# Fundamentals of Artificial Intelligence

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

**COMP307/AIML420**

**Search 2**

Dr. Heitor Murilo Gomes
heitor.gomes@vuw.ac.nz
http://www.heitorgomes.com

# Information

- Assignment 1 (due on week 5 - 27 March 2024)
- Extension requests (use the Submission system)
- Teaching evaluation (Heitor)
- Helpdesks starting from 2pm until 4pm (Thursday until next Wednesday)

# On the last lecture…

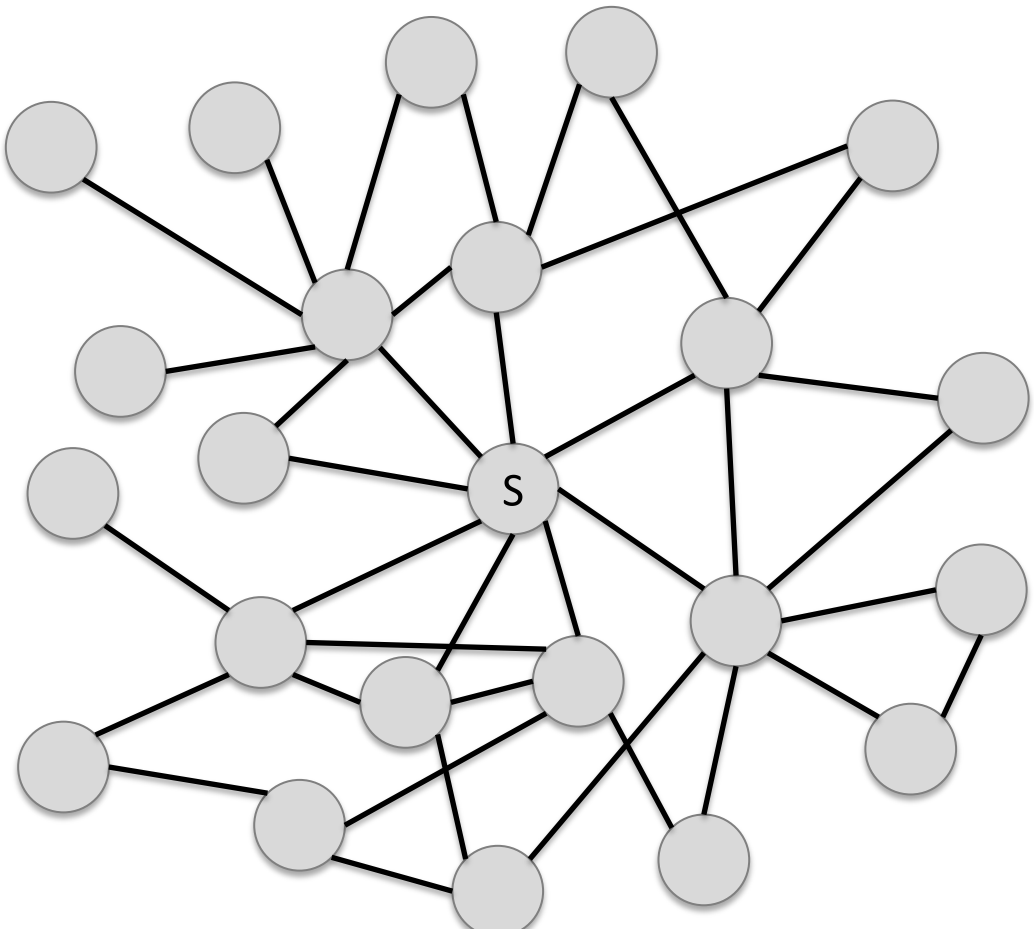
- **Abstracting** the problem is fundamental
- **Selecting** an appropriate Search algorithm
- **Defining** ***h(n)*** may not always be trivial
- Focus on finding the ***path*** **from S to E**
- See Chapter 3 [1]

[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence a modern approach. 4th edition

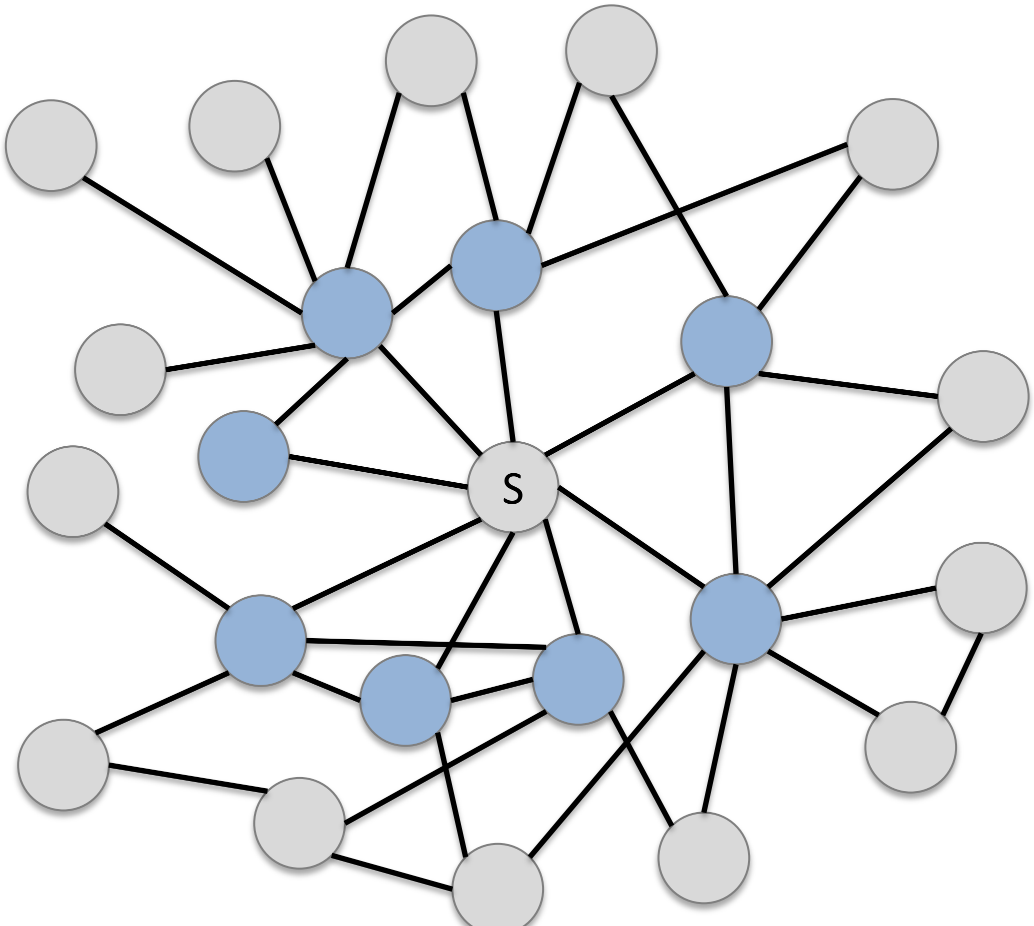
# Beam search

- Extends BFS, instead of exploring all possible paths the exploration is limited (**beam width**)
- More efficient on large search spaces
- Not complete and not optimal ☹
- Which paths should be explored?
  - Evaluation function, heuristic function (if possible) and random

