AARHUS UNIVERSITY

Data Analytics and Machine Learning

Global Search Part 1

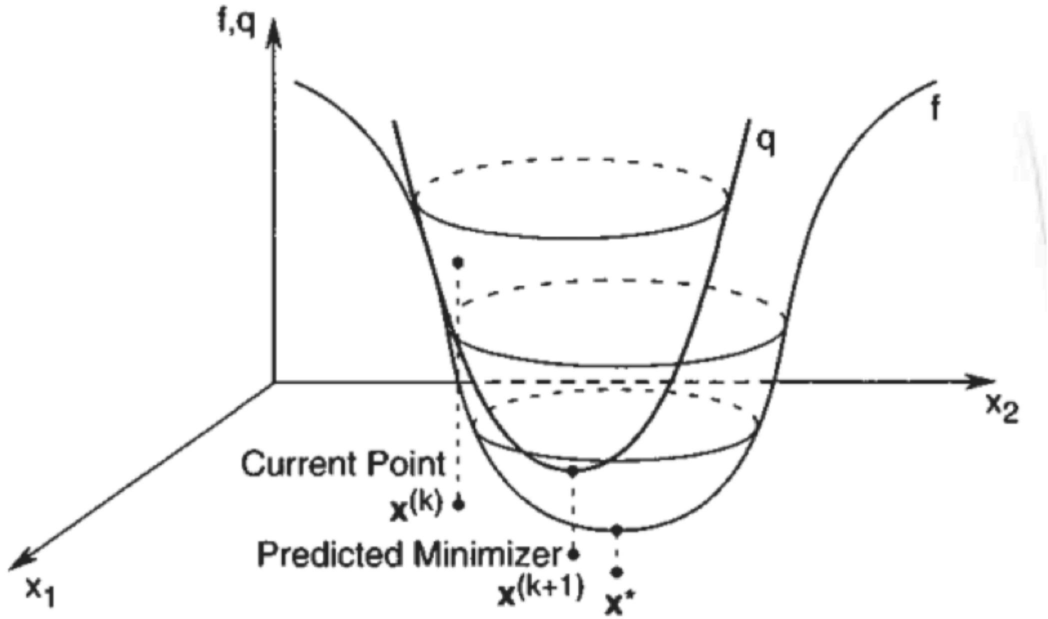
Henrik Karstoft
Carl Schultz
Alexandros Iosifidis

TODAY'S OUTLINE

quick recap

1. preliminaries
2. simulated annealing
3. particle swarm optimisation

what are Quasi-Newton methods?

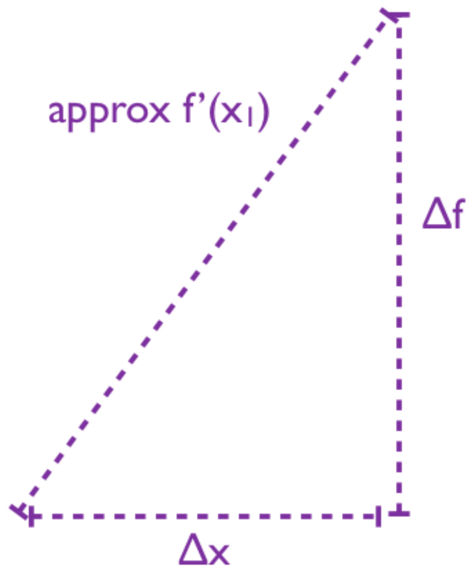


recap Quasi-Newton

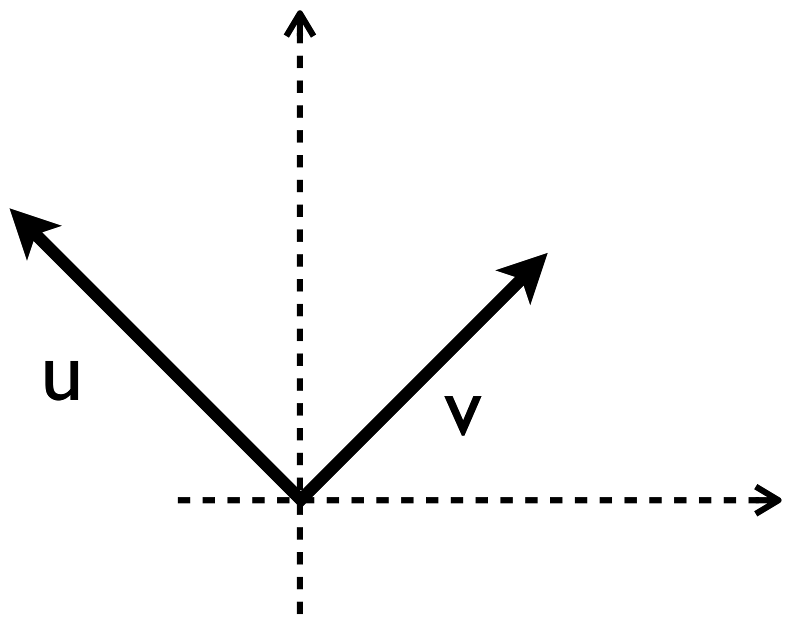
- replace Jakobian/Hessian with approximation

$$X^{(i+1)} = X^{(i)} - B^{(i)-1} \nabla f(X^{(i)})$$

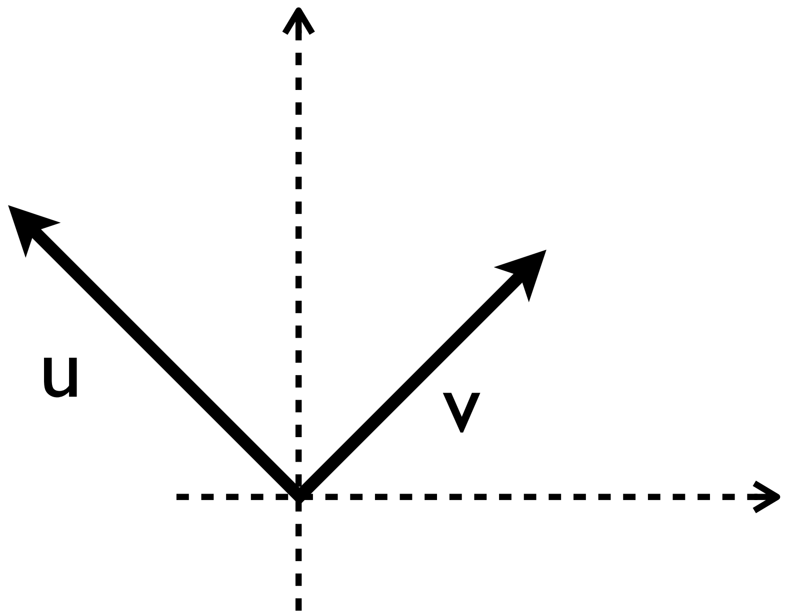
- they generalise secant method
- fast and more robust than Newton



what's the **difference** between orthogonal and linearly independent?



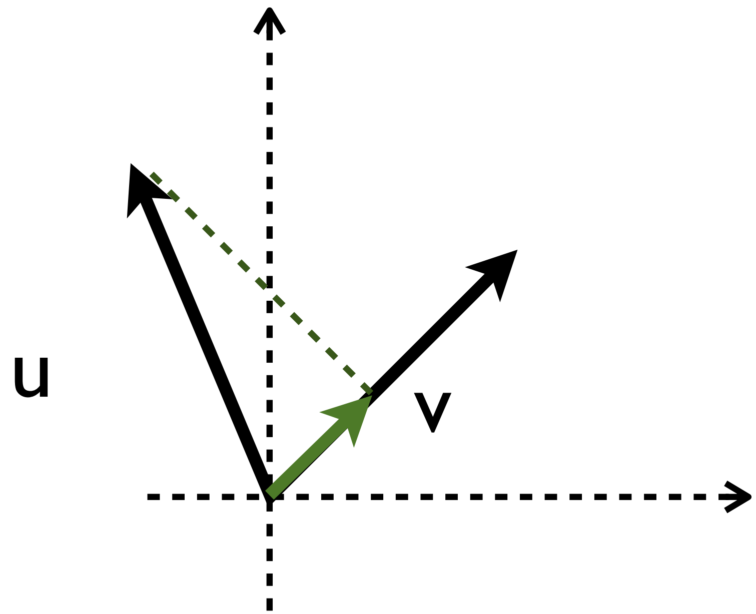
orthogonal



$u \cdot v = 0$

(u,v are perpendicular)

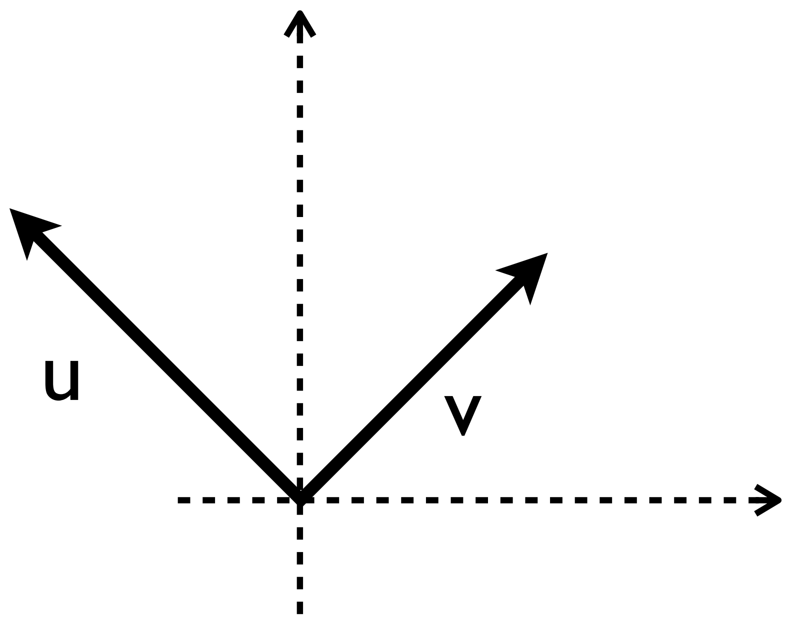
linearly independent



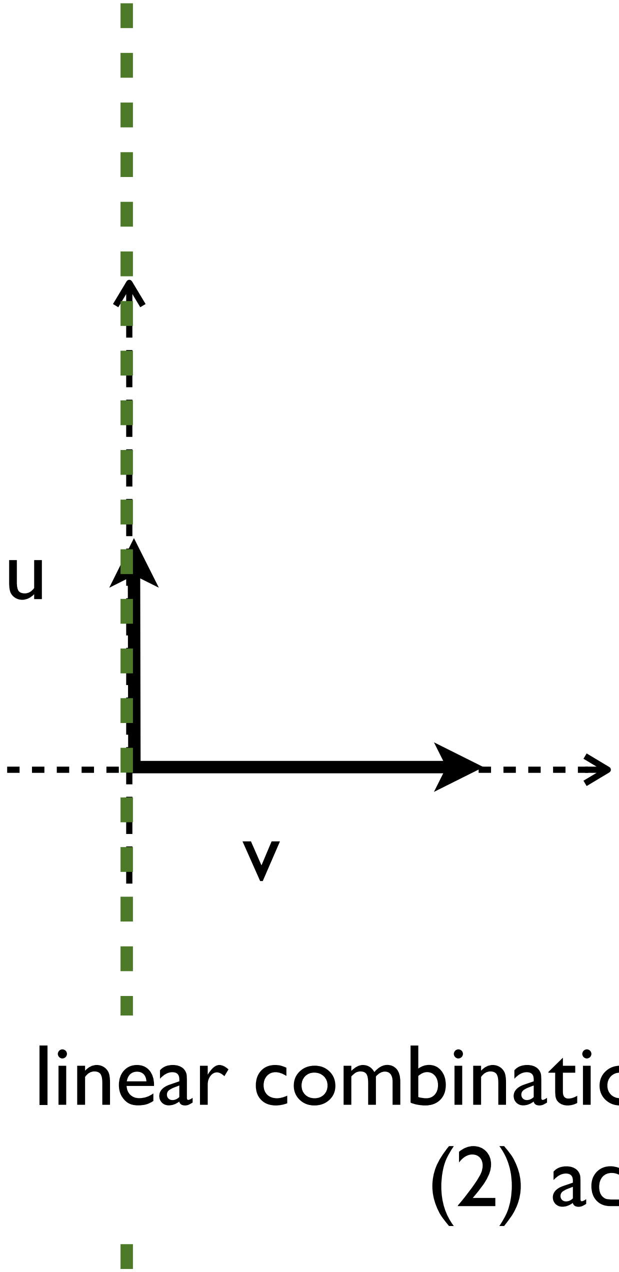
not necessarily true that

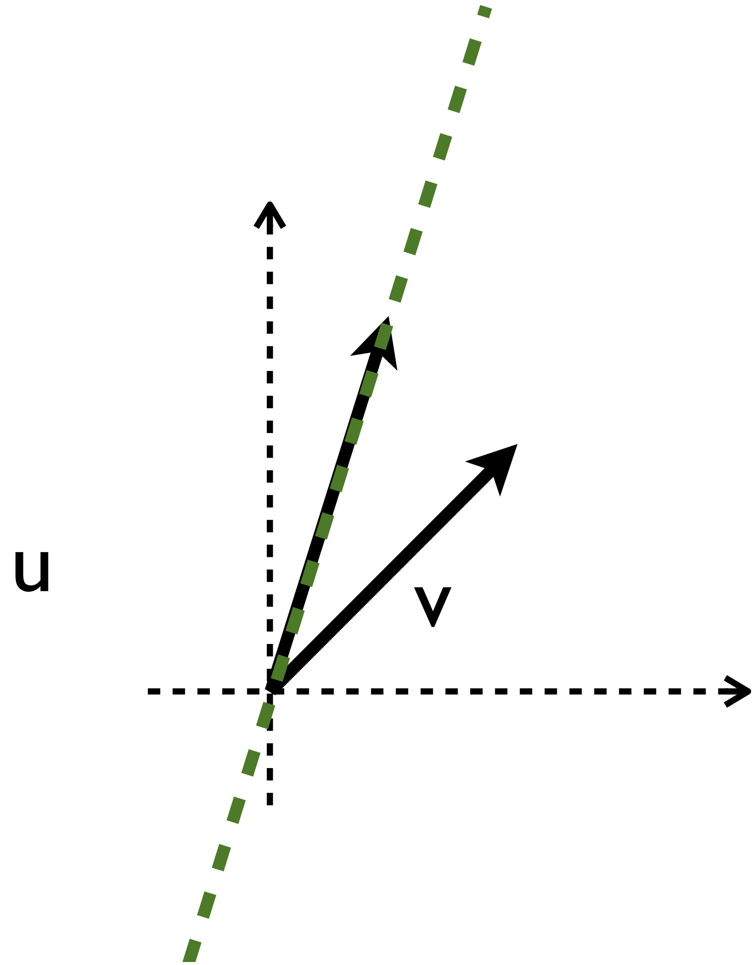
$u \cdot v = 0$

what's the **similarity** between orthogonal and linearly independent?



