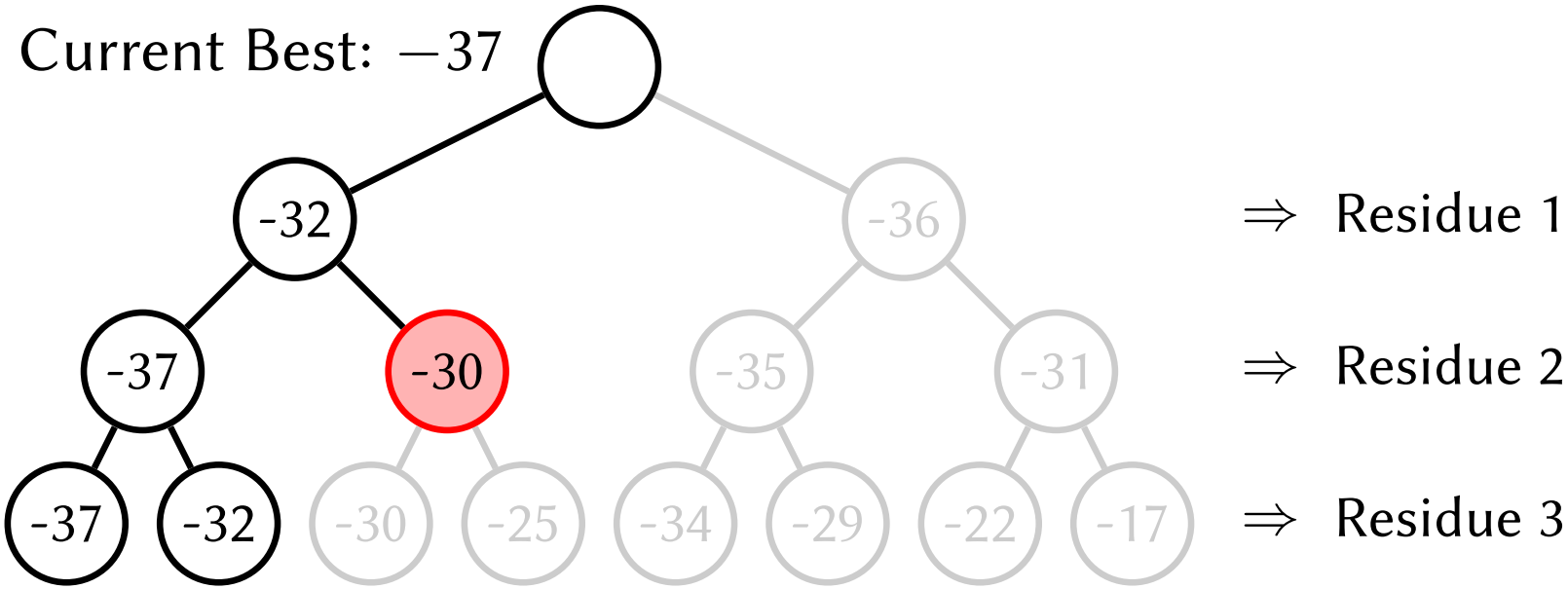
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

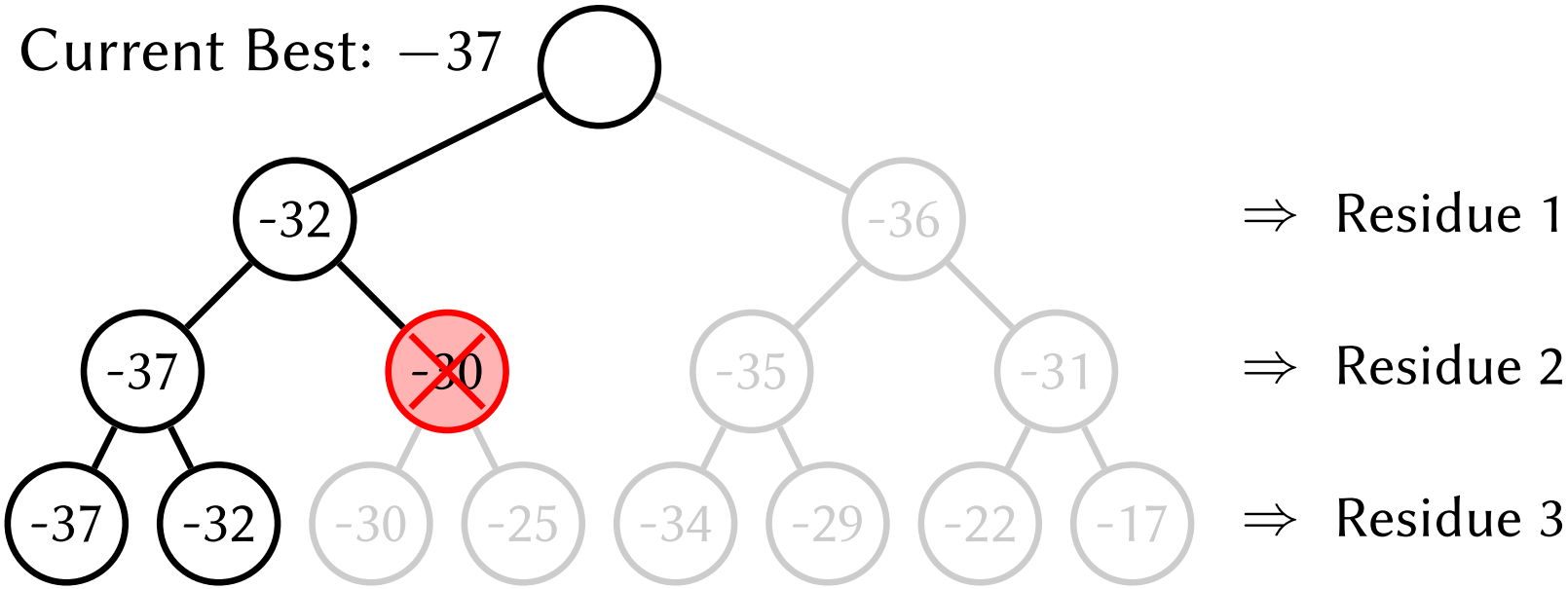
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

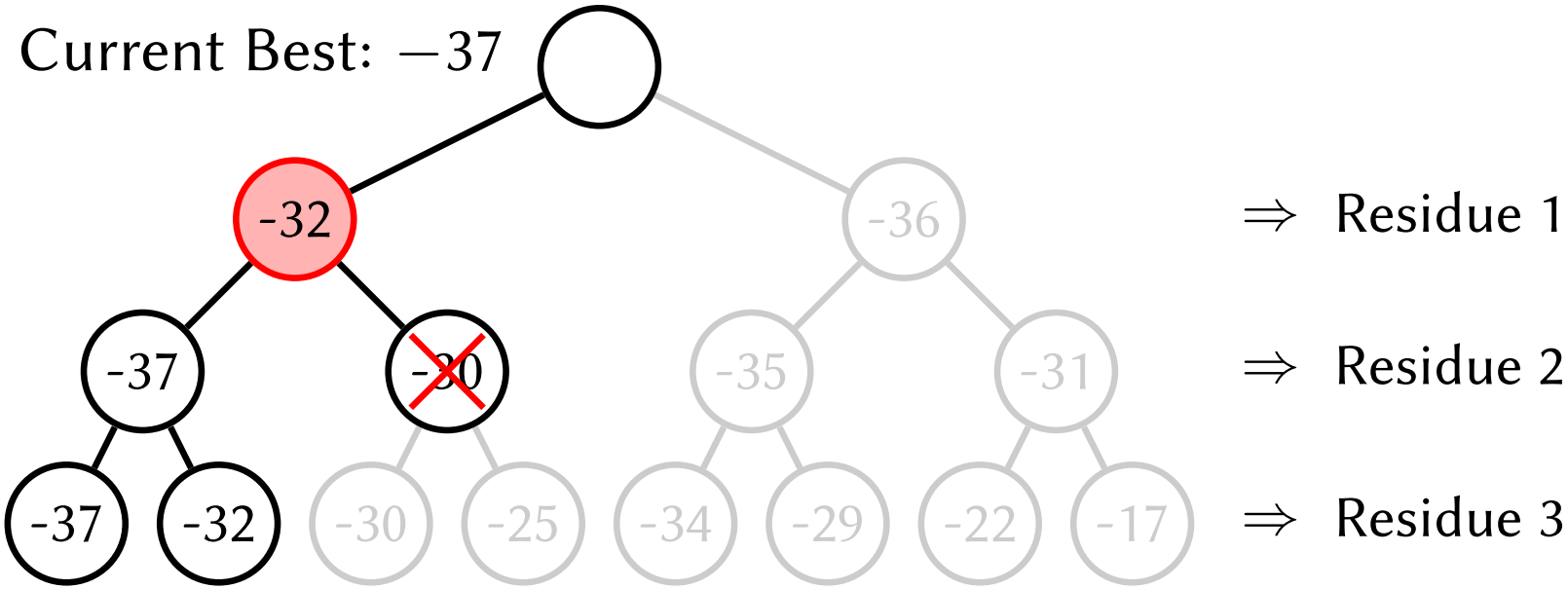
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

