Übungen zu Algorithmen und Programmentwicklung für die Biologische Chemie

Michael T. Wolfinger

Research group for Bioinformatics and Computational Biology
Institute for Theoretical Chemistry
Währingerstrasse 17, 1090 Vienna, Austria

Sommersemester 2025

Optimization

Optimization problems

Definition: A ***combinatorial*** **optimization problem** consists of

- an arbitrary ***finite*** set X: the **solution space**
- a **cost function** $f : X \to \mathbb{R}$,
  which assigns the cost $f(x)$ to each $x \in X$.

Task: Find a solution.

Def.: A **solution** of the COP (X, f) is an $x_0 \in X$ with minimal cost, i.e.

$$f(x_0) = \min_{x \in X} f(x)$$

Optimization problems: Examples

Example 1:

- $X = \{1, \ldots, 100\}$
- $f : X \to \mathbb{R}$, $x \mapsto |x - 42|$

optimal solution is 42 at minimum cost $f(42) = 0$.

Example 2: Travelling Salesperson Problem.

- Input: distances between n cities.
- The solution set X consists of all tours.
  (a tour visits every city exactly once and returns to the start city)
- Cost function f yields the length of the tour.

Example disctances:

	A	B	C	D
A	-	2	4	2
B	2	-	3	3
C	4	3	-	1
D	2	3	1	-

Brute force—Generate and minimize

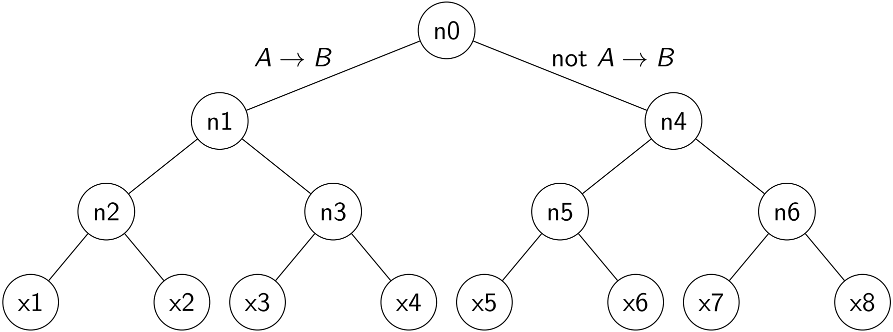
Algorithmic "strategy" that works without any knowledge of the problem's structure

$best \leftarrow +\infty$;
$x_0 = \varnothing$;
for $x \in X$ **do**
    $cost \leftarrow f(x)$;
    if $cost < best$ **then**
        $best \leftarrow cost$;
        $x_0 \leftarrow x$;
    end
end

Branch and Bound

Systematic enumeration of the solution space

- **Enumerate** solutions by systematically deciding on features of the solutions $\Rightarrow$ search tree
  Example: Salesperson travels **A**$\rightarrow$**B** vs. **not A**$\rightarrow$**B**.
- Each **leaf** represents a **solution**
- Each **inner node** represents a **partial solution**
- Each **subtree** represents a sub-space of the **solution space**



Exploration of the solution space

Depth First Search

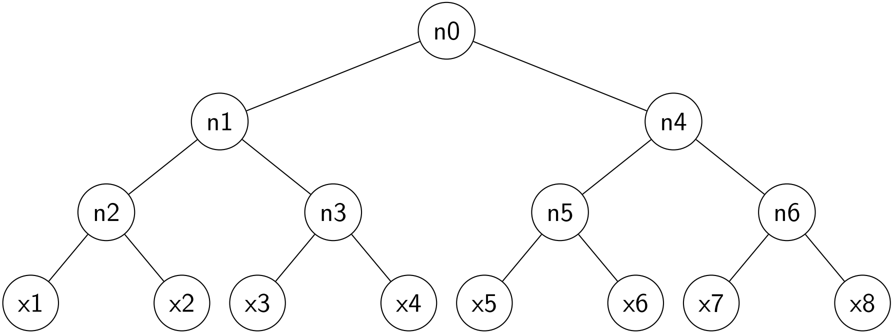
At inner node, generate all children and push them onto a **stack**.

Continue with node on top of stack (**LIFO**).

Breadth First Search

At inner node, generate all children and add them to a **queue**.

Continue with first node in the queue (**FIFO**).



Branch and Bound optimization

Idea:

- systematically search through the search tree, depth-first
- skip subtrees that cannot improve the currently best solution

Ingredients:

- **Enumeration strategy** (how to generate search tree)
- **Lower bound cost function** g on partial solutions at inner nodes. Bounds cost of all possible solutions of the subtree.

For example: TSP. g yields the length of the beginning of a tour: a lower bound for the possible total length.

Branch and Bound, basic algorithm

```
best ← +∞; INIT(S);
PUSH(S,X);
while not EMPTY(S) do
    A=POP(S);
    if A is consistent then
        if g(A) < best then
            if A is a solution then
                best ← g(A);
            else
                (B, C)=Split(A);
                PUSH(S,B) ;
                PUSH(S,C) ;
            end
        end
    end
end
```

Branch and Bound, recursive formulation

```
bound = infty

def BAB( A ):
    if A is consistent and g(A) < bound :
        if A is a solution :
            bound = f( A )
        else:
            (B,C) = Split( A )
            BAB( B )
            BAB( C )

BAB( S )
```

In general, recursion...

means that a function calls itself (directly or indirectly)

```
fac :: Int -> Int
fac 0 = 1
fac n = n * fac (n-1)

Example:
 (fac 4) --> 24
  | ^
  | | 24
  V |
 (4 * (fac 3))
       | ^
       | | 6
       V |
      (3 * (fac 2)
            | ^
            | | 2
            V |
           (2 * (fac 1))
                 | ^
                 | | 1
                 V |
                (1 * (fac 0))
                      | ^
                      | | 1
                      V |
                       1
```

Assignment 2: Optimal Distribution of Authorities

Go to:

`https://github.com/TBIAPBC/APBC2025/tree/master/A2`

This assignment is due on 28 April 10:00 AM

(GitHub pull request must be placed by this time at latest)

Happy Hacking!

Next meeting: 29 April 2025