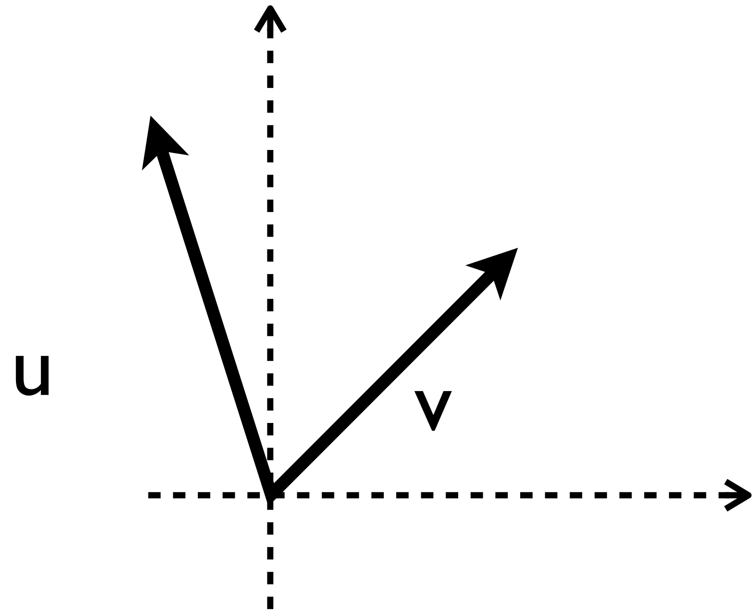
if u,v are linearly independent, cannot describe u as linear combination of v (and vice versa)



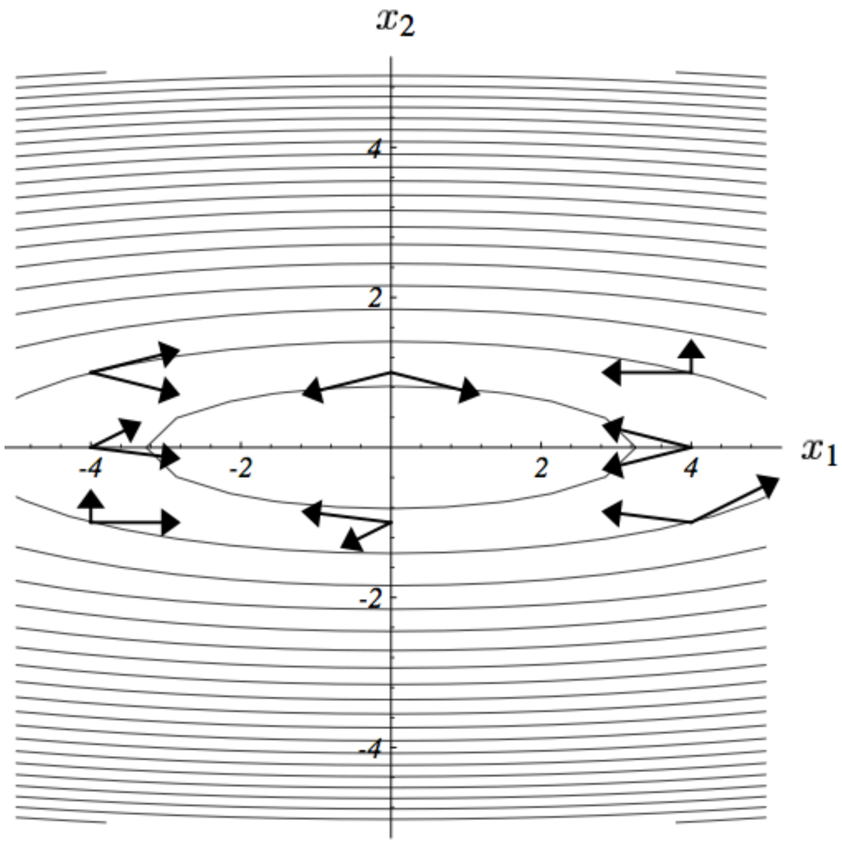


linear combination: can (1) scale vectors, and (2) add them together

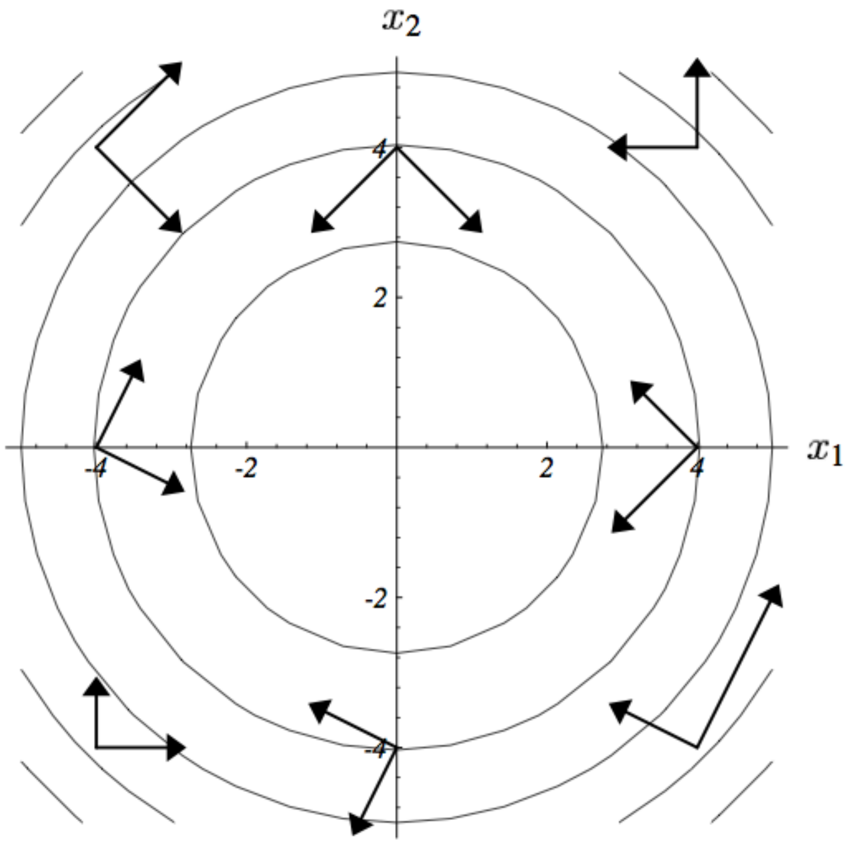
what is "A-conjugate"?



A-orthogonal directions ("A-conjugate")



(a)

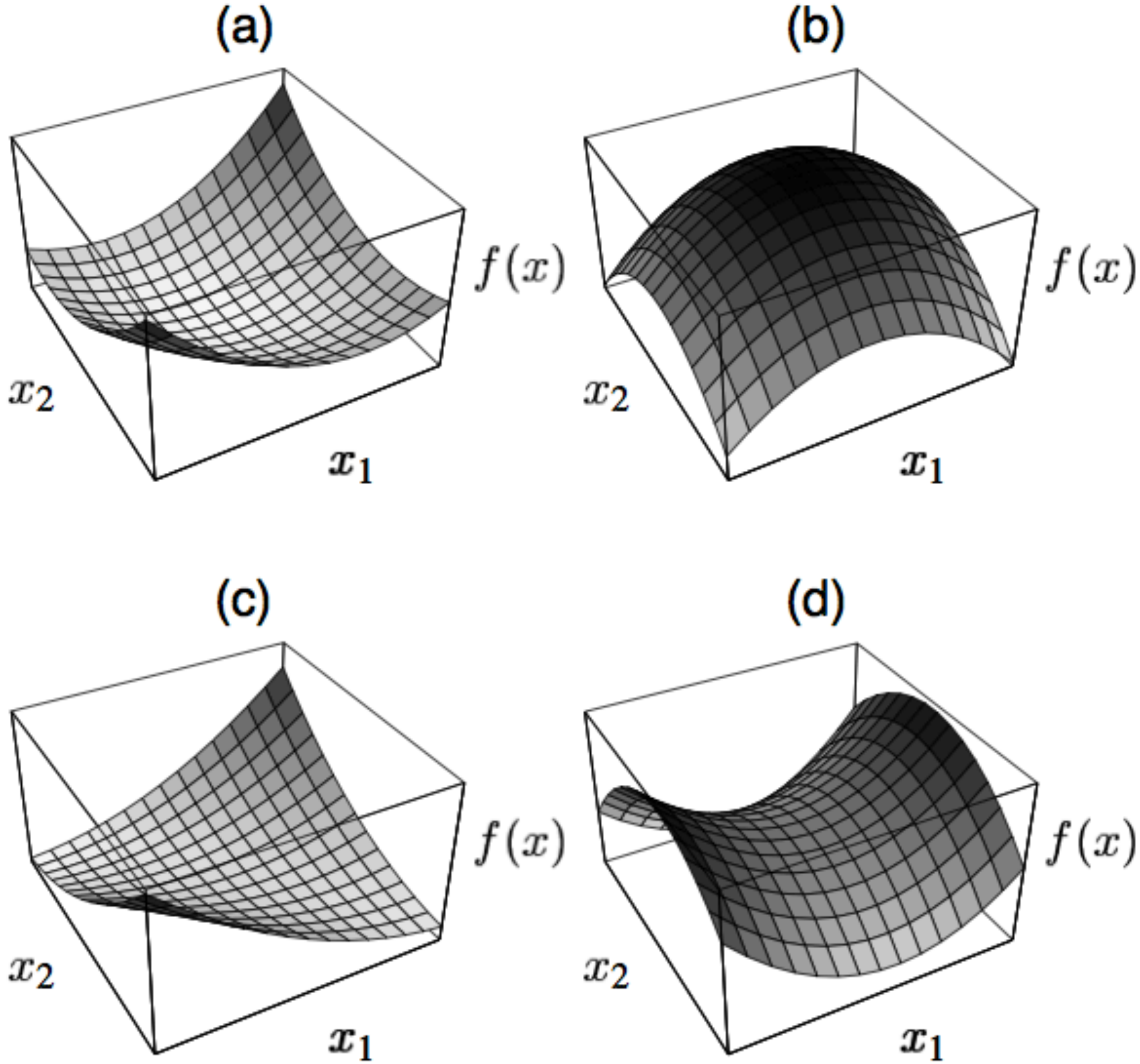


(b)

