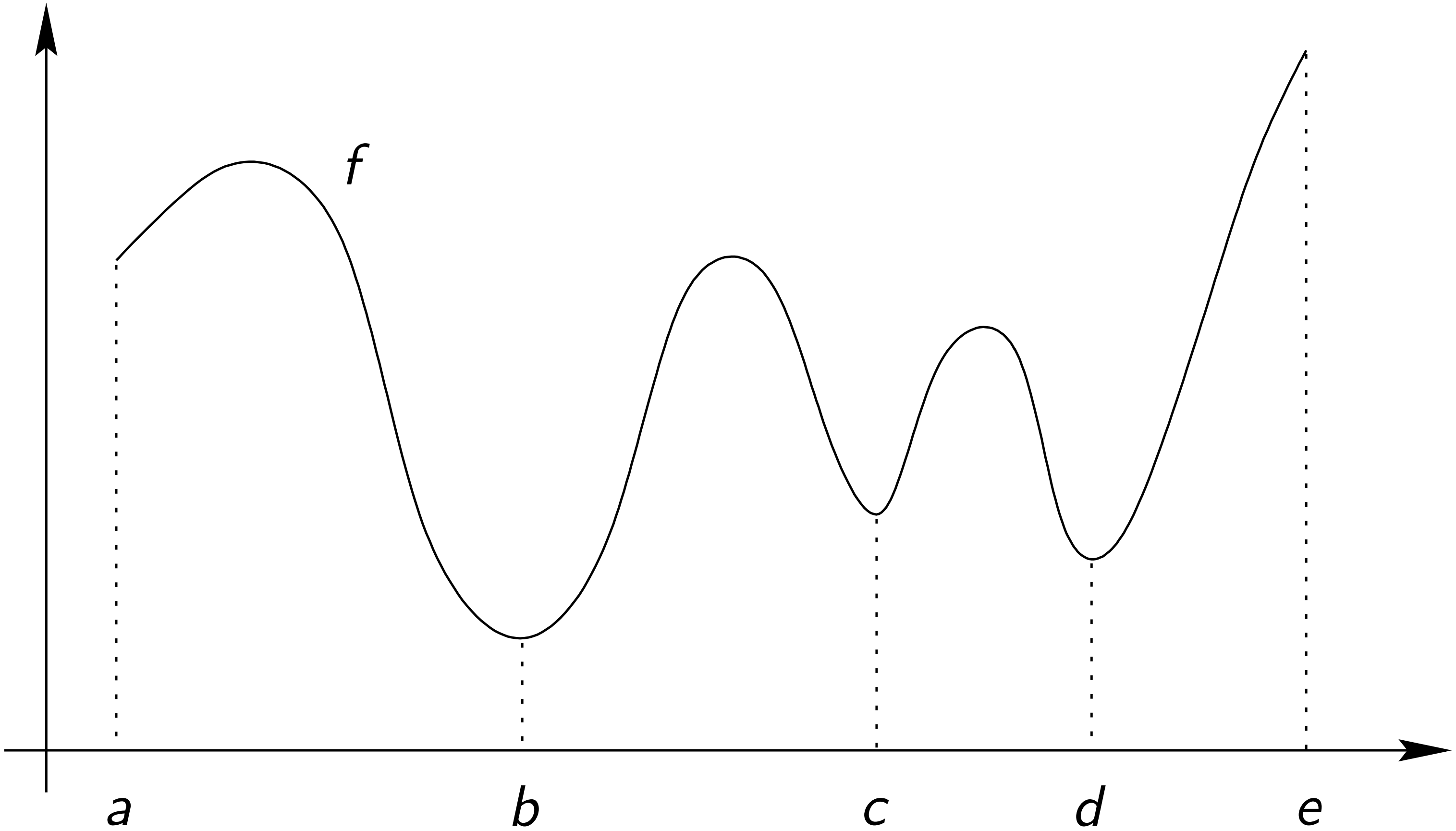
Optimization: Global optimization

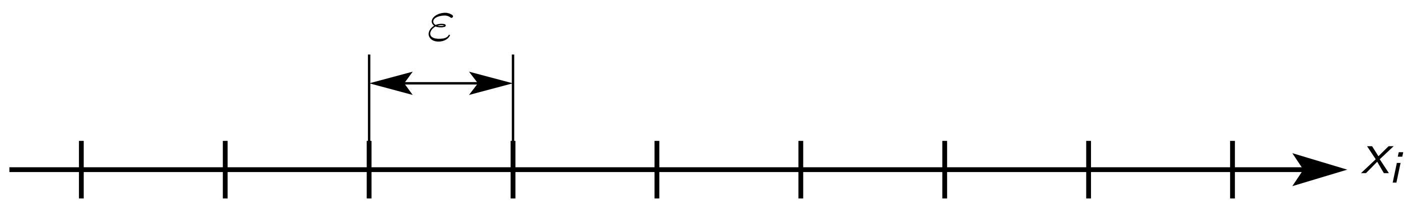
Global optimization

Local minimum versus global minimum



Finding global minimum $\rightarrow$ computationally hard problem
(cf. grid search)

Complexity of grid search



- Grid size: ε
  Number of variables: n
  $\rightarrow$ complexity $\sim \left(\dfrac{1}{\varepsilon}\right)^n$
- Example: $\varepsilon = 10^{-4}$, $n = 50 \rightarrow$ complexity $\sim 10^{200}$
- So what ... just use fast computer with parallel processors

Complexity of grid search (continued)

- Processor at size of proton
