



UNIVERSITY OF
BIRMINGHAM

ARTIFICIAL INTELLIGENCE 1
OPTIMISATION AND CONSTRAINTS
HANDLING

DR LEONARDO STELLA

SCHOOL OF COMPUTER SCIENCE

2023/2024 - Week 10

AIMS OF THE SESSION

This session aims to help you:

- Understand the weaknesses in Hill Climbing and how to mitigate them
- Explain the steps involved in Simulated Annealing
- Deal with constraints in optimisation problems

OUTLINE

- 1 Recap: Informed Search and Optimisation
- 2 Hill Climbing: Variants
- 3 Simulated Annealing
- 4 Constraint Handling

OUTLINE

- 1 Recap: Informed Search and Optimisation
- 2 Hill Climbing: Variants
- 3 Simulated Annealing
- 4 Constraint Handling

OUTLINE

- 1 Recap: Informed Search and Optimisation
- 2 Hill Climbing: Variants**
- 3 Simulated Annealing
- 4 Constraint Handling

WEAKNESSES OF HILL CLIMBING

- Last week, we discussed pros and cons of hill climbing
- Can we mitigate some of its weaknesses?
 - **W**: it can run for ever if the problem is not properly formulated
 - **S**: we run the algorithm for a maximum number of iterations m
 - Indeed, we used m to determine its time complexity: $\mathcal{O}(mnp)$
 - **W**: not guaranteed to be complete, nor optimal
 - **S**: as long as we are satisfied with a good solution, it can find it quickly
 - **W**: it can get stuck in local maxima and plateaus
 - **S**: there are variants of the algorithm to deal with this

HILL CLIMBING: VARIANTS

- Let us consider an n -dimensional problem
- The gradient determines the uphill moves along the n dimensions
- **Stochastic Hill Climbing**
 - This variant chooses at random among the **uphill moves**
 - The probability of picking a specific move can depend on the **steepness**
 - Converges **more slowly** than steepest ascent
 - Can find **higher quality** solutions
- **First-Choice Hill Climbing**
 - Implements stochastic hill climbing by randomly generating successors until a better successor than the current state is generated
 - It performs **well** when a state has many successors
- **Random-Restart Hill Climbing**
 - Generates a series of hill-climbing searches from random initial states
 - It stops when a goal is found
 - It is **complete** with probability 1 as it will eventually generate a goal state as initial state

OUTLINE

- 1 Recap: Informed Search and Optimisation
- 2 Hill Climbing: Variants
- 3 Simulated Annealing**
- 4 Constraint Handling

SIMULATED ANNEALING

- A hill climbing algorithm that **never makes downhill moves** can get stuck in local maxima (or minima)
- A purely random walk will eventually find the global maximum, but will be **extremely inefficient**
- **Simulated Annealing** combines these two approaches:
 - Generate initial candidate solution at random
 - Generate neighbour solutions, **pick one at random (instead of best)**
 - If the move improves the current solution, it is always accepted
 - Otherwise, the algorithm accepts a worse move with **a probability less than 1 that decreases over time**
 - Terminate when **a min temperature is reached** or **solution does not change in value**
- Differently from hill climbing, simulated annealing allows bad moves

SIMULATED ANNEALING: ACCEPTING BAD MOVES

- How should we set the probability of accepting bad moves?
 - **High** probability in the beginning, similar to random search (explore)
 - **Lower** as time passes, similar to hill climbing (exploit)
- In metallurgy, **annealing** is the process used to temper or harden metals by heating them to a **high** temperature
- Then, **gradually** cooling them until they reach a low-energy state
- Simulated annealing uses a similar principle to accept worse moves
 - The algorithm accepts worse moves with some probability less than 1
 - The probability **decreases exponentially** the worse the move is
 - The probability decreases as the **temperature goes down**

PROBABILITY FUNCTION: QUALITY

- We model the probability using a thermodynamics-inspired function:

$$e^{\Delta E/T}$$

- 'The probability **decreases exponentially** the worse the move is'. This is determined by the numerator in the exponential:

$$\Delta E = E(x_{\text{new}}) - E(x_{\text{current}}) \leq 0,$$

where $E(x_{\text{new}})$ is the energy (quality) of the new random state and $E(x_{\text{current}})$ is the energy (quality) of the current state

