Solving NP-complete&hard problems: part 2

Petr Kurapov

Fall 2024

Agenda

- Exact solution
  - Brute force search
  - Branch and bound
  - Dynamic programming, memoization
- Approximation (today)
  - Greedy strategy
  - Heuristics, local search
  - Monte-Carlo
  - Meta-heuristics (genetic alg, annealing, ant colony, etc.)

Vangelis Paschos, An overview on polynomial approximation of NP-hard problems. https://hal.archives-ouvertes.fr/hal-00186549/document

Motivation: SAT solvers

- Conflict-driven clause learning (CDCL)
- **Conditioning** – set x_i to a concrete value, and:
  - Remove all clauses containing x_i as satisfied.
  - Remove all $\overline{x_i}$ terms as true.
  - $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3})$
  - $\varphi \wedge x_1 = (x_2 \vee x_3)$
- **Resolution** – replace 2 clauses with one "resolvent":
  - $\varphi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4)(x_2 \vee x_4 \vee x_5)$
  - $\varphi = (x_1 \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee x_4 \vee x_5)$

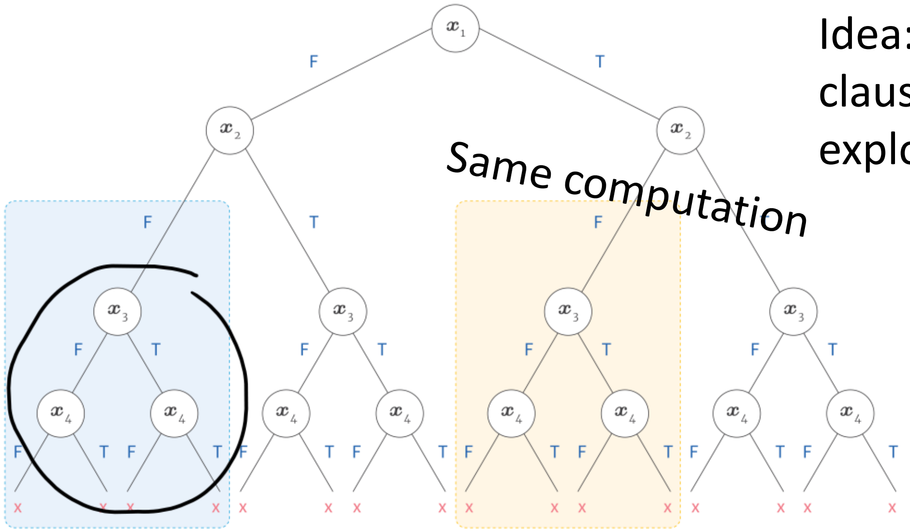
When applied to a single term (e.g., $(x_1 \vee \overline{x_3} \vee x_4) \wedge x_4$) – *unit resolution* – can produce more unit clauses. Recursive application = *unit propagation*.

SAT solvers

- 2-SAT is solved polynomially, using the resolution process & unit propagation.
  - Choosing the first value at random triggers a resolution chain.
- Directional resolution for 3-SAT – similar idea:
  - Sort clauses into bins (e.g., bin 1 contains all clauses having x_1 in them).
  - Apply resolutions with some variable ordering.
  - Each new level generates 2(K -1) new clauses – not very efficient.

SAT solvers: DPLL & CDCL

- Davis–Putnam–Logemann-Loveland (DPLL)
  - Some computations in the tree are redundant.
  - Embed unit propagation into tree search – each time we make a decision, apply resolution.
  - $\varphi \wedge x_1 = (x_2 \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_2 \vee x_4) \wedge (\overline{x_2} \vee x_3)$
  - Next step will trigger unit propagation

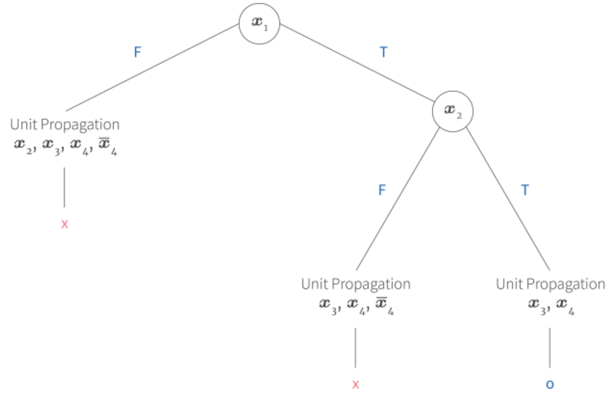


Same computation

redundant

Idea: add a clause to avoid exploration

Examples from: https://rbcborealis.com/research-blogs/tutorial-10-sat-solvers-ii-algorithms/



$$\varphi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_2} \vee x_3)$$

A machine program for theorem-proving https://dl.acm.org/doi/10.1145/368273.368557

Conversion to CNF

- Using de-Morgan and distributive laws:
  - Simple and correct
  - Do not introduce new variables
  - BUT: may lead to exponentially large formula
- Create an equisatisfiable formula
- Sudoku transformation example

Approach

- Decide on what to model with clauses (natural for sudoku – model positions as variables; instead – each combination and value as variable)
- Example

Greedy algorithms

- Much simpler than dynamic programming:
  - Choose best option at each step
- Can give an optimal solution
  - Optimal substructures
- Covered by another course in great detail

Approximation

- Polynomial time approximation algorithms to get a feasible (guaranteed to be close to optimal, depending on a problem) solution
- Approximation coefficient $\frac{C}{C^*} \leq \rho(n)$ for minimalization problem

Approximation: vertex cover example

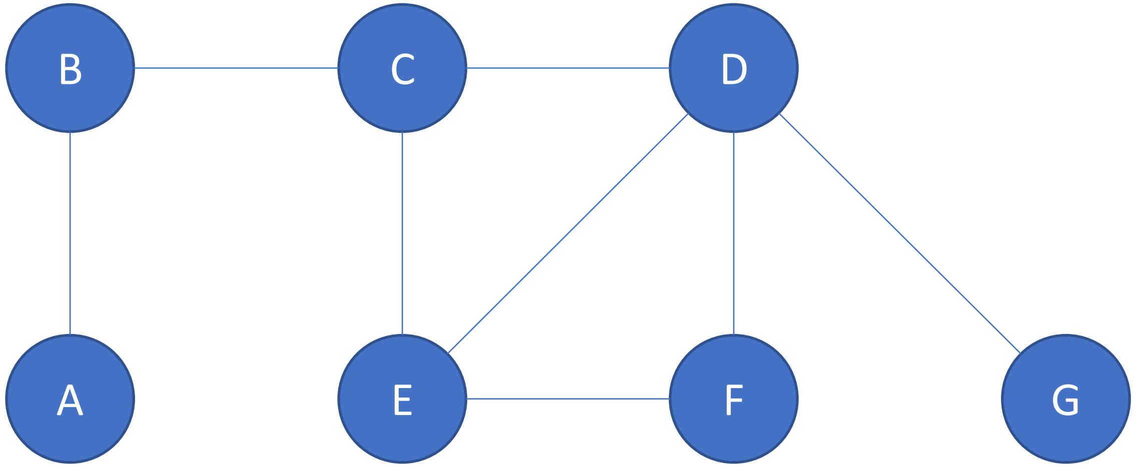
- Vertex cover: $G = (V, E), V_{cov} \subseteq V$: $\forall (u, v) \in E, u \in V_{cov}\ or\ v \in V_{cov}$
- Optimal vertex cover – minimal cardinality

Zongjie Ma, et al., Random Walk in Large Real-World Graphs for Finding Smaller Vertex Cover. https://core.ac.uk/download/pdf/143897847.pdf

Approximation: vertex cover example

- Vertex cover: $G = (V, E), V_{cov} \subseteq V$: $\forall(u, v) \in E, u \in V_{cov}$ or $v \in V_{cov}$
- Optimal vertex cover – minimal cardinality