- $\#$ processors $= \#$ protons in universe
- Clock period $=$ time for light to traverse proton
- Run time is $10 \times$ current age of universe

Number of evaluations $\approx 10^{168} \quad \leftrightarrow \quad 10^{200}$ required

$\Rightarrow$ grid search fails even for relatively small-sized problems

Global optimization

Global optimization methods:

- Random search
- Multi-start local optimization
- Simulated annealing
- Genetic algorithm

Characteristics:

- Convergence to global minimum not guaranteed
- But yield “good” solutions on the average

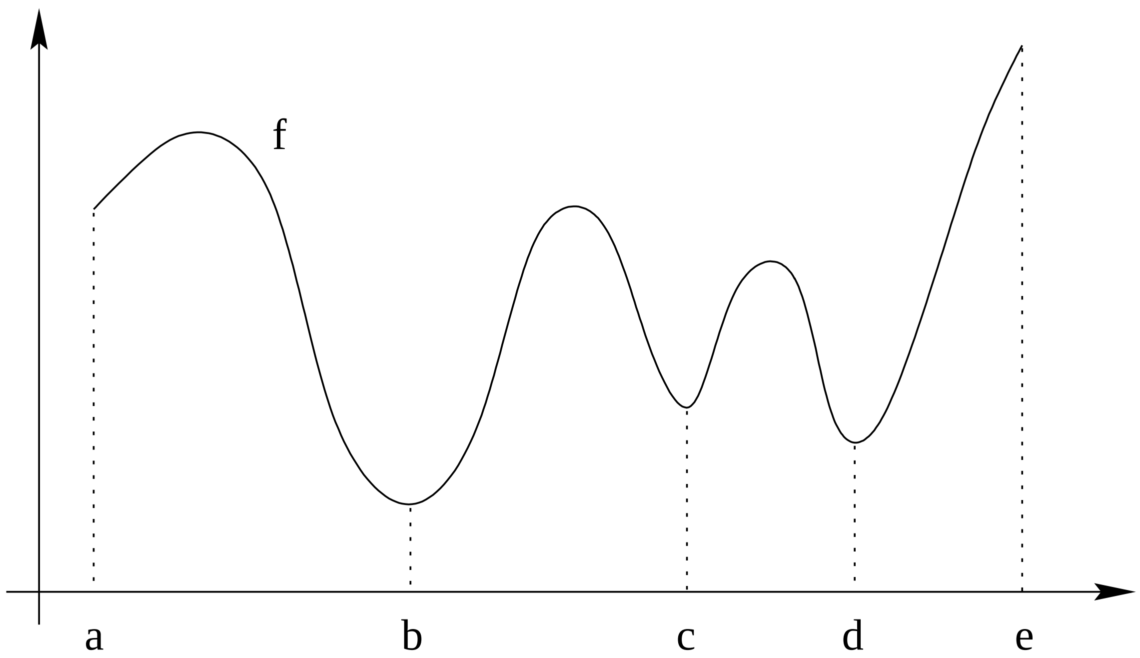
Random search & Multi-start local minimization