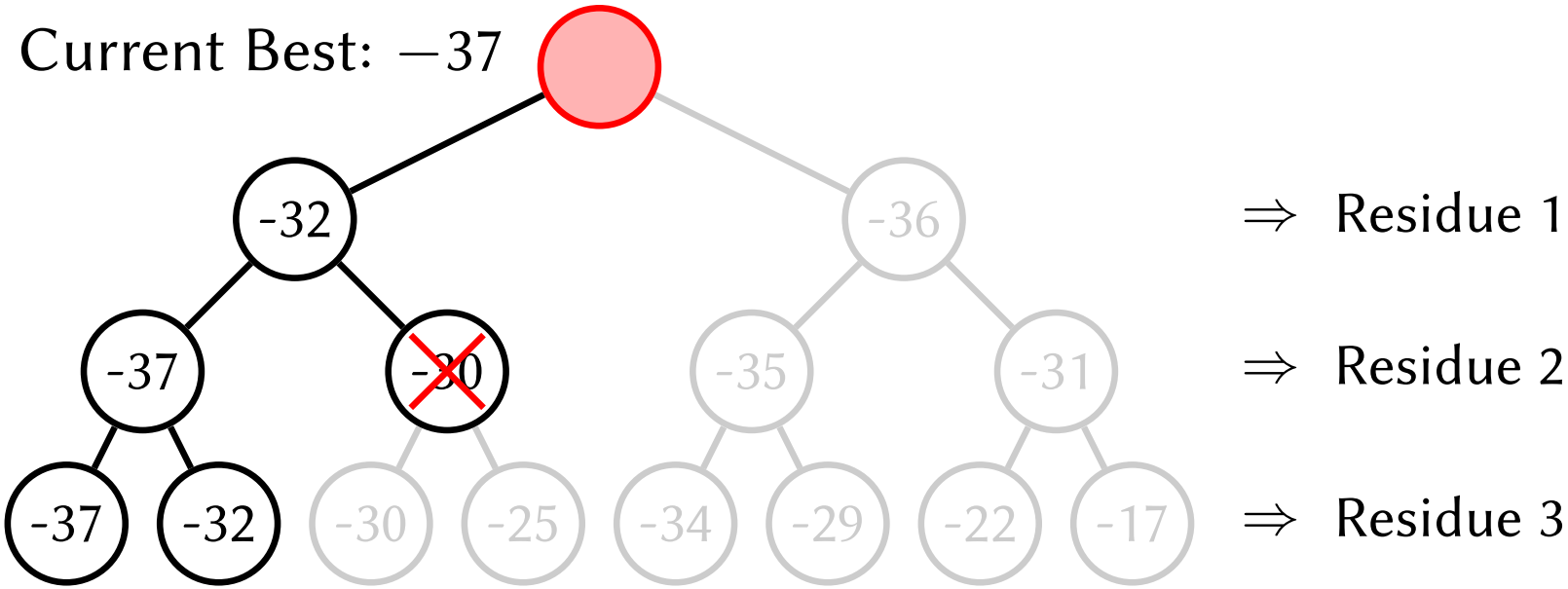
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

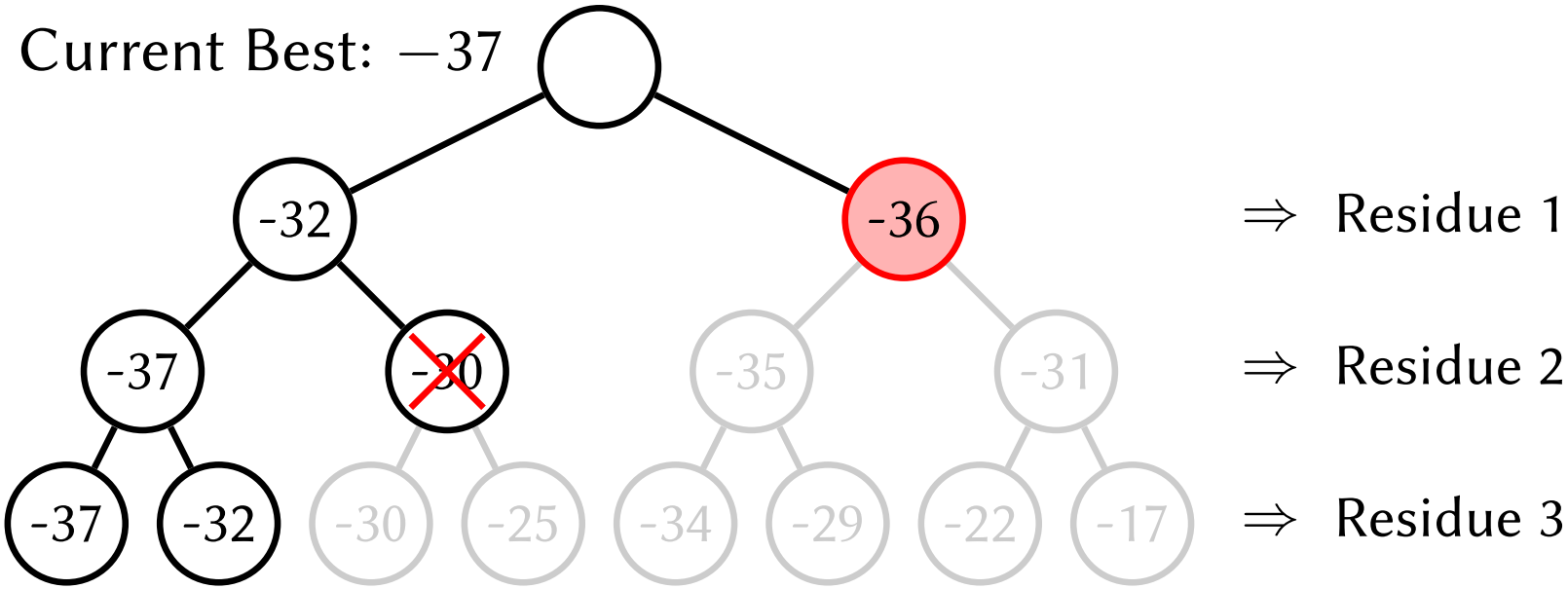
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

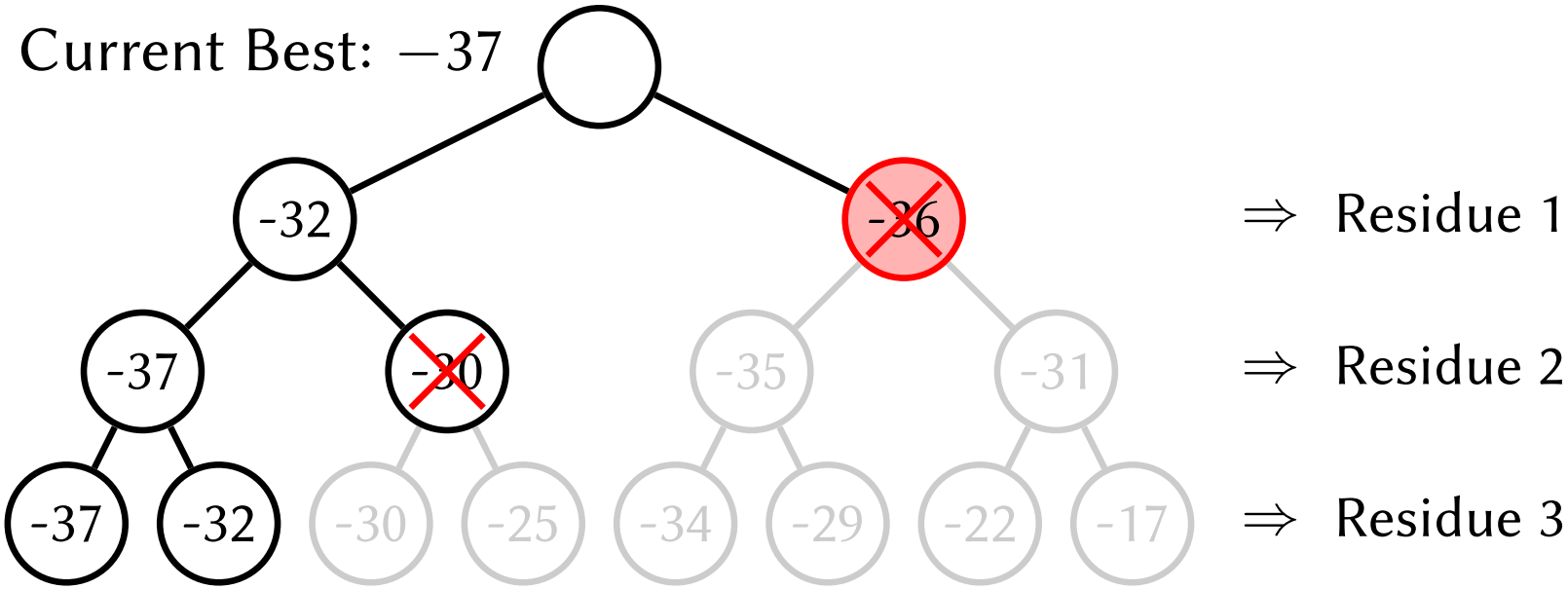
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

