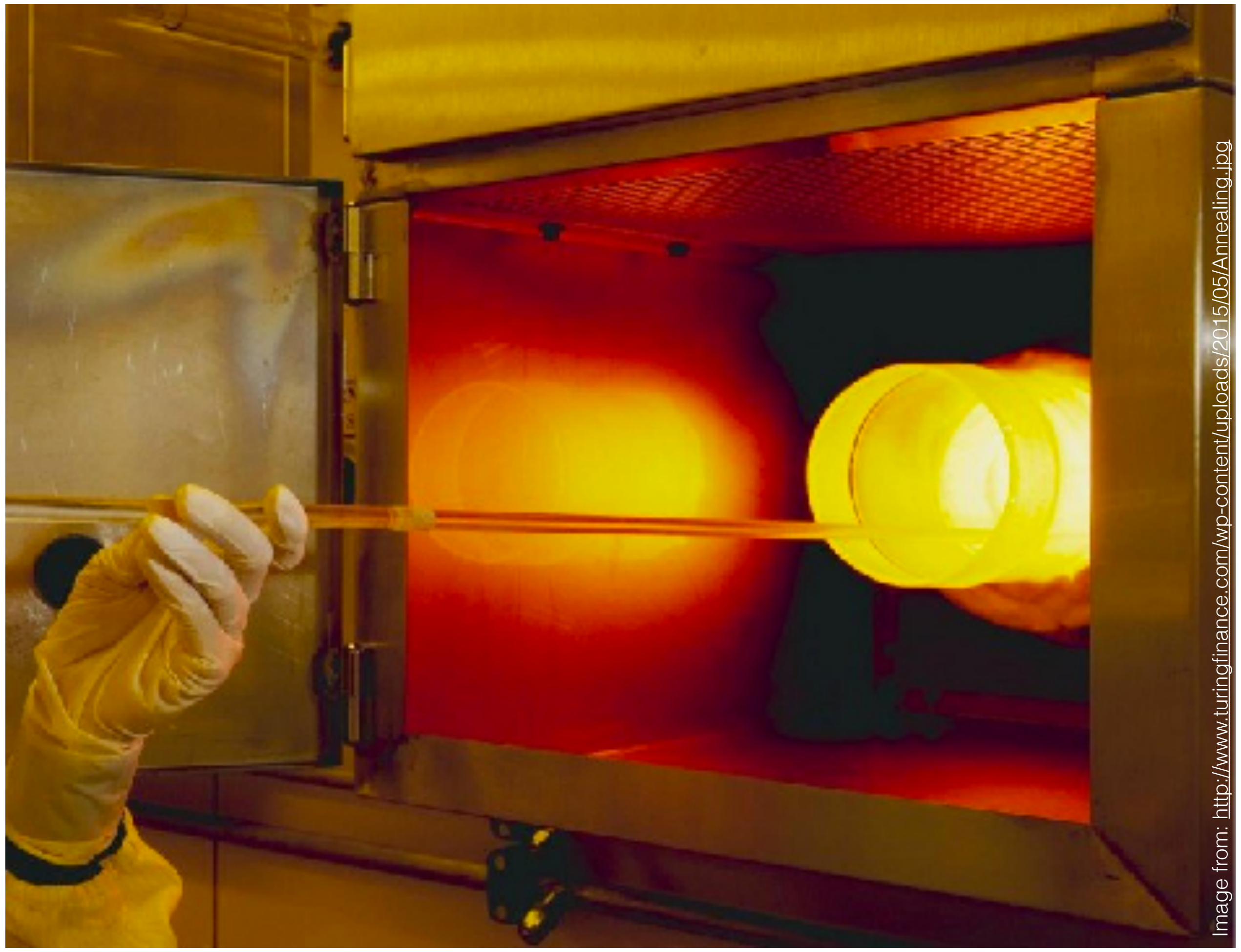
CO3091 - Computational Intelligence and Software Engineering