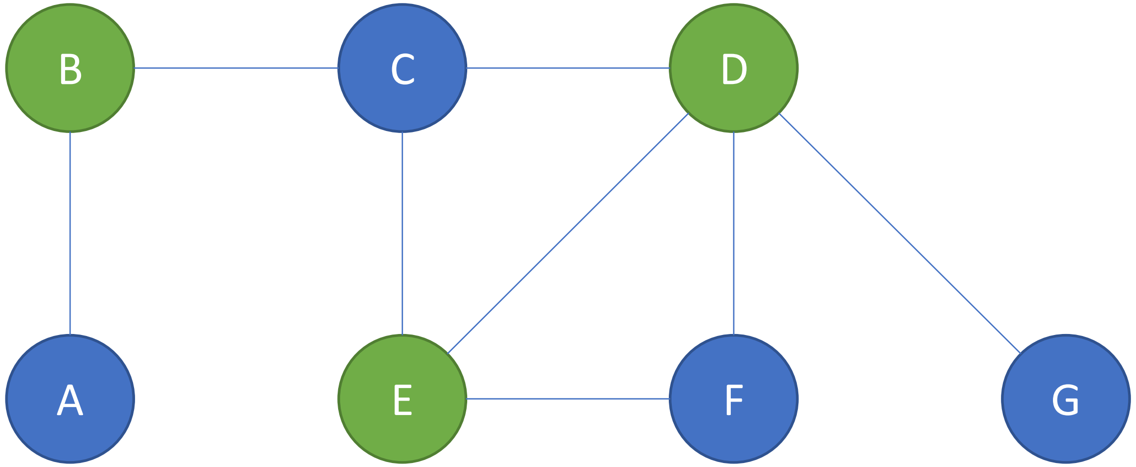
Optimal solution?



Approximation: vertex cover example

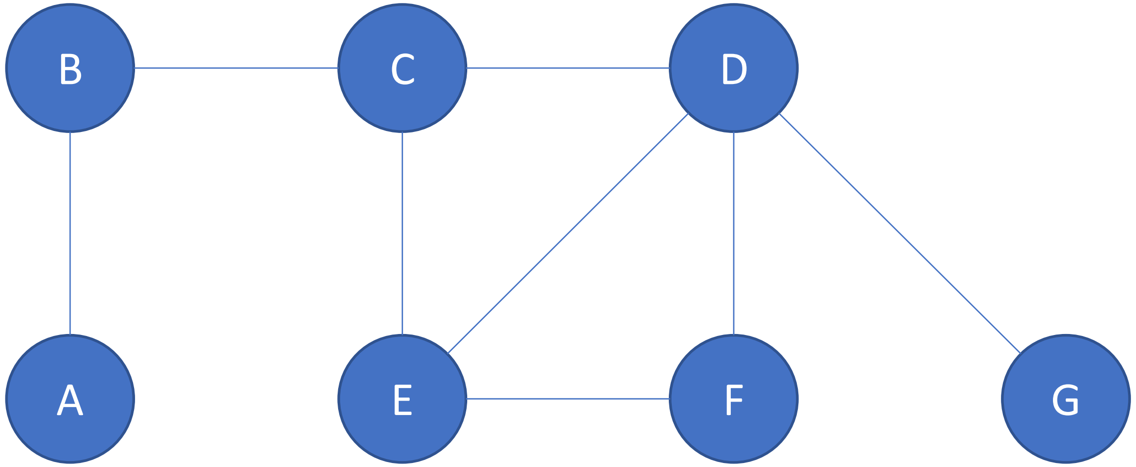
- Vertex cover: $G = (V, E), V_{cov} \subseteq V$: $\forall (u, v) \in E, u \in V_{cov}$ or $v \in V_{cov}$
- Optimal vertex cover – minimal cardinality

Optimal solution



Approximation: vertex cover example

- $G = (V, E)$
- $COV = \emptyset$
- $E_{left} = E$



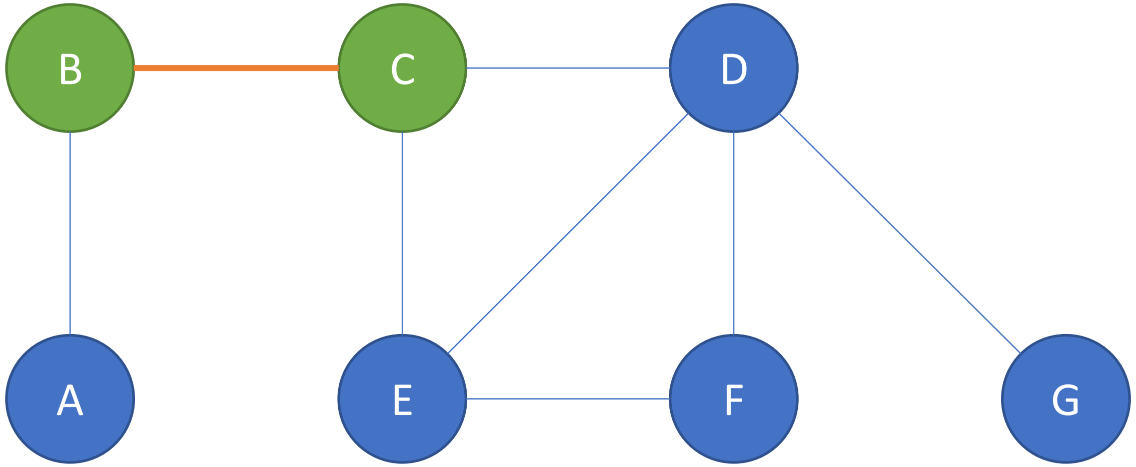
Example source: [1]

Go through edges in E_{left} adding to COV and removing incident edges

Approximation: vertex cover example

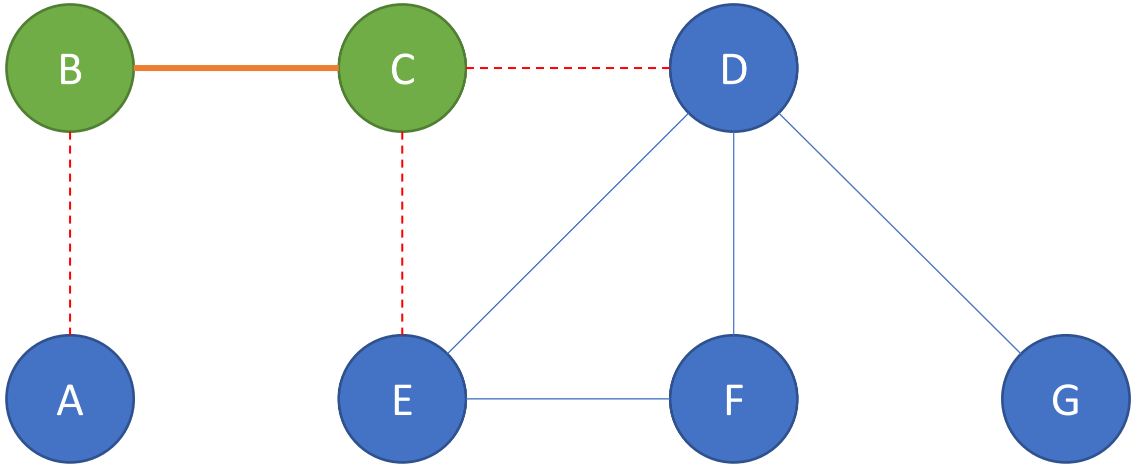
- $G = (V, E)$
- $COV = \{B, C\}$
- $E_{left} = E$