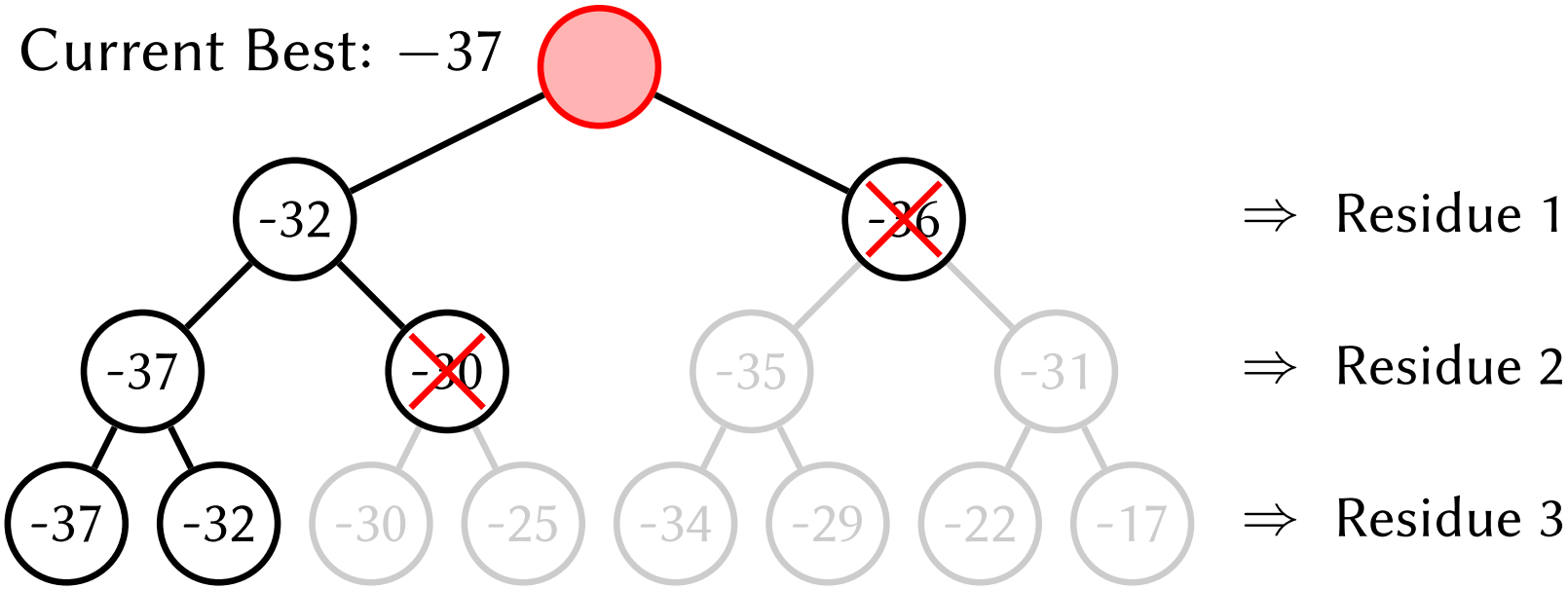
Traditional Branch-and-bound Search



Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

Traditional Branch-and-bound Search



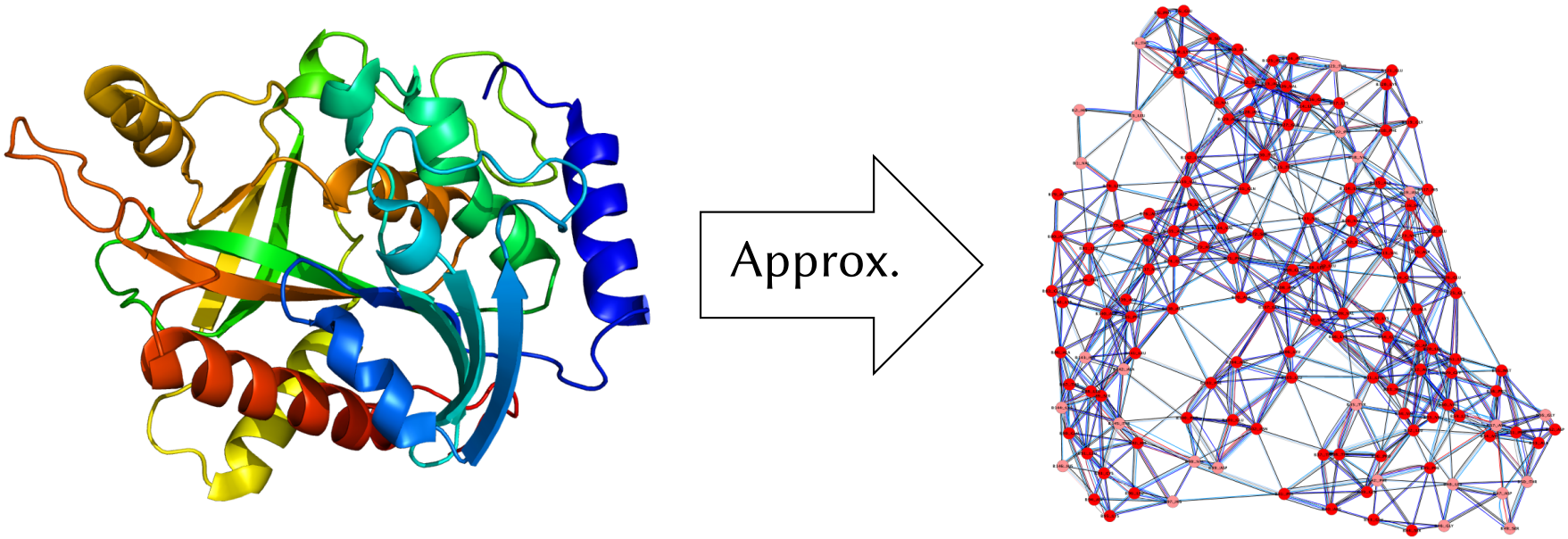
Heuristic function

Heuristic function $h(x)$ returns a lower bound of the energy of leaf nodes in the subtree root at x.

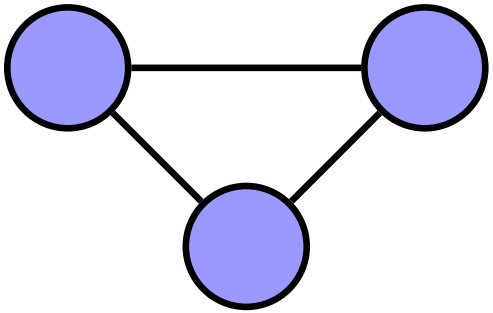
Residue Interaction Network

Approximation

Simplify the energy model by only considering the interaction between residues whose energy term is greater than a threshold.



Problem of Traditional Branch-and-bound



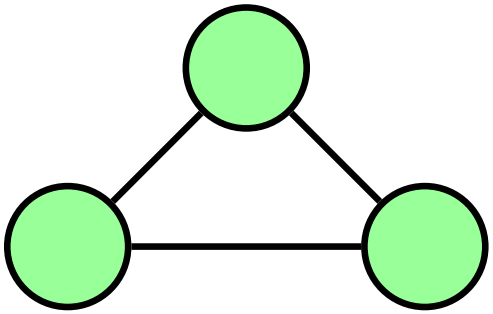
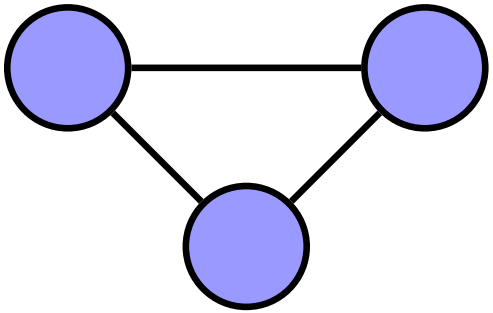
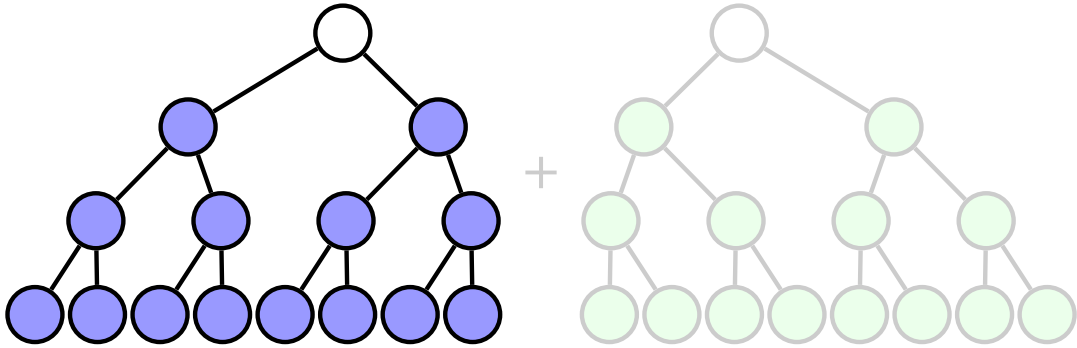