# Beam search: intuition

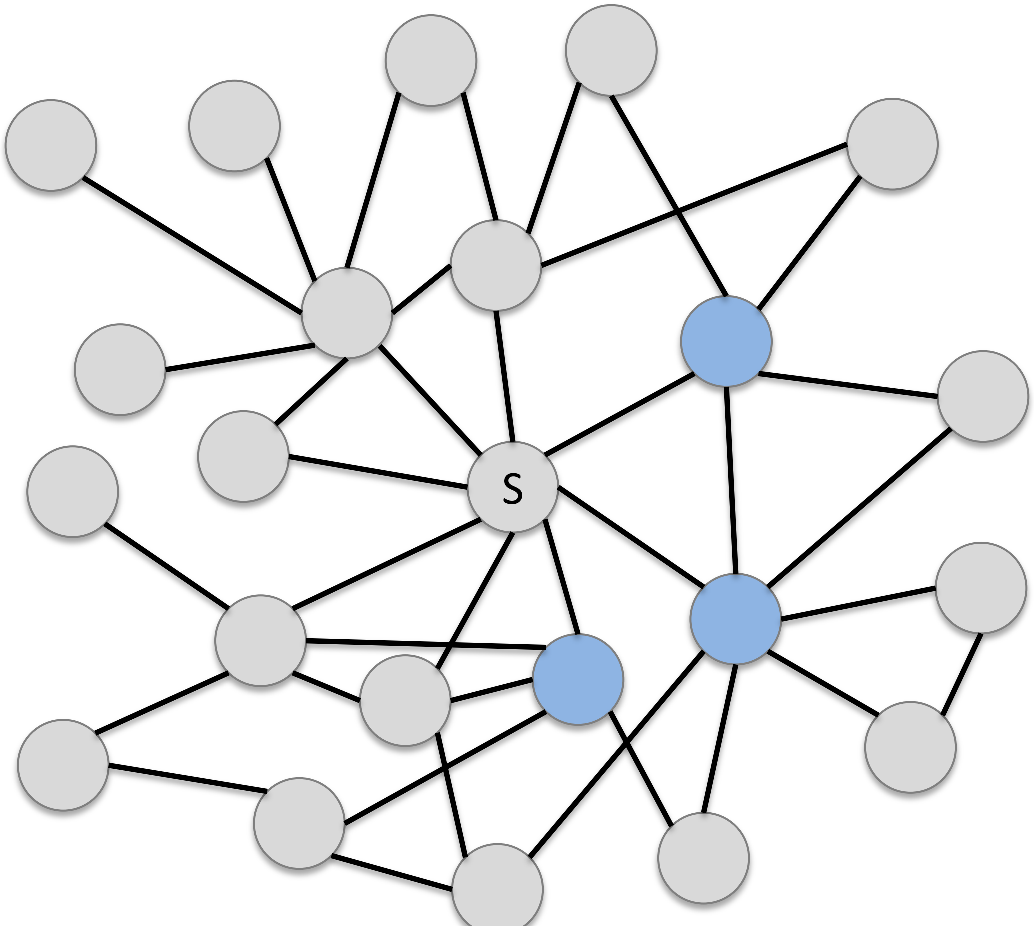


# Beam search: intuition



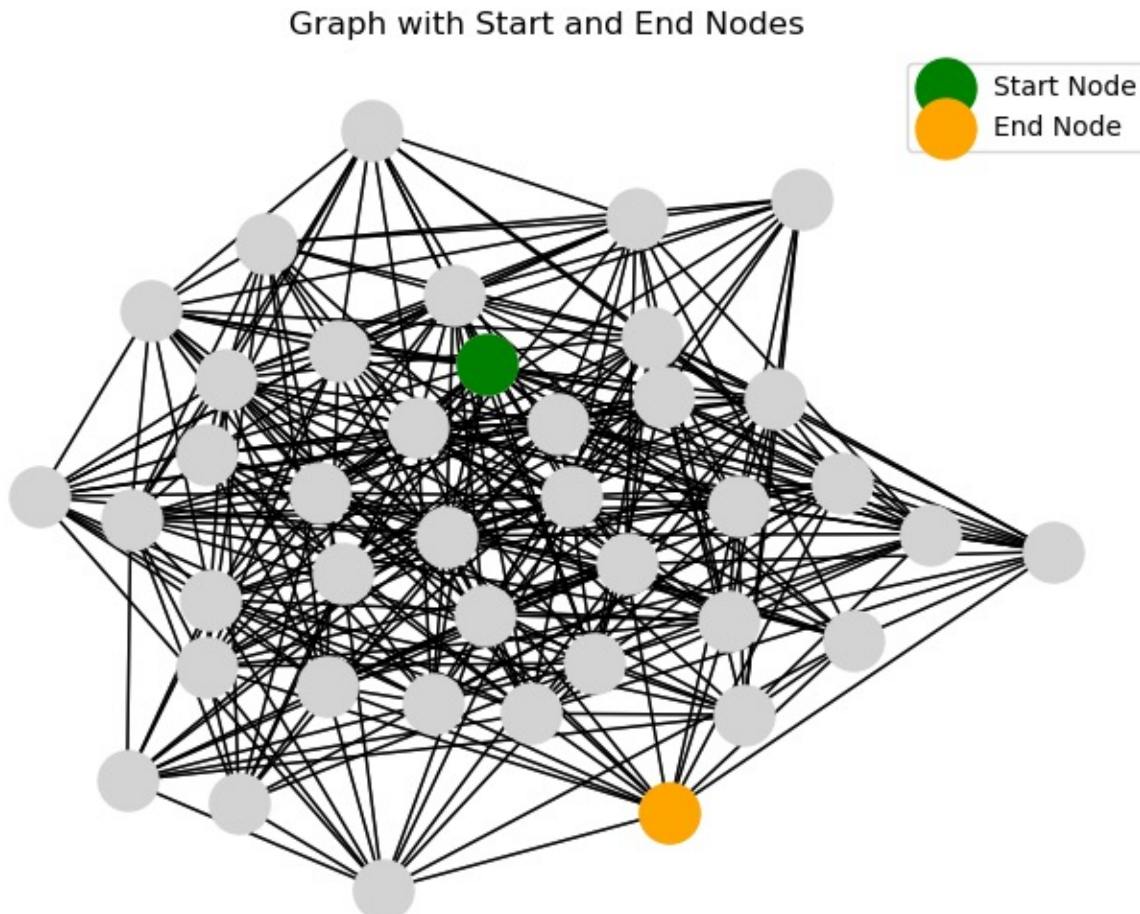

- BFS add all neighbours to the frontier

# Beam search: intuition

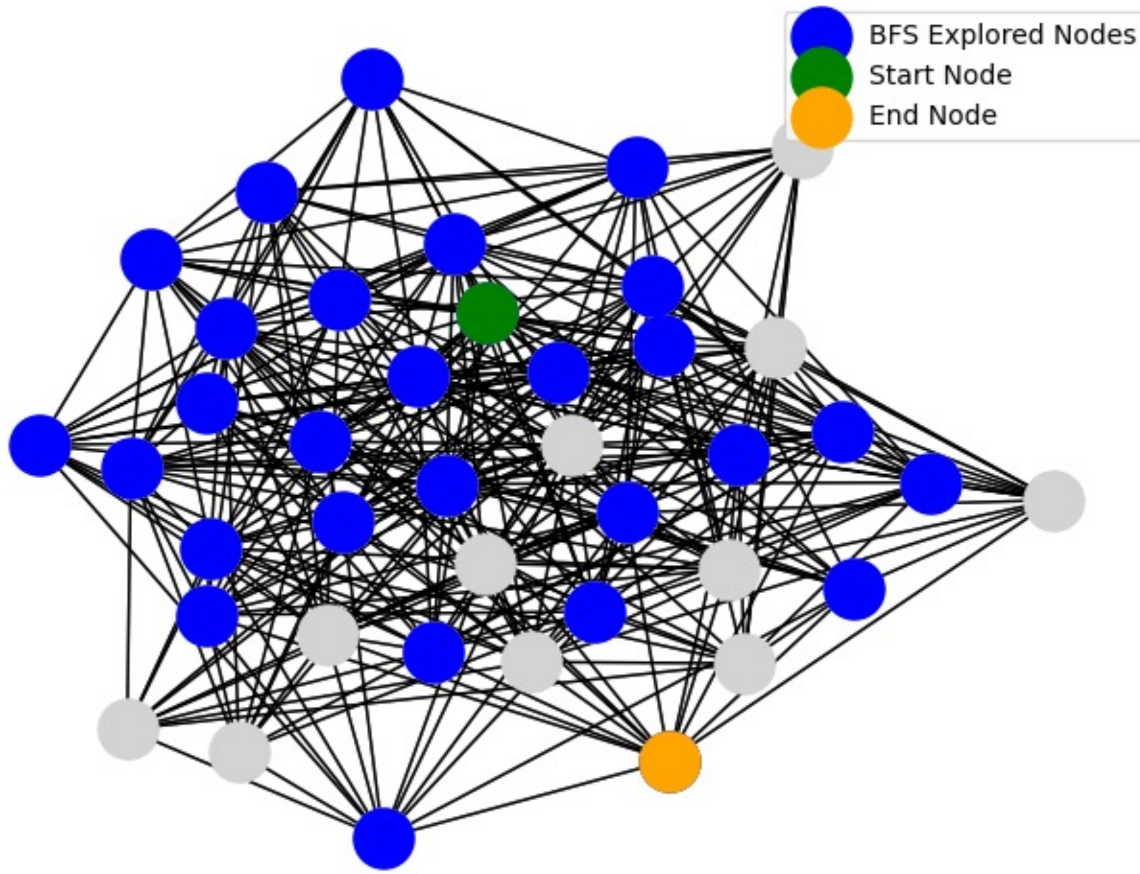



- BFS add all neighbours to the frontier
- Beam search add just some neighbours (beam width)