Lecture 03



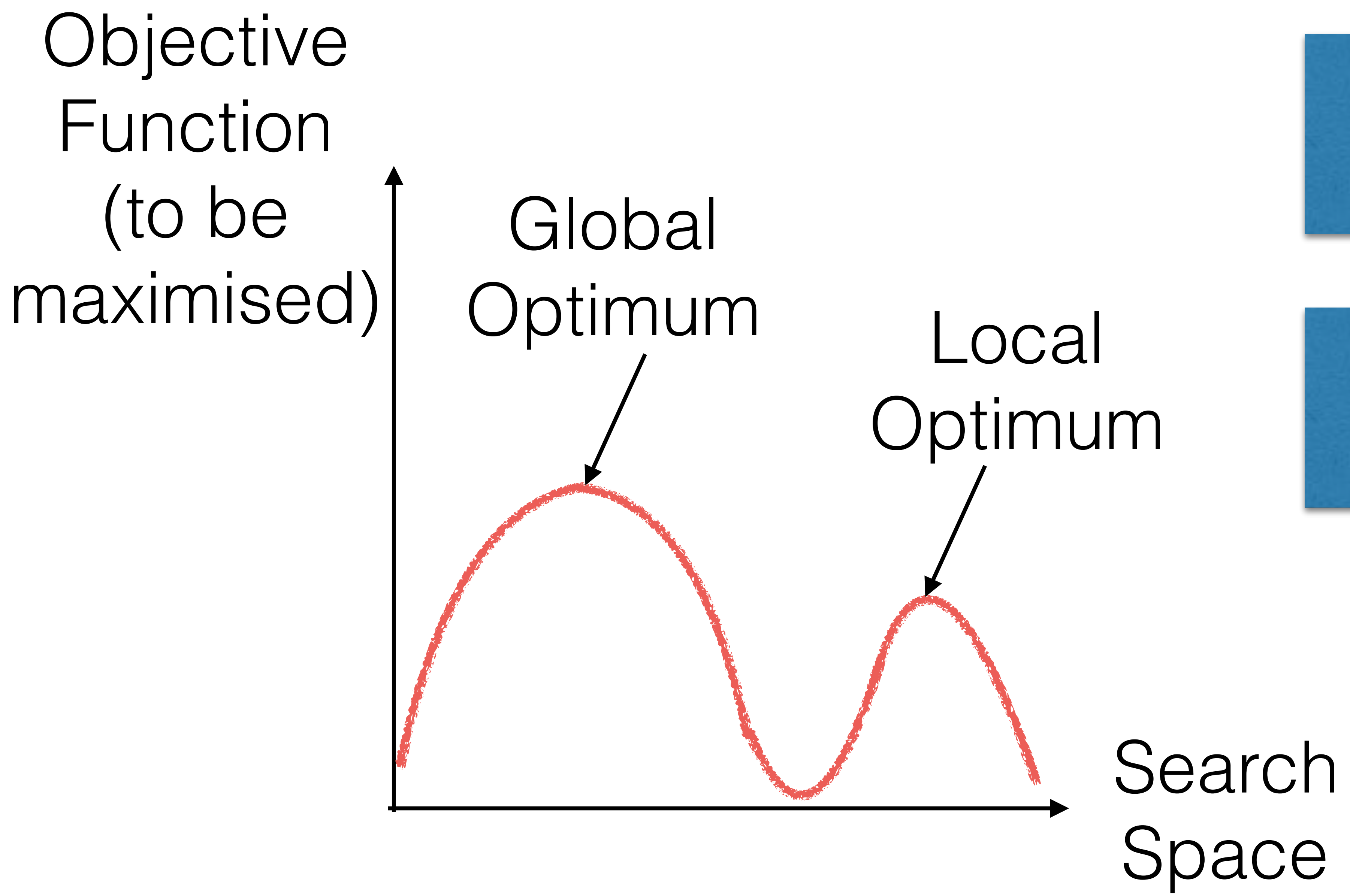
Simulated Annealing

Leandro L. Minku

Overview

- Motivation for Simulated Annealing
- Simulated Annealing
- Examples of Applications

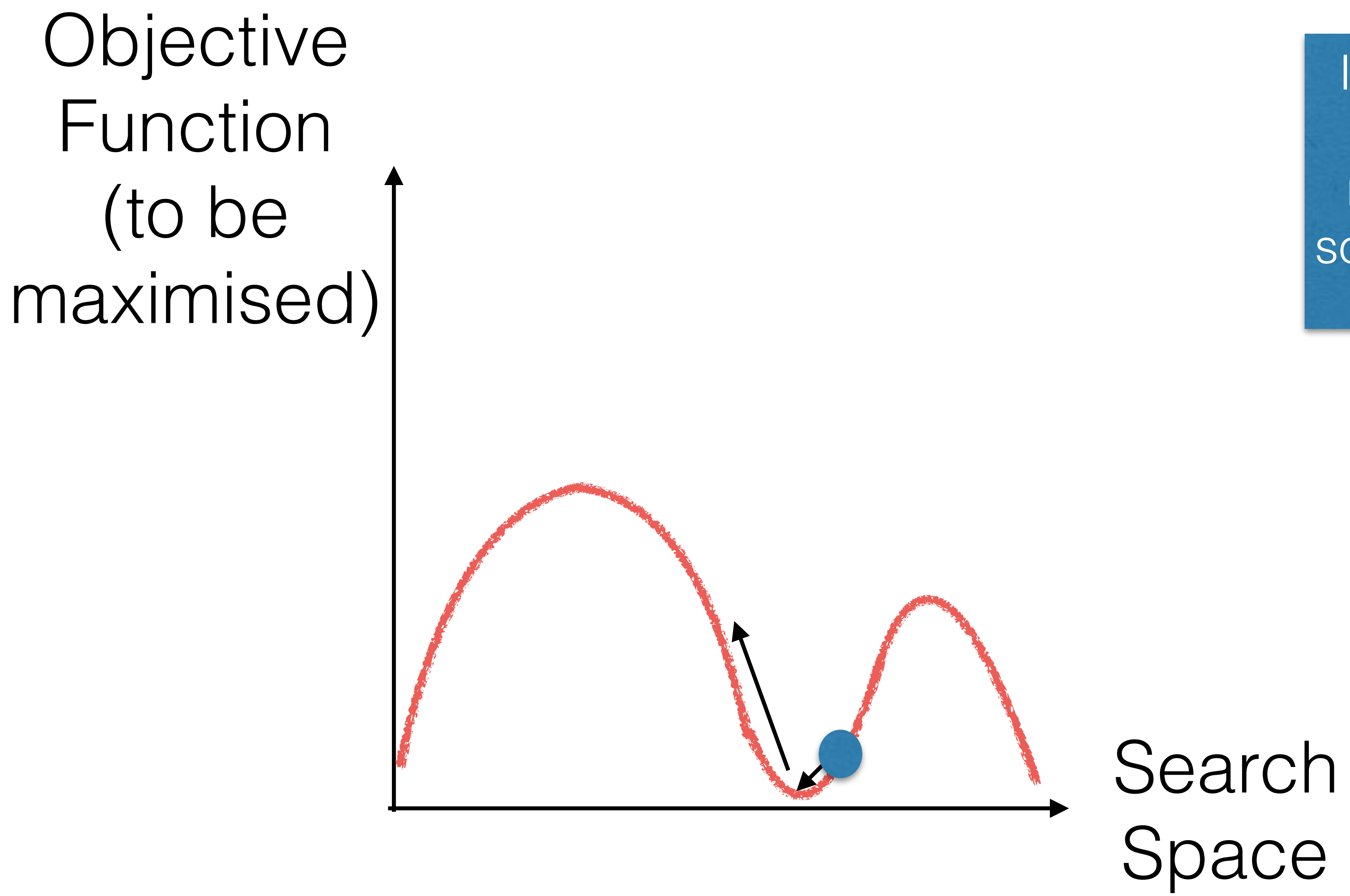
Motivation



Hill-climbing may get trapped in a local optimum.

Heuristic = informed guess

Motivation



If we could sometimes accept a downward move, we would have some chance to move to another hill.

Hill-Climbing

Hill-Climbing (assuming maximisation)

1. current_solution = generate initial solution randomly

2. Repeat:

    2.1 generate neighbour solutions (differ from current solution by a single element)

    2.2 best_neighbour = get highest quality neighbour of current_solution

    2.3 If quality(best_neighbour) <= quality(current_solution)

        2.3.1 Return current_solution

    2.4 current_solution = best_neighbour

In simulated annealing, instead of taking the best neighbour, we pick a random neighbour.

Hill-Climbing

Hill-Climbing (assuming maximisation)

1. current_solution = generate initial solution randomly

2. Repeat:

    2.1 generate neighbour solutions (differ from current solution by a single element)

    2.2 best_neighbour = get highest quality neighbour of current_solution

    2.3 If quality(best_neighbour) <= quality(current_solution)

        2.3.1 Return current_solution

    2.4 current_solution = best_neighbour

Simulated annealing will give some chance to accept a bad neighbour.