Figure: Residue Interaction Network

VS

Problem of Traditional Branch-and-bound





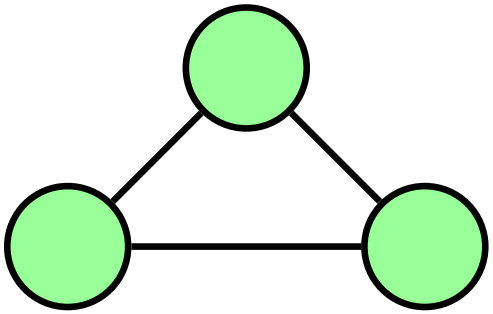
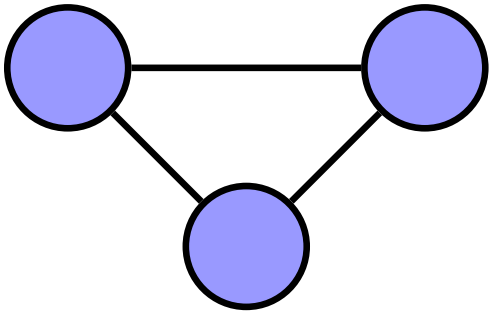
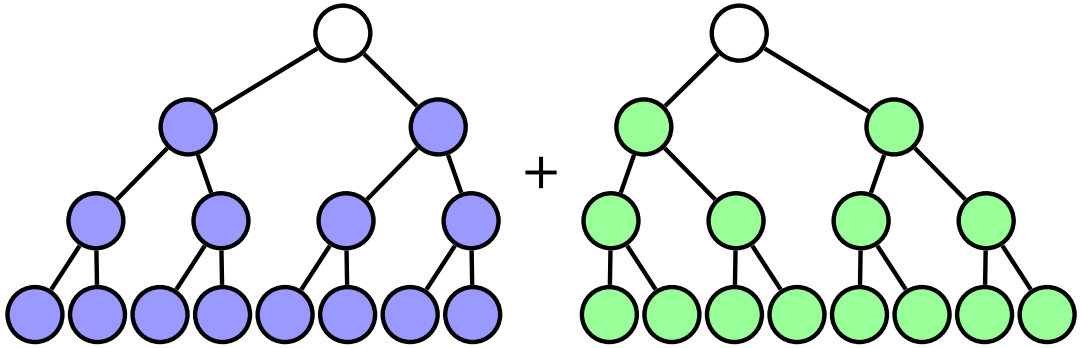


Figure: Residue Interaction Network

VS

Problem of Traditional Branch-and-bound





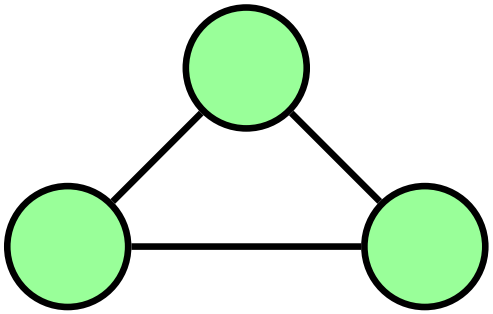
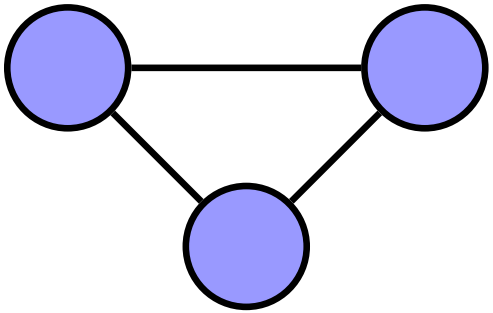
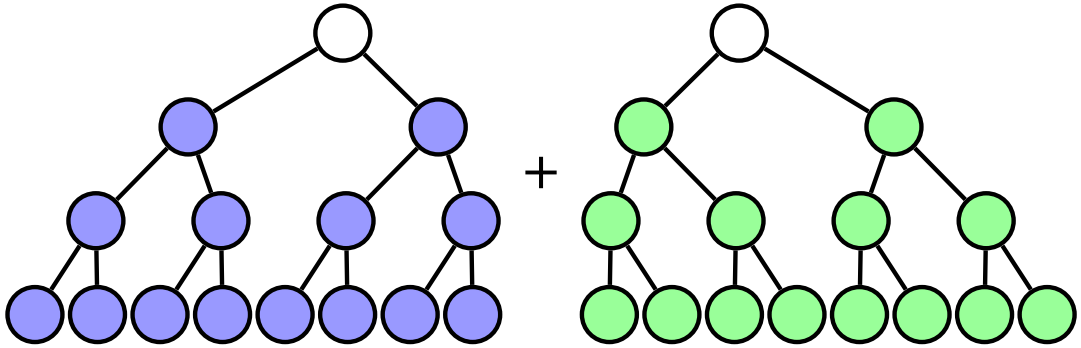


Figure: Residue Interaction Network

VS

Problem of Traditional Branch-and-bound





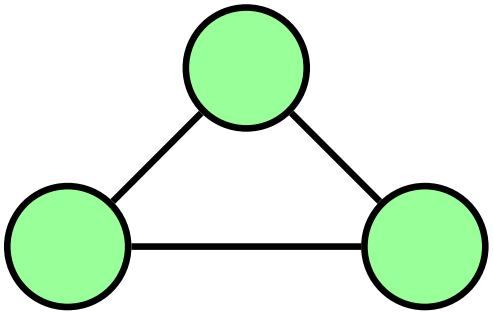
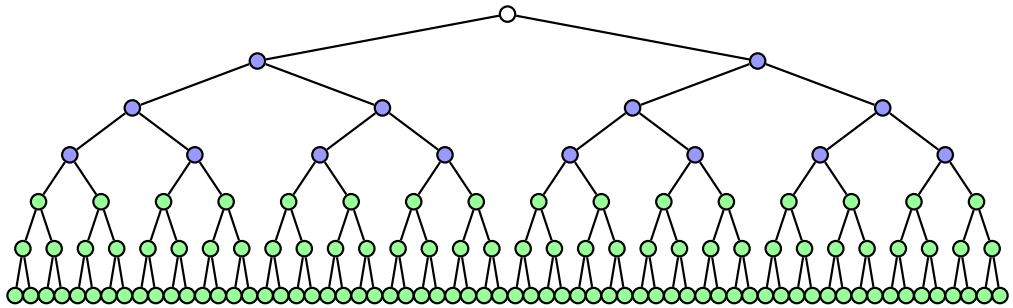


Figure: Residue Interaction Network

VS



Problem of Traditional Branch-and-bound

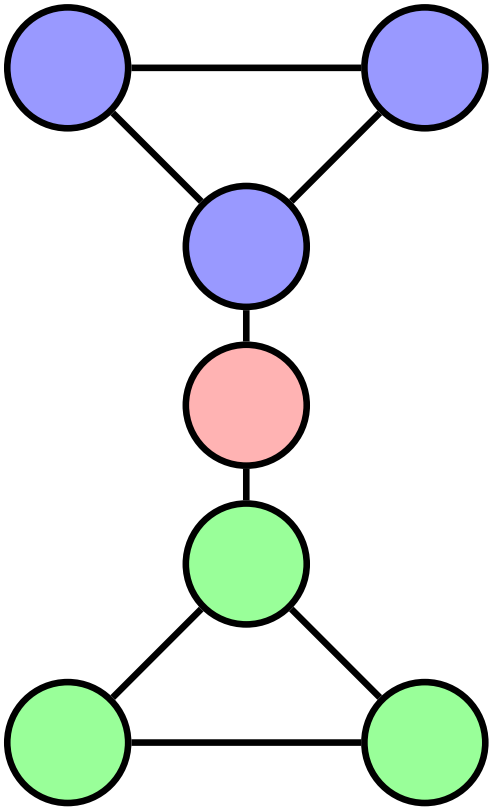
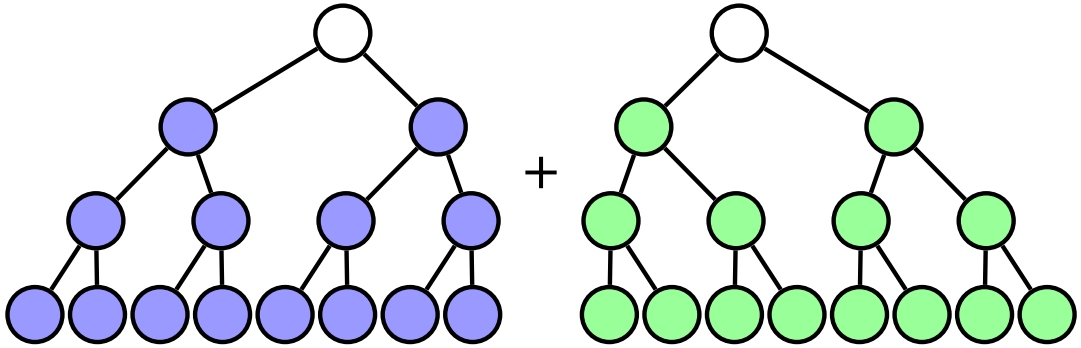
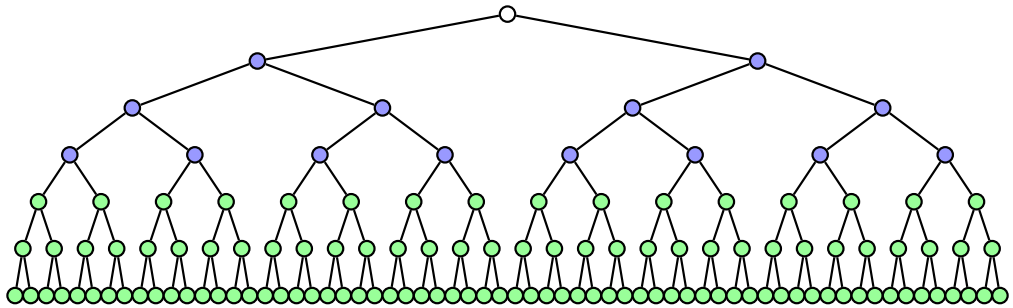