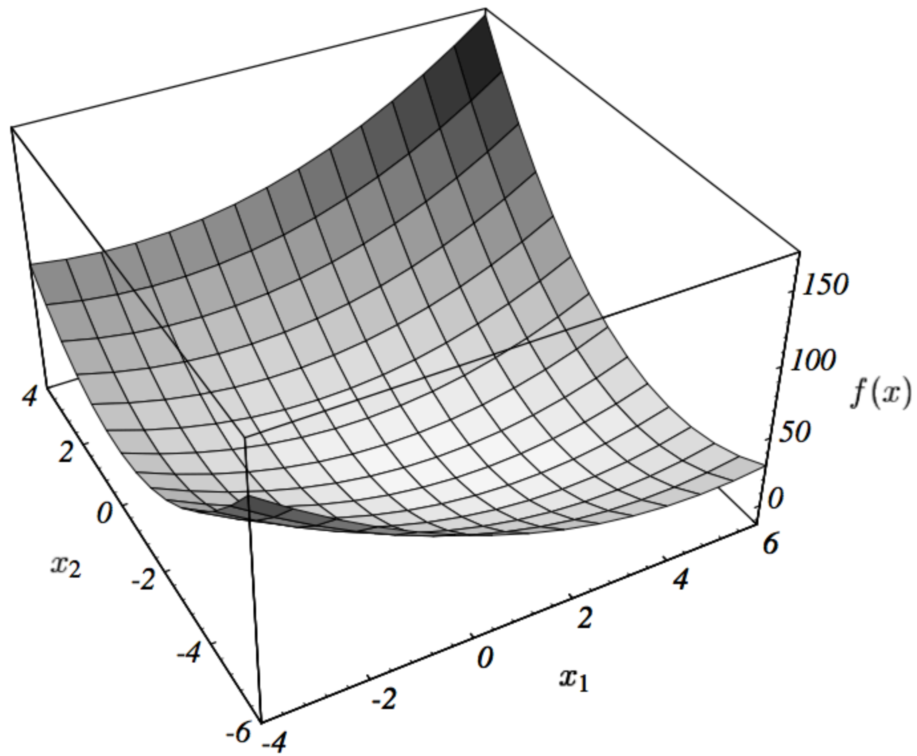
pairs of vectors (u,v) are A-orthogonal.... because these are orthogonal

$$u^T A v = 0$$

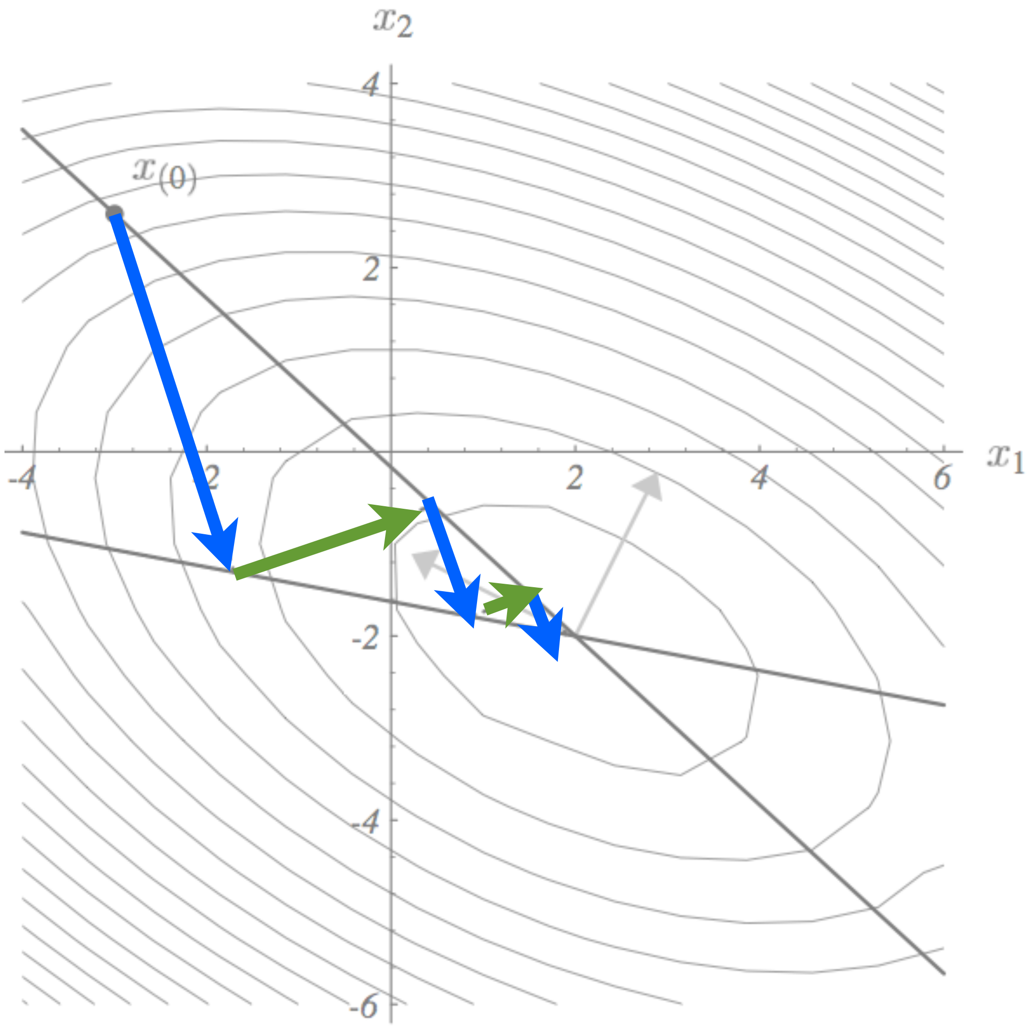
what is positive-definiteness?



what's the difference between: **steepest descent** and **conjugate gradient method?**