Simulated Annealing

Simulated Annealing (assuming maximisation)

1. current_solution = generate initial solution randomly

2. Repeat:

   2.1 generate neighbour solutions (differ from current solution by a single element)

   2.2 rand_neighbour = get random neighbour of current_solution

   2.3 If quality(rand_neighbour) <= quality(current_solution)

      2.3.1 With some probability,
      current_solution = rand_neighbour

   Else current_solution = rand_neighbour

Simulated Annealing

Simulated Annealing (assuming maximisation)

1. current_solution = generate initial solution randomly

2. Repeat:

   2.1 generate neighbour solutions (differ from current solution by a single element)

   2.2 rand_neighbour = get random neighbour of current_solution

   2.3 If quality(rand_neighbour) <= quality(current_solution)

      2.3.1 With some probability,
      current_solution = rand_neighbour

   Else current_solution = rand_neighbour

How Should the Probability be Set?

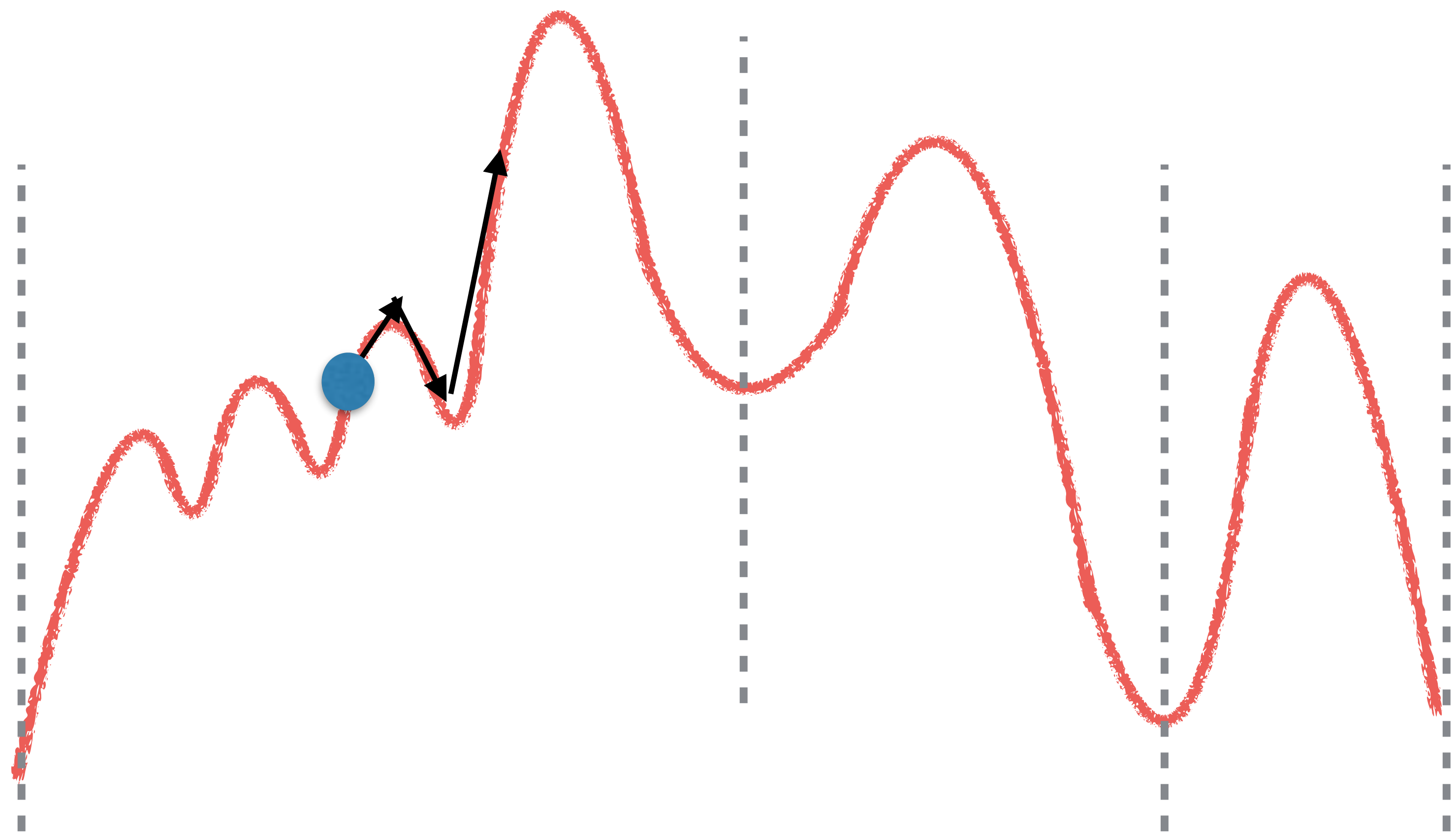
- Probability to accept solutions with much worse quality should be lower.
  - We don't want to be dislodged from the optimum.

- High probability in the beginning.
  - More similar effect to random search.
  - Allows us to explore the search space.

- Lower probability as time goes by.
  - More similar effect to hill-climbing.
  - Allows us to exploit a hill.

How to Decrease the Probability?