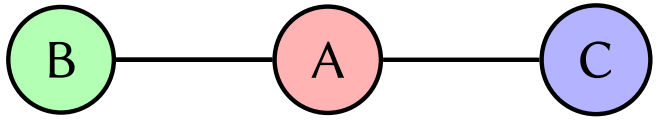
Figure: Residue Interaction Network



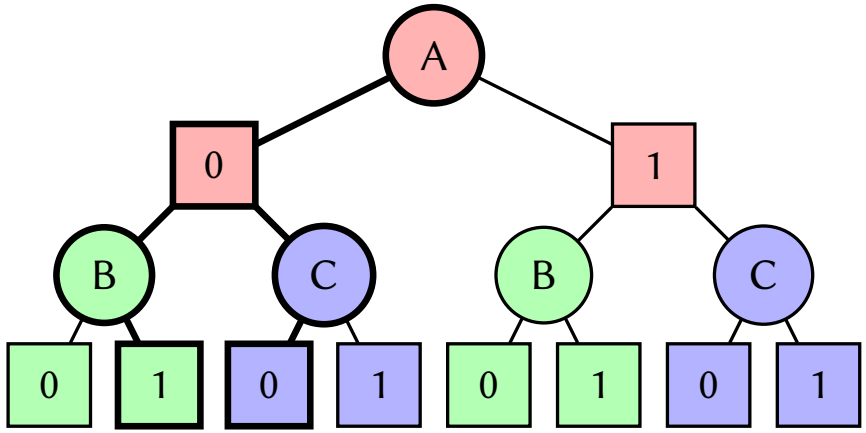
VS



AND/OR Search Space



(a) Residue Interaction Network



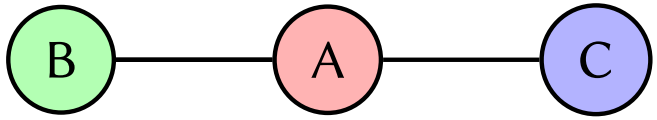
(b) AND/OR Search Space

Node value

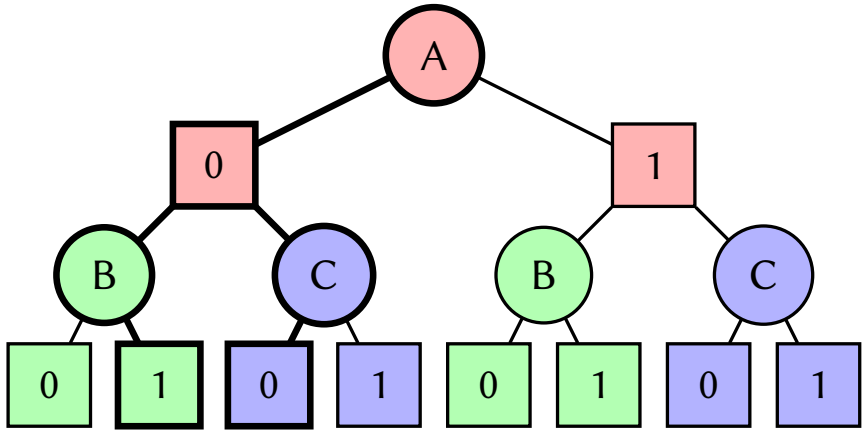
$$v_x = \begin{cases} \sum_{y \in \text{child}(x)} v_y, & \text{if } x \text{ is an AND node;} \\ \min_{y \in \text{child}(x)} e(y) + v_y, & \text{if } x \text{ is an OR node.} \end{cases}$$

energy terms between y and ancestors of y

AND/OR Search Space



(a) Residue Interaction Network



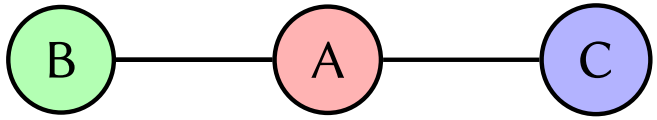
(b) AND/OR Search Space

Node value

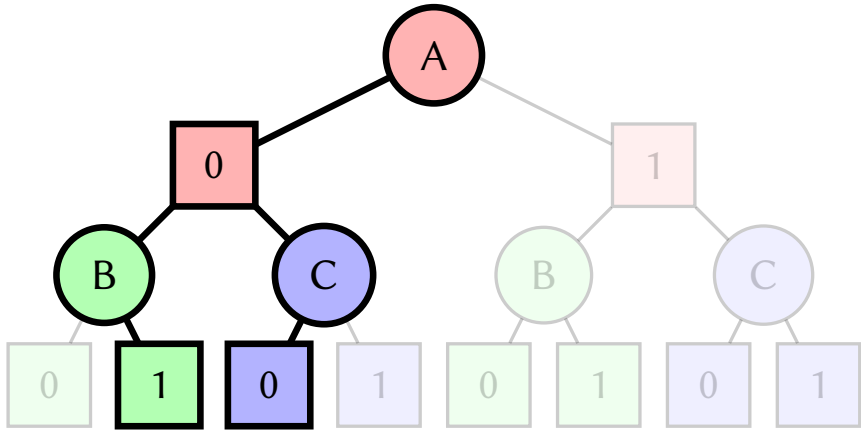
$$v_x = \begin{cases} \sum_{y \in \text{child}(x)} v_y, & \text{if } x \text{ is an AND node;} \\ \min_{y \in \text{child}(x)} e(y) + v_y, & \text{if } x \text{ is an OR node.} \end{cases}$$

energy terms between y and ancestors of y

AND/OR Search Space



(a) Residue Interaction Network



(b) AND/OR Search Space

Node value

$$v_x = \begin{cases} \sum_{y \in \text{child}(x)} v_y, & \text{if } x \text{ is an AND node;} \\ \min_{y \in \text{child}(x)} e(y) + v_y, & \text{if } x \text{ is an OR node.} \end{cases}$$

energy terms between y and ancestors of y

- AND/OR branch-and-bound search can find GMEC efficiently
- Protein design models are not perfect:
  - errors in energy functions and structure of protein templates
  - assumptions of rigid backbone and discrete side-chain conformations
- The GMEC solution is not sufficient in practice
- Sub-optimal (e.g., *second-best*, ...) conformations are required

Finding the k best solutions

- add a threshold to prevent from pruning sub-optimal solutions
- maintain k node values for each node, i.e., array $v_x[1], v_x[2], \ldots, v_x[k]$

Finding Sub-optimal Solutions

Problems

- AND/OR branch-and-bound search can find GMEC efficiently
- Protein design models are not perfect:
  - errors in energy functions and structure of protein templates
  - assumptions of rigid backbone and discrete side-chain conformations
- The GMEC solution is not sufficient in practice
- Sub-optimal (e.g., *second-best*, ...) conformations are required

Finding the k best solutions

- add a threshold to prevent from pruning sub-optimal solutions
- maintain k node values for each node, i.e., array $v_x[1], v_x[2], \ldots, v_x[k]$

Finding Sub-optimal Solutions

Problems

- AND/OR branch-and-bound search can find GMEC efficiently
- Protein design models are not perfect:
  - errors in energy functions and structure of protein templates
  - assumptions of rigid backbone and discrete side-chain conformations
- The GMEC solution is not sufficient in practice
- Sub-optimal (e.g., *second-best*, ...) conformations are required

Finding the k best solutions

- add a threshold to prevent from pruning sub-optimal solutions
- maintain k node values for each node, i.e., array $v_x[1], v_x[2], \ldots, v_x[k]$

- AND/OR branch-and-bound search can find GMEC efficiently
- Protein design models are not perfect:
  - errors in energy functions and structure of protein templates
  - assumptions of rigid backbone and discrete side-chain conformations
- The GMEC solution is not sufficient in practice
- Sub-optimal (e.g., *second-best*, ...) conformations are required