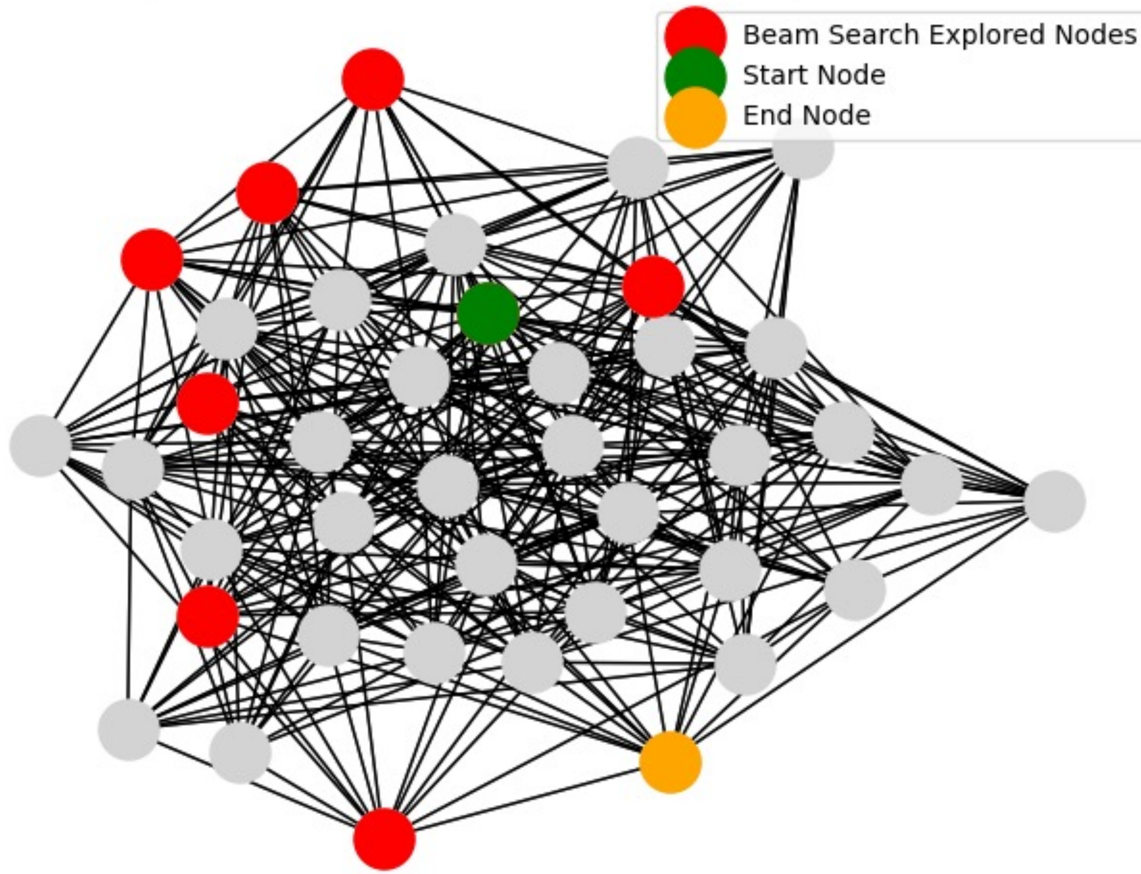
# Beam search: example



# Beam search: example



# Beam search: example



# Beam search: applications

- **Beam search** allow us to maintain **tractability** in **large state-spaces**

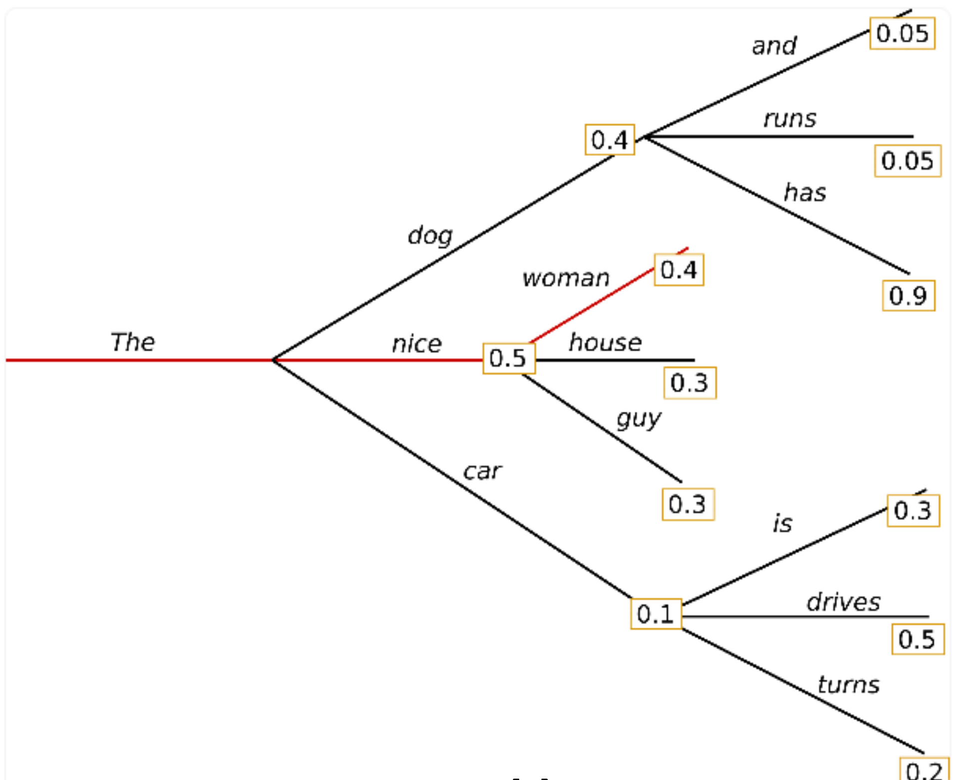
- Practical applications includes:
  - **text generation**
  - **machine translation**
  - **…**

- Let's say we want to generate a text sentence*

* Grossly overlooking some details so that we maintain sanity

# Beam search: Text generation

- One approach is to use Greedy search
  - Selects the word with the highest probability as the next word in the sentence