Random search:

- Select random points, uniformly distributed over feasible set
- Evaluate objective function values
- Keep point with lowest value

Multi-start local minimization:

- Select random starting points, uniformly distributed over feasible set
- Run local minimization algorithm
- Keep point with lowest value



Simulated annealing

Simulated annealing:

$$\min_x f(x) \quad \text{s.t.} \quad x \in \mathcal{G}$$

$\rightarrow$ mimic annealing of metal ($f =$ energy)

- Molten metal, liquid $\rightarrow$ particles arranged randomly
- Cooling $\rightarrow$ movement of particles restricted
- Slow cooling $\rightarrow$ crystal-like structure
  (minimum energy configuration)
- Fast cooling $\rightarrow$ glass-like/defects in structure
  (metastable, locally optimal structure)

- Descent algorithm = fast cooling, local optimization
- Simulated annealing = slow cooling, more global optimization

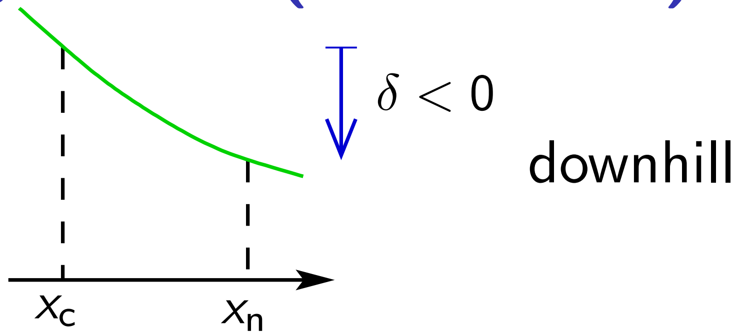
Basic simulated annealing algorithm

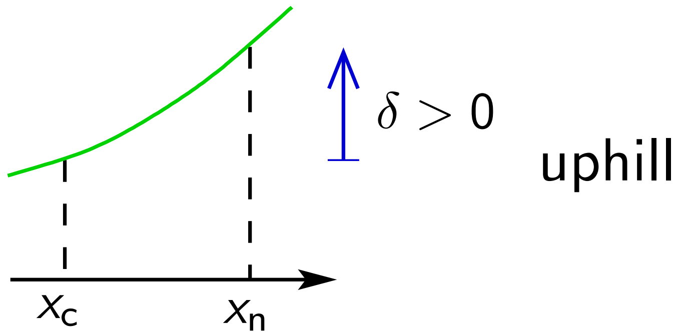
- Initialization:
  select initial point x_{current}, initial temperature $T > 0$
- Iteration:
  generate random point $x_{\text{new}} \in \mathcal{G}$ in neighborhood of x_{current}
  define $\delta = f(x_{\text{new}}) - f(x_{\text{current}})$
  if $\delta < 0$ then
      $x_{\text{current}} \leftarrow x_{\text{new}}$
  else
      select random number $r \in (0, 1)$
      if $r < \exp(-\delta/T)$ then
          $x_{\text{current}} \leftarrow x_{\text{new}}$

  every m steps: cooling (e.g. $T \leftarrow \alpha T$, $0 < \alpha < 1$)
- Stop if stopping criterion is satisfied (e.g. maximum number of steps reached)

Basic simulated annealing algorithm (continued)

. . .
$\delta = f(x_{\text{new}}) - f(x_{\text{current}})$
if $\delta < 0$ then
  $x_{\text{current}} \leftarrow x_{\text{new}}$
else
  select random number $r \in (0, 1)$
  if $r < \exp(-\delta/T)$ then
    $x_{\text{current}} \leftarrow x_{\text{new}}$
. . .