Finding the k best solutions

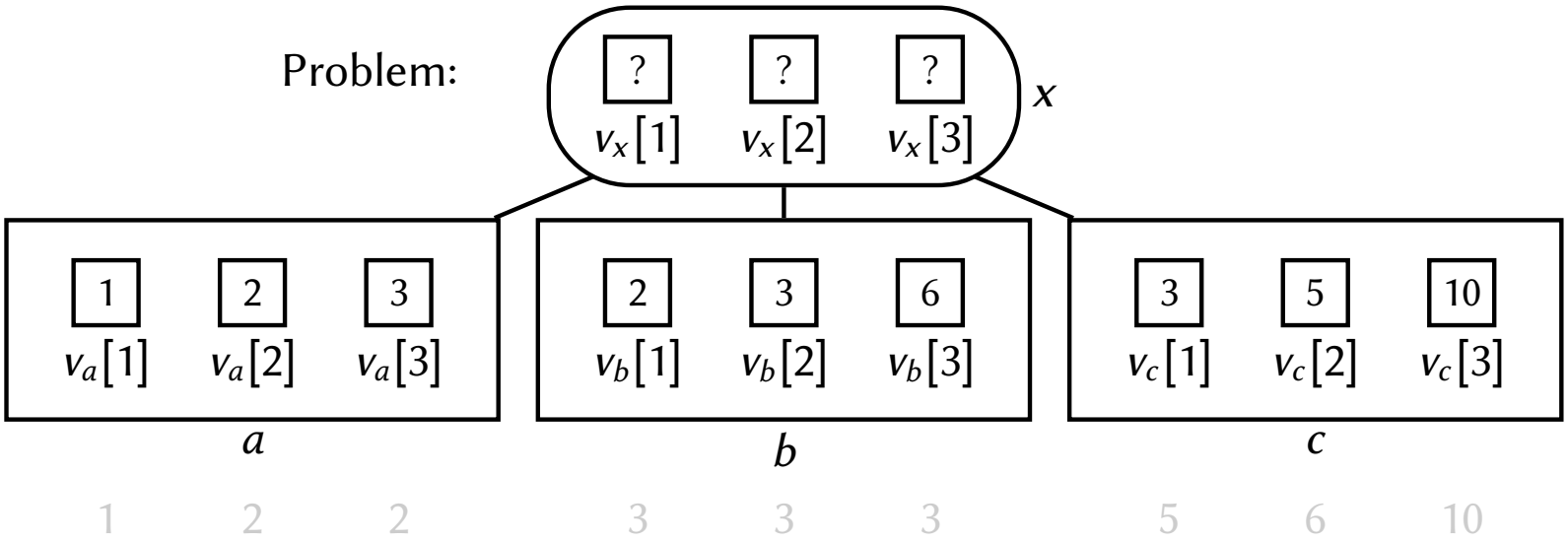
- add a threshold to prevent from pruning sub-optimal solutions
- maintain k node values for each node, i.e., array $v_x[1], v_x[2], \ldots, v_x[k]$

- AND/OR branch-and-bound search can find GMEC efficiently
- Protein design models are not perfect:
  - errors in energy functions and structure of protein templates
  - assumptions of rigid backbone and discrete side-chain conformations
- The GMEC solution is not sufficient in practice
- Sub-optimal (e.g., *second-best*, ...) conformations are required

Finding the k best solutions

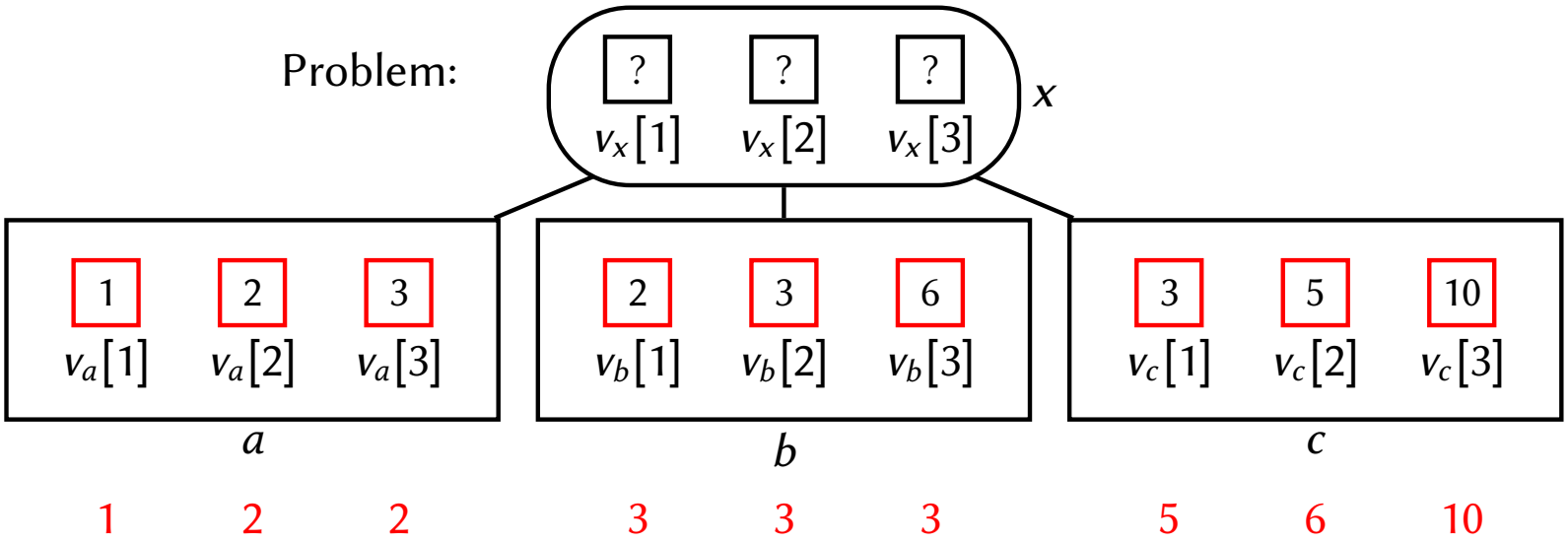
- add a threshold to prevent from pruning sub-optimal solutions
- maintain k node values for each node, i.e., array $v_x[1], v_x[2], \ldots, v_x[k]$

Computing $v[\cdot]$ values for OR nodes



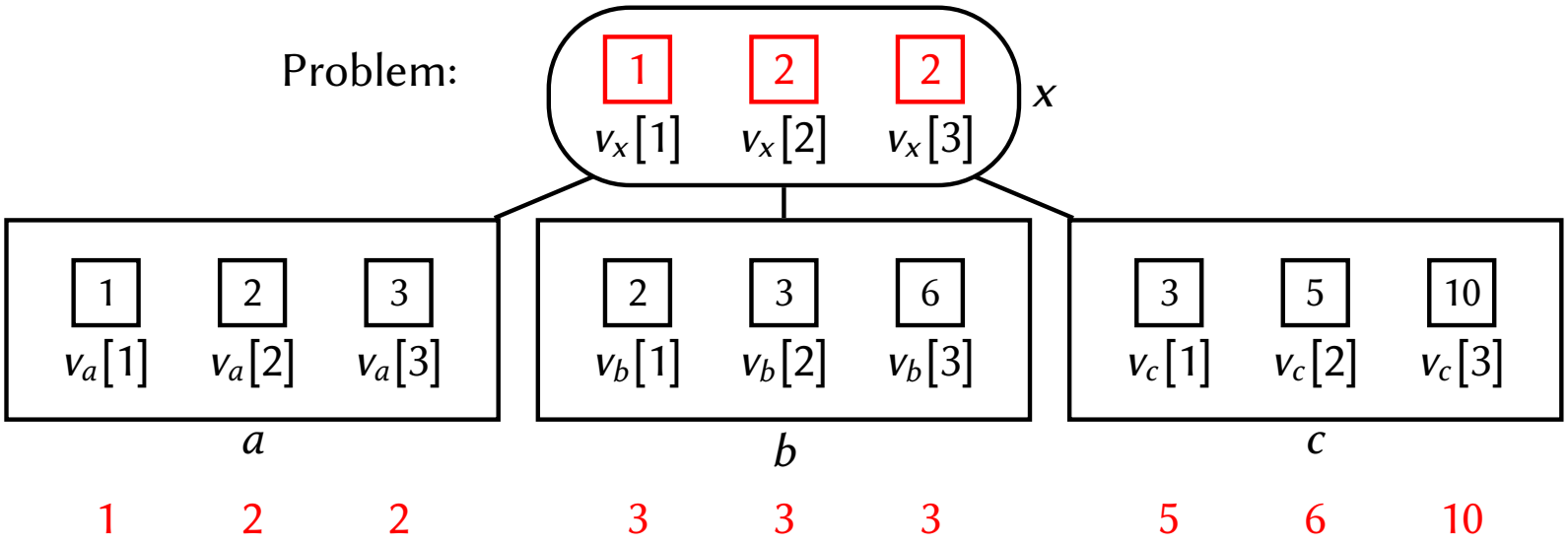
- Find k best rotamer assignments for x;
- Sort all the v values for the child nodes of x;
- Keep k smallest node values