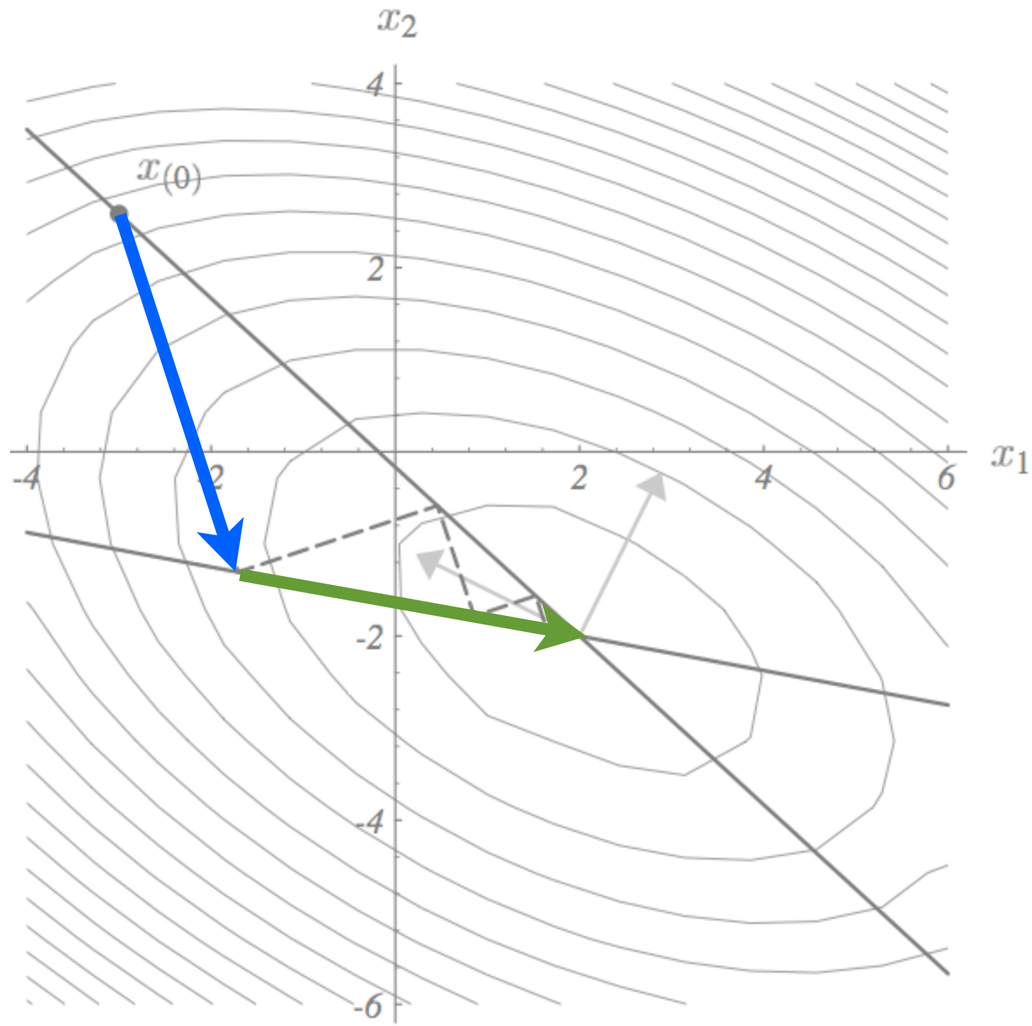


steepest descent



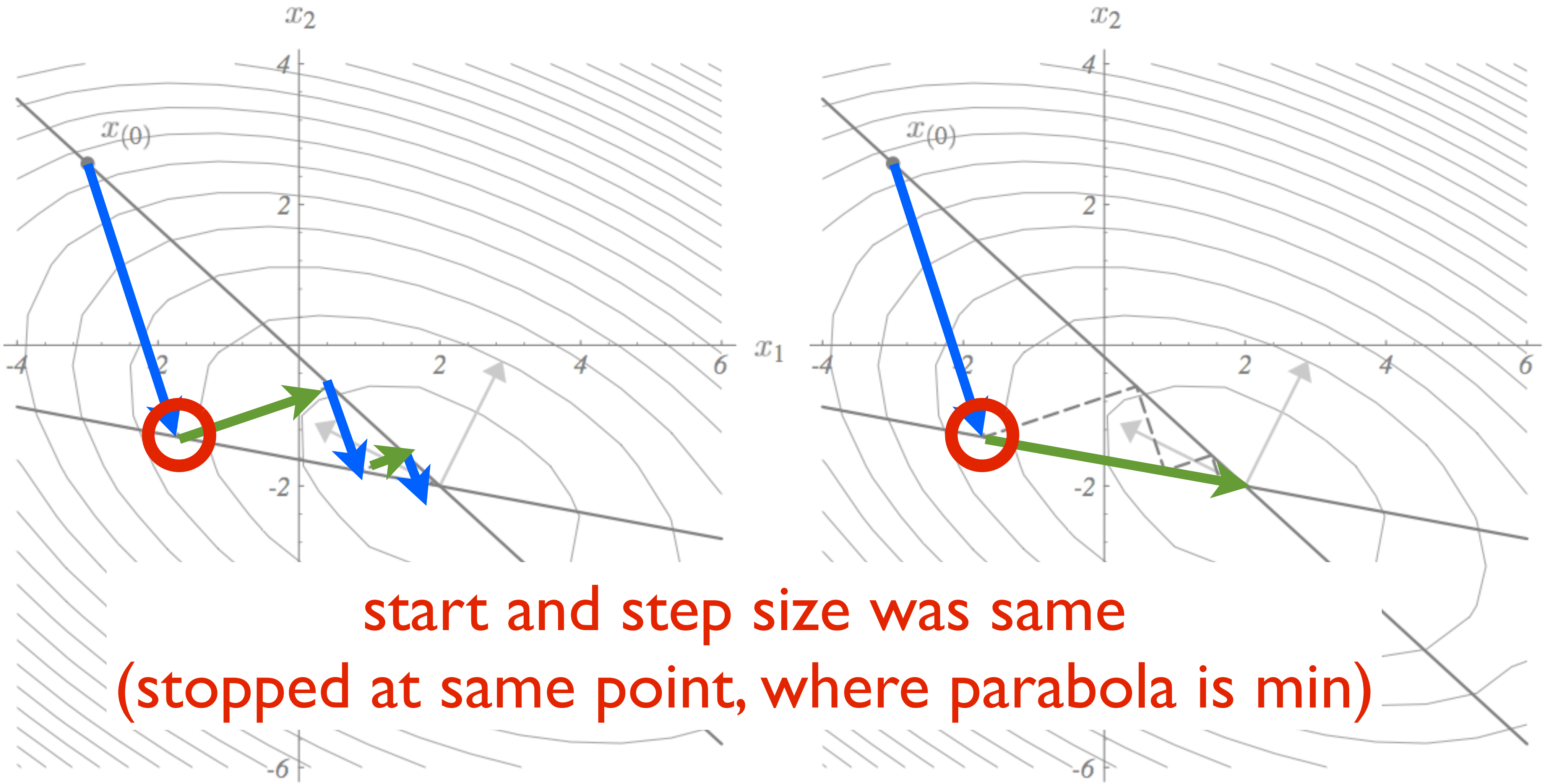
always turns orthogonal

conjugate gradients



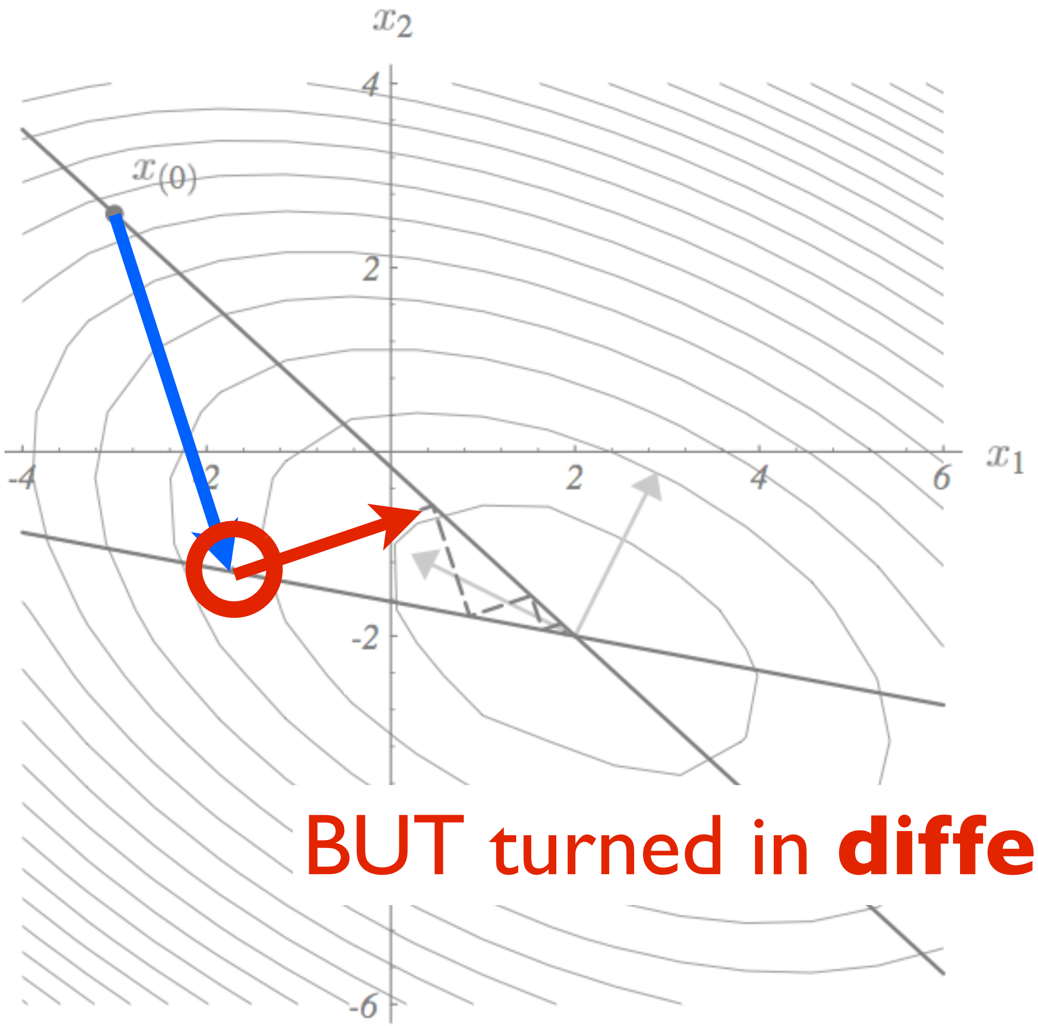
always turns A-orthogonal

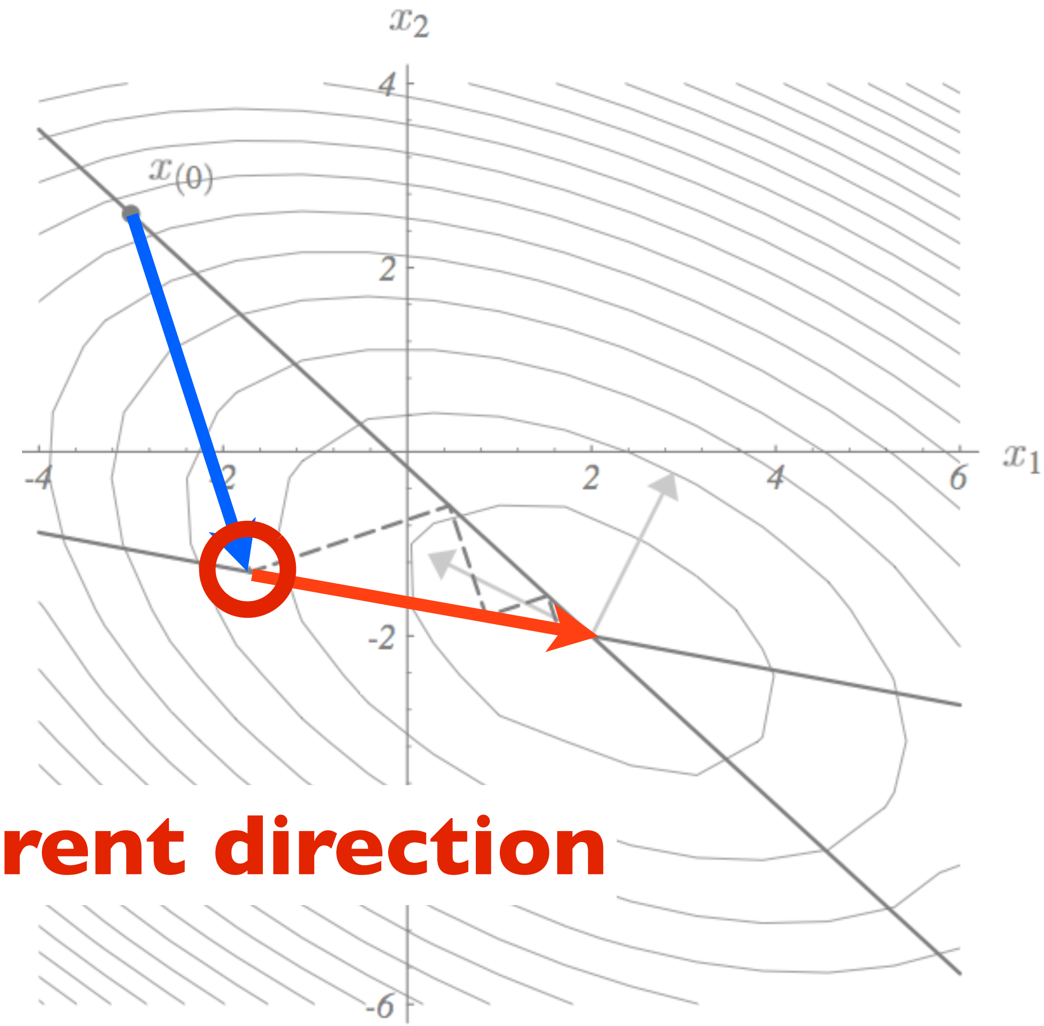
steepest descent | conjugate gradients



always turns orthogonal | always turns A-orthogonal steepest descent

conjugate gradients





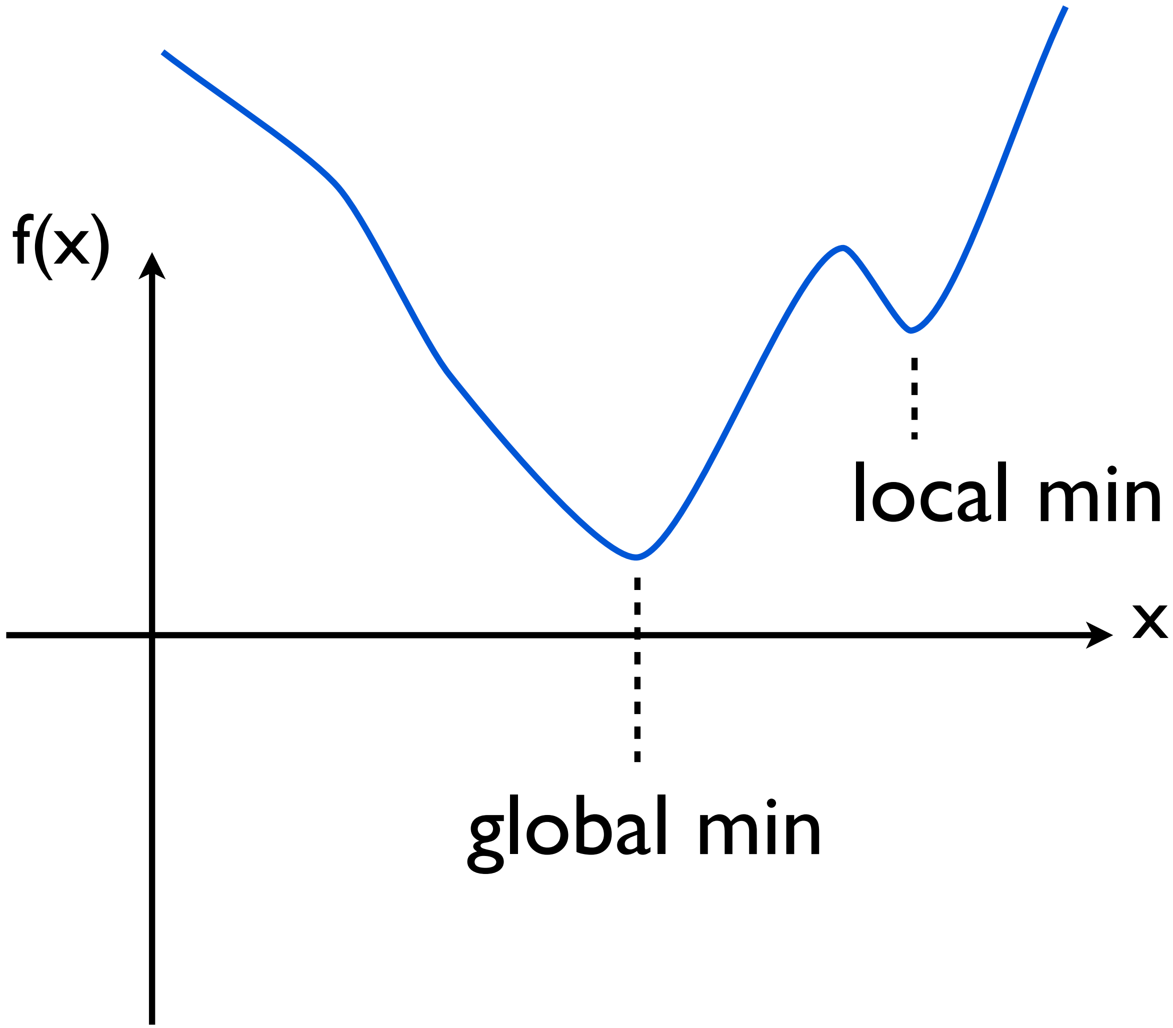
BUT turned in **different direction**

always turns orthogonal