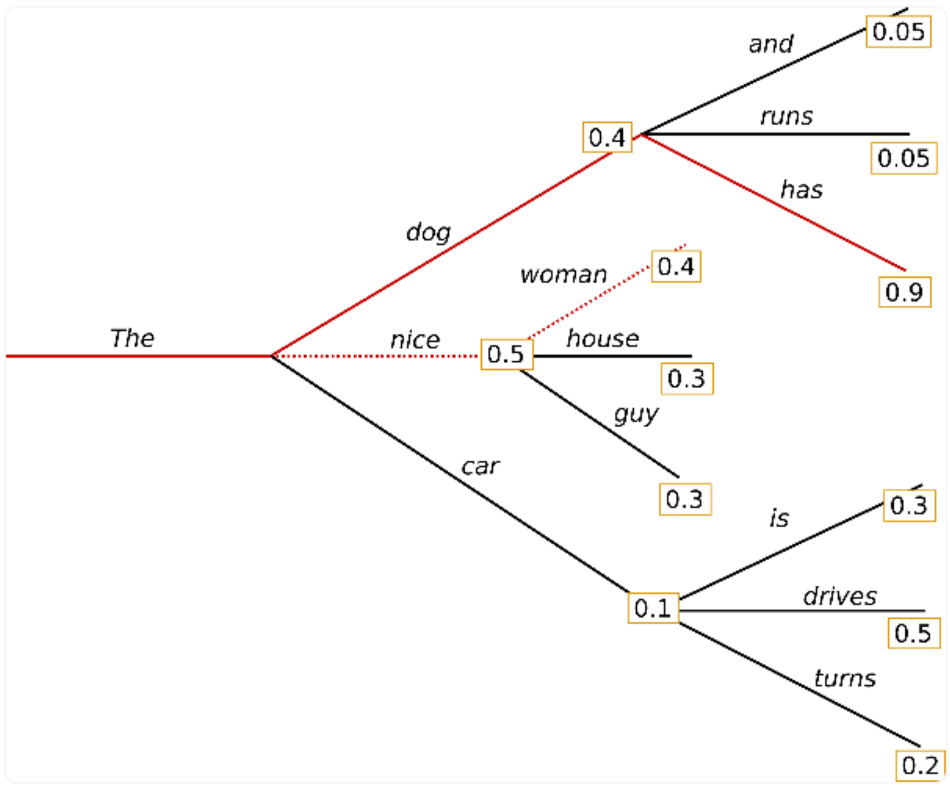



Source [2]

[2] https://huggingface.co/blog/how-to-generate

# Beam search: Text generation

- Using **Beam search**, we reduce the risk of missing "hidden" high probability word sequences* (e.g. beam width = 2)




Source [2]

[2] https://huggingface.co/blog/how-to-generate

* The product of all the words prob. in the sequence

# Local Search & Optimization

- Sometimes we **don’t care about the path** only the **solution**

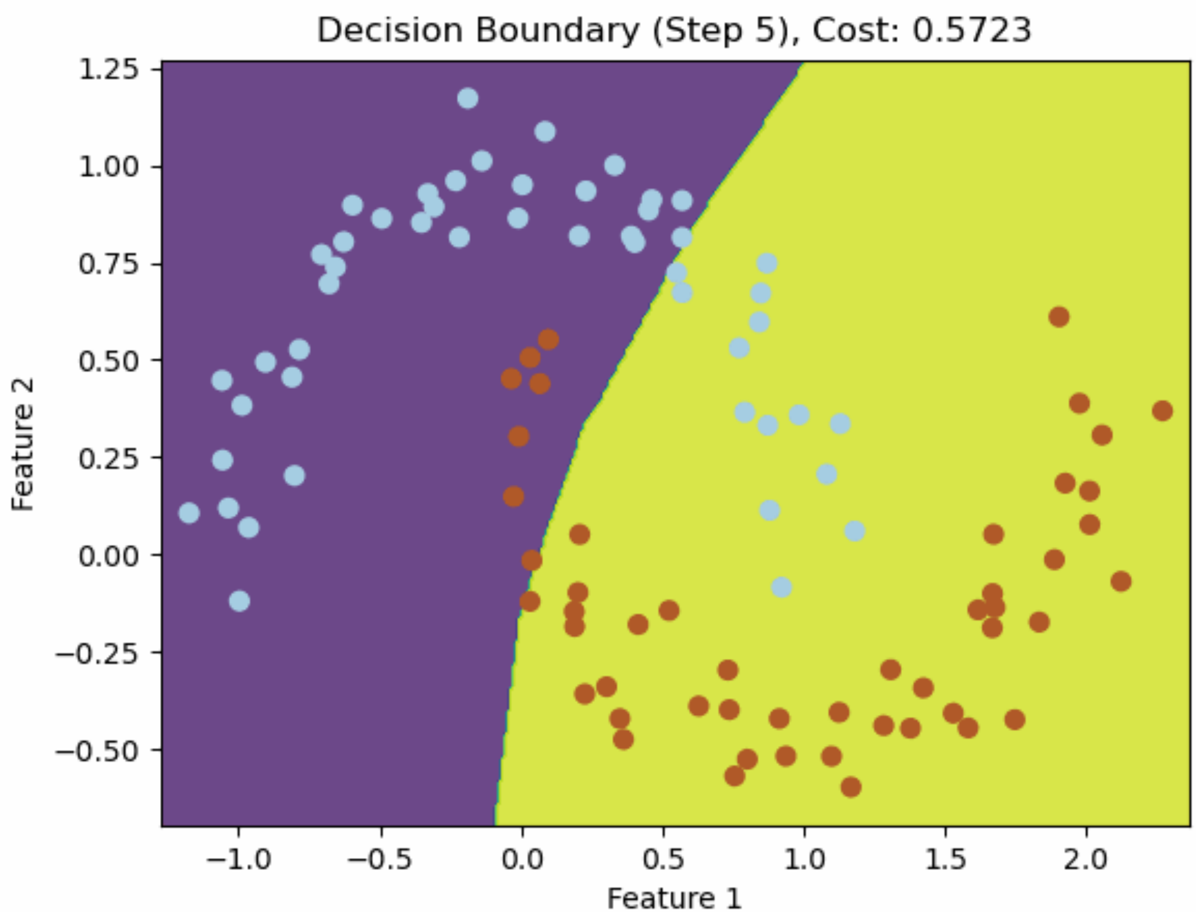
- We define a **problem** and **iteratively** attempt to **optimize intermediary solutions**.

- Examples:
  - **Job scheduling**: manufacturing, project management, or CPU scheduling → assign tasks to resources while optimizing criteria i.e. minimizing total time to complete all tasks or maximize resource utilization.