Uphill move if $r < \exp(-\delta/T)$

Value of $\exp(-\delta/T)$:

$$\begin{array}{lcl} \delta > 0,\ \text{large} & \to & \pm 0 \\ \delta > 0,\ \text{small} & \to & \pm 1 \\ & & \\ T\ \text{high} & \to & \pm 1 \\ T\ \text{low} & \to & \pm 0 \end{array}$$

Basic simulated annealing algorithm (continued)

Uphill move if $r < \exp(-\delta/T)$

Value of $\exp(-\delta/T)$:

$$
\begin{array}{lcl}
\delta > 0, \text{ large} & \rightarrow & \pm 0 \\
\delta > 0, \text{ small} & \rightarrow & \pm 1 \\
& & \\
T \text{ high} & \rightarrow & \pm 1 \\
T \text{ low} & \rightarrow & \pm 0
\end{array}
$$

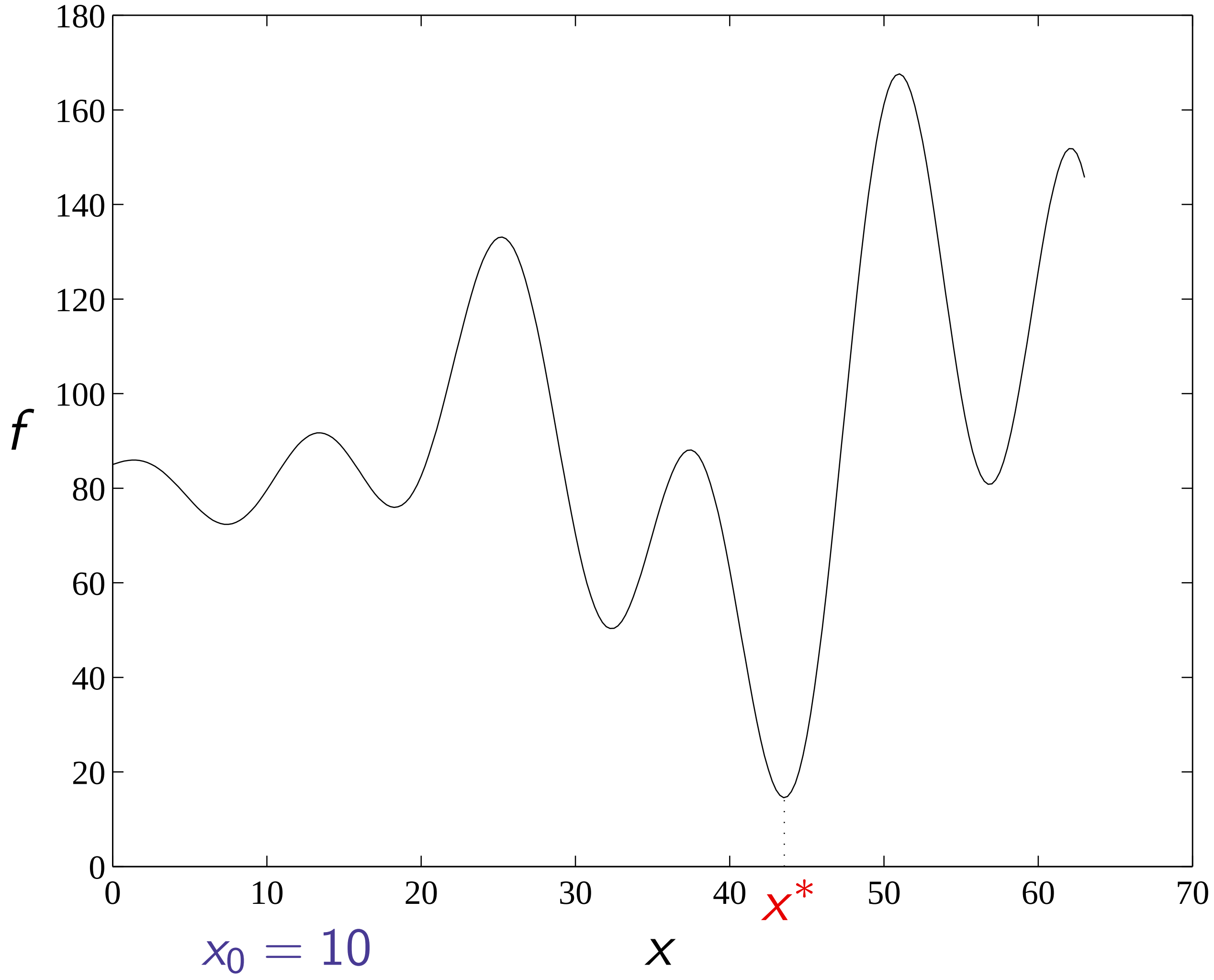
Summary:

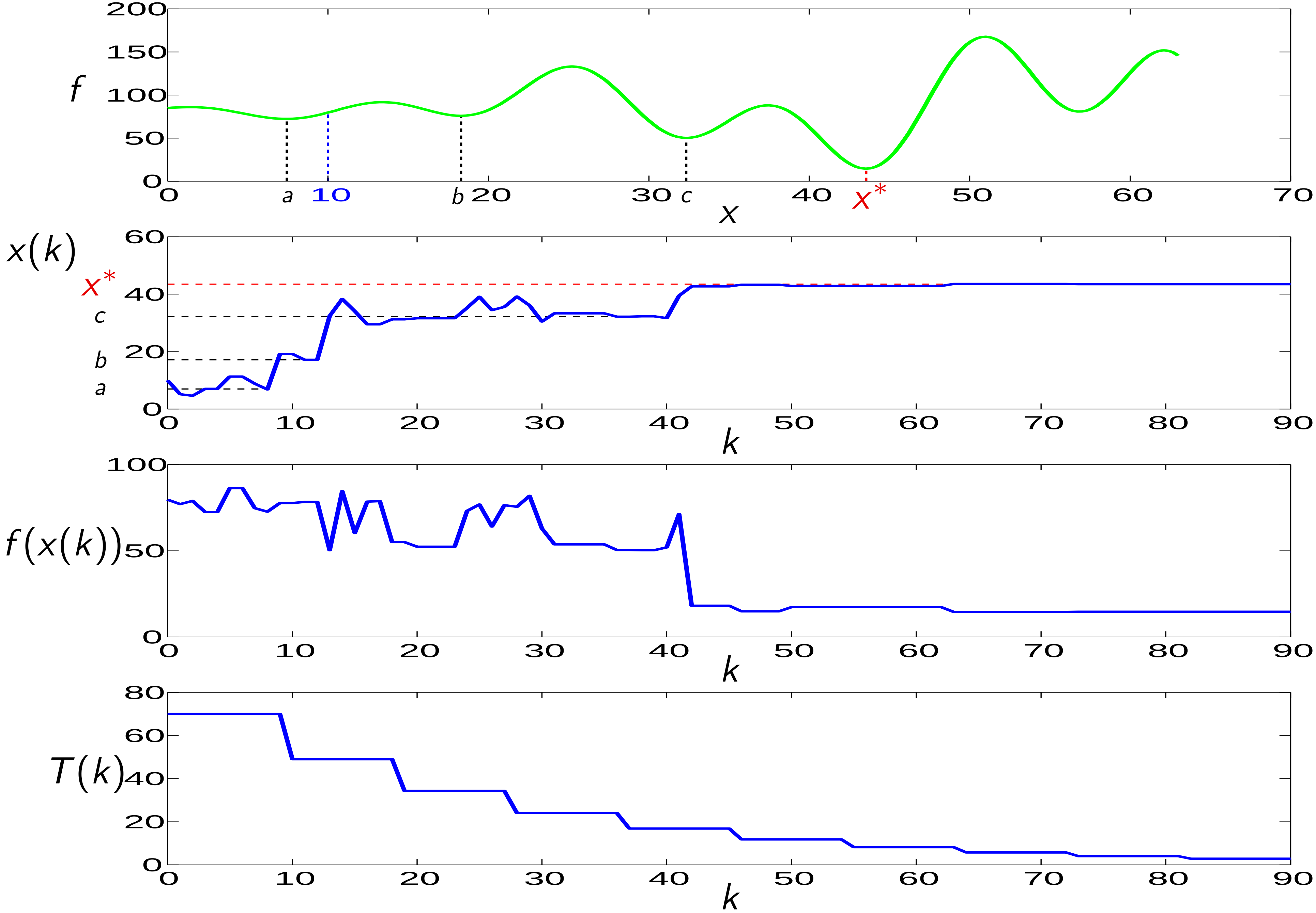
- Sometimes up-hill move (which increases f) is accepted
- Small increases more likely to be accepted
- T high $\rightarrow$ most up-hill moves accepted
- T low $\rightarrow$ most up-hill moves rejected