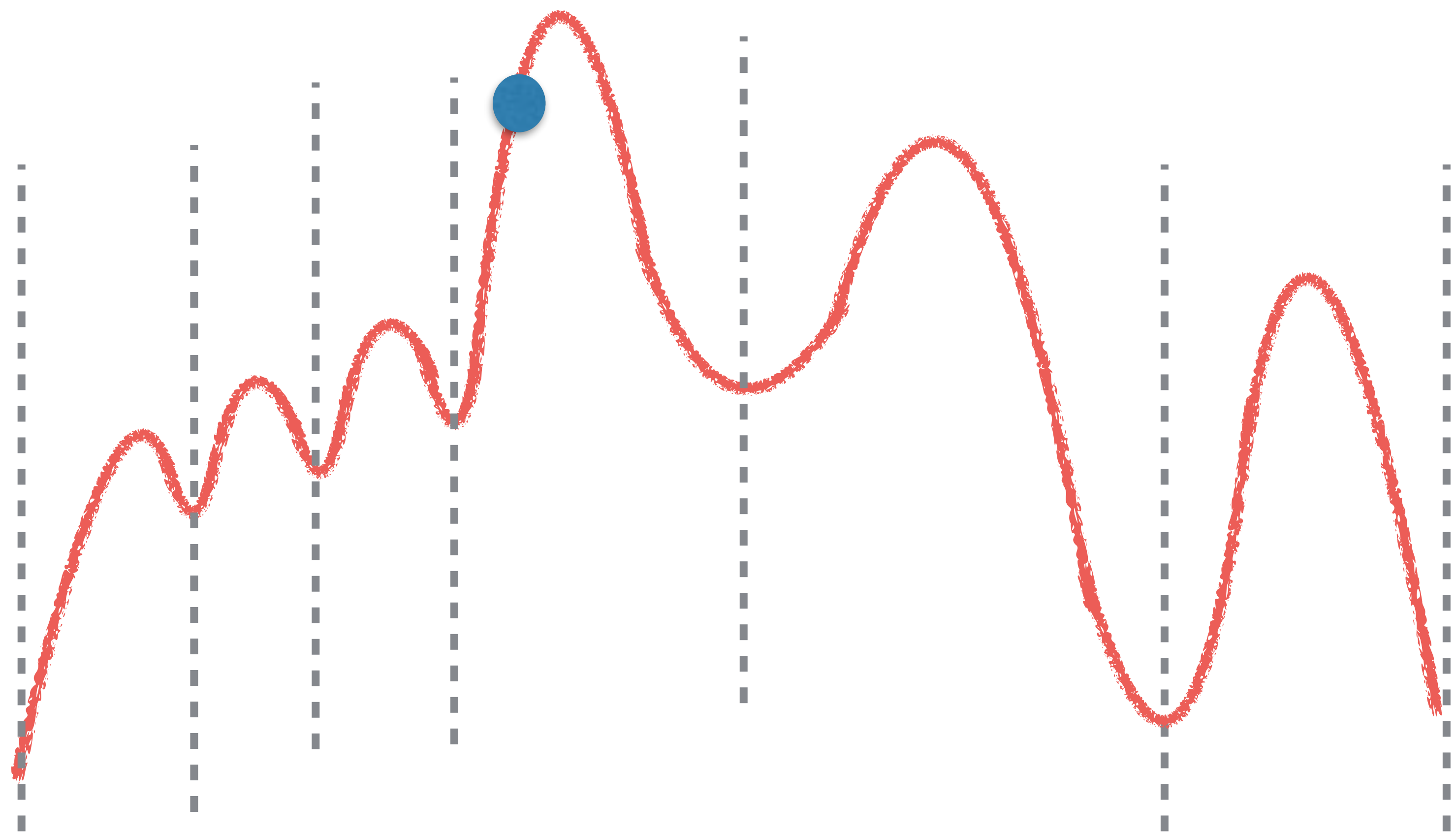
- We would like to decrease the probability slowly.



If you decrease the probability slowly, you start to form basis of attraction, but you can still walk over small hills initially.

How to Decrease the Probability?

- We would like to decrease the probability slowly.



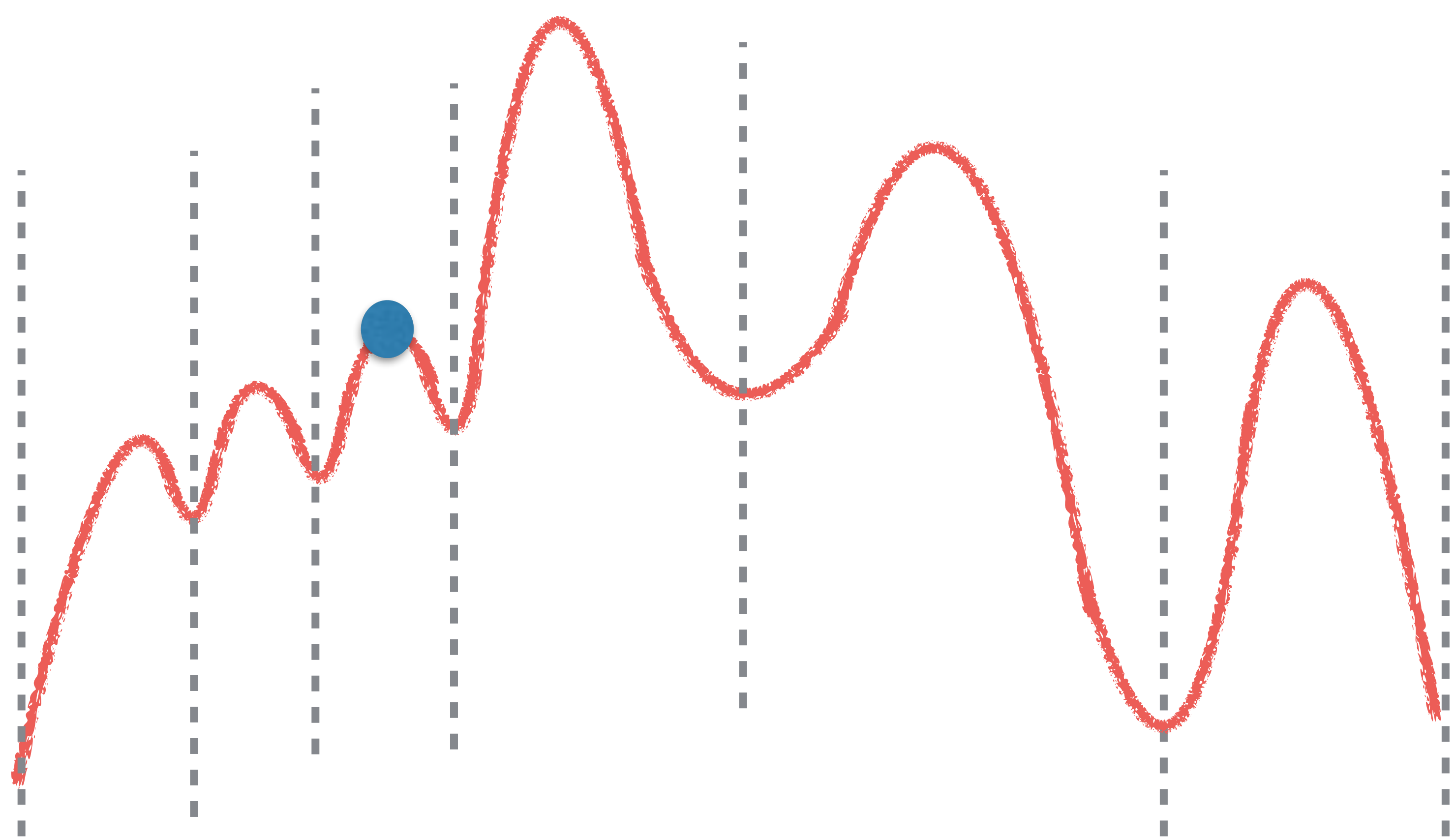
As the probability decreases further, the small hills start to form basis of attraction too.