always turns A-orthogonal

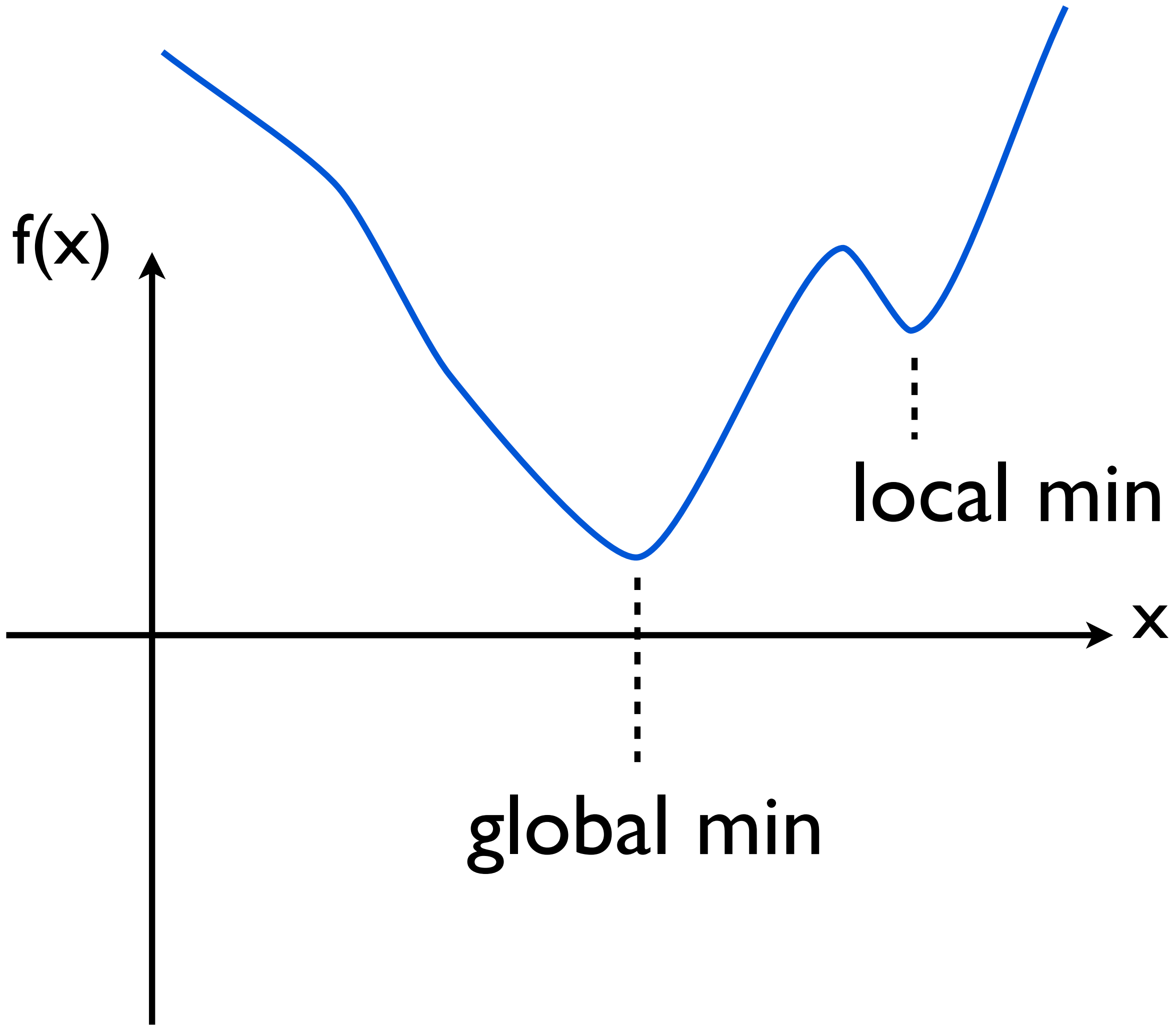
Part 1
some preliminaries

global vs local optimisation



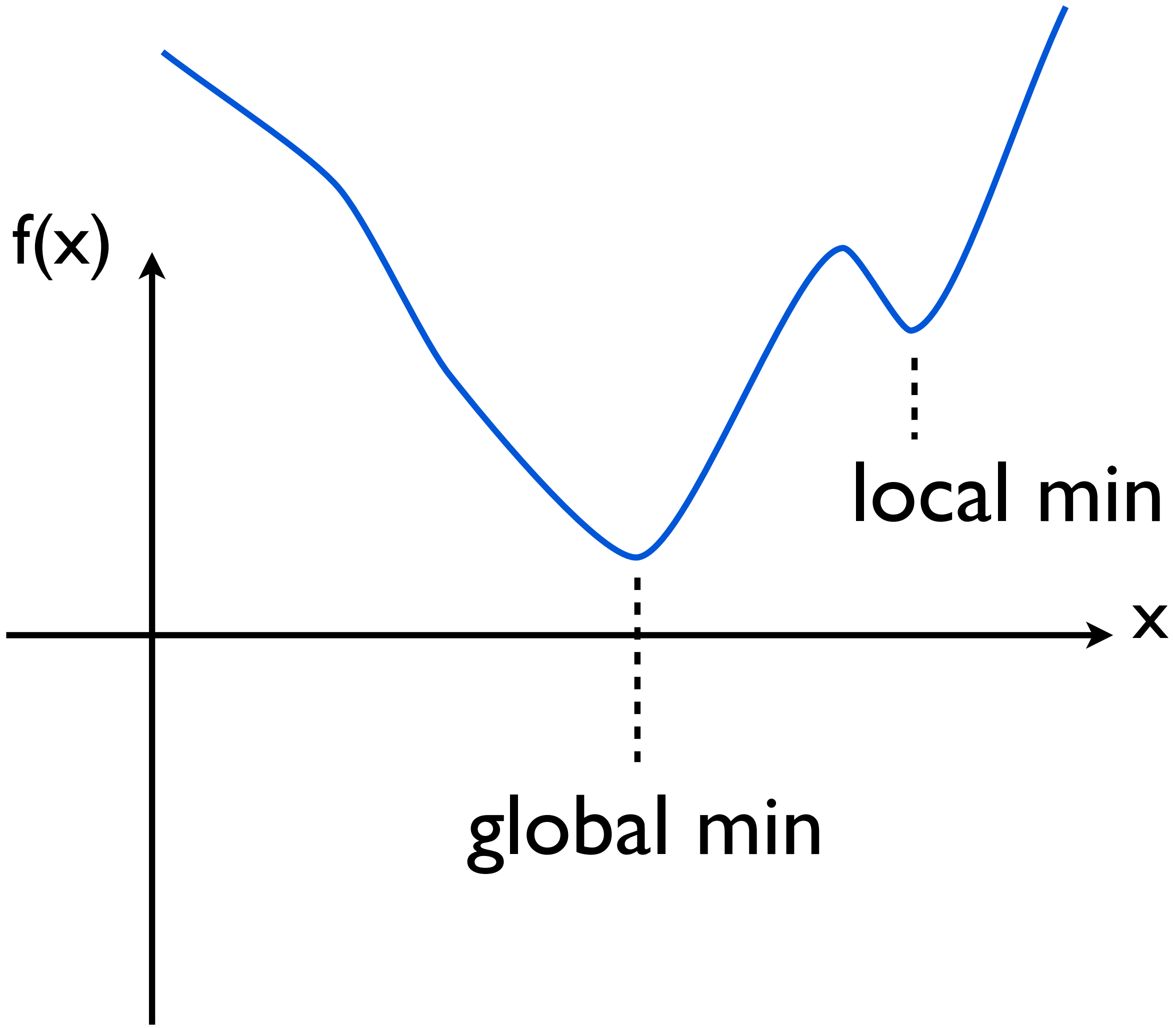
what is local optimisation? what is global optimisation?

global vs local optimisation



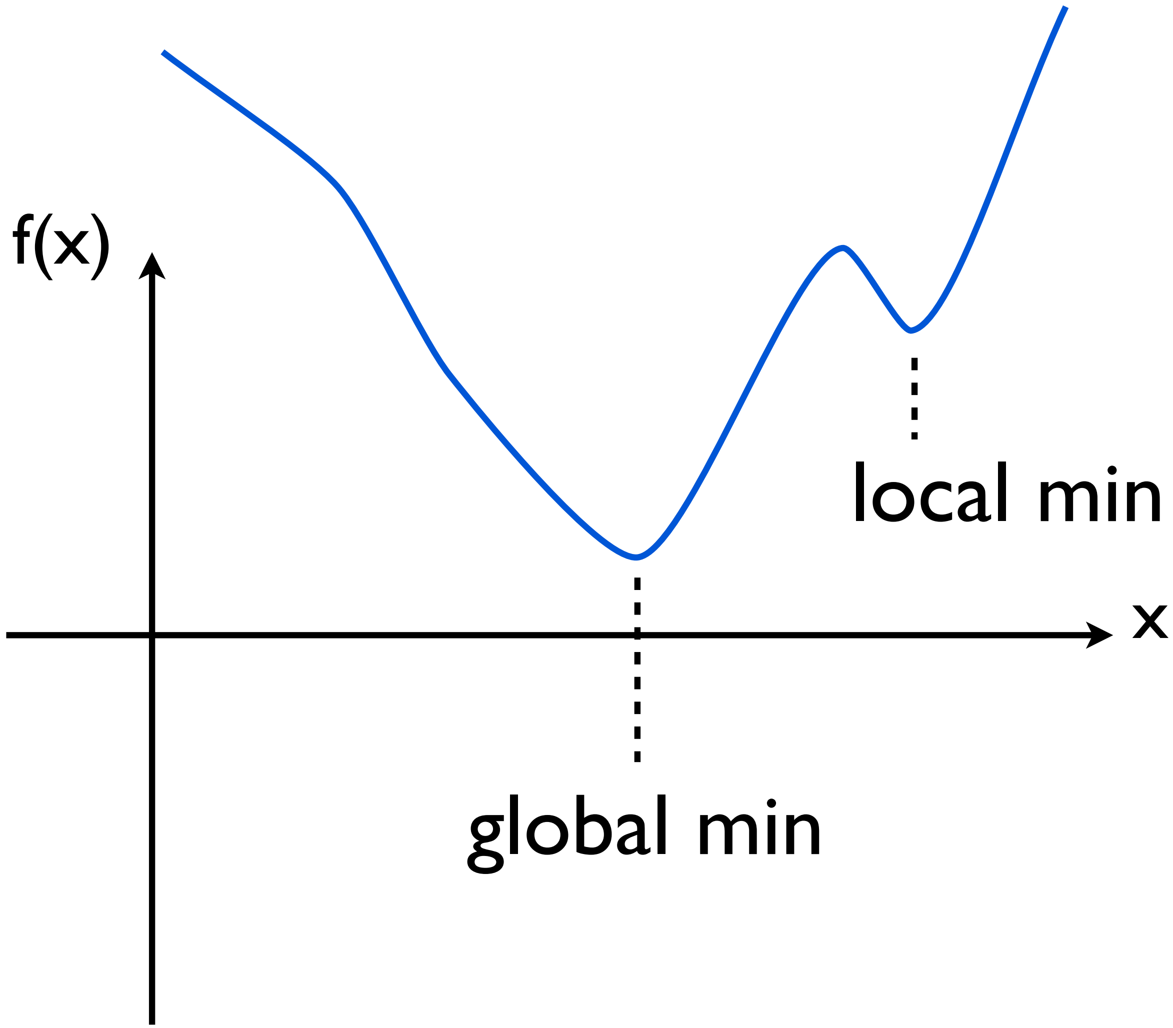
local opt algorithms find local optima only
(tend to be fast)

global vs local optimisation



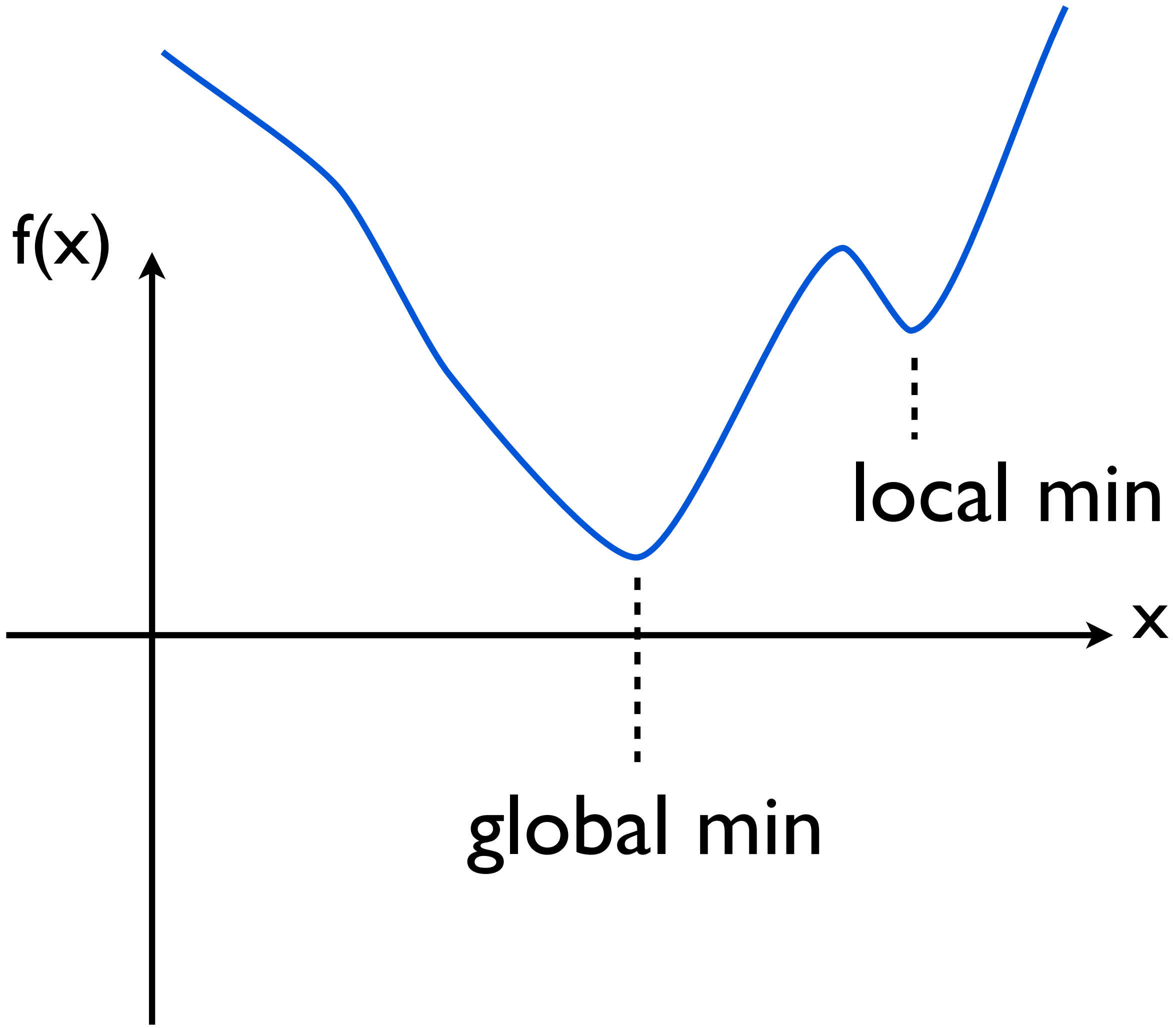
global opt algorithms have mechanism to not get stuck in local optima