Example: Simulated annealing

$$\min_x \quad 85 + (1-x)\sin\frac{x}{5} + x\cos\frac{x}{2}$$

$$\text{s.t.} \quad x \in [0, 63]$$





200
150
100
50
0
f
0
a
10
b
20
30
c
40
x*
50
60
70
x
x(k)
60
x*
40
c
20
b
a
0
10
20
30
40
50
60
70
80
90
k
100
f(x(k))
50
0
k
80
60
T(k)
40
20
0
k

Simulated annealing (continued)

Characteristics of simulated annealing algorithm:

- Does not use derivatives
- Sometimes accepts move which increases f
- Can also be used for problems with discrete or real-valued parameters
- Uses probabilistic transition rules
- Can escape from local minima!!

Genetic algorithms

$$\max_x f(x) \quad \text{s.t.} \quad x \in G$$

$\rightarrow$ mimic evolution in biology ($f =$ fitness)

Code $x \in G$ as binary string

Basic genetic algorithm (assume $f > 0$):

- Initialization: set up initial population
- Iteration: create new generation
  - select parents
    probability $\sim$ relative fitness: $\dfrac{f_i}{\sum f_i}$
  - create off-spring (e.g. 1-point cross-over):

$$\begin{array}{ccc|cc} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{array} \quad \rightarrow \quad \begin{array}{ccc|cc} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{array}$$