- **Case 1: Very bad move.** If $E(x_{\text{new}}) \ll E(x_{\text{current}})$, then $e^{\Delta E/T} \approx 0$, i.e., a lower probability to accept the move
- **Case 2: Not too bad move.** If $|E(x_{\text{new}}) - E(x_{\text{current}})| \approx 0$, then $e^{\Delta E/T} > 0$, i.e., a higher probability to accept the move

PROBABILITY FUNCTION: TEMPERATURE

- 'The probability decreases as the **temperature goes down**'. This is determined by the denominator $T > 0$
- At start, we set the temperature to a high value $T(0) = \mathcal{T} \gg 0$
- An update rule (**schedule**) is used to reduce it, for instance:

$$T(t+1) = \alpha T(t),$$

where $T(t+1)$ and $T(t)$ represent the values of the temperature at this iteration and at the next iteration, respectively, and $\alpha \approx 1$

- If the temperature decreases slowly enough, simulated annealing will **find the global maximum with probability 1**
- This is a property of the Boltzman distribution $e^{\Delta E/T}$, i.e., the probability is **concentrated on the global maxima**

PROBABILITY FUNCTION: TEMPERATURE

- We model the probability using a thermodynamics-inspired function:

$$e^{\Delta E/T}$$

- How does the temperature affect the probability?
- **Case 1: High temperature.** If $T \gg 0$, then $e^{\Delta E/T} \approx 1$, i.e., a higher probability to accept the move
- **Case 2: Low temperature.** If $T \approx 0$, then $e^{\Delta E/T} \approx 0$, i.e., a lower probability to accept the move

SIMULATED ANNEALING

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
current \leftarrow *problem*.INITIAL
for $t = 1$ **to** ∞ **do**
 $T \leftarrow \text{schedule}(t)$
 if $T = 0$ **then return** *current*
 next \leftarrow a randomly selected successor of *current*
 $\Delta E \leftarrow \text{VALUE}(\text{current}) - \text{VALUE}(\text{next})$
 if $\Delta E > 0$ **then** *current* \leftarrow *next*
 else *current* \leftarrow *next* only with probability $e^{\Delta E/T}$

PERFORMANCE OF SIMULATED ANNEALING

- Let us evaluate the performance of simulated annealing
 - **Completeness:** simulated annealing **is not complete**, as it depends on the problem formulation and design of the algorithm
 - **Optimality:** simulated annealing **is not optimal**, as it depends on the termination criteria and the schedule
 - **Time complexity:** it depends on the schedule and the termination criterion
 - **Space complexity:** depends on how the design variable is represented in the algorithm

SIMULATED ANNEALING

- Pros:
 - Can usually find a **good (near optimal) solution** in a reasonable amount of time
 - It avoids getting stuck **in poor local maxima and plateau** by combining exploration and exploitation
- Cons:
 - **Not guaranteed** to be complete, nor optimal
 - Time/space complexity is **problem- and representation-dependent**

OUTLINE

- 1 Recap: Informed Search and Optimisation
- 2 Hill Climbing: Variants
- 3 Simulated Annealing
- 4 Constraint Handling**

APPLYING OPTIMISATION ALGORITHMS

- To apply hill climbing and simulated annealing we need to specify:
- **Optimisation problem formulation**
 - Design variable and search space
 - Objective function
 - Constraints
- **Algorithm operators**
 - Representation
 - Initialisation procedure
 - Neighbourhood operator
- **Strategy to deal with constraints**
 - 1 By carefully designing algorithm operators
 - 2 By modifying the objective function

CONSTRAINT HANDLING: ALGORITHM OPERATORS

- Pros:
 - **Infeasible** candidate solutions **will not be generated**, facilitating the search for optimal solutions
- Cons:
 - **May be difficult** to design, **problem-dependent**
 - Can **restrict the search space too much**, making it harder to find the optimal solution

APPROACH 1: DEATH PENALTY

- Given the canonical formulation of an optimisation problem:

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, n. \end{aligned}$$

- We avoid infeasible solutions by modifying the objective function

$$\min \quad f(x) + Q(x),$$

where the penalty term $Q(x)$ is defined as:

$$Q(x) \begin{cases} 0, & \text{if } x \text{ is feasible,} \\ C, & \text{otherwise.} \end{cases}$$

- The term C is a large positive constant, ensuring infeasible solutions are worse than feasible solutions

APPROACH 1: DEATH PENALTY

- Given the canonical formulation of an optimisation problem:

$$\begin{aligned} \min \quad & f'(x), \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, n. \end{aligned}$$

- We can also use the following formulation:

$$f'(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ C, & \text{otherwise.} \end{cases}$$

- The term C is a large positive constant, like before
- Weakness: all infeasible solutions have the **same penalty**
- Algorithms **will struggle** to find feasible solutions in regions of the state space dominated by infeasible solutions

APPROACH 2: LEVELS OF INFEASIBILITY

- Given the canonical formulation of an optimisation problem:

$$\begin{aligned} \min \quad & f(x) + Q(x), \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, n. \end{aligned}$$

- The idea is to distinguish the objective value of infeasible solutions to help the algorithm find a feasible solution

$$\begin{aligned} Q(x) = & C g_1(x) + C g_2(x) + \dots + C g_m(x) \\ & + C |h_1(x)| + C |h_2(x)| + \dots + C |h_n(x)|. \end{aligned}$$

- The **objective function itself** will guide the algorithm towards feasible solutions (and near-optimal solutions, too!)

APPROACH 2: LEVELS OF INFEASIBILITY

- Given the canonical formulation of an optimisation problem:

$$\begin{aligned} \min \quad & f(x) + Q(x), \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, n. \end{aligned}$$

- We should only add the penalties corresponding to violated constraints

$$\begin{aligned} Q(x) = & v_{g_1} C |g_1(x)| + v_{g_2} C |g_2(x)| + \dots + v_{g_m} C |g_m(x)| \\ & + v_{h_1} C |h_1(x)| + v_{h_2} C |h_2(x)| + \dots + v_{h_n} C |h_n(x)|, \end{aligned}$$

where $v_{g_i} = 1$ if g_i is violated and $v_{g_i} = 0$ otherwise,
and $v_{h_j} = 1$ if h_j is violated and $v_{h_j} = 0$ otherwise

APPROACH 2: LEVELS OF INFEASIBILITY

- Given the canonical formulation of an optimisation problem:

$$\begin{aligned} \min \quad & f(x) + Q(x), \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, n. \end{aligned}$$

- We should only add the penalties corresponding to violated constraints

$$\begin{aligned} Q(x) = & v_{g_1} C g_1(x)^2 + v_{g_2} C g_2(x)^2 + \dots + v_{g_m} C g_m(x)^2 \\ & + v_{h_1} C h_1(x)^2 + v_{h_2} C h_2(x)^2 + \dots + v_{h_n} C h_n(x)^2. \end{aligned}$$

- Squared values** make the distinction between objective values even larger, distinguishing different infeasible solutions more

CONSTRAINT HANDLING: OBJECTIVE FUNCTION

- Pros:
 - Usually easier to design
- Cons:
 - Algorithms have to search for feasible solutions to design

FINAL REMARKS: COMPLETENESS

- If we use a strategy that never enables infeasible solutions to be generated, then hill climbing and simulated annealing are **complete**
- Otherwise:
 - Hill climbing **is not complete** if the objective function has local optima (a local optimum may be infeasible)
 - Simulated annealing **is not guaranteed** to find a feasible solution within a reasonable amount of time; it might terminate before finding a feasible solution

References



Russell, A. S., and Norvig, P., *Artificial Intelligence A Modern Approach*, 4th Edition. Prentice Hall.



Chapter 4 – “Search in Complex Environments”, Section 4.1 “Local Search and Optimization Problems” up to and including Section 4.1.2 “Simulated Annealing”.

AIMS OF THE SESSION

You should now be able to:

- Understand the weaknesses in Hill Climbing and how to mitigate them
- Explain the steps involved in Simulated Annealing
- Deal with constraints in optimisation problems



Thank you!