Go through edges in E_{left} adding to COV and removing incident edges



Approximation: vertex cover example

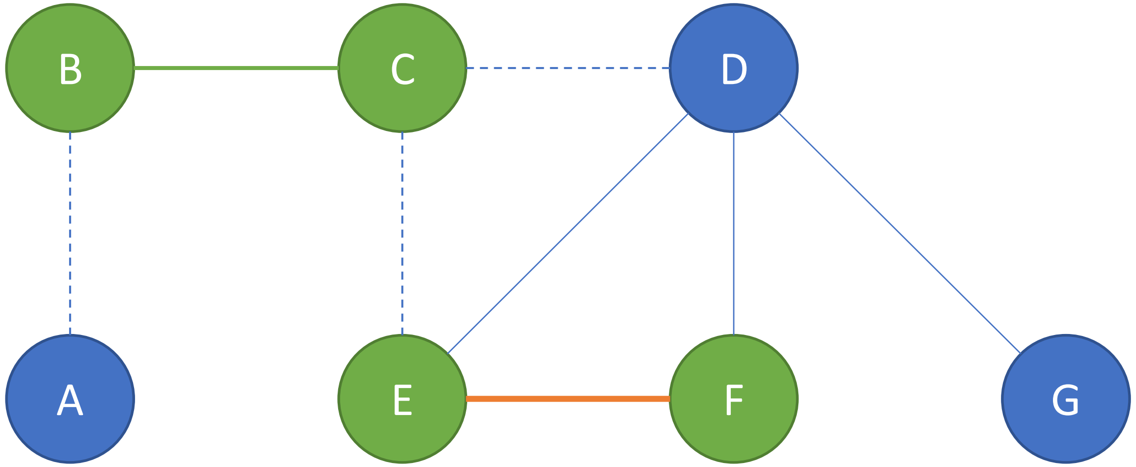
- $G = (V, E)$
- $COV = \{B, C\}$
- $E_{left} = E / \{A, B\}, \{C, E\}, \{C, D\}$



Go through edges in E_{left} adding to COV and removing incident edges

Approximation: vertex cover example

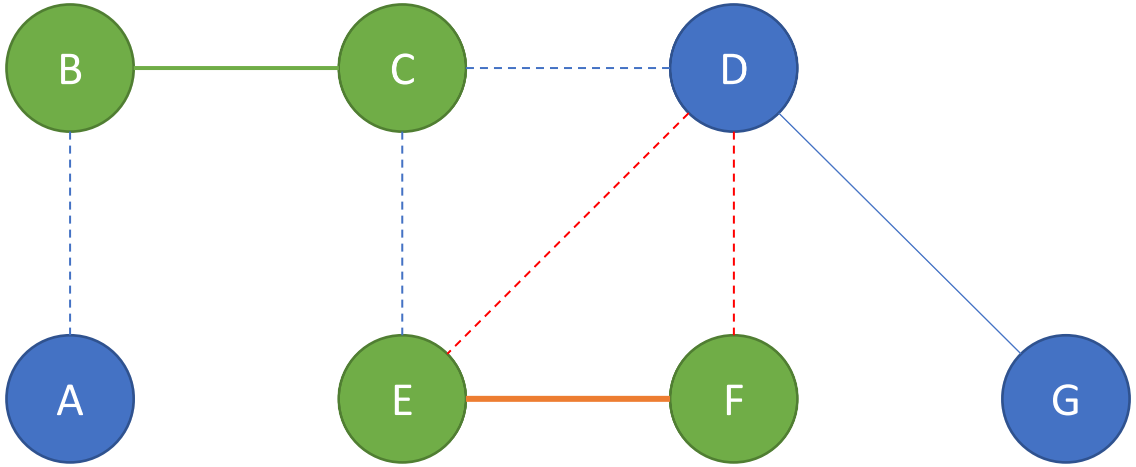
- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}$
- $E_{left} = E_{left}$



Go through edges in E_{left} adding to COV and removing incident edges

Approximation: vertex cover example

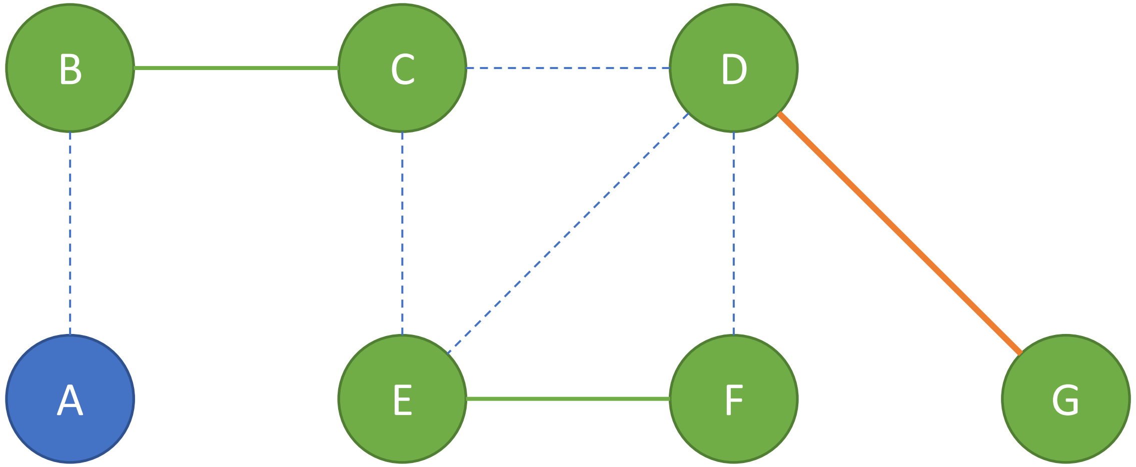
- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}$
- $E_{left} = E_{left} / \{E, D\}, \{F, D\}$



Go through edges in E_{left} adding to COV and removing incident edges

Approximation: vertex cover example

- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}, \{D, G\}$
- $E_{left} = E_{left} / \{D, G\}$



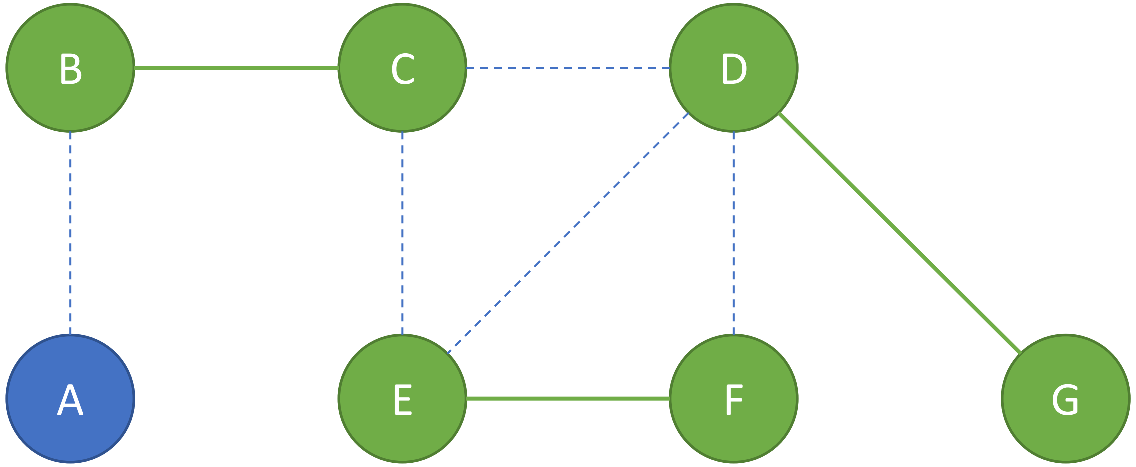
Go through edges in E_{left} adding to COV and removing incident edges

Approximation: vertex cover example

- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}, \{D, G\}$
- $E_{left} = E_{left} / \{D, G\}$

Coverage – 6 vertices

Go through edges in E_{left} adding to COV and removing incident edges



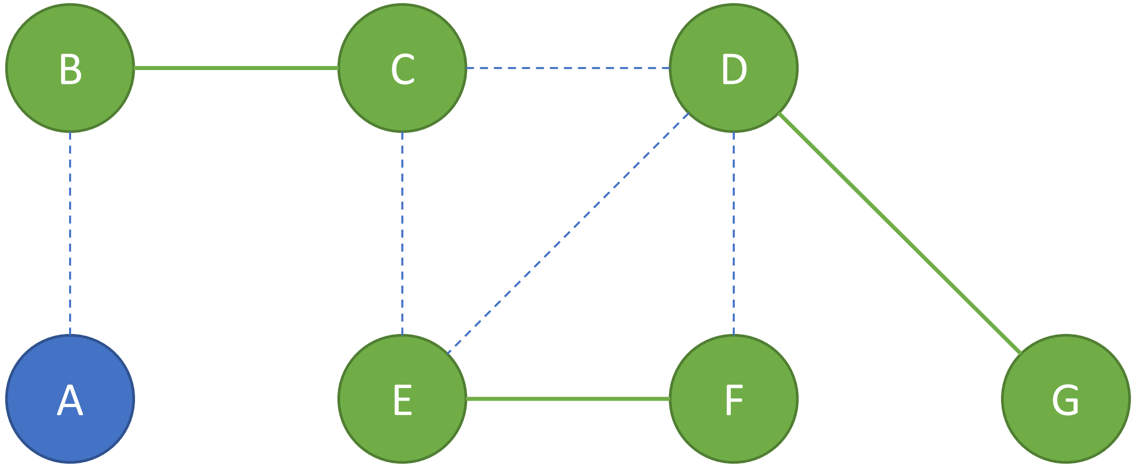
Approximation: vertex cover example

- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}, \{D, G\}$
- $E_{left} = E_{left} / \{D, G\}$

Coverage – 6 vertices

Complexity?