global vs local optimisation



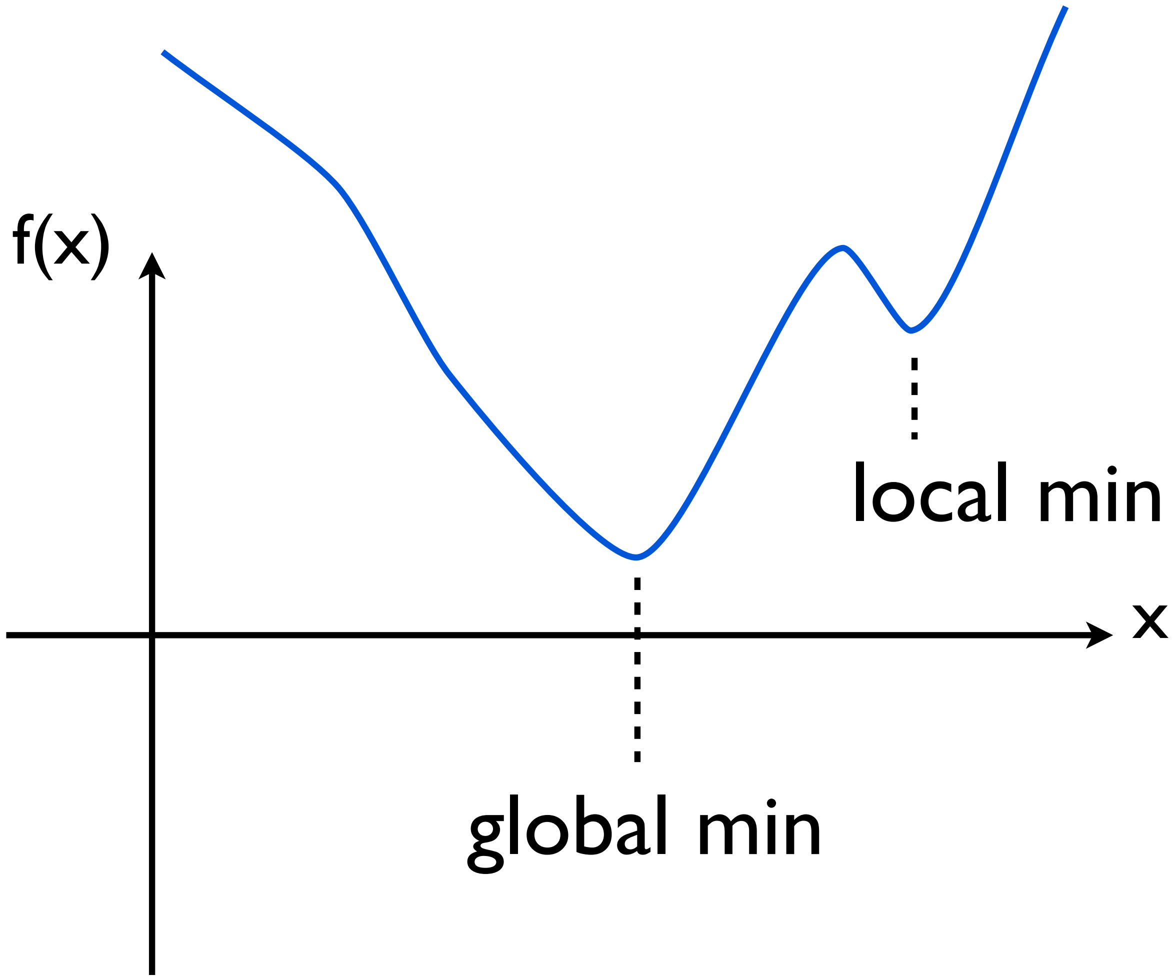
do global opt algorithms guarantee global min?

global vs local optimisation



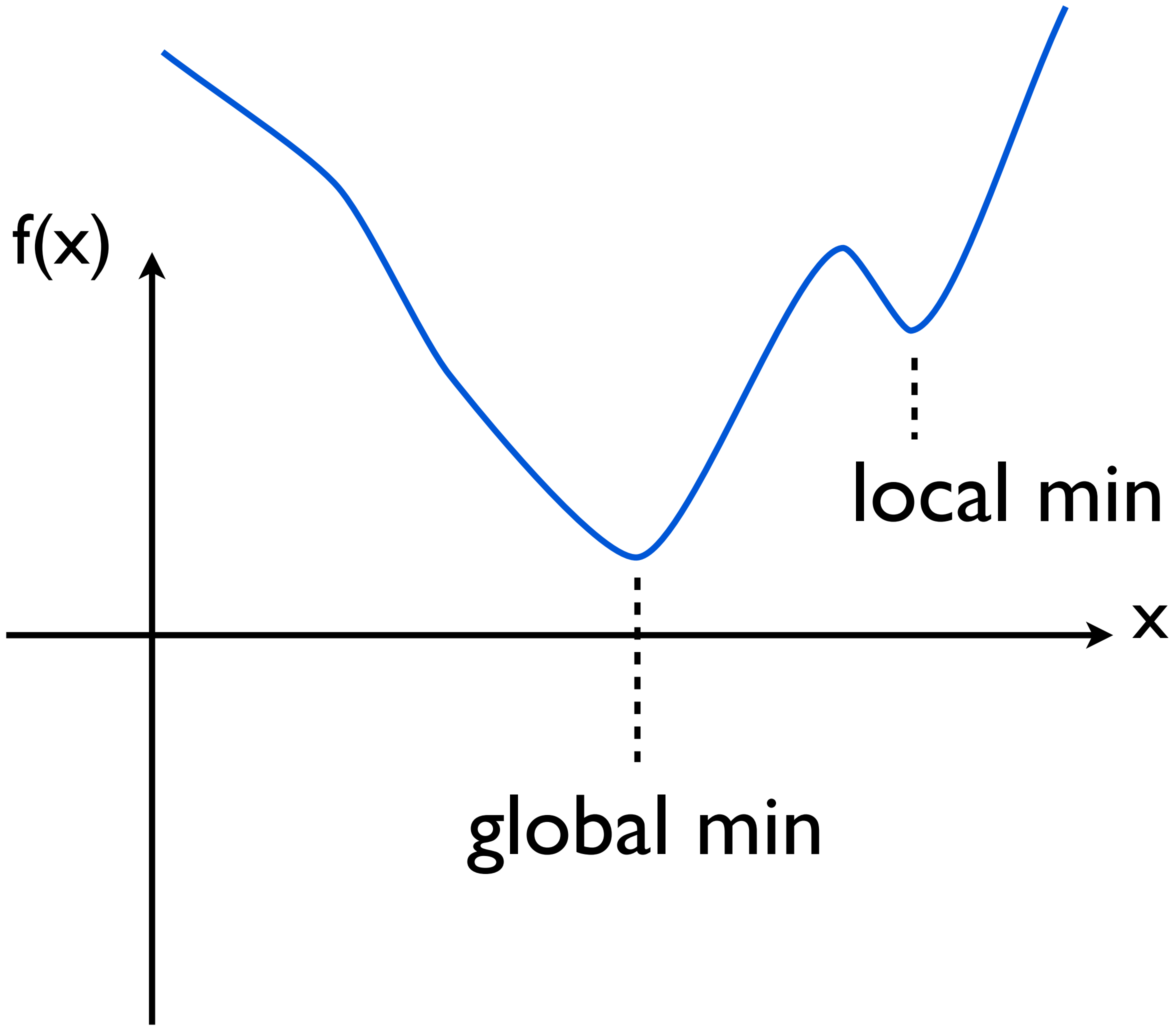
most do not guarantee finding global optimum, but methods do exist (they can take a very very long time)

local search

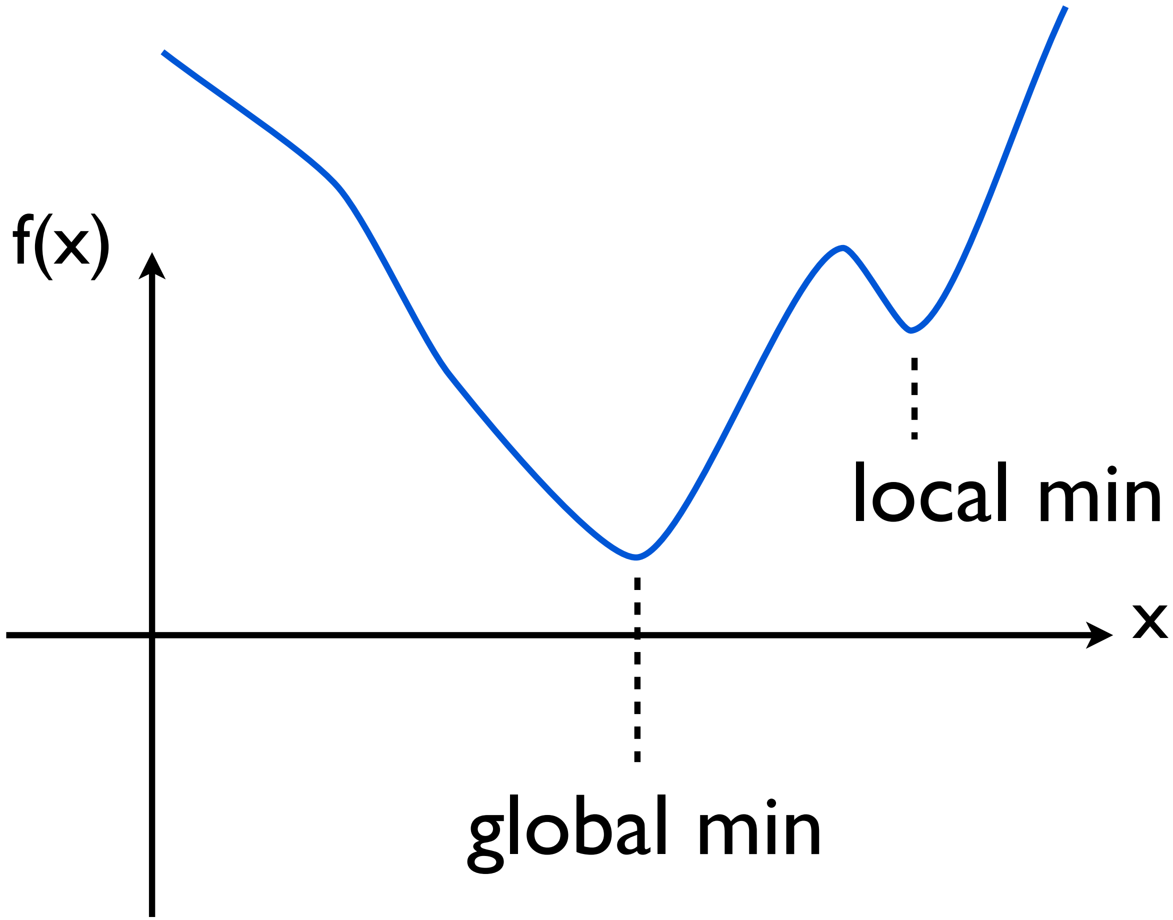


terminology may vary, but concept is important: each step, algorithm picks from local "neighbourhood" of candidate solutions - i.e. next candidate somehow related to current candidate

deterministic vs stochastic

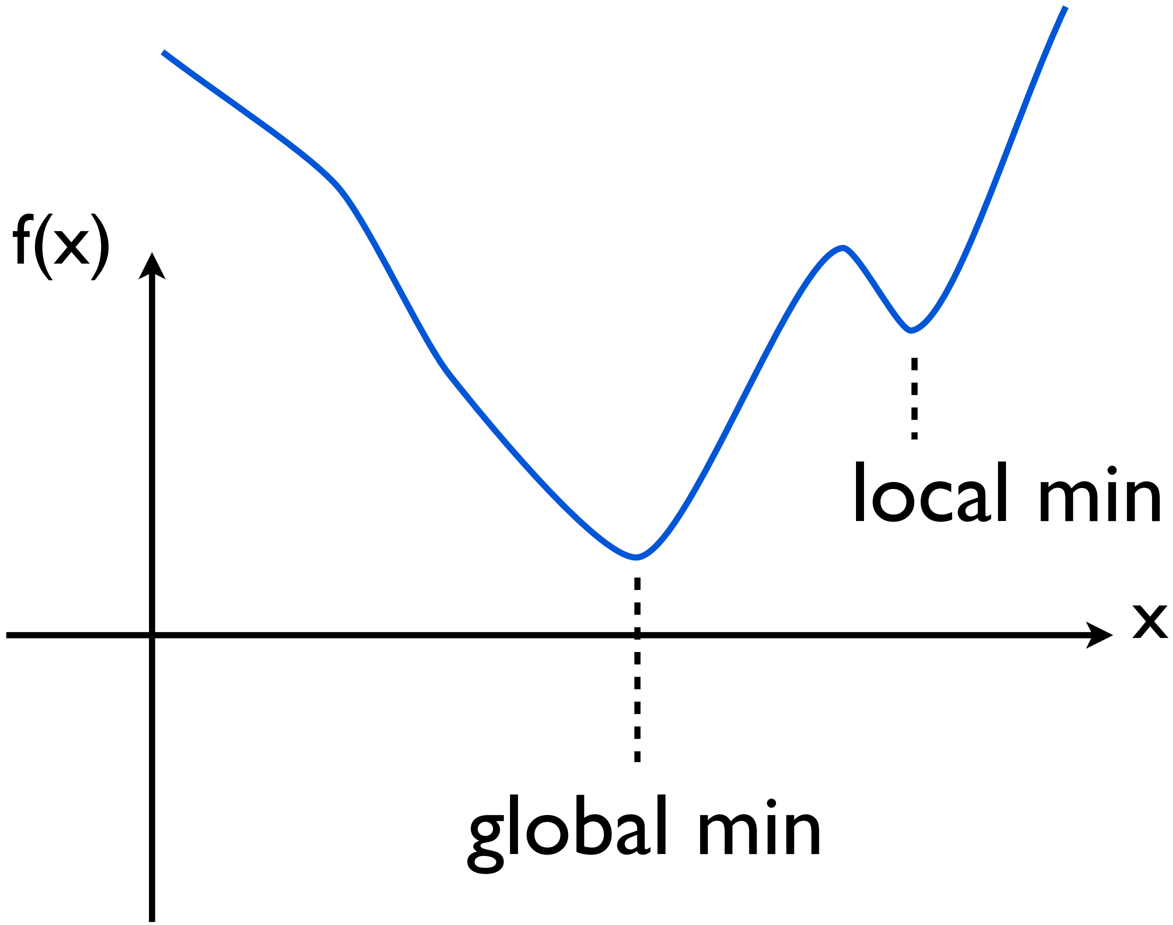


deterministic vs stochastic



deterministic: no randomness in search - always produces same output from initial input (e.g. from same initial candidate solution $x^{(0)}$)