  - **Circuit design**: optimize the layout of components on a chip

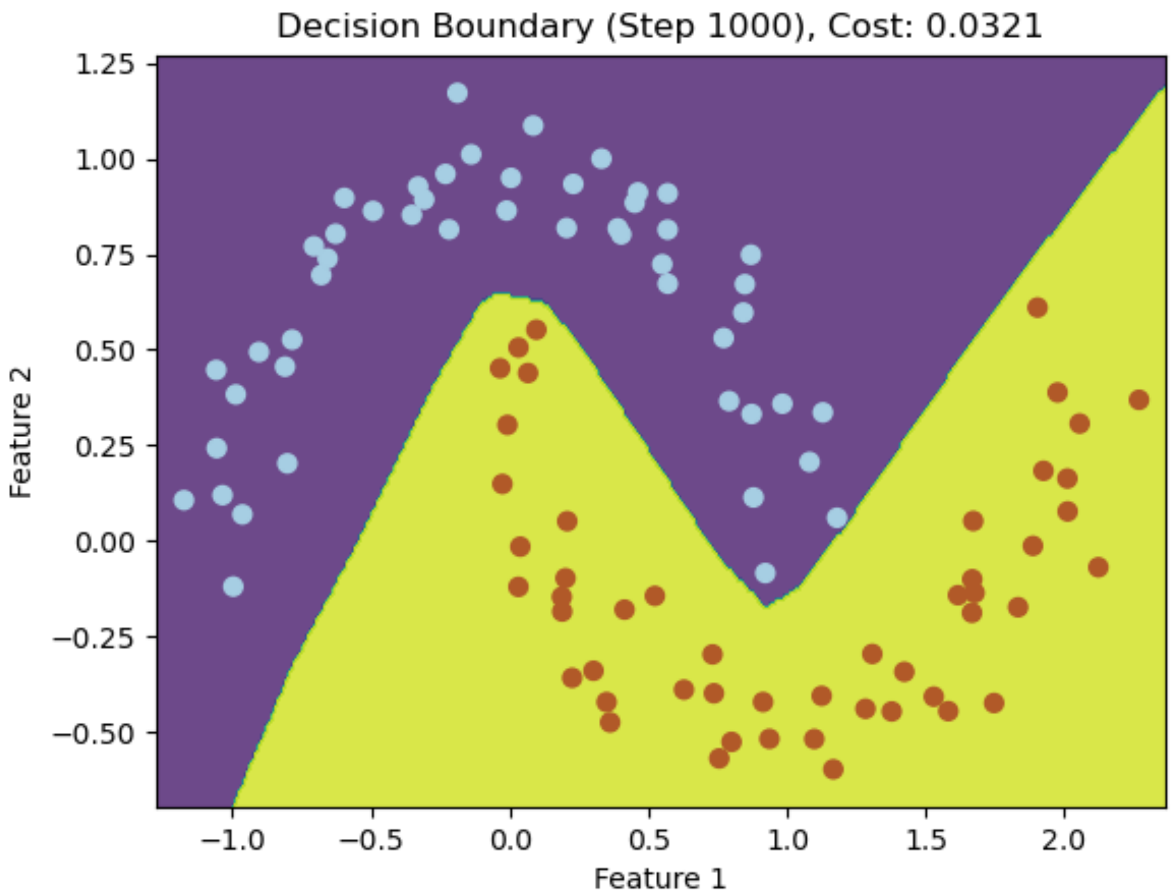
# Local Search & Optimization

- Examples:
  - **Neural networks**

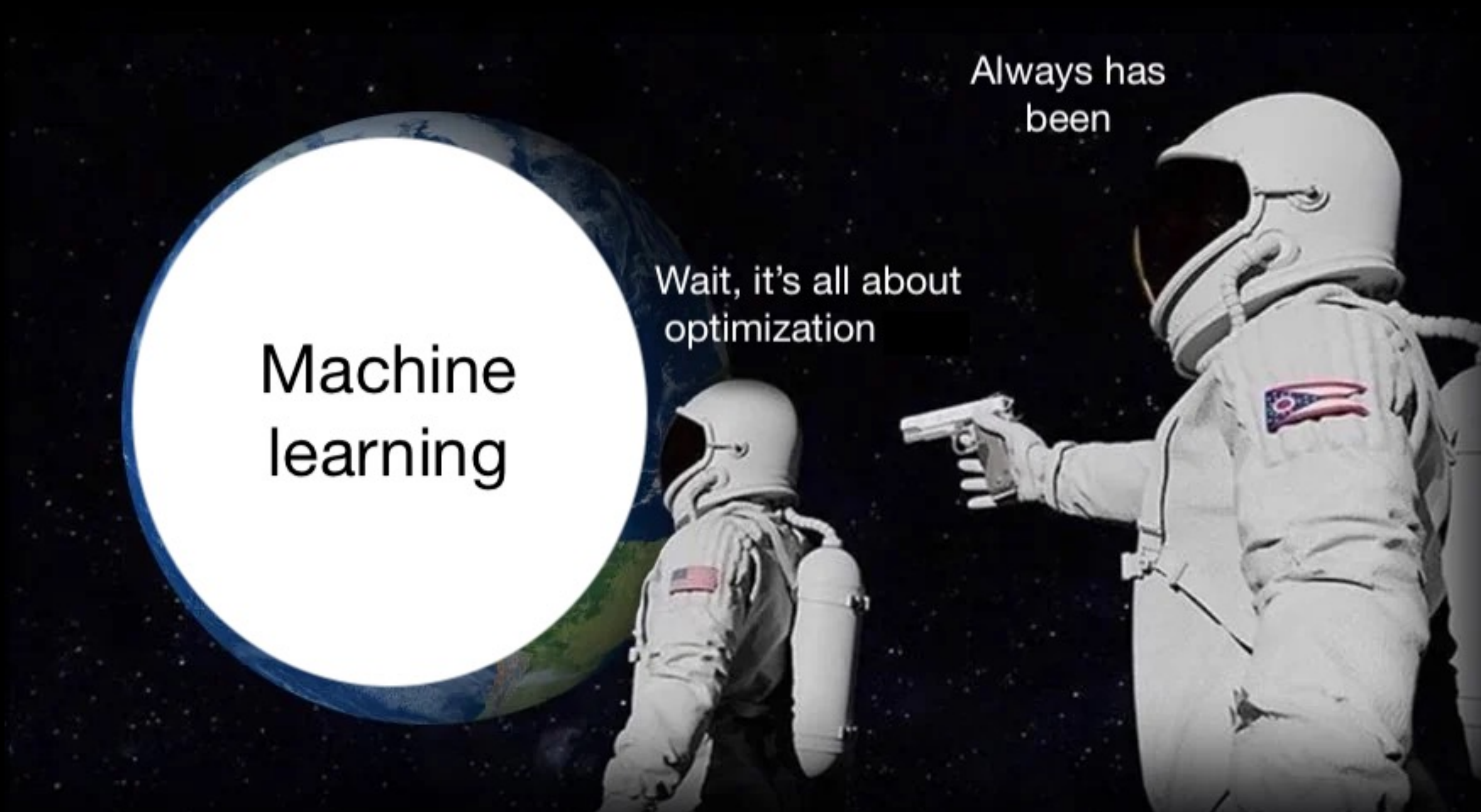


# Local Search & Optimization

- Examples:
  - **Neural networks**



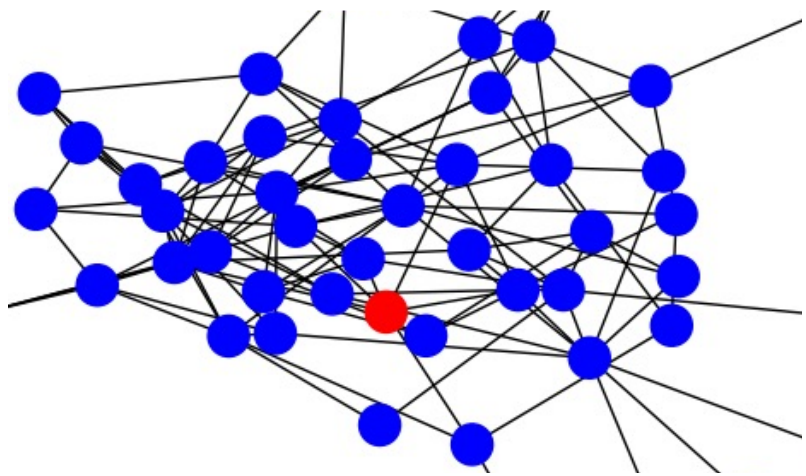
# Local Search & Optimization



# Local Search & Optimization

- **Local search** methods commonly operate on a **single node** (current state), and often can only **move to its neighbors**

?

?  ?

?

# Local Search & Optimization

- **Objective function**: <cost, loss, fitness, utility, ...> function
- **State-space landscape**: location and elevation