Go through edges in E_{left} adding to COV and removing incident edges

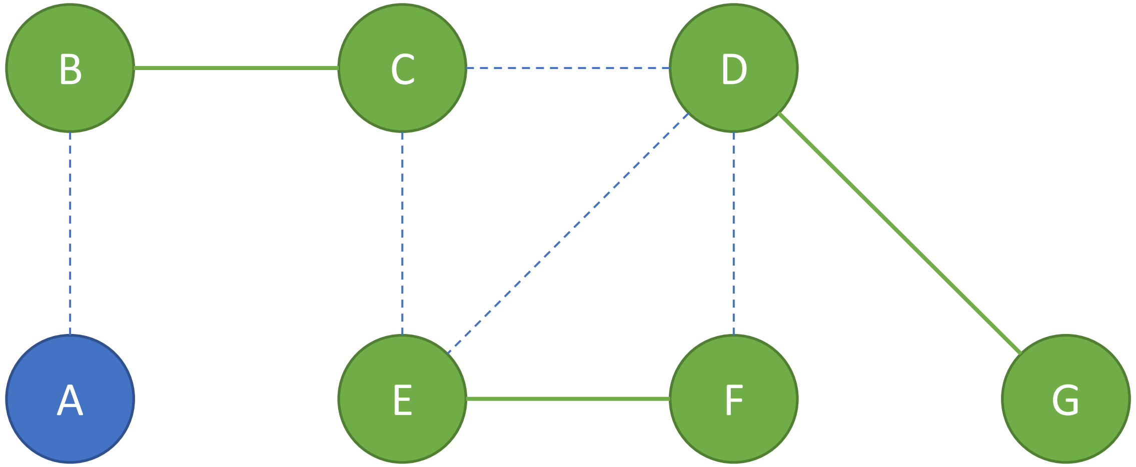


Approximation: vertex cover example

- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}, \{D, G\}$
- $E_{left} = E_{left} / \{D, G\}$

Coverage – 6 vertices

Complexity $\boldsymbol{O(|V| + |E|)^*}$



*using adjacency list

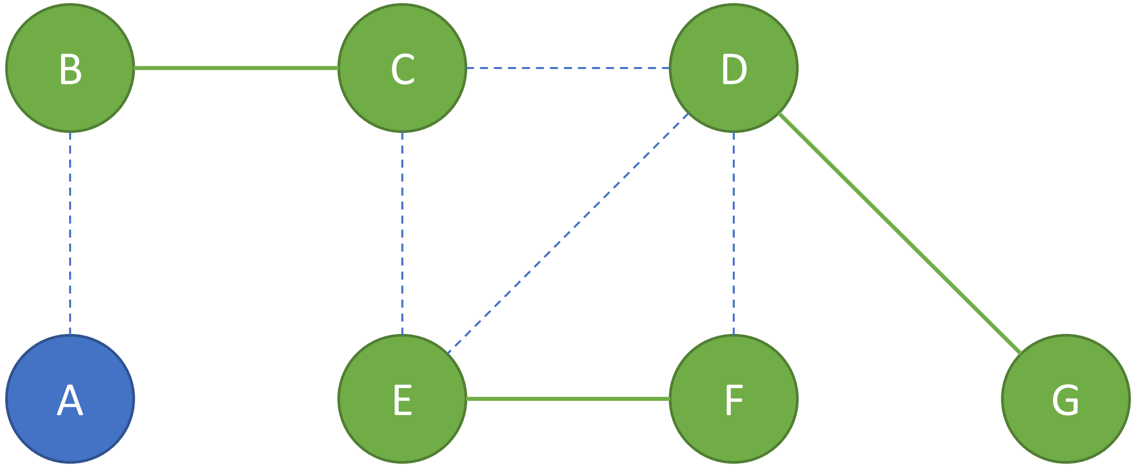
Approximation: vertex cover example

- $G = (V, E)$
- $COV = \{B, C\}, \{E, F\}, \{D, G\}$
- $E_{left} = E_{left} / \{D, G\}$

Coverage – 6 vertices

Complexity $O(|V| + |E|)^*$

How bad the solution is?



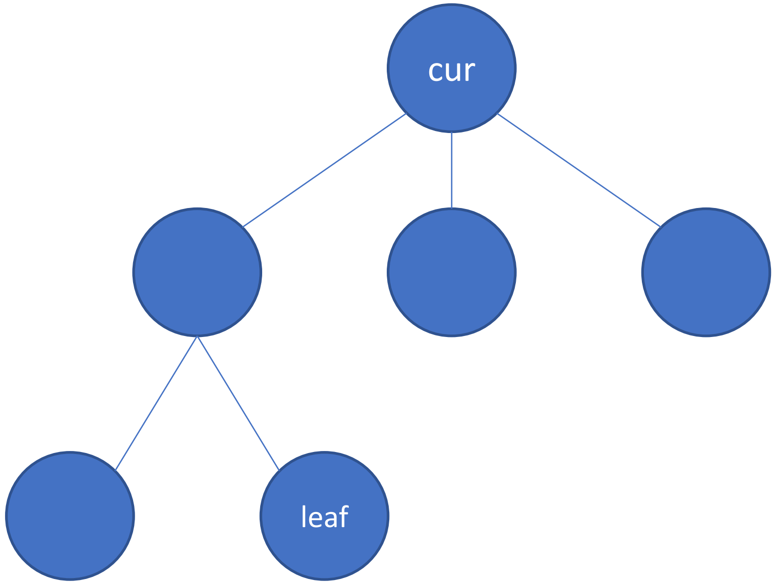
- Go through edges in E_{left} adding to COV and removing incident edges

*using adjacency list

Monte-Carlo methods

- Idea: random sampling
- Monte-Carlo tree search (MCTS) – best first search:
  - Selection – traverse tree to leaf node using selection strategy
  - Expansion – store one or more children to a leaf node
  - Simulation – play the rest of the game to get a result
  - Backpropagation – propagate the result upwards