Computing $v[\cdot]$ values for OR nodes



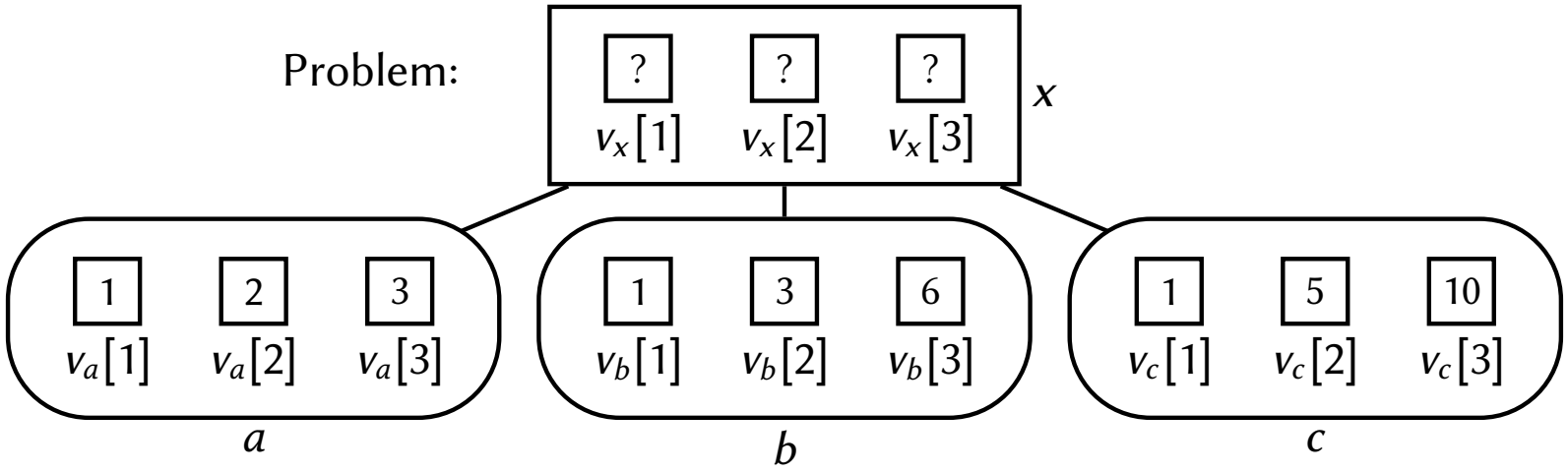
- Find k best rotamer assignments for x;
- Sort all the v values for the child nodes of x;
- Keep k smallest node values

Computing $v[\cdot]$ values for OR nodes



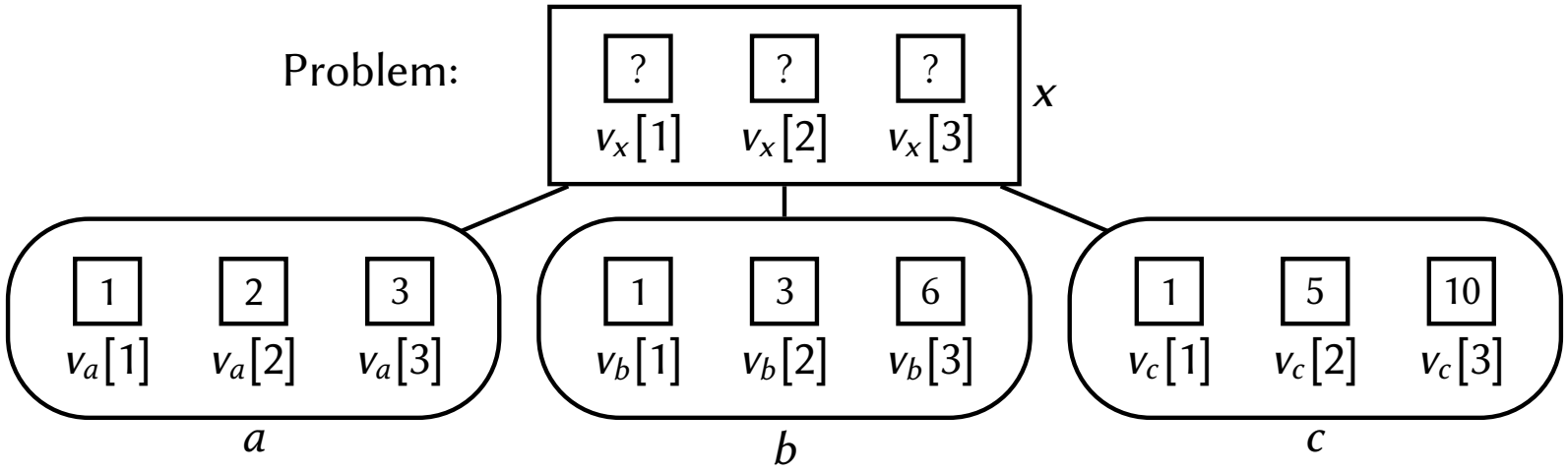
- Find k best rotamer assignments for x;
- Sort all the v values for the child nodes of x;
- Keep k smallest node values

Computing $v[\cdot]$ values for AND nodes



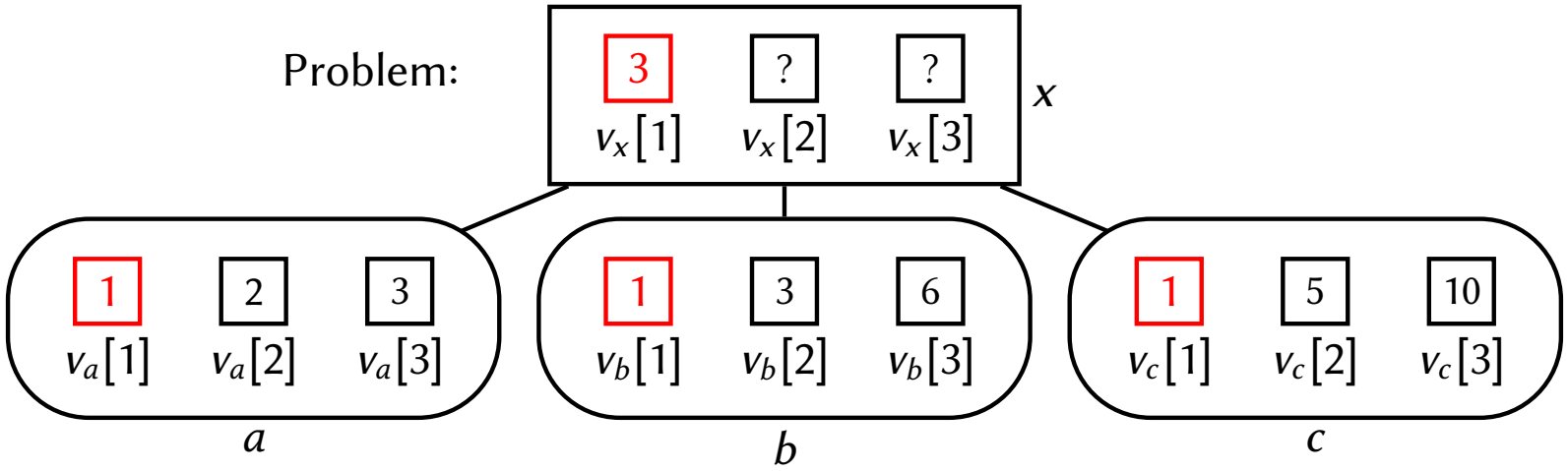
- Want to find k best ways to merge the sub-problems;
- $v_x[i]$ should be the sum of $v_a[i_a] + v_b[i_b] + v_c[i_c]$;
- Obviously $v_x[1] = v_a[1] + v_b[1] + v_c[1]$, hard for $v_x[2]$ and $v_x[3]$;
- Brute force algorithm: $O(k^d)$
- Our algorithm: $O(dk \log(dk))$
  - Develop a new method to reduce the complexity exponentially
  - Use binary heap as the data-structure for efficiency