But if you do so slowly enough, you give time to wander to the higher value hills before starting to exploit.

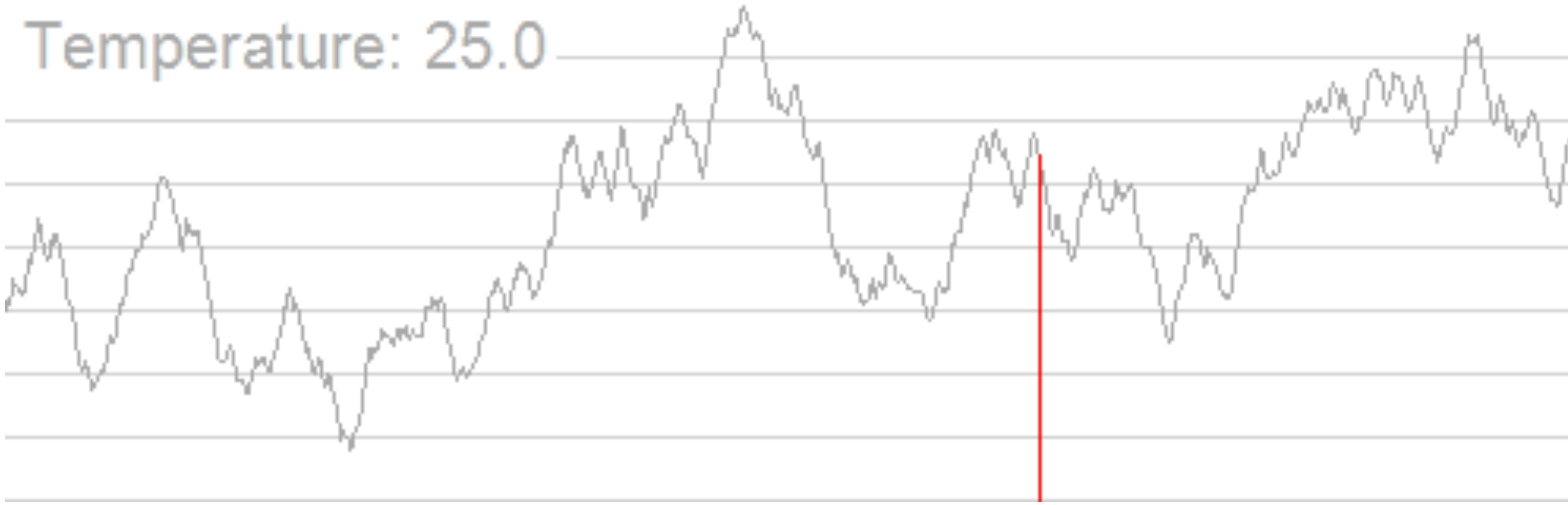
So, you can find the global optimum!

How to Decrease the Probability?

- We would like to decrease the probability slowly.



If you decrease too quickly, you can get trapped in local optima.



[By Kingpin13 - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=25010763]

Simulated Annealing

Simulated Annealing (assuming maximisation)

1. current_solution = generate initial solution randomly

2. Repeat:

    2.1 generate neighbour solutions (differ from current solution by a single element)

    2.2 rand_neighbour = get random neighbour of current_solution

    2.3 If quality(rand_neighbour) <= quality(current_solution)

        2.3.1 With some probability,
        current_solution = rand_neighbour

        Else current_solution = rand_neighbour

    2.4 Reduce probability

Metallurgy Annealing

- A blacksmith heats the metal to a very high temperature.
- When heated, the steel's atoms can move fast and randomly.

Image from: http://www.stormthecastle.com/indeximages/sting-steel-thumb.jpg

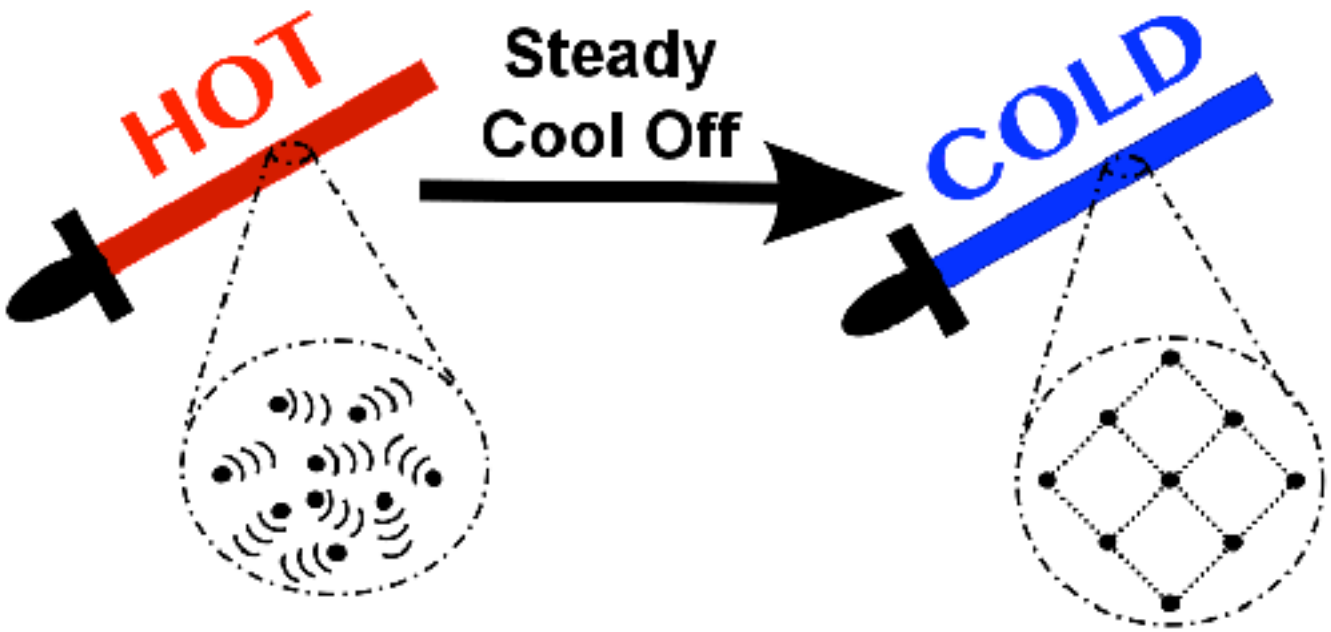


Image from: http://2.bp.blogspot.com/--kOlrodykkg/UbfVZ0_I5HI/AAAAAAAAAJ4/0rQ98g6tDDA/s1600/annealingAtoms.png

- The blacksmith then lets it cool down slowly.
- If cooled down at the right speed, the atoms will settle in nicely.
- This makes the sword stronger than the untreated steel.