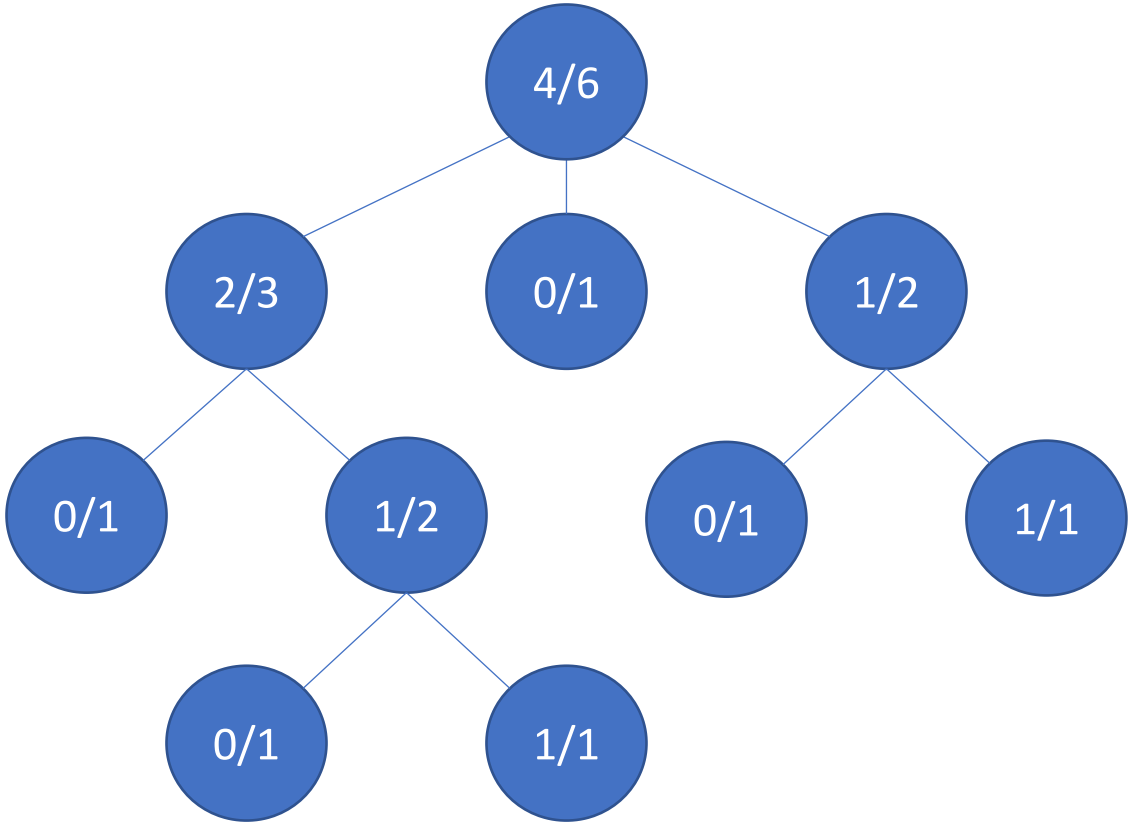
- Search tree
  - Each node represents a state
  - Current value v_i + visit count n_i



PROGRESSIVE STRATEGIES FOR MONTE-CARLO TREE SEARCH. https://dke.maastrichtuniversity.nl/m.winands/documents/pMCTS.pdf

Monte-Carlo methods

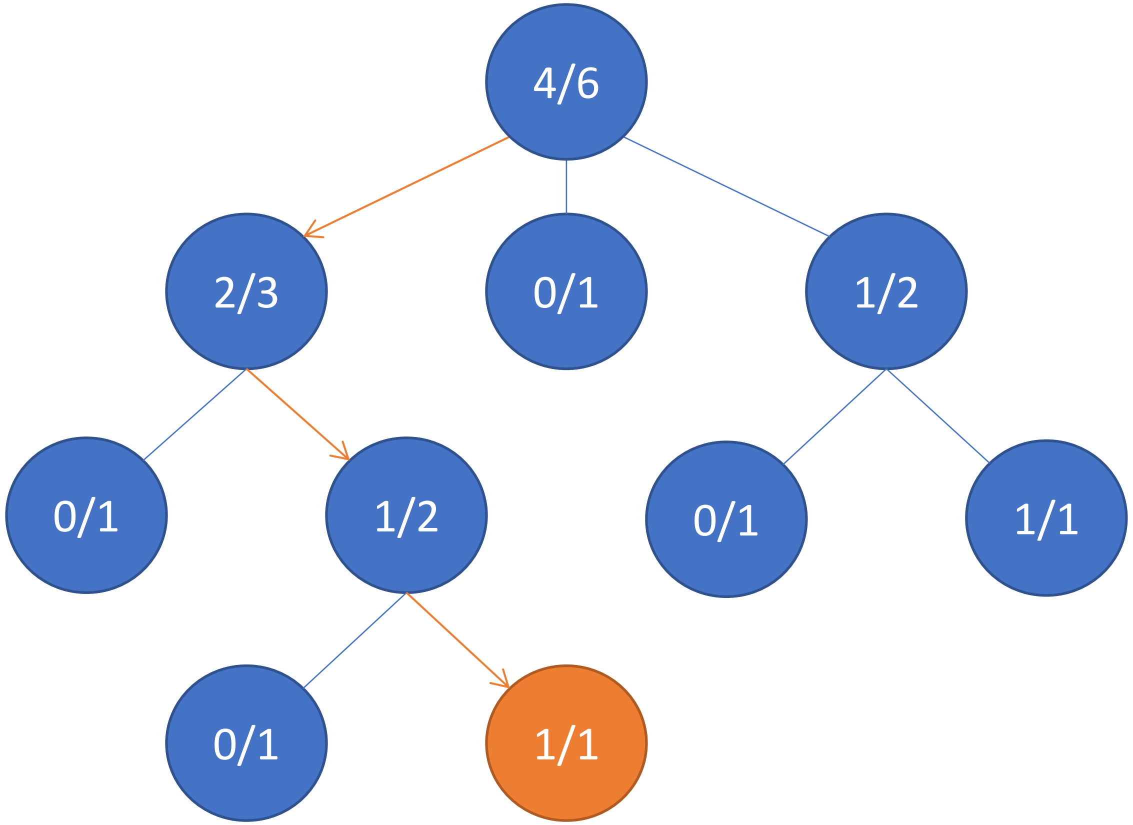
- Monte-Carlo tree search (MCTS) – best first search:
  - **Selection – traverse tree to leaf node using selection strategy**
  - Expansion – store one or more children to a leaf node
  - Simulation – play the rest of the game to get a result
  - Backpropagation – propagate the result upwards



Example source: [5]