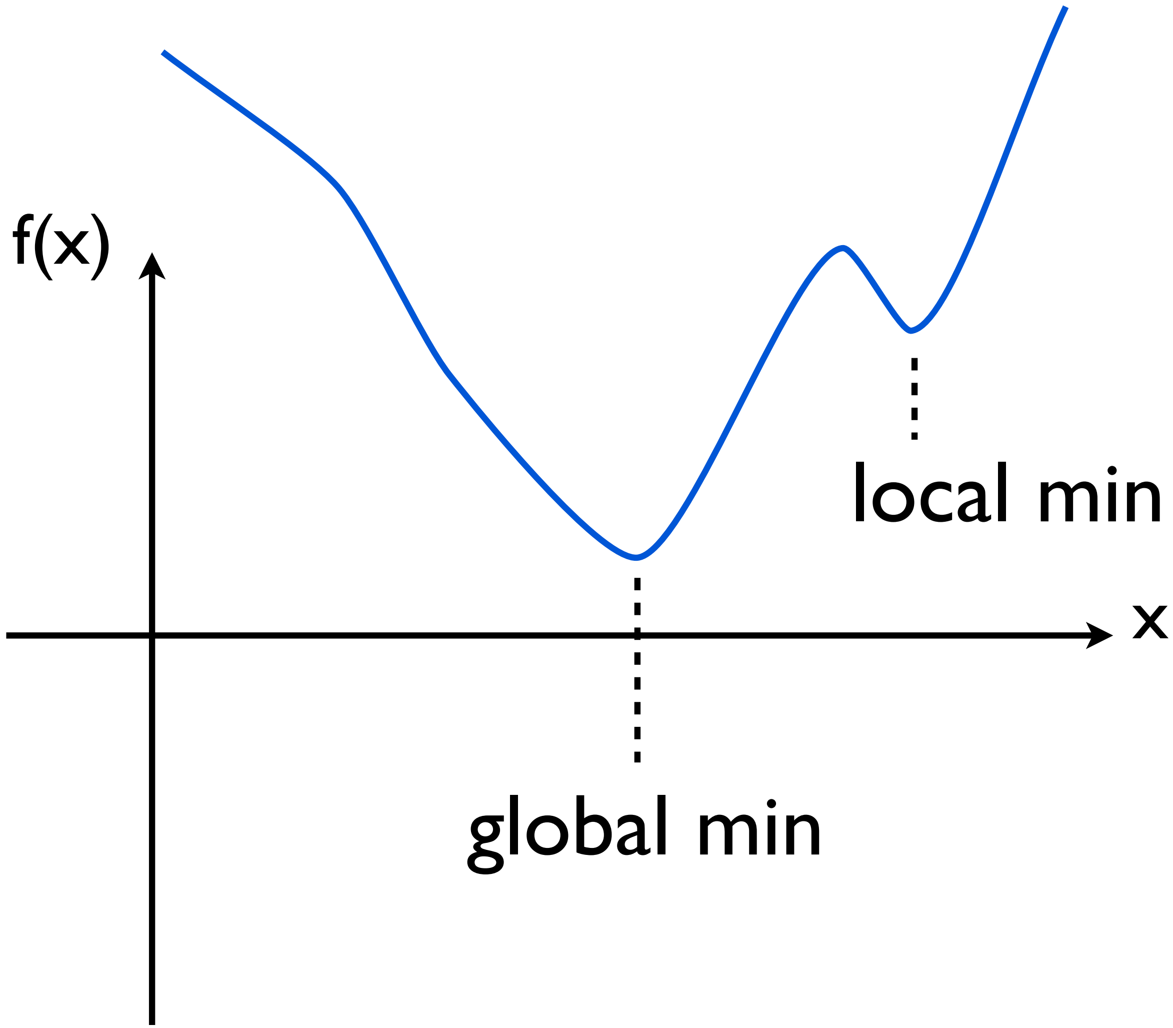
deterministic vs stochastic



stochastic: adds some randomness to search, e.g. randomly select from neighbourhood of candidate solutions

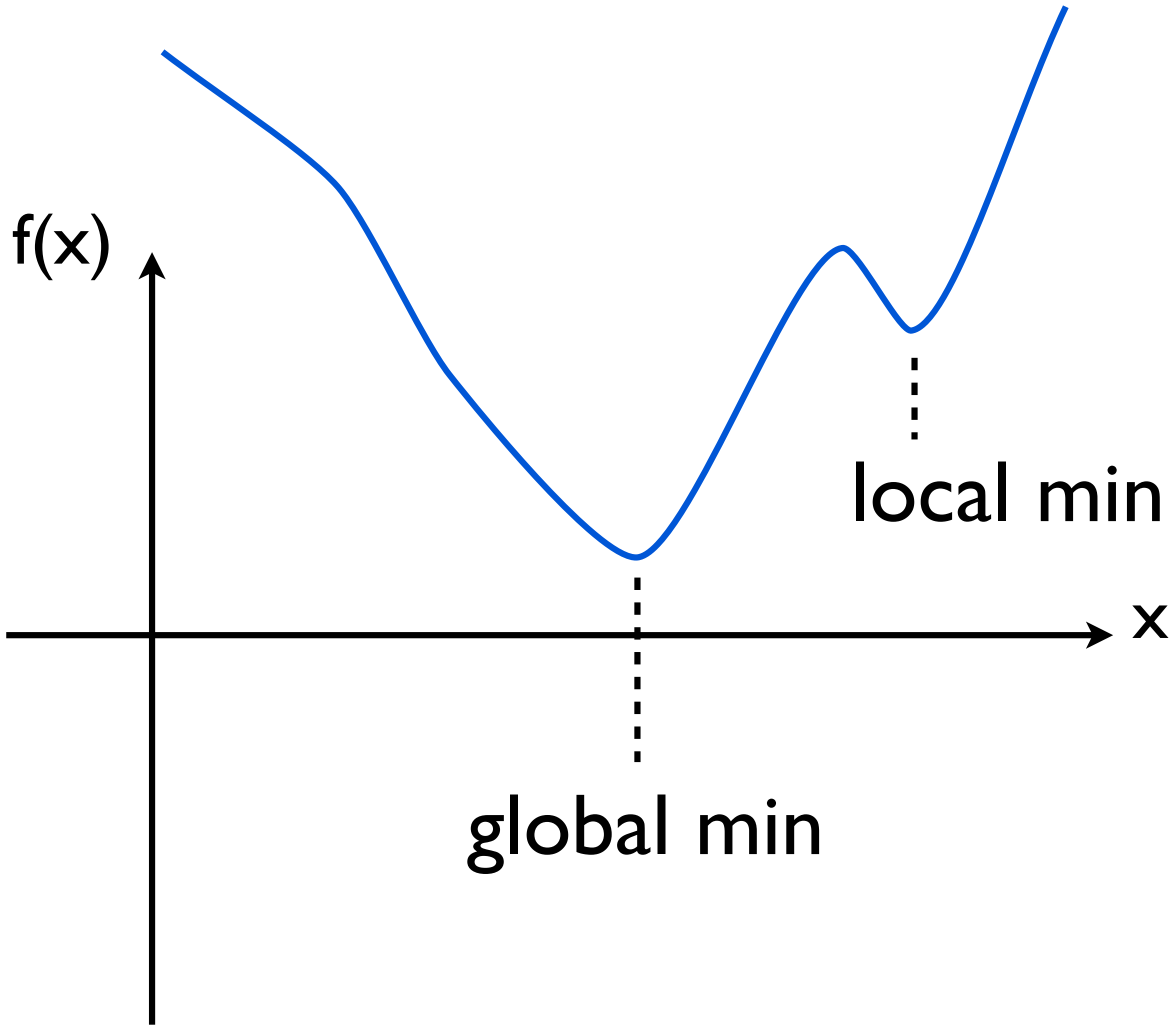
why introduce randomness?

deterministic vs stochastic

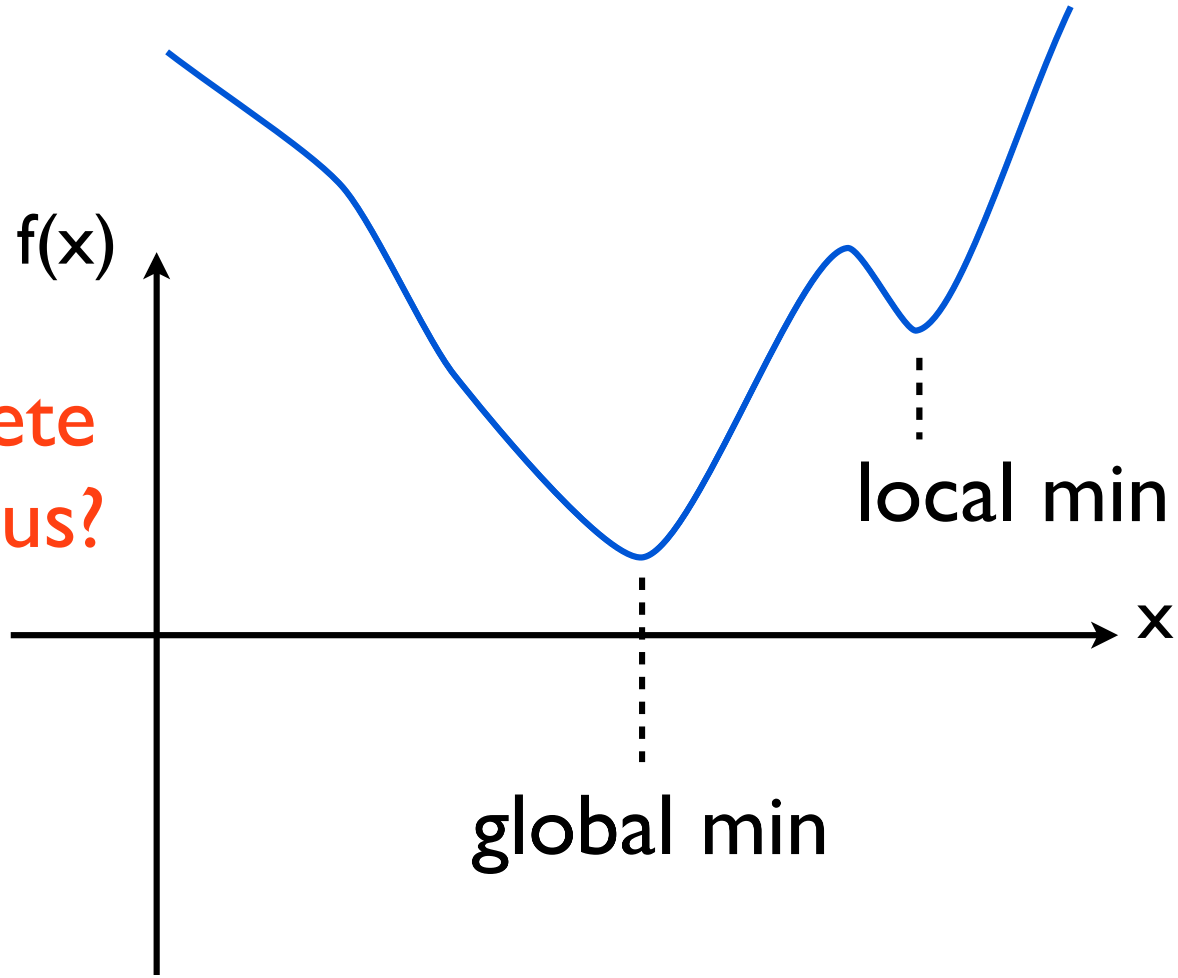


(hopefully) faster, more robust, one mechanism to get out of local optima - some researchers disagree