Probability Function

Probability of accepting a solution of equal or worse quality, inspired by thermodynamics:

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

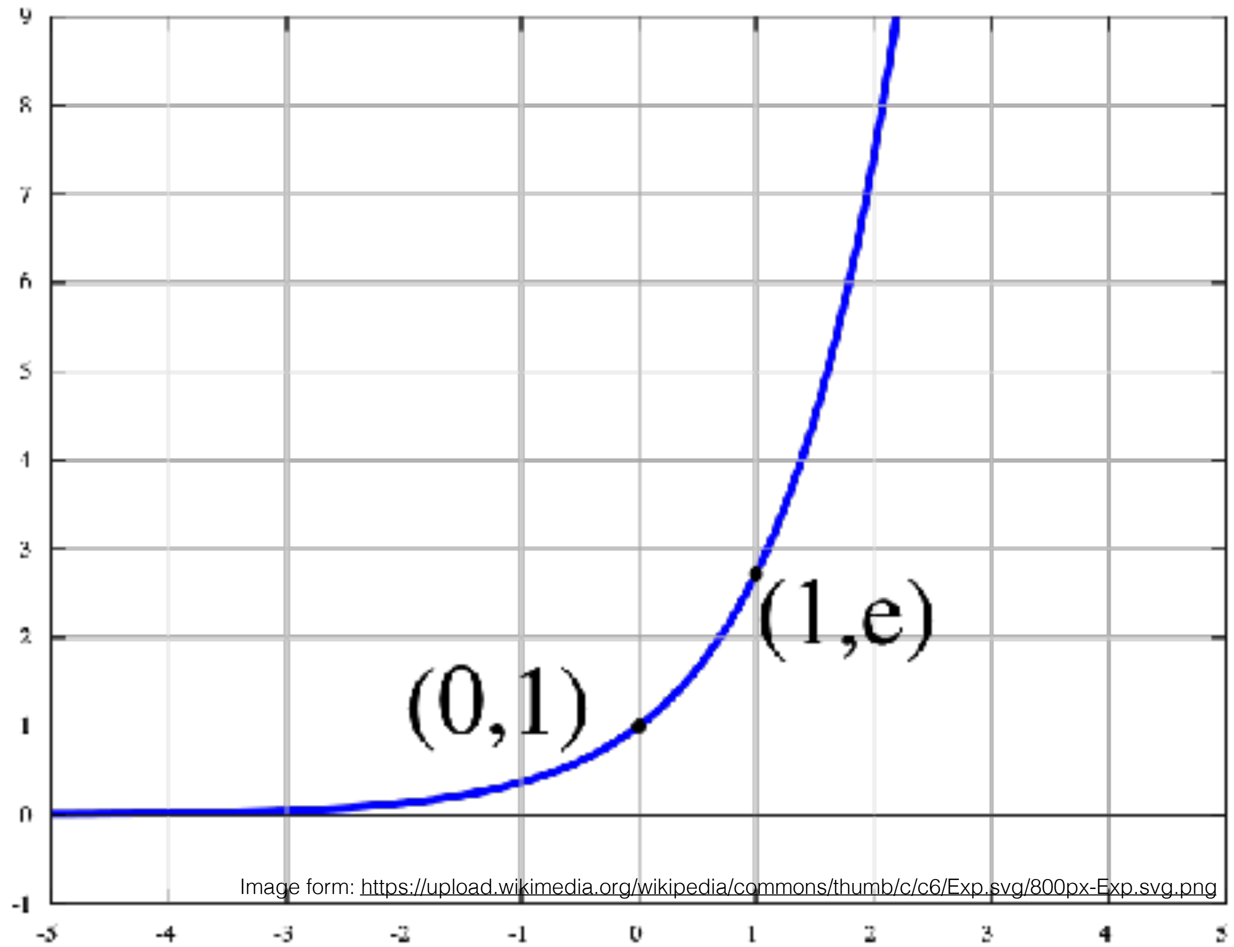
ΔE = quality(rand_neighbour) - quality(current_solution)
(<=0)

Assuming maximisation…

T = temperature
(>0)