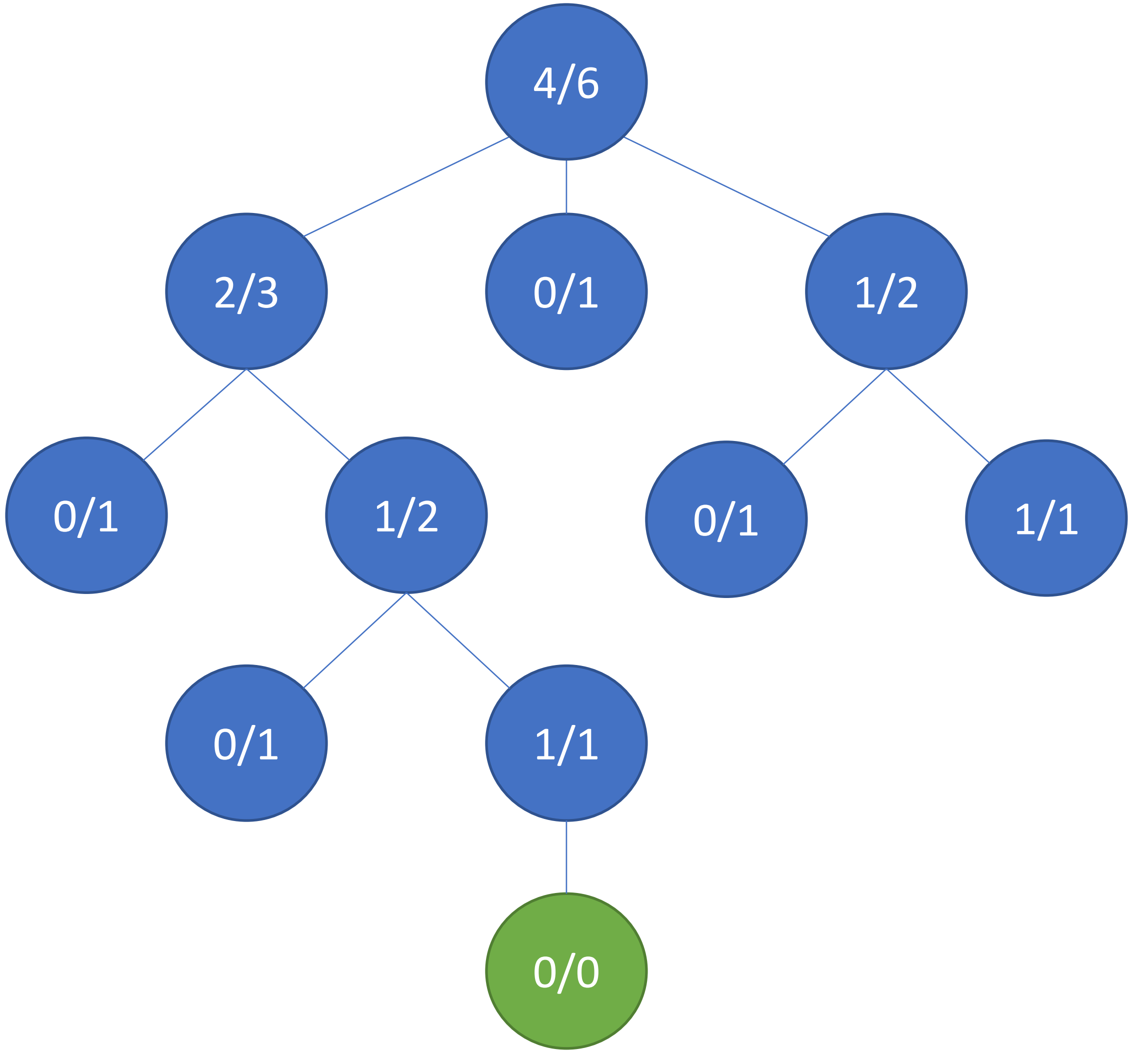
Monte-Carlo methods

- Monte-Carlo tree search (MCTS) – best first search:
  - **Selection – traverse tree to leaf node using selection strategy**
  - Expansion – store one or more children to a leaf node
  - Simulation – play the rest of the game to get a result
  - Backpropagation – propagate the result upwards



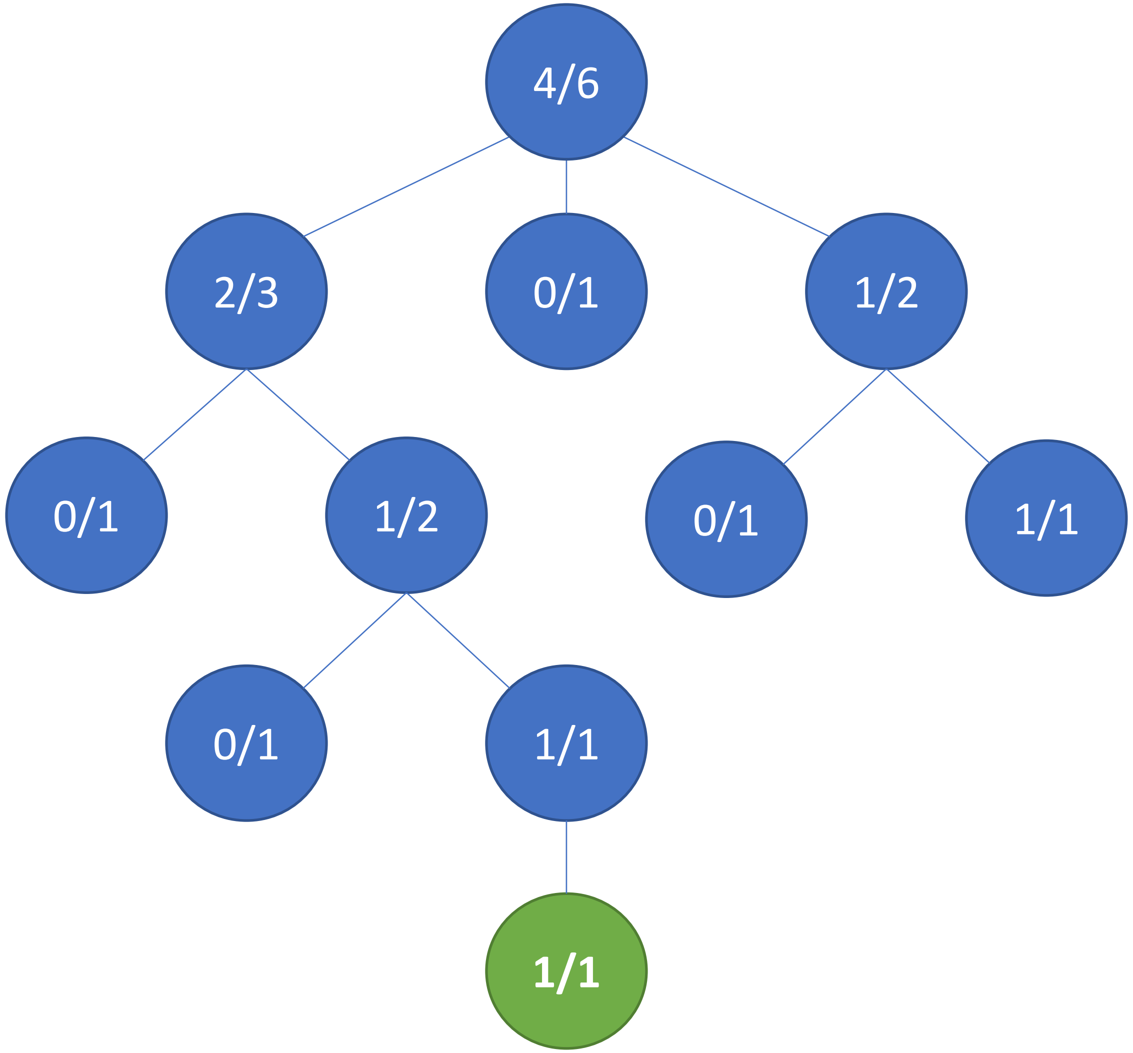
Monte-Carlo methods

- Monte-Carlo tree search (MCTS) – best first search:
  - Selection – traverse tree to leaf node using selection strategy
  - **Expansion – store one or more children to a leaf node**
  - Simulation – play the rest of the game to get a result
  - Backpropagation – propagate the result upwards



