

Introduction to Program Synthesis (WS 2024/25)

Chapter 3.1 - Traditional Methodologies (Naive Search)

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany



Center for
Artificial Intelligence

RWTHAACHEN
UNIVERSITY

Traditional Methodologies

Naive Search

- ▶ Naive search methods → Easy to implement
 - ▶ Lack of sophisticated search strategies
- ▶ Use of such methods is usually entails a high computational effort
- ▶ Can be applied to a wide range of search problems
 - ▶ Program synthesis can be considered a search problem
 - ▶ Search problems can be considered decision problems

Traditional Methodologies

Naive Search

- ▶ Naive search → Commonly performed in enumerative or stochastic fashion
 - ▶ Have been successfully applied to many problems in the past
- ▶ **Exhaustive Search** → Enumerative search problem
 - ▶ Iterative or recursive search method
 - ▶ Prone to combinatorial explosion
- ▶ **Monte-Carlo Search** → Stochastic search problem
 - ▶ Search is based on random sampling
 - ▶ Pure random search → divergent method without any guarantee of convergence or success

Traditional Methodologies

Naive Search

candidate	— potential solution that can replace the current (best) solution
configuration	— setting of a candidate
solution	— valid and complete solution that satisfies all constraints
partial solution	— incomplete solution that does not fulfill all constraints
feasible solution	— complete or partial solution that adheres all constraints
optimal solution	— best (ideal) possible solution
dead end	— point where further extension would cause violating constraints
backtrack	— return to a previous decision point
search space	— space of all possible permutations and choices

Table: Basic terminologies used for the description of search problems and algorithms

Traditional Methodologies

Naive Search

- ▶ Characteristics of a search problem:
 - ▶ Set of states $S_1, S_2, \dots, S_N \rightarrow$ State space
 - ▶ Start state $S_0 \rightarrow$ State after initialization
 - ▶ Goal state $S_G \rightarrow$ Ideal solution
 - ▶ Successor function $\delta \rightarrow$ Transition operator

Traditional Methodologies

Naive Search

Definition (Search Problem)

Given a set X of candidate solutions, and a property $P : X \rightarrow \{\text{True}, \text{False}\}$, find a $x \in X$ such that $P(x)$ is True.

Traditional Methodologies

Enumerative Search: Considerations

- ▶ No efficient solution method exists \rightarrow Exhaustive Search
- ▶ Evaluation of each possibility in sequential fashion
 - ▶ Also known as **brute force search (BFS)**
- ▶ BFS has two operations that are performed with a search problem P and candidate c
 - ▶ $\text{valid}(\mathcal{P}, c) \rightarrow$ Checks the validity of candidate solutions
 - ▶ $\text{output}(\mathcal{P}, c) \rightarrow$ Outputs the optimal solution
 - ▶ $\text{first}(\mathcal{P}) \rightarrow$ Returns the first element for the iteration
 - ▶ $\text{next}(\mathcal{P}) \rightarrow$ Returns the next element considered in the iteration
 - ▶ $\text{terminate}(\mathcal{P}, c, n) \rightarrow$ Checks terminate conditions

Traditional Methodologies

Enumerative Search: Characteristics

Algorithm Brute Force Search

```
1:  $c \leftarrow \text{first}(\mathcal{P})$ 
2:  $n \leftarrow 0$ 
3: repeat
4:   if  $\text{valid}(\mathcal{P}, c)$  then  $\text{output}(\mathcal{P}, c)$ 
5:   end if
6:    $c \leftarrow \text{next}(\mathcal{P})$ 
7:    $n \leftarrow n + 1$ 
8: until  $\text{terminate}(\mathcal{P}, c, n) == \text{true}$ 
```

▷ *Until termination criteria not triggered*

Traditional Methodologies

Enumerative Search: Historical Example

- ▶ **Enigma machine** → cipher device
 - ▶ Developed and used in the early - mid 20th century
 - ▶ Encrypted military, diplomatic and commercial communication
 - ▶ Widely used by the Nazis in the second world war
- ▶ Encryption via rotors that manipulate electric pathways
 - ▶ Additional plugboard for letter swapping
 - ▶ Increases cryptographic strength