Exponential Function

$e^{\Delta E/T}$

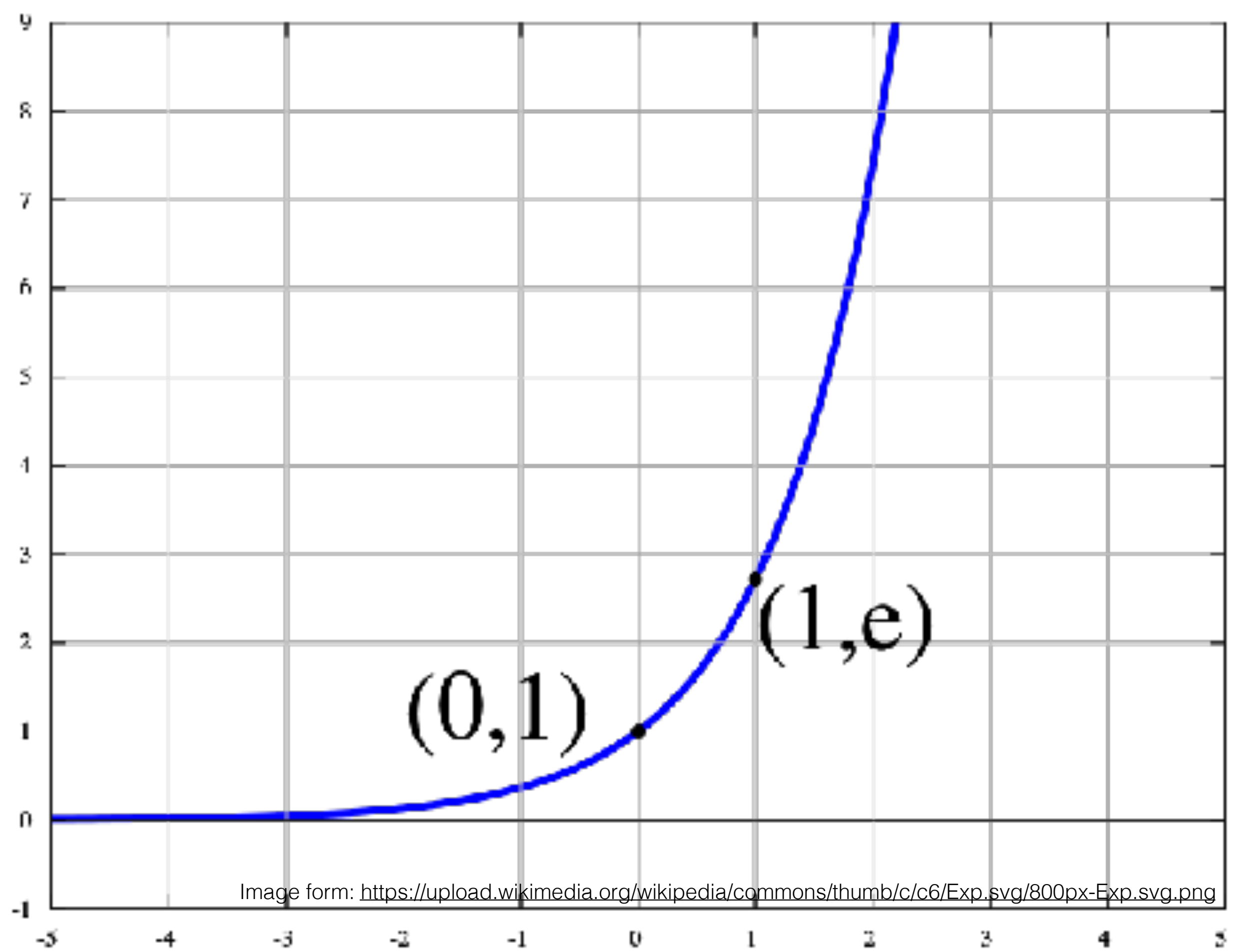


$\Delta E/T$

Exponential Function

$e^{\Delta E/T}$

$e^{\Delta E/T}$



$\Delta E/T$

Image form: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Exp.svg/800px-Exp.svg.png

Exponential Function

$e^{\Delta E/T}$

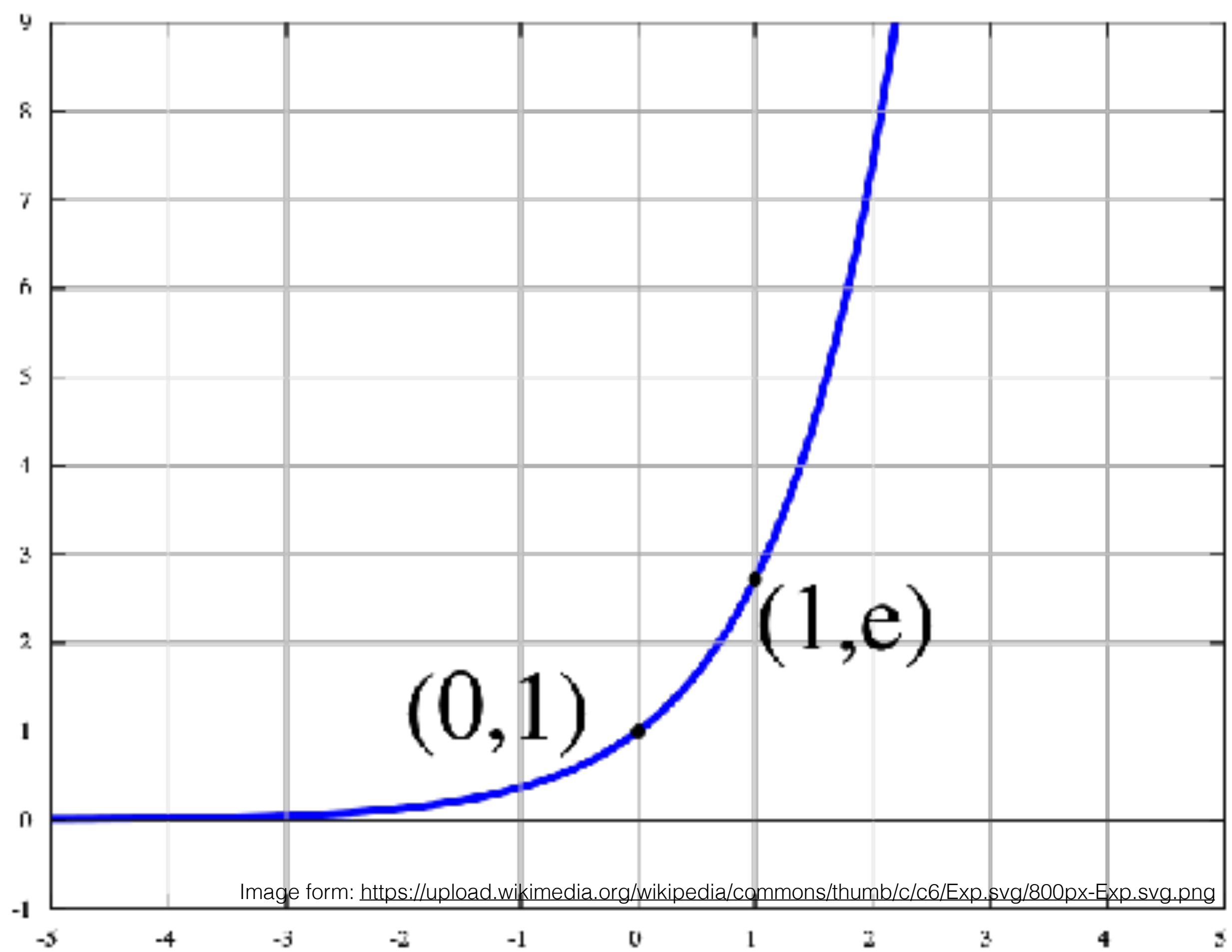


Image form: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c6/Exp.svg/800px-Exp.svg.png

$\Delta E/T$

$e^{\Delta E/T}$

But never reaches zero

How Does ΔE Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

ΔE = quality(rand_neighbour) - quality(current_solution)

(<=0)

Assuming maximisation…

T = temperature

(>0)

The worse the neighbour is in comparison to the current solution, the less likely to accept it.

How Does ΔE Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

But never reaches zero

ΔE = quality(rand_neighbour) - quality(current_solution)

(<=0)

Assuming maximisation…

T = temperature

(>0)

We always have some probability to accept a bad neighbour, no matter how bad it is.

How Does ΔE Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

ΔE = quality(rand_neighbour) - quality(current_solution)
(<=0)

Assuming maximisation…

T = temperature
(>0)

The better the neighbour is, the more likely to accept it.

How Should the Probability be Set?