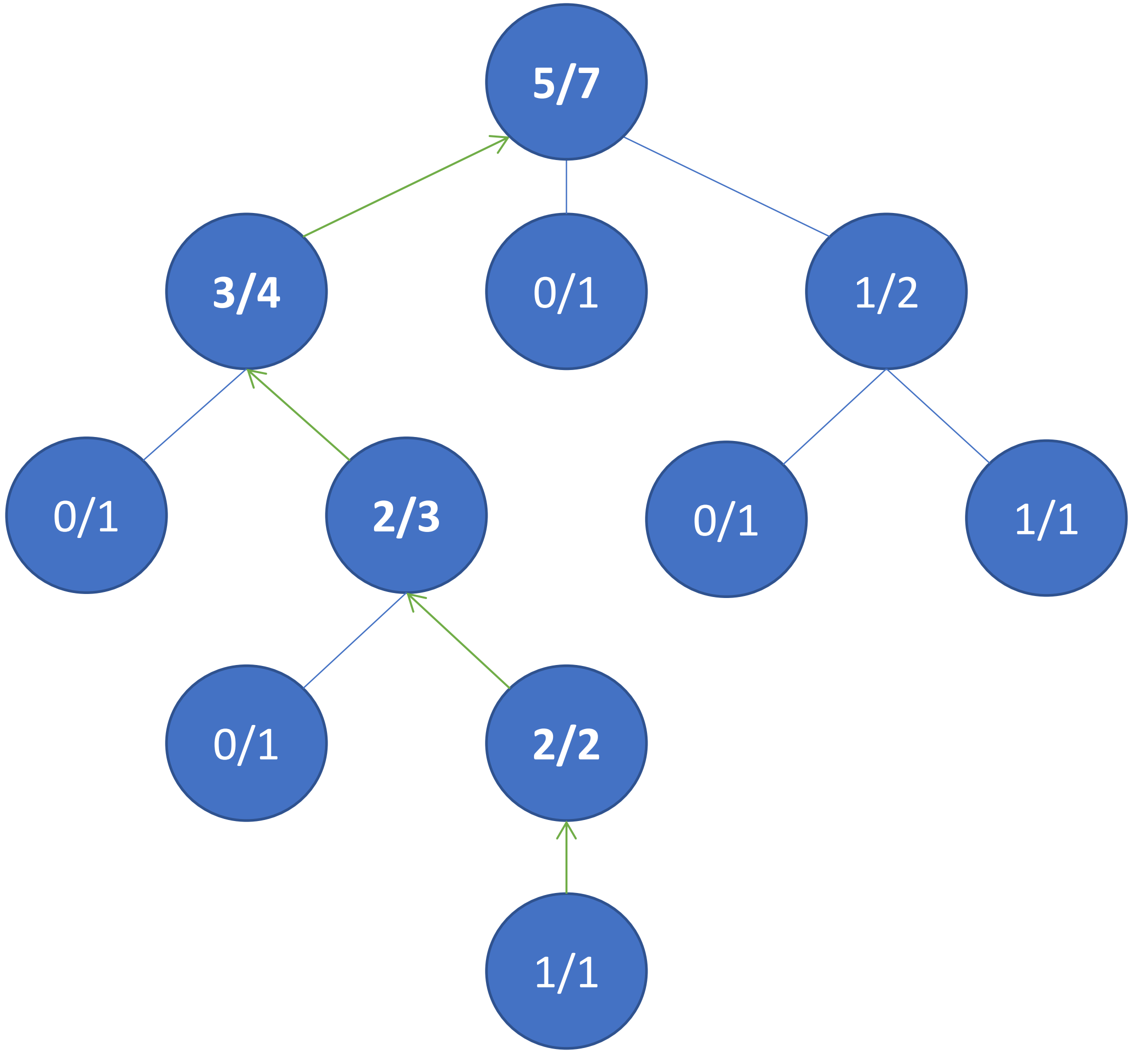
Monte-Carlo methods

- Monte-Carlo tree search (MCTS) – best first search:
  - Selection – traverse tree to leaf node using selection strategy
  - Expansion – store one or more children to a leaf node
  - **Simulation – play the rest of the game to get a result**
  - Backpropagation – propagate the result upwards



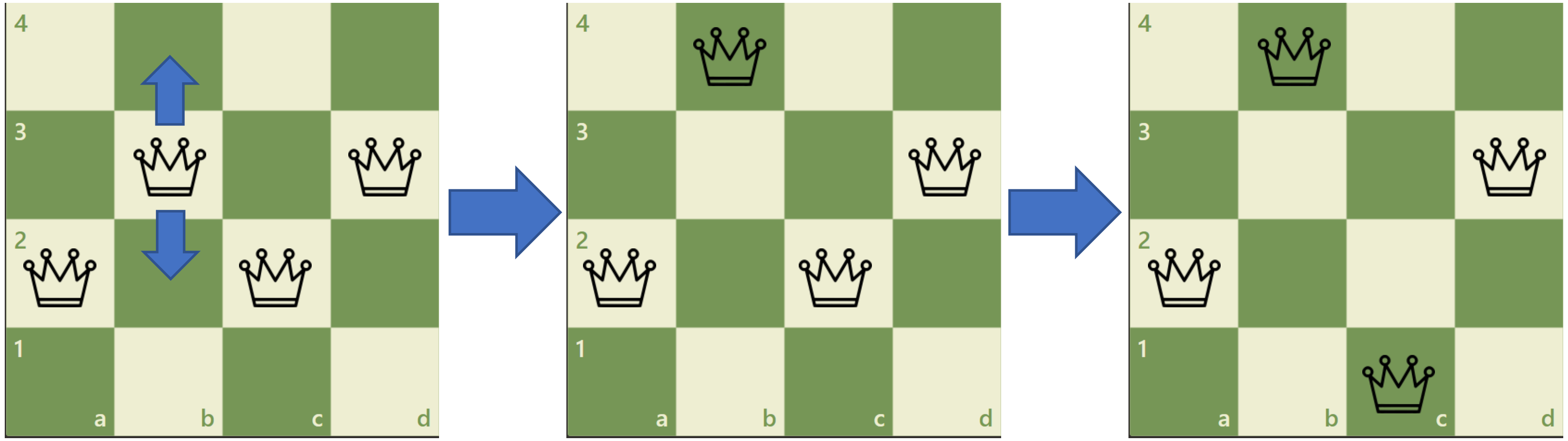
Monte-Carlo methods

- Monte-Carlo tree search (MCTS) – best first search:
  - Selection – traverse tree to leaf node using selection strategy
  - Expansion – store one or more children to a leaf node
  - Simulation – play the rest of the game to get a result
  - **Backpropagation – propagate the result upwards**



Local search

- Use a candidate solution as a starting point
- Iteratively move to neighbor solution trying to improve the result



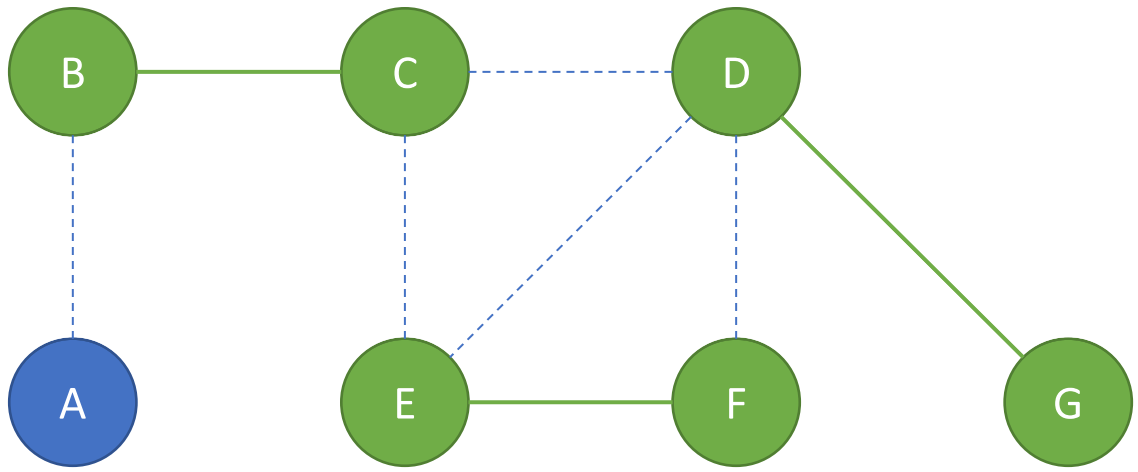
5 conflicts

2 conflicts

0 conflicts

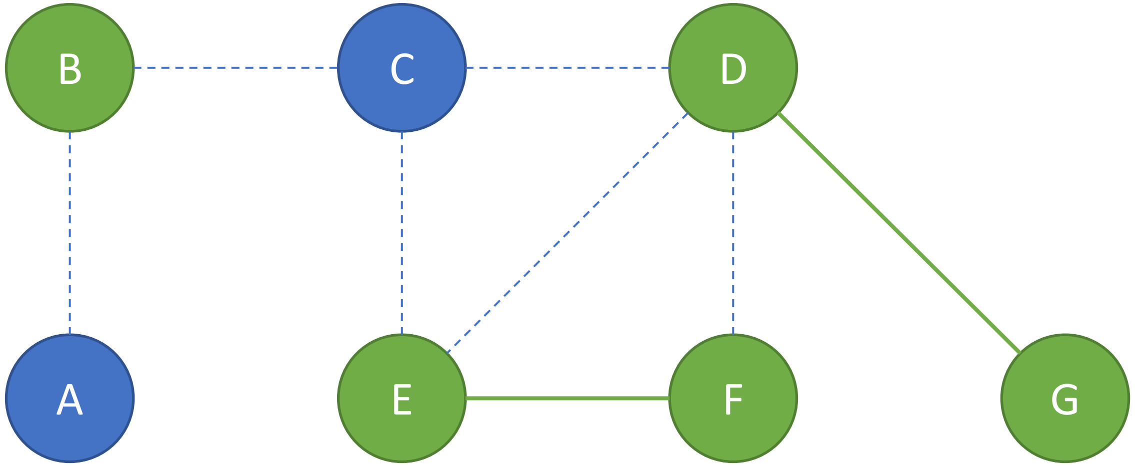
Local search

- Use a candidate solution as a starting point
- Iteratively move to neighbor solution trying to improve the result
- B-C, E-F, D-G