Figure: Enigma cipher machine (Source: Wikimedia)

Traditional Methodologies

Enumerative Search: Historical Example

- ▶ **Turing bomb** → Machine that is able to break the Enigma
 - ▶ Developed by Alan Turing during WWII
 - ▶ Relies on electric-mechanical exhaustive search
- ▶ Successfully decrypted intercepted German radio messages
 - ▶ Exhaustive search in the *setting space* of the Enigma
 - ▶ Enigma used in WWII had 158,962,555,217,826,360,000 $\approx 10^{20}$ different settings

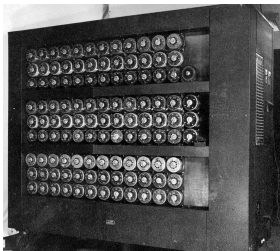


Figure: Turing bomb at Bletchley Park in times of WWII (UK) (Source: Wikimedia)

Traditional Methodologies

Recursive Enumeration

Definition (Enumerative Problem)

A enumeration problem \mathcal{P} is characterized with:

- ▶ An arbitrary alphabet Σ
- ▶ A relation over strings or symbols \mathcal{R} : $\mathcal{R} \subseteq \Sigma^* \times \Sigma^*$

An enumerative search algorithm V produces a sequence y for the input x given to V without duplicates and for $z \in y \iff (x, z) \in \mathcal{R}$.

Traditional Methodologies

Recursive Enumeration

Listing: Sub-sequences of 1 .. 4 in military order

```
1
2
3
4
1 2
1 3
1 4
2 3
2 4
3 4
1 2 3
1 2 4
1 3 4
2 3 4
1 2 3 4
```

Listing: Sub-sequences of 1 .. 4 in lexicographic order

```
1
1 2
1 2 3
1 2 3 4
1 2 4
1 3
1 3 4
1 4
2
2 3
2 3 4
2 4
3
3 4
4
```

- ▶ The number of subsequences of 1 .. n grows exponentially with n
 - ▶ Doubles when n is incremented by 1

Traditional Methodologies

Enumerative Search: Recursive Enumeration

- ▶ Traditional exhaustive search → executed in iterative fashion
- ▶ Recursive search → Enumerative search performed with recursion
 - ▶ Base case is defined as **goal state** or **dead end**
 - ▶ Preferred for enumerative search problems solved with (data) structures well suited for recursion → Tree, Graph, ...

Traditional Methodologies

Recursion

► Advantages:

- Allows problems to be solved in a concise and straight-forward way
- More readable code that is easier to understand
- Less time needed to implement solvers for certain problems
- Dynamic properties

► Disadvantages:

- Mostly more time and space requirements
- Overhead caused by a high number of function calls → can be optimized with **tail recursion**

Traditional Methodologies

Enumerative Search: Recursive Enumeration

Algorithm Enumerative Recursive Search (ERS)

Arguments

S_n : Starting or intermediate state

Return

S_g : Goal or base case

```
1: procedure ERS( $S_n$ )
2:   if  $S_n == G$  then                                ▷ Base case 1: Intermediate state is goal state G
3:     output  $S_n$ 
4:     return  $S_n$ 
5:   else if  $S_n == \text{null}$  then                          ▷ Base case 2: Intermediate state does not exist
6:     return  $S_n$ 
7:   else
8:      $S_{n+1} \leftarrow \delta(S_n)$                     ▷ Execute transition function to obtain next element
9:     ERS( $S_{n+1}$ )                                     ▷ Perform recursion step
10:  end if
11: end procedure
```

Traditional Methodologies

Enumerative Search: Backtracking

- ▶ **Backtracking** → Stepping back to a prior decision point
 - ▶ Making use of recursion to explore all valid possibilities
- ▶ Recursive function → Calls itself until base case is reached
- ▶ Backtracking rejects candidates that cannot fulfill the conditions required for the solution
 - ▶ Extension of recursion by using an acceptance function α
 - ▶ Let S be a state then α is defined as $\alpha : S \rightarrow \{\text{True}, \text{False}\}$