- **Probability to accept solutions with much worse quality should be lower.**
  - **We don't want to be dislodged from the optimum.**

- High probability in the beginning.
  - More similar effect to random search.
  - Allows us to explore the search space.

- Lower probability as time goes by.
  - More similar effect to hill-climbing.
  - Allows us to exploit a hill.

How Does T Affect the Probability?

Probability of accepting a solution of **equal or worse quality**:

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

<=0

ΔE = quality(rand_neighbour) - quality(current_solution)

(<=0)

Assuming maximisation…

T = temperature

(>0)

How Does T Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

<=0

ΔE = quality(rand_neighbour) - quality(current_solution)
(<=0)

Assuming maximisation…

T = temperature
(>0)

If T is higher, the probability of accepting the neighbour is higher.

How Does T Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

<=0

ΔE = quality(rand_neighbour) - quality(current_solution)
(<=0)

Assuming maximisation…

T = temperature
(>0)

If T is lower, the probability of accepting the neighbour is lower.

How Does T Affect the Probability?

Probability of accepting a solution of equal or worse quality:

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

<=0

ΔE = quality(rand_neighbour) - quality(current_solution)
(<=0)

Assuming maximisation…

T = temperature
(>0)

So, reducing the temperature over time would
reduce the probability of accepting the neighbour.

How Should the Temperature be Set?

- High probability in the beginning.
  - More similar effect to random search.
  - Allows us to explore the search space.
- Lower probability as time goes by.
  - More similar effect to hill-climbing.
  - Allows us to exploit a hill.


Image from: http://static.comicvine.com/uploads/original/13/130470/2931473-151295.jpg

How to Set and Reduce T?

- T starts with an initially high pre-defined value (parameter of the algorithm).
- There are different update rules (schedules)…
- Update rule:
  - $T = \alpha T$,

    α is close to, but smaller than, 1

    e.g., $\alpha = 0.95$

Simulated Annealing

Simulated Annealing (assuming maximisation)

Input: initial temperature Ti

1. current_solution = generate initial solution randomly

2. T = Ti

3. Repeat:

    3.1 generate neighbour solutions (differ from current solution by a single element)

    3.2 rand_neighbour = get random neighbour of current_solution

    3.3 If quality(rand_neighbour) <= quality(current_solution)

        3.3.1 With probability $e^{\Delta E/T}$,

            current_solution = rand_neighbour

        Else current_solution = rand_neighbour

    3.4 **T = schedule(T)**

Simulated Annealing

Simulated Annealing (assuming maximisation)

Input: initial temperature Ti, minimum temperature Tf

1. current_solution = generate initial solution randomly

2. T = Ti

3. **Repeat until a minimum temperature Tf is reached or until the current solution "stops changing"**:

    3.1 generate neighbour solutions (differ from current solution by a single element)

    3.2 rand_neighbour = get random neighbour of current_solution

    3.3 If quality(rand_neighbour) <= quality(current_solution)

        3.3.1 With probability $e^{\Delta E/T}$,

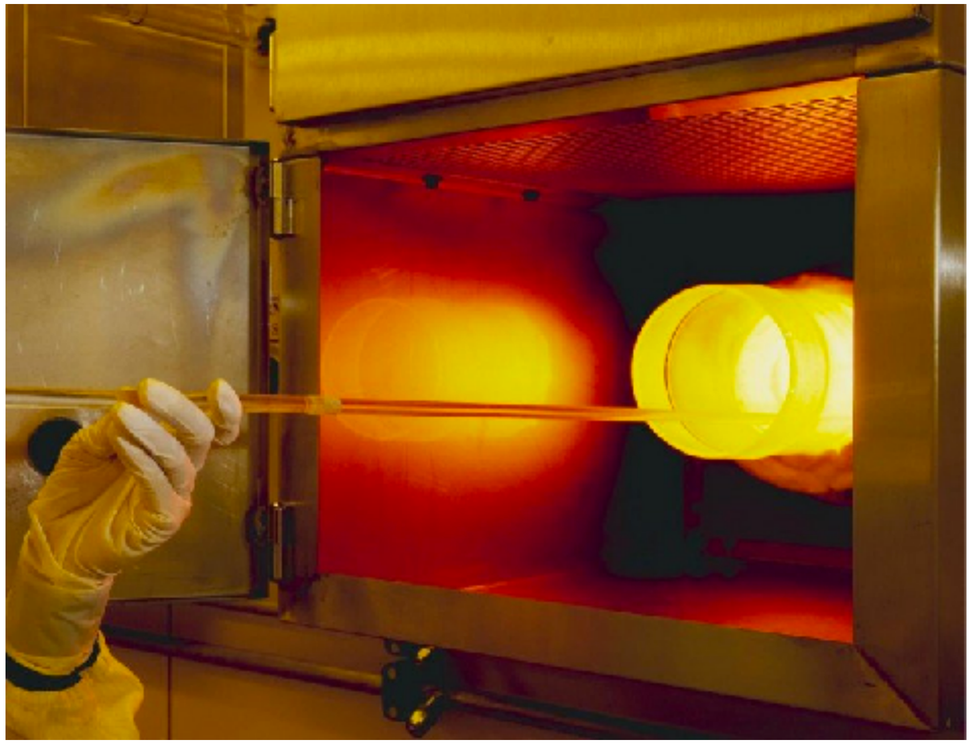
            current_solution = rand_neighbour

        Else current_solution = rand_neighbour

    3.4 **T = schedule(T)**

Local Search

- Simulated annealing can also be considered as a local search, as it allows to move only to neighbour solutions.
- However, it has mechanisms to try and escape from local optima.



Examples of Applications

- Several engineering problems, e.g.: VLSI (Very-Large-Scale Integration).
    - Process of creating an integrated circuit by combining thousands of transistors into a single chip.
    - Decide placement of transistors.
    - Objectives: reduce area, wiring and congestion.

Image from: https://upload.wikimedia.org/wikipedia/commons/9/94/VLSI_Chip.jpg

- Software engineering problems:
    - Component selection and prioritisation for the next release problem.
    - Software quality prediction.

Where Are We?

So far…

- Optimisation problems
- Brute force
- Hill climbing
- Simulated annealing

Next class: surgery.

Please revise the lectures before the surgery!

Further Reading

http://readinglists.le.ac.uk/lists/D888DC7C-0042-C4A3-5673-2DF8E4DFE225.html

Stuart J. Russell, Peter Norvig, John F. Canny
Artificial intelligence: a modern approach
Section 4.1: Local Search Algorithms and Optimization Problems - Simulated Annealing
Pearson Education
2014