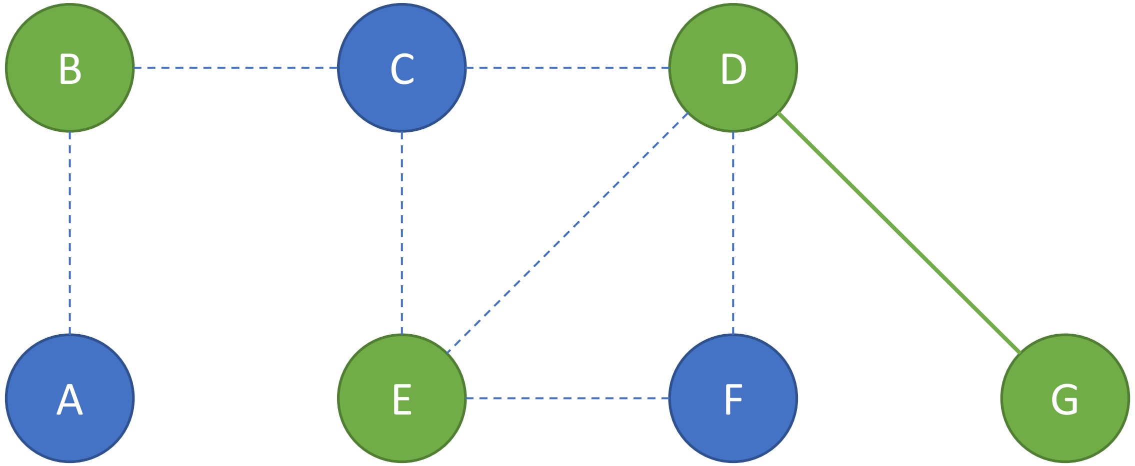
Local search

- Use a candidate solution as a starting point
- Iteratively move to neighbor solution trying to improve the result
- **B-C**, E-F, D-G



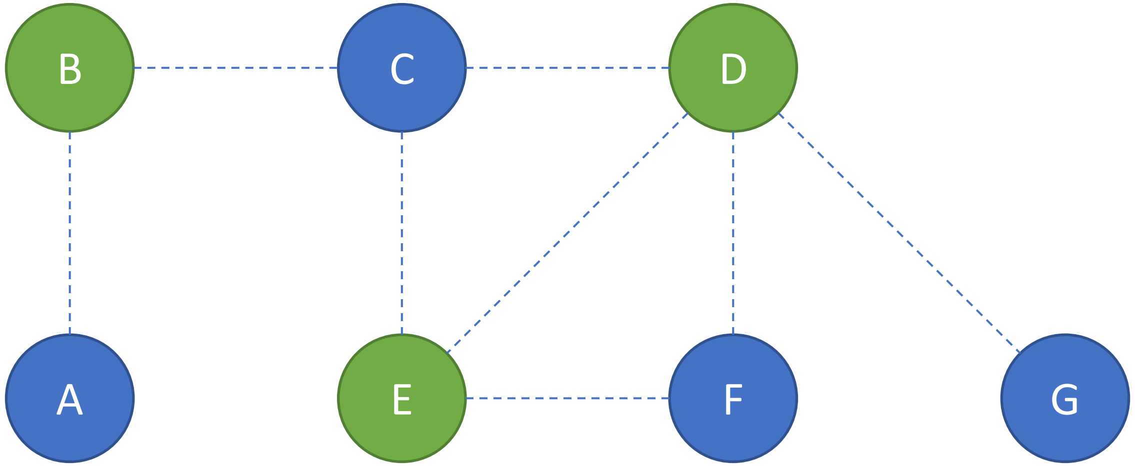
Local search

- Use a candidate solution as a starting point
- Iteratively move to neighbor solution trying to improve the result
- B-C, **E-F**, D-G



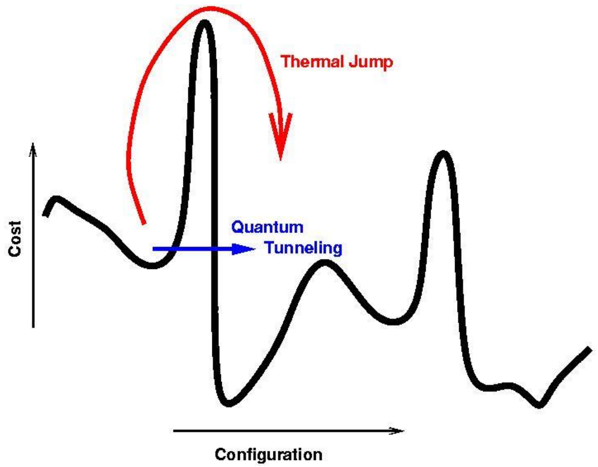
Local search

- Use a candidate solution as a starting point
- Iteratively move to neighbor solution trying to improve the result
- B-C, E-F, **D-G**



Meta heuristics: Annealing

- Optimization problem: $F(\overline{x}) \to min, x = (x_1, \dots x_n)$, i.e. TSP
- With a defined permutation operator using probability:
- $P\left(\overline{x^*} \to \overline{x_{i+1}} \mid \overline{x_i}\right) = \begin{cases} 1, F\left(\overline{x^*}\right) < F(\overline{x_i}) \\ e^{\frac{F\left(\overline{x^*}\right) - F(\overline{x_i})}{T}} \end{cases}, T \to 0$
- Sharp peaks – quantum annealing simulation:
  - $e^{-\frac{\sqrt{\Delta}\omega}{\Gamma}}, \omega(width) \ll \sqrt{\Delta}(height)$



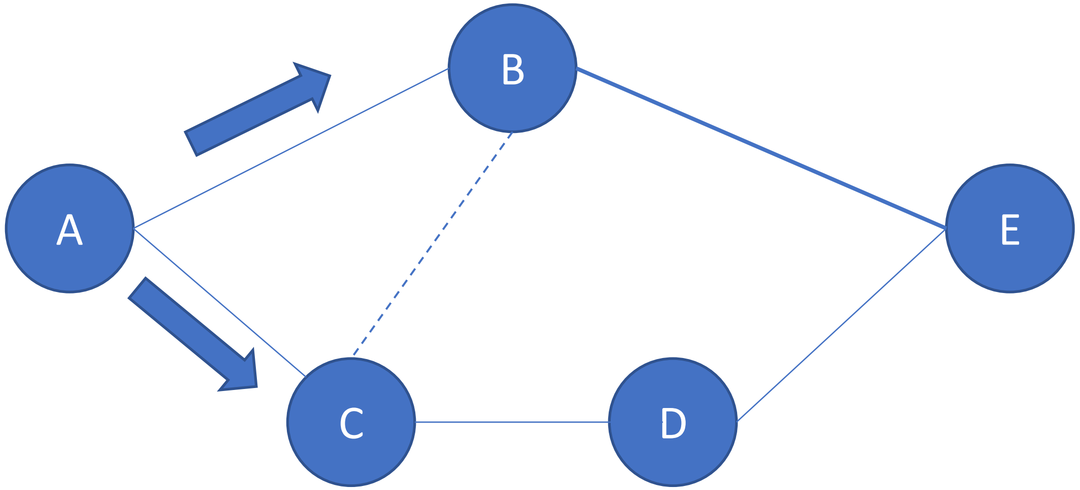
Example source: [wiki]

Meta heuristics: Genetic algorithm

- Initial population – solutions
- Fitness function – determine how good the solution is
- Selection – select fittest to pass their “genes” to the next population
- Crossover – exchange parent properties (genes) to generate offspring
- Mutation – change properties at random
- Example

Meta heuristics: Ant colony

- Solving a problem finding a “good” path through a graph
- Coordinated effort of multiple agents
- Use pheromone to guide search



While:
  Construct ant solutions
  Apply local search (optional)
  Update pheromones

M.Dorigo, et al. Ant colony optimization. https://ieeexplore.ieee.org/abstract/document/4129846

Resources

- [1] Introduction to Algorithms, Thomas H. Cormen, chapters 16, 35
- [2] Random Walk in Large Real-World Graphs for Finding Smaller Vertex Cover (pub)
- [3] Bernhard Reus. Limits of Computation: From a Programming Perspective (link)
- [4] Approximation Algorithms for NP-Hard Problems https://www.utdallas.edu/~dzdu/cs6363/unit5.pdf
- [5] MCTS article
- [6] Handbook of Satisfiability (link)

BACKUP