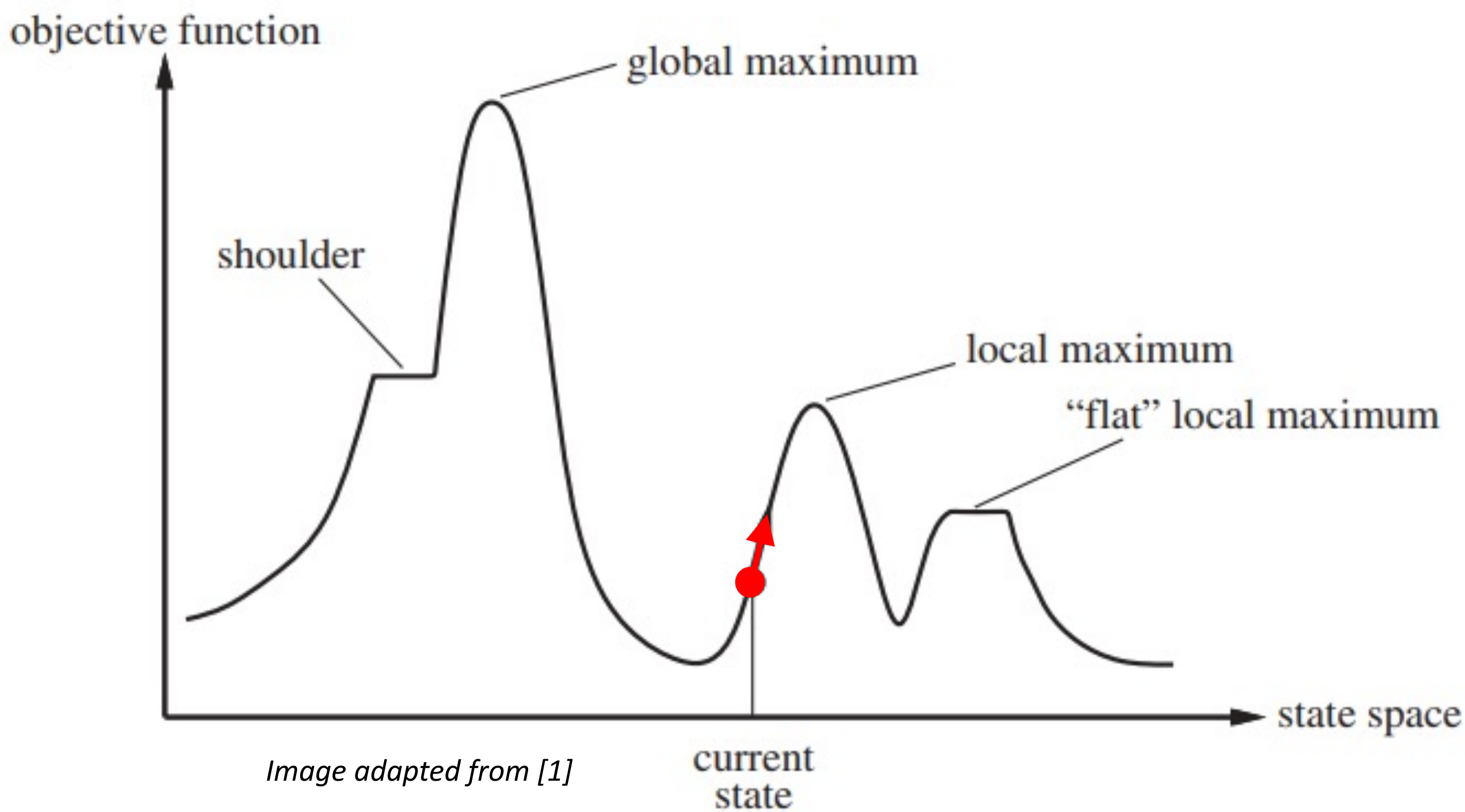



*Image adapted from [1]*



[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence a modern approach. 4th edition

# Hill-climbing

**function** HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

  *current* $\leftarrow$ MAKE-NODE(*problem*.INITIAL-STATE)
  **loop do**
    *neighbor* $\leftarrow$ a highest-valued successor of *current*
    **if** neighbor.VALUE $\leq$ current.VALUE **then return** *current*.STATE
    *current* $\leftarrow$ *neighbor*

*Adapted from [1]*

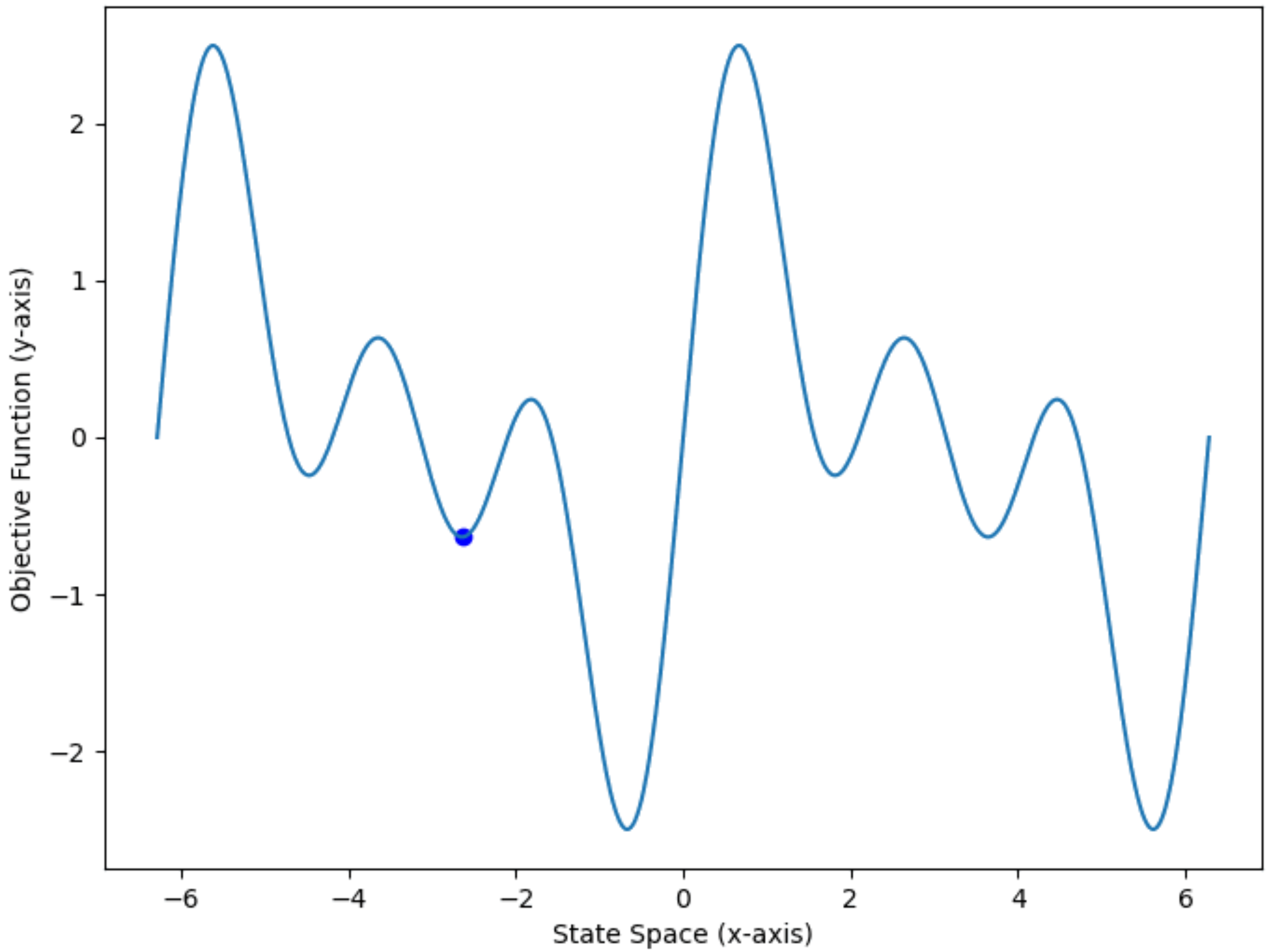
- **Iteratively** moves in the **direction of increasing** (or decreasing) **value** (uphill or downhill)

- Stop when no neighbor has a better value (higher or lower)



[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence a modern approach. 4th edition

# Hill-climbing




**Hill-climbing** example
Moving towards local maximum

# Simulated Annealing

- One drawbacks of **Hill-climbing** is that it **cannot make downhill** movements which can be **beneficial in overall**
- It can get “stuck” on local maximum
- **Simulated annealing** combines **Hill-climbing** with **random walk**
- This allow us to explore other parts of the state space

# Simulated Annealing

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to “temperature”

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← next.VALUE – current.VALUE
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