Traditional Methodologies

Enumerative Search: Historical Example

S - Start State
C - Checkpoint
T - Terminal
G - Goal State

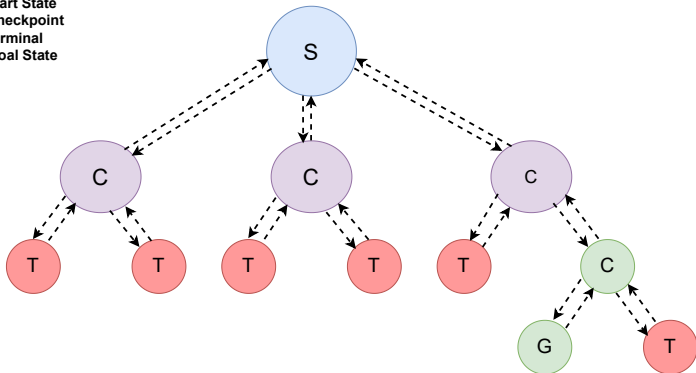


Figure: Backtracking

Traditional Methodologies

Exhaustive Search

Algorithm Enumerative Recursive Search (ERS) with backtracking

Arguments

S_n : Starting or intermediate state

Return

S_g : Goal or base case

```
1: procedure ERS( $S_n$ )
2:   if  $S_n == G$  then                                ▷ Base case 1: Intermediate state is goal state G
3:     output  $S_n$ 
4:     return  $S_n$ 
5:   else if  $S_n == \text{null}$  then                          ▷ Base case 2: Intermediate state does not exist
6:     return  $S_n$ 
7:   else
8:      $S_{n+1} \leftarrow \delta(S_n)$ 
9:     if  $\alpha(S_{n+1}) == \text{true}$  then                    ▷ Backtracking: Call of rejection function
10:      ERS( $S_{n+1}$ )                                     ▷ Recursive step if solution is accepted
11:    else
12:      return  $S_n$                                        ▷ Return checkpoint state in case of rejection
13:    end if
14:  end if
15: end procedure
```

Traditional Methodologies

Monte-Carlo Search

- ▶ **Monte-Carlo Search (MCS)** → Based on Monte-Carlo Sampling
 - ▶ Also known as random search or Monte-Carlo method
- ▶ Foundation of randomized search heuristics
 - ▶ Family of black-box optimization and search algorithms
 - ▶ Local search methods, evolutionary algorithms, ...

Traditional Methodologies

Monte-Carlo Search

- ▶ Randomized search methods → Commonly performed iteratively
- ▶ New samples are drawn from a predefined probability distribution
 - ▶ **Uniform** and **normal distribution** are common choices
 - ▶ Evaluation with a cost function i.e. $\mathcal{C} : \mathbb{R}^n \rightarrow \mathbb{R}$
 - ▶ Termination criterion → number of iterations or ideal cost function value

Traditional Methodologies

Enumerative Search: Recursive Enumeration

Algorithm Simple Monte Carlo Search (S-MCS)

```
1:  $P \sim U$  ▷ Sample initial state
2: repeat ▷ Until termination criteria not triggered
3:    $Q \sim U$  ▷ Sample new search point
4:   ▷ Evaluate cost and check potential replacement
5:   if  $\mathcal{C}(Q)$  better than  $\mathcal{C}(P)$  then
6:      $P \leftarrow Q$  ▷ Replace best immediate search result
7:   end if
8: until  $P$  meets termination criterion
9: return  $P$  ▷
```

Traditional Methodologies

Monte-Carlo Search

- ▶ Success strongly depended on the size of the solution space
 - ▶ If the percentage s of solutions in the search space is known, the success probability after performing n trials can be calculated with counterprobability
 - ▶ $s = 3, n = 50 \rightarrow 1 - 0.97^{50} \approx 0.78 = 78\%$