Introduction to Program Synthesis (WS 2024/25)

Chapter 3.3 - Traditional Methodologies (Enumerative and Stochastic search)

Dr. rer. nat. Roman Kalkreuth

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

Center for
Artificial Intelligence

RWTH AACHEN
UNIVERSITY

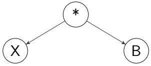
Traditional Methodologies
Traditional Methodologies: Stochastic HC

- Basic search methods for SR $\rightarrow$ **Stochastic hill climbing** and **enumerative branching**
- Stochastic HC $\rightarrow$ Requires a neighbourhood function $\mathcal{N}$
- Adapting enumerative search to SR $\rightarrow$ Depth-first search with **deep cloning**
  - $\rightsquigarrow$ Consider neighbours sampled by $\mathcal{N}$ as potential next steps

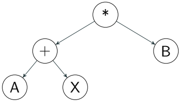
Traditional Methodologies

Traditional Methodologies: Stochastic HC

- **Neighbourhood function** $\mathcal{N}$ $\rightarrow$ replace a leaf node with a randomly generated subtree: $\mathcal{N}(\Psi) \mapsto \Psi'$



$$\Psi = X * B$$



$$\Psi' = (A + X) * B$$

- Stochastic hill climbing with restart is used as a search algorithm:
  - $\rightsquigarrow$ $\mathcal{R}$ $\rightarrow$ **Replacement function**: Selects a neighbour **with better cost** value **uniformly at random**
  - $\rightsquigarrow$ If no neighbour has a better cost value $\rightarrow$ Return and keep $\mathcal{P}$ (no replacement occurs)

Algorithm Stochastic Hill Climbing (S-HC)

while $t < trials$ **do**
    $t \leftarrow i \leftarrow x \leftarrow 0$
    $\mathcal{P} \sim U$ ▷ *Sample initial point*
    $\mathcal{C}(\mathcal{P})$ ▷ *Calculate cost of $\mathcal{P}$*
    while $i < iterations$ **do**
        while $x < neighbours$ **do**
            $\mathcal{Q} \leftarrow \mathcal{N}(\mathcal{P}, n)$ ▷ *Sample n neighbours*
            $\texttt{Prediction}(\mathcal{Q})$ ▷ *Make predictions*
            $\texttt{Evaluation}(\mathcal{Q}, \mathcal{C})$ ▷ *Evaluate neighbours*
            $\mathcal{P} \leftarrow \mathcal{R}(\mathcal{P}, \mathcal{Q})$ ▷ *Replace with better partial solution or keep $\mathcal{P}$*
            $x \leftarrow x + 1$
        end while
        $i \leftarrow i + 1$
    end while
    $t \leftarrow t + 1$
end while
return $\mathcal{P}$ ▷

Traditional Methodologies
Traditional Methodologies: Steepest HC

- Steepest HC $\rightarrow$ Modification of stochastic HC
- Selects the best (steepest) improvement among the sampled neighbours
- Suitable for landscapes that are well suited for *greedy* approaches

Traditional Methodologies

Traditional Methodologies: Improvement Strategies

- Hill climbing is prone to getting stuck in **local optima**
- Requires advanced strategies to prevent **search stagnation**:
  - $\leadsto$ **Backtracking** $\rightarrow$ Rejection of infeasible partial solutions
  - $\leadsto$ **Backjuming** $\rightarrow$ Stochastic roll-backs in the space of partial solutions
  - $\leadsto$ **Population-based approach** $\rightarrow$ Simultaneous evolving partial solutions

$\leadsto$ Enumerative search $\rightarrow$ Expensive in large symbolic search spaces

- $\leadsto$ **Branch and Bound** $\rightarrow$ Reject solution that are not located in the range of predefined lower/upper bounds
- $\leadsto$ Specific **pre-evaluation** of candidate solution
- $\leadsto$ Can be integrated into **enumerative recursive search** (ERS)

Traditional Methodologies

Traditional Methodologies: Branch and Bound

- Branch and bound prunes the search space
  - $\rightsquigarrow$ Clipping partial solutions within semantic boundaries
- Rejection of infeasible solutions
- Assumption that following feasible search trajectories is more effective

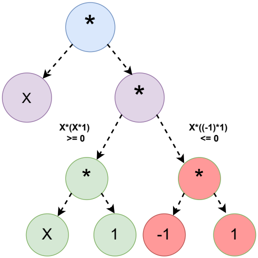


Figure: Branch-and-Bound example: Let $f(x) = x^2 \mapsto \mathbb{R}_{\geq 0}, x \in \mathbb{R}$ be the objective function. Candidate solutions are bounded in $\mathbb{R}_{\geq 0}$. The red branch will be rejected since the evaluation will lead to negative semantic values in $\mathbb{R}^-$.

Traditional Methodologies

Traditional Methodologies: Branch and Bound

Algorithm ERS with Branch and Bound

Arguments
$\mathcal{S}_n$: Initial or intermediate state
$\mathcal{P}$: Problem instance
$\mathcal{B}_l$: Lower bound
$\mathcal{B}_u$: Upper bound
$\mathcal{G}$: Goal state
s: Number of steps
Return
$\mathcal{S}_g$: Goal or base case

procedure ERS($\mathcal{S}_n$, $\mathcal{P}$, $\mathcal{B}_{lower}$, $\mathcal{B}_{upper}$, $n_{neighbour}$)
  $\Omega \leftarrow \mathcal{N}(s)$ ▷ *Determine the neighbourhood*
  for ω `in` Ω **do** ▷ *Consider each neighbour*
    $p_\omega \leftarrow$ `Prediction`(ω) ▷ *Make prediction*
    $c_\omega \leftarrow \mathcal{C}(\omega, \mathcal{P})$ ▷ *Calculate cost*
    $b_{lower}, b_{upper} \leftarrow$ `boundaries`(c_ω) ▷ *Determine the bounds*
    if $\mathcal{B}_l \leq b_{lower}$ **and** $\mathcal{B}_u \geq b_{upper}$ **then** ▷ *Check against the global bounds*
      if $S_n == \mathcal{G}$ **then** ▷ *If goal state is reached*
        `output`$(\mathcal{S}_n)$ ▷ *Output the solution*
      else ▷ *Consider the current step as next one*
        $\mathcal{S}_{n+1} \leftarrow (\omega, c)$
        `ERS`$(\mathcal{S}_{n+1}, \ldots)$ ▷ *Trigger next recursion step*
      end if
    end if
  end for
end procedure