Computing $v[\cdot]$ values for AND nodes



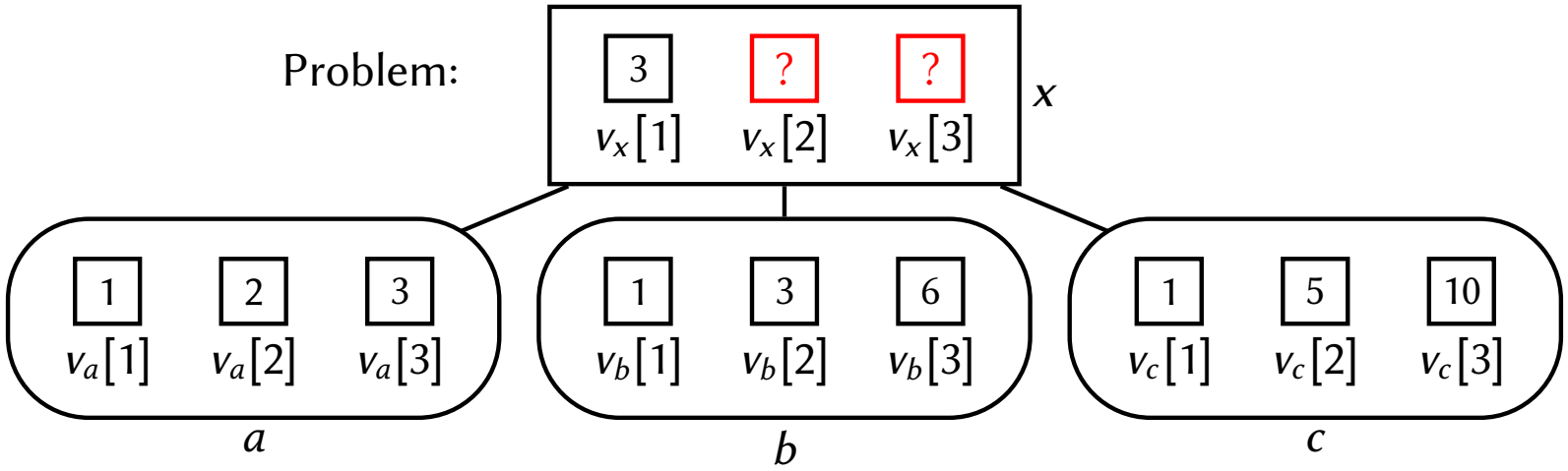
- Want to find k best ways to merge the sub-problems;
- $v_x[i]$ should be the sum of $v_a[i_a] + v_b[i_b] + v_c[i_c]$;
- Obviously $v_x[1] = v_a[1] + v_b[1] + v_c[1]$, hard for $v_x[2]$ and $v_x[3]$;
- Brute force algorithm: $O(k^d)$
- Our algorithm: $O(dk \log(dk))$
  - Develop a new method to reduce the complexity exponentially
  - Use binary heap as the data-structure for efficiency

Computing $v[\cdot]$ values for AND nodes



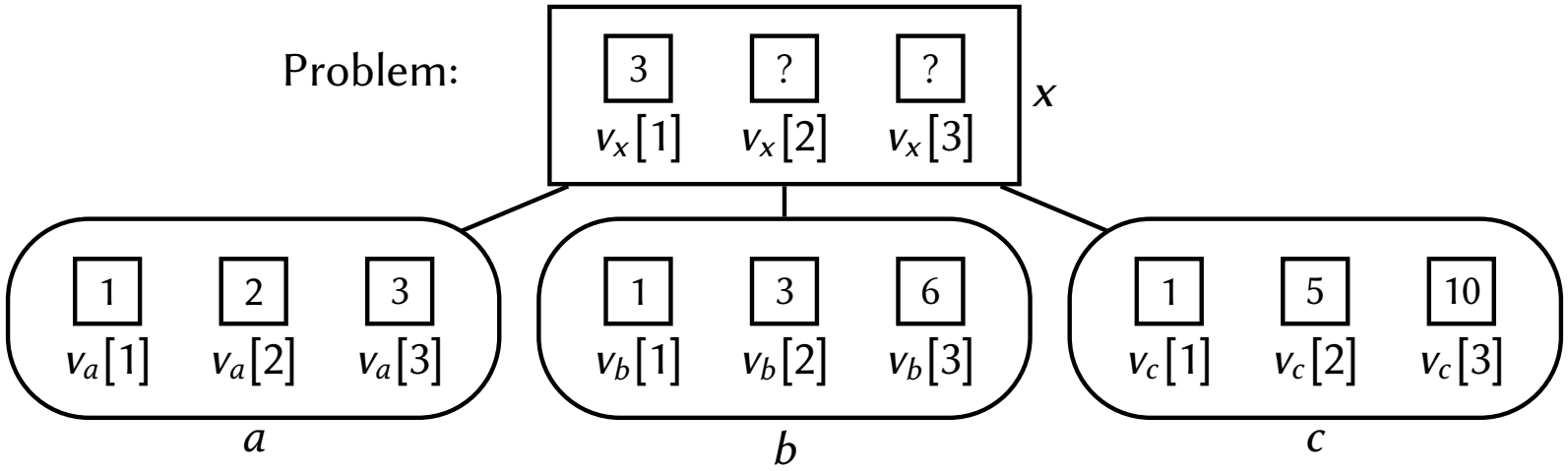
- Want to find k best ways to merge the sub-problems;
- $v_x[i]$ should be the sum of $v_a[i_a] + v_b[i_b] + v_c[i_c]$;
- Obviously $v_x[1] = v_a[1] + v_b[1] + v_c[1]$, hard for $v_x[2]$ and $v_x[3]$;
- Brute force algorithm: $O(k^d)$
- Our algorithm: $O(dk \log(dk))$
  - Develop a new method to reduce the complexity exponentially
  - Use binary heap as the data-structure for efficiency

Computing $v[\cdot]$ values for AND nodes



- Want to find k best ways to merge the sub-problems;
- $v_x[i]$ should be the sum of $v_a[i_a] + v_b[i_b] + v_c[i_c]$;
- Obviously $v_x[1] = v_a[1] + v_b[1] + v_c[1]$, hard for $v_x[2]$ and $v_x[3]$;
- Brute force algorithm: $O(k^d)$
- Our algorithm: $O(dk \log(dk))$
  - Develop a new method to reduce the complexity exponentially
  - Use binary heap as the data-structure for efficiency

Computing $v[\cdot]$ values for AND nodes



- Want to find k best ways to merge the sub-problems;
- $v_x[i]$ should be the sum of $v_a[i_a] + v_b[i_b] + v_c[i_c]$;
- Obviously $v_x[1] = v_a[1] + v_b[1] + v_c[1]$, hard for $v_x[2]$ and $v_x[3]$;
- Brute force algorithm: $O(k^d)$
- Our algorithm: $O(dk \log(dk))$
  - Develop a new method to reduce the complexity exponentially
  - Use binary heap as the data-structure for efficiency

Environment

Experiment

- OSPREY platform from *Donald Lab* (Duke University)
- daoopt AOBB framework from UC Irvine
- Native sequence recovery (core redesign)
- Large rotamer library for better accuracy
- DEE/A* as the comparison

Environment Information

CPU: Intel Xeon™ E5-1620 3.6GHz

Memory: 4G Memory Limitation

Environment

Experiment

- OSPREY platform from *Donald Lab* (Duke University)
- daoopt AOBB framework from UC Irvine
- Native sequence recovery (core redesign)
- Large rotamer library for better accuracy
- DEE/A* as the comparison

Environment Information

CPU: Intel Xeon™ E5-1620 3.6GHz

Memory: 4G Memory Limitation

Dataset

- 23 previously unsolvable protein redesign cases
- 5 previously solvable protein redesign cases
- 6 full protein redesign cases

Results

- solve *21 out of 23 problems* that was previously unsolvable;
- provide a large speedup by several orders of magnitude for previously solvable problems;
- solve all full protein redesign problems while DEE/A* exceeds 4G memory limitation.

Dataset

- 23 previously unsolvable protein redesign cases
- 5 previously solvable protein redesign cases
- 6 full protein redesign cases

Results

- solve *21 out of 23 problems* that was previously unsolvable;
- provide a large speedup by several orders of magnitude for previously solvable problems;
- solve all full protein redesign problems while DEE/A* exceeds 4G memory limitation.

Dataset

- 23 previously unsolvable protein redesign cases
- 5 previously solvable protein redesign cases
- 6 full protein redesign cases

Results

- solve *21 out of 23 problems* that was previously unsolvable;
- provide a large speedup by several orders of magnitude for previously solvable problems;
- solve all full protein redesign problems while DEE/A* exceeds 4G memory limitation.

Running time of AND/OR branch-and-bound algorithm

Running time of our optimized OSPREY

Running time of traditional OSPREY

Conformation Space

PDB	Space size	OSPREY	cOSPREY	AOBB
1IQZ	7.11e+17	1,824,235	40,217	117
2COV	1.14e+10	317	21	1
3FGV	6.44e+12	59,589	5,091	< 1
3DNJ	5.11e+12	7,469	570	3
2FHZ	1.83e+18	3,475,716	70,783	13

Time is measured in millisecond.

Conclusion and Future Work

Our Contribution

- Apply AND/OR branch-and-bound search to protein design
- Design an algorithm to find sub-optimal solutions
- Test AOBB search in native sequence recovery experiment

Results

- Memory efficient
- Speedup by several orders of magnitude comparing to A*/DEE

Future Work

- Using AND/OR best-first search for sub-optimal solutions
- Parallelizing AOBB on a super computer

Acknowledgement

Discussion

- Prof. Alexander Ihler
- Dr. Lars Otten

Funding

- National Basic Research Program of China
- National Natural Science Foundation of China
- China's Youth 1000-Talent Program.

Question and Answer

Thank you!