  - mutation: flip bits with probability p_{mut}

$$1\ \underline{0}\ 1\ 1\ 1 \quad \rightarrow \quad 1\ 1\ 1\ 1\ 1$$

- Stop if maximum number of generations reached

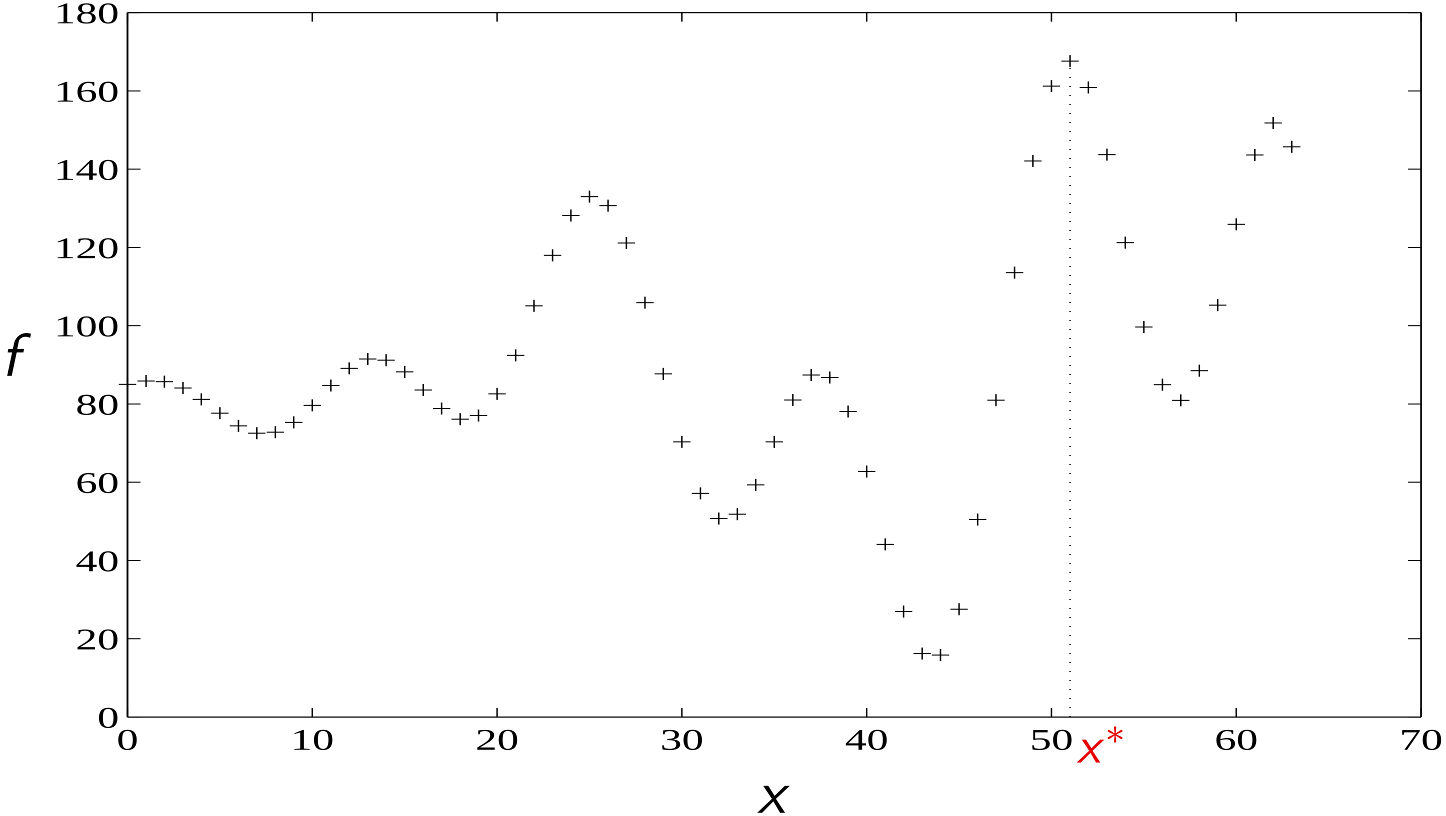
Example: Genetic algorithm

$$\max_x \; 85 + (1-x)\sin\frac{x}{5} + x\cos\frac{x}{2}$$

$$\text{s.t.} \quad x \in \{0, 1, \dots, 63\}$$

Code x as 6-bit string

e.g. $24 = 16 + 8 = 2^4 + 2^3 \;\rightarrow\; 0\,1\,1\,0\,0\,0$



Example: Genetic algorithm (continued)

Initial population:

i	x_i	b_i						$f(b_i)$	$\frac{f(b_i)}{\sum_i f(b_i)}$
1	25	0	1	1	0	0	1	132.96	0.20
2	23	0	1	0	1	1	1	117.98	0.18
3	15	0	0	1	1	1	1	88.22	0.14
4	6	0	0	0	1	1	0	74.40	0.11
5	38	1	0	0	1	1	0	86.76	0.13
6	62	1	1	1	1	1	0	151.82	0.23

Average f_i: 108.69, max f_i: 151.82

Selection of parents $\sim \dfrac{f(b_i)}{\sum f(b_i)}$:

$$p_1 = b_1,\ p_2 = b_2,\ p_3 = b_2,\ p_4 = b_6,\ p_5 = b_6,\ p_6 = b_5$$

Example: Genetic algorithm (continued)

Cross-over:

i	parent p_i	child c_i
1	0 1 1 0 0 | 1	0 1 1 0 0 | 1
2	0 1 0 1 1 | 1	0 1 0 1 1 | 1
3	0 1 0 | 1 1 1	0 1 0 | 1 1 0
4	1 1 1 | 1 1 0	1 1 1 | 1 1 1
5	1 1 | 1 1 1 0	1 1 | 0 1 1 0
6	1 0 | 0 1 1 0	1 0 | 1 1 1 0

Mutation with $p_{\text{mut}} = 0.01$ per bit:

expected # mutations: $0.01 \cdot 6 \cdot 6 = 0.36$ bits

c_5: $1\,1\,0\,\underline{1}\,1\,0 \;\rightarrow\; 1\,1\,0\,0\,1\,0$

Example: Genetic algorithm (continued)

New generation:

i	x_i	b_i						$f(b_i)$	$\frac{f(b_i)}{\sum_i f(b_i)}$
1	25	0	1	1	0	0	1	132.96	0.19
2	23	0	1	0	1	1	1	117.98	0.17
3	22	0	1	0	1	1	0	105.08	0.14
4	63	1	1	1	1	1	1	145.69	0.20
5	50	1	1	0	0	1	0	161.22	0.23
6	46	1	0	1	1	1	0	50.46	0.07

Average f_i: 118.90, max f_i: 161.22

...

Genetic algorithms (continued)

Characteristics of genetic algorithms:

- Do not use derivatives
- Work with a coding of the feasible set
- Search from a population of points
- Use probabilistic transition rules
- Can escape from local minima!!

Summary

- Global optimization: not tractable
- Use algorithms that yield “good” solutions on the average:
    - Multi-start local optimization
    - Simulated annealing
    - Genetic algorithm