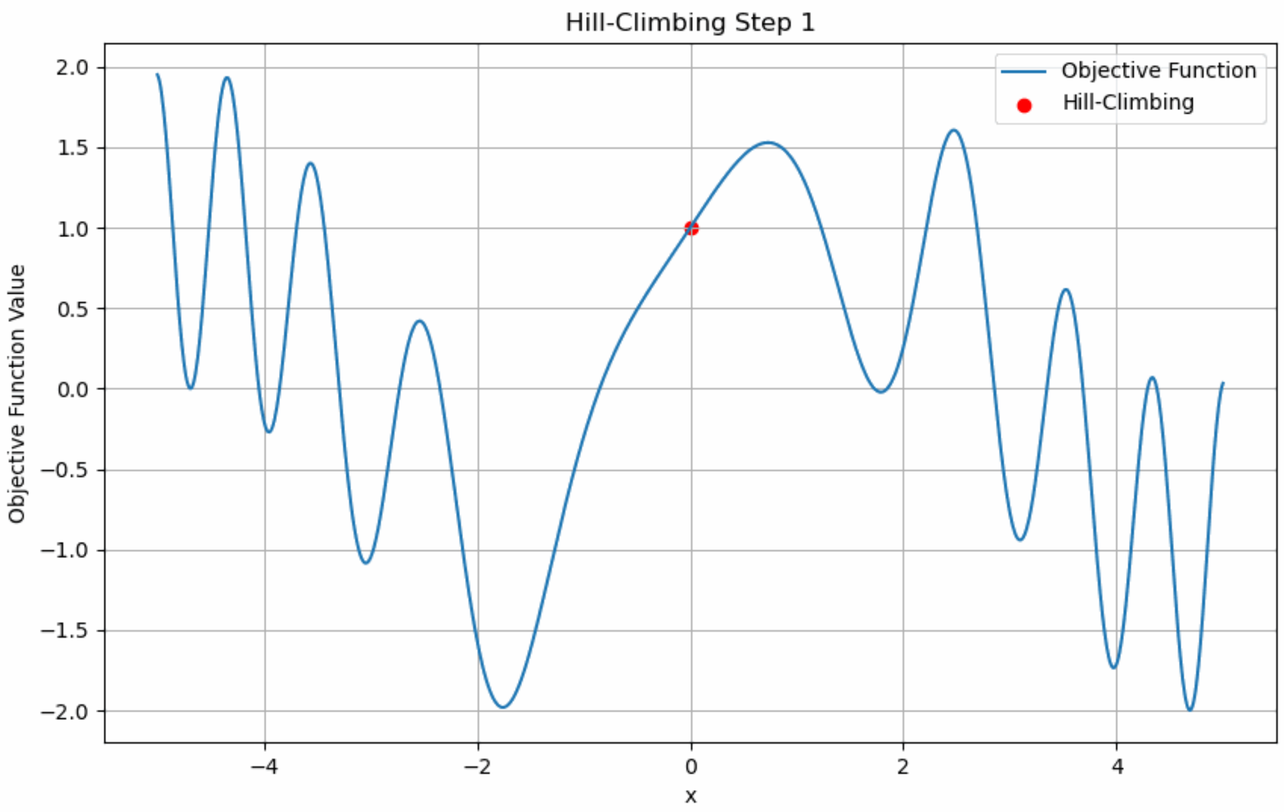
*Adapted from [1]*

- Selects the next move randomly, if it improves, accept it
- Else, accept it with probability $e^{\Delta E/T}$

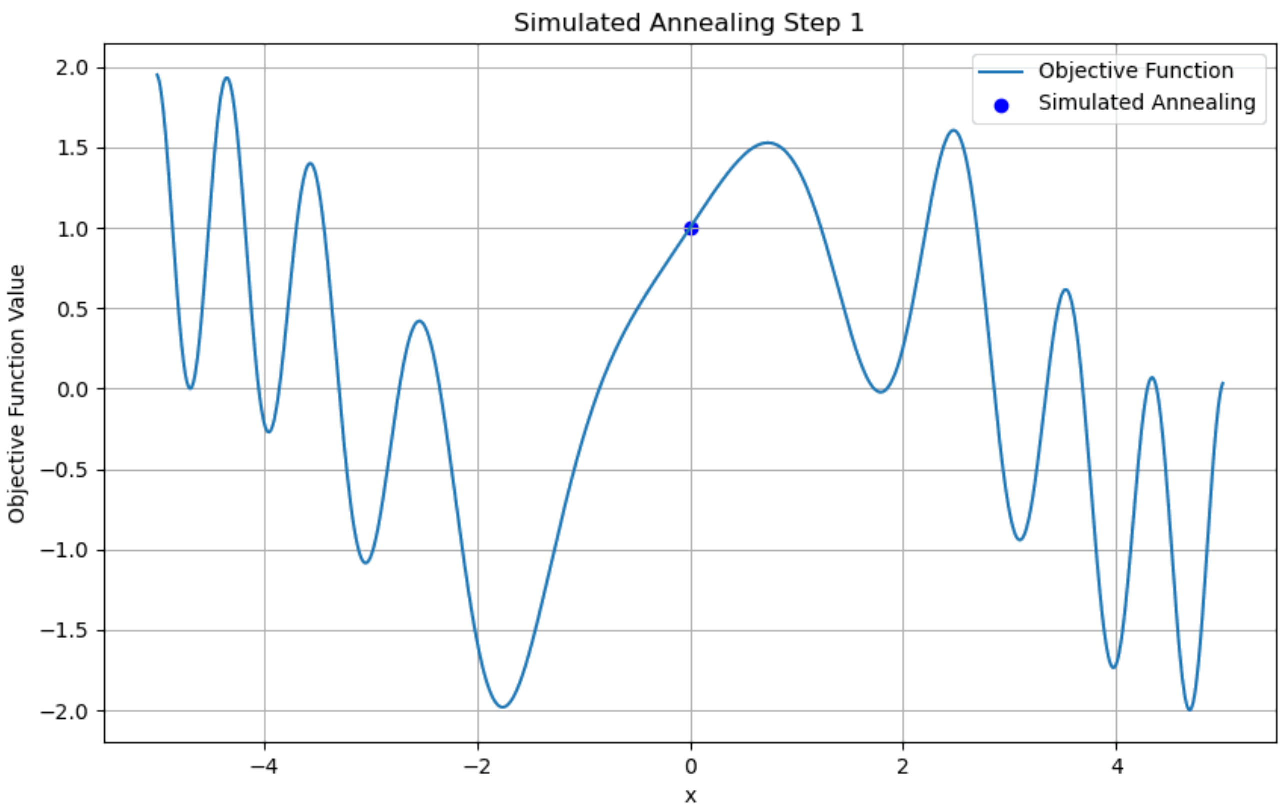


[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence a modern approach. 4th edition

# Hill-climbing



# Simulated Annealing



# Simulated Annealing: some intuition

**function** SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  **inputs**: *problem*, a problem
    *schedule*, a mapping from time to "temperature"

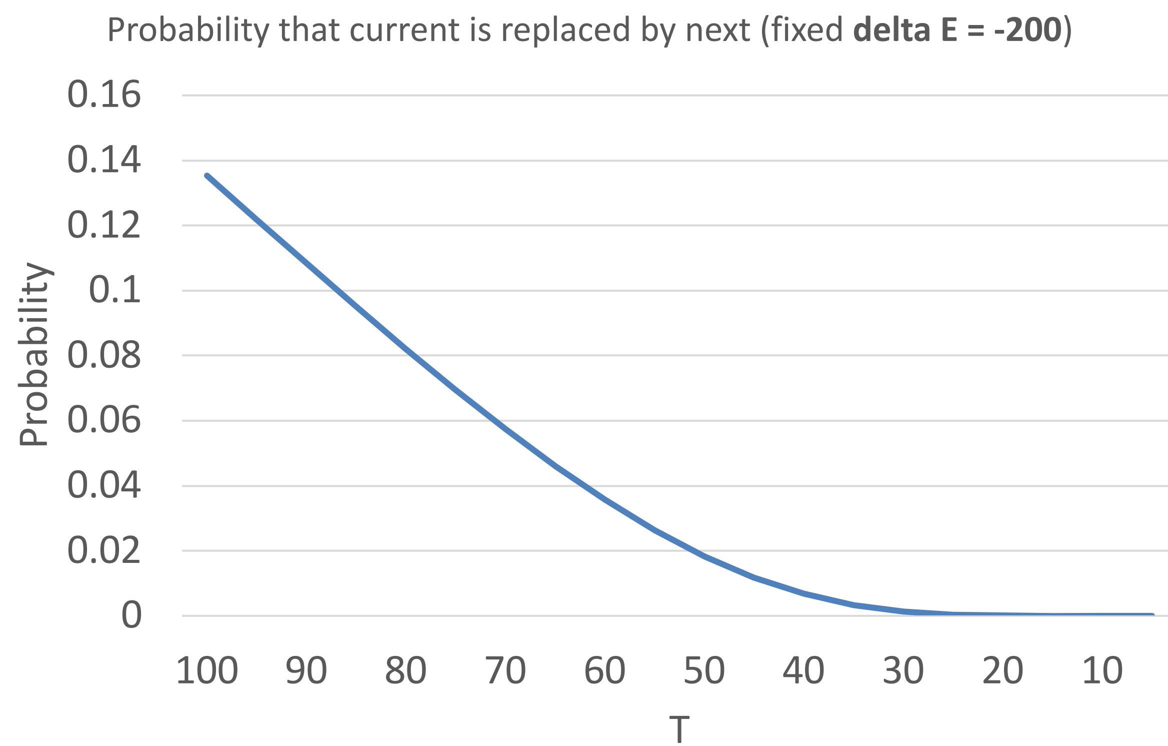
  *current* $\leftarrow$ MAKE-NODE(*problem*.INITIAL-STATE)
  **for** $t = 1$ **to** $\infty$ **do**
    $T \leftarrow$ *schedule*($t$)
    **if** $T = 0$ **then return** *current*
    *next* $\leftarrow$ a randomly selected successor of *current*
    $\Delta E \leftarrow$ *next*.VALUE – *current*.VALUE
    **if** $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
    **else** *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

**if** $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
**else** *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

# Simulated Annealing: some intuition

**if** $\Delta E > 0$ **then** *current* $\leftarrow$ *next*
**else** *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

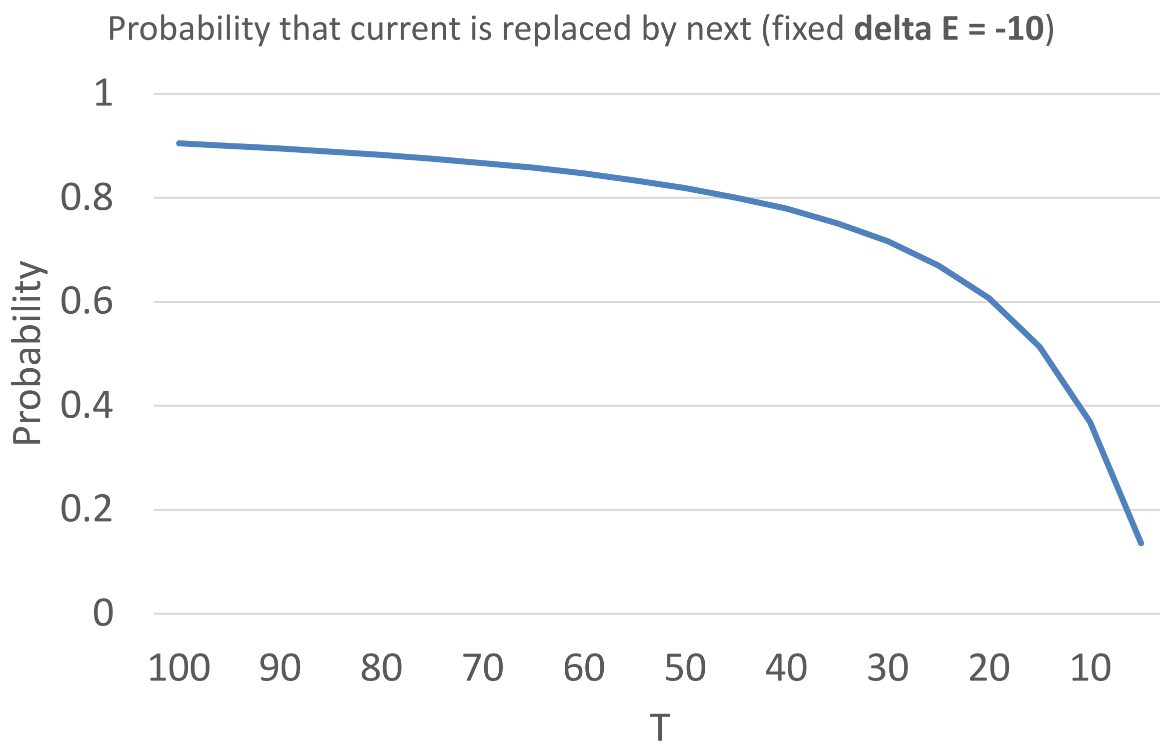
| delta E | T | delta E / T | e ^ (delta E / T) |
|---|---|---|---|
| -200 | 100 | -2 | 0.135335283 |
| -200 | 95 | -2.1052632 | 0.121813614 |
| -200 | 90 | -2.2222222 | 0.108368023 |
| -200 | 85 | -2.3529412 | 0.095089077 |
| -200 | 80 | -2.5 | 0.082084999 |
| -200 | 75 | -2.6666667 | 0.069483451 |
| -200 | 70 | -2.8571429 | 0.057432619 |
| -200 | 65 | -3.0769231 | 0.046100888 |
| -200 | 60 | -3.3333333 | 0.035673993 |
| -200 | 55 | -3.6363636 | 0.026347981 |
| -200 | 50 | -4 | 0.018315639 |
| -200 | 45 | -4.4444444 | 0.011743628 |
| -200 | 40 | -5 | 0.006737947 |
| -200 | 35 | -5.7142857 | 0.003298506 |
| -200 | 30 | -6.6666667 | 0.001272634 |
| -200 | 25 | -8 | 0.000335463 |
| -200 | 20 | -10 | 4.53999E-05 |
| -200 | 15 | -13.333333 | 1.6196E-06 |
| -200 | 10 | -20 | 2.06115E-09 |
| -200 | 5 | -40 | 4.24835E-18 |



# Simulated Annealing: some intuition

**if** $\Delta E > 0$ **then** $current \leftarrow next$
**else** $current \leftarrow next$ only with probability $e^{\Delta E/T}$

| delta E | T | delta E / T | e ^ (delta E / T) |
|---|---|---|---|
| -10 | 100 | -0.1 | 0.904837418 |
| -10 | 95 | -0.1052632 | 0.900087626 |
| -10 | 90 | -0.1111111 | 0.894839317 |
| -10 | 85 | -0.1176471 | 0.889009765 |
| -10 | 80 | -0.125 | 0.882496903 |
| -10 | 75 | -0.1333333 | 0.875173319 |
| -10 | 70 | -0.1428571 | 0.8668779 |
| -10 | 65 | -0.1538462 | 0.857403919 |
| -10 | 60 | -0.1666667 | 0.846481725 |
| -10 | 55 | -0.1818182 | 0.833752918 |
| -10 | 50 | -0.2 | 0.818730753 |
| -10 | 45 | -0.2222222 | 0.800737403 |
| -10 | 40 | -0.25 | 0.778800783 |
| -10 | 35 | -0.2857143 | 0.751477293 |
| -10 | 30 | -0.3333333 | 0.716531311 |
| -10 | 25 | -0.4 | 0.670320046 |
| -10 | 20 | -0.5 | 0.60653066 |
| -10 | 15 | -0.6666667 | 0.513417119 |
| -10 | 10 | -1 | 0.367879441 |
| -10 | 5 | -2 | 0.135335283 |



# Local Beam Search

- Focus on the solution, not the path
- It works as a **parallel search**, where the nodes added to the frontier can be abandoned
- In practical terms, we keep a fixed number of “options” or “candidates” in the frontier to be explored next
- We can’t backtrack

# Summary

- See Chapter 4 (precisely 4.1 Local Search and Optimization problems) [1]
- What about Gradient Descent?!
- What about Genetic Algorithms?!
- Convex optimization, Dynamic programming, Branch and bound, …

[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence a modern approach. 4th edition

# Coming up next…

- Probability theory and Neural Networks (next week)
- History AI (Friday Tutorial) – Prof Mengjie Zhang