discrete vs continuous optimisation task



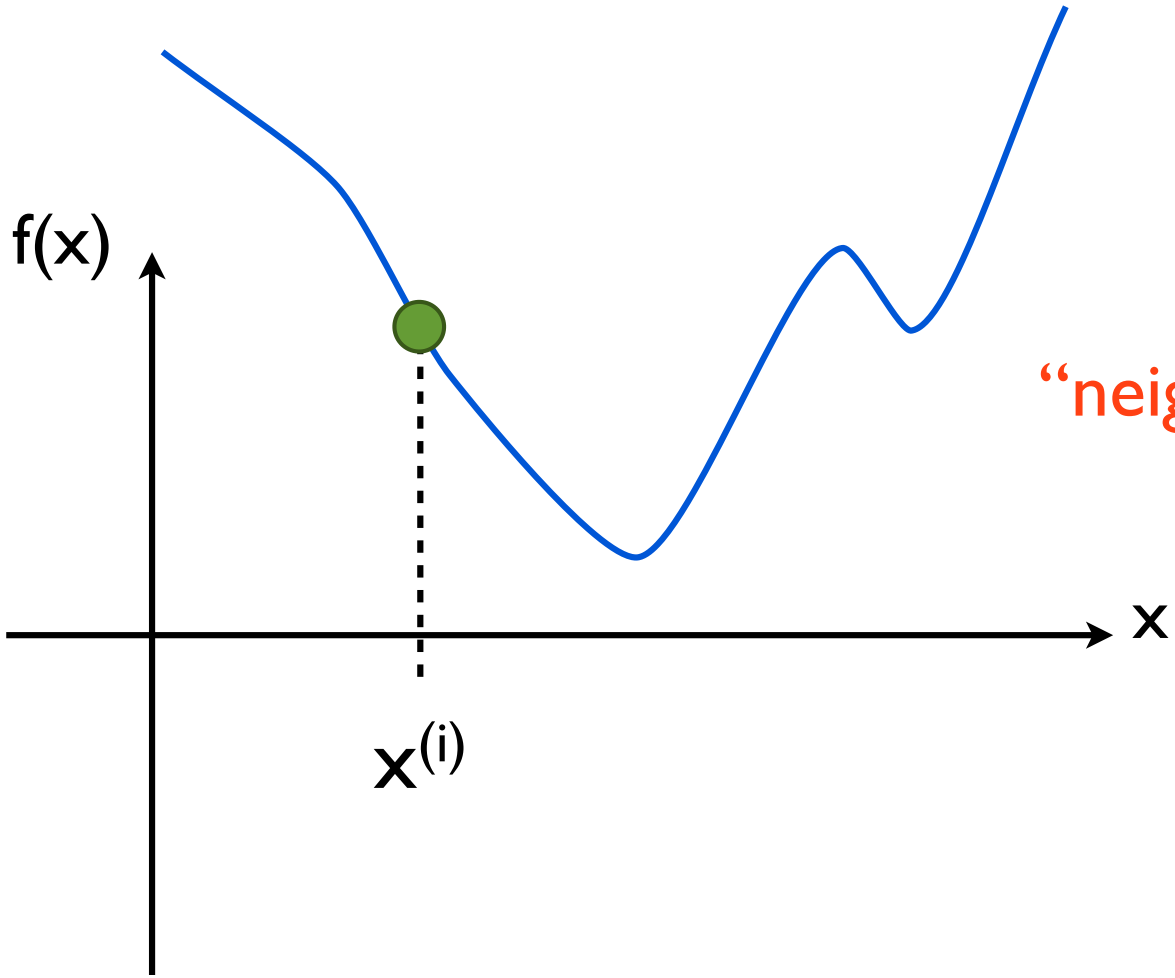
discrete vs continuous optimisation task



is this discrete or continuous?

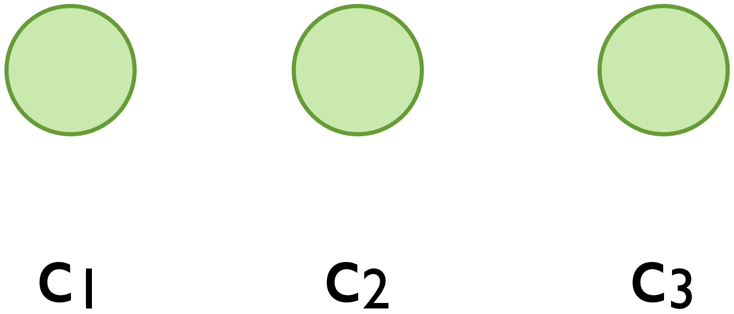
depends on whether optimisation variables are discrete or continuous

discrete vs continuous optimisation task



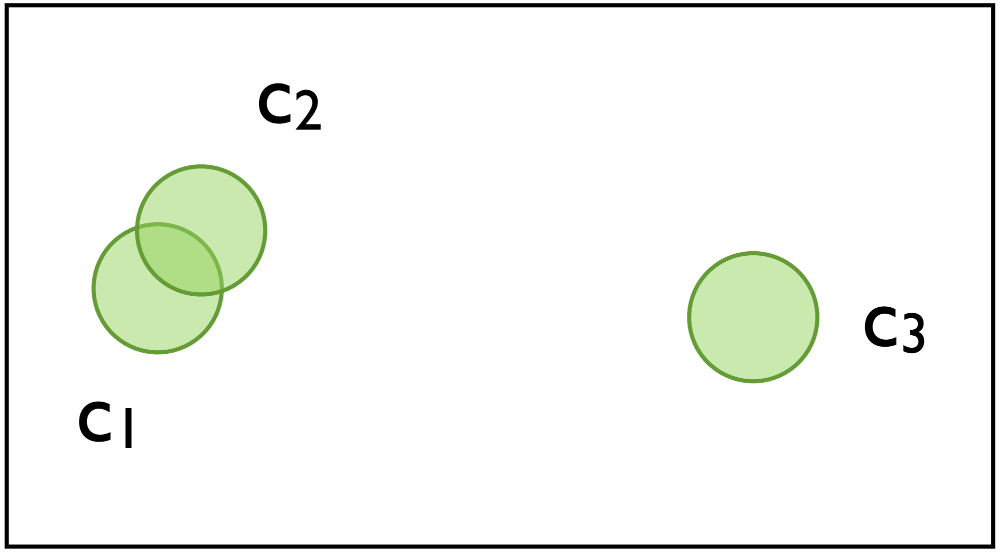
what is “neighbourhood” of $x^{(i)}$?

depends on whether optimisation variables are discrete or continuous can three same-sized circles all touch each other?



is this discrete
or continuous?

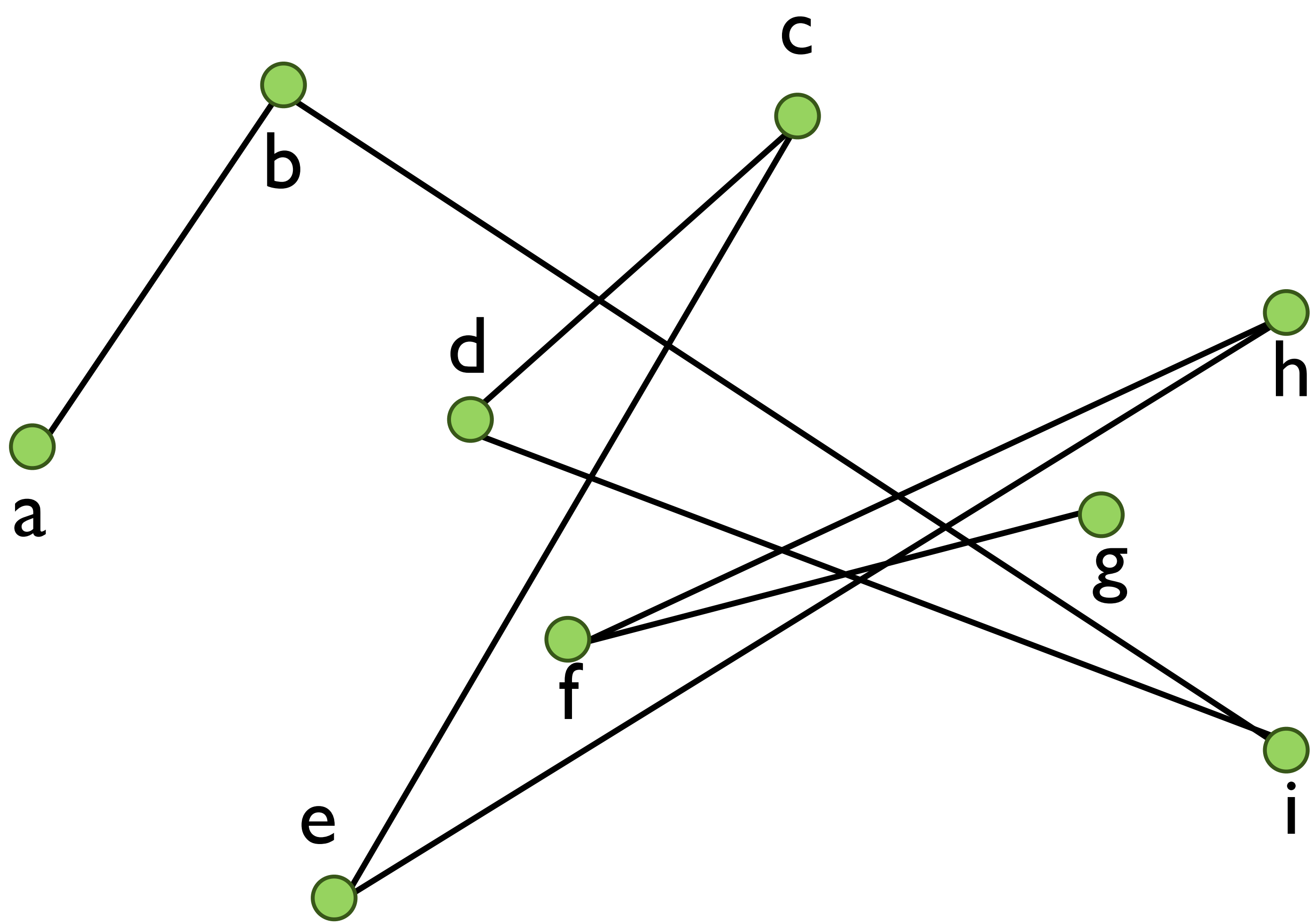
can three same-sized circles all touch each other?



candidate solution $x^{(i)}$

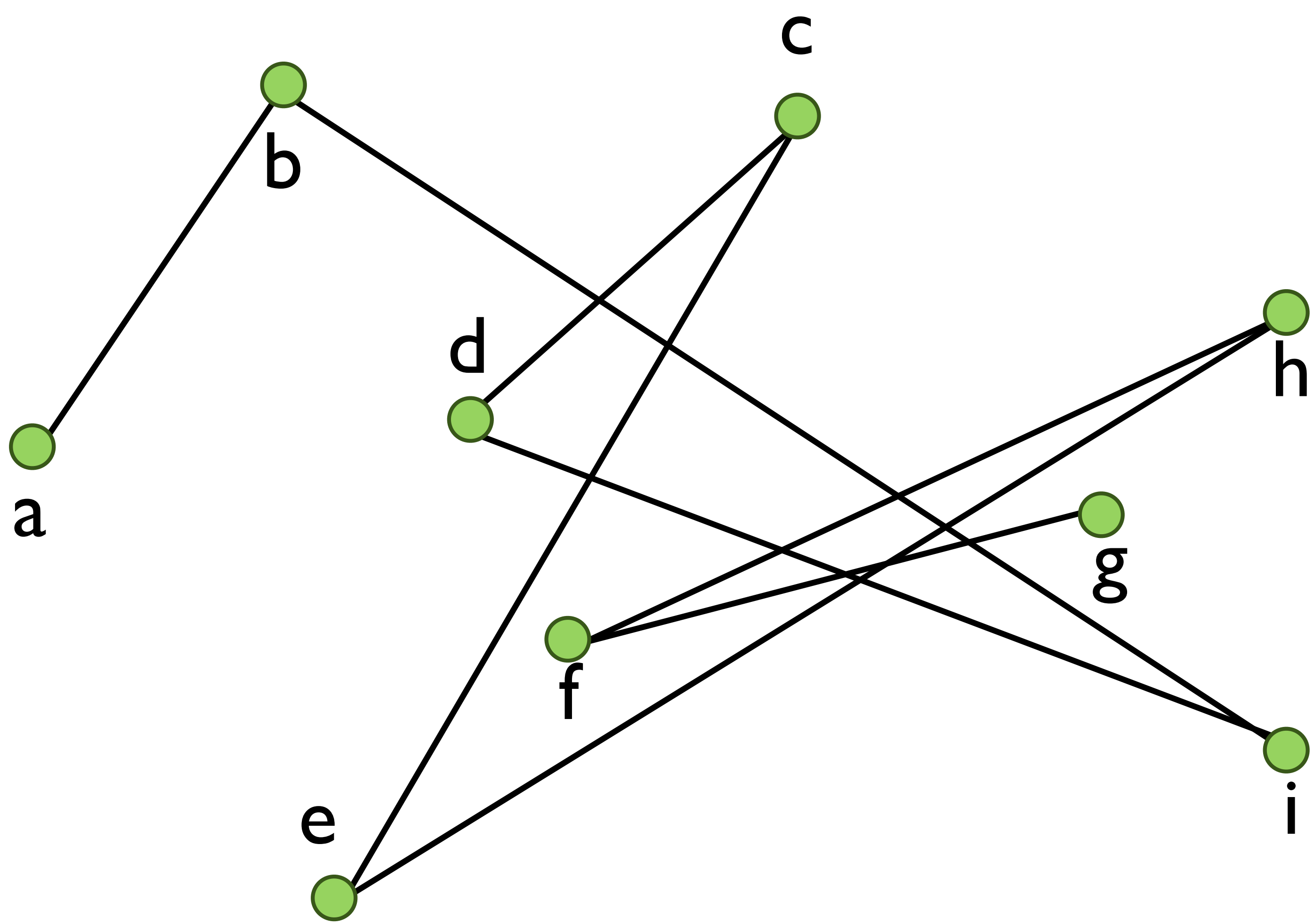
what is “neighbourhood” of $x^{(i)}$?

travelling salesman problem



what is candidate solution $X^{(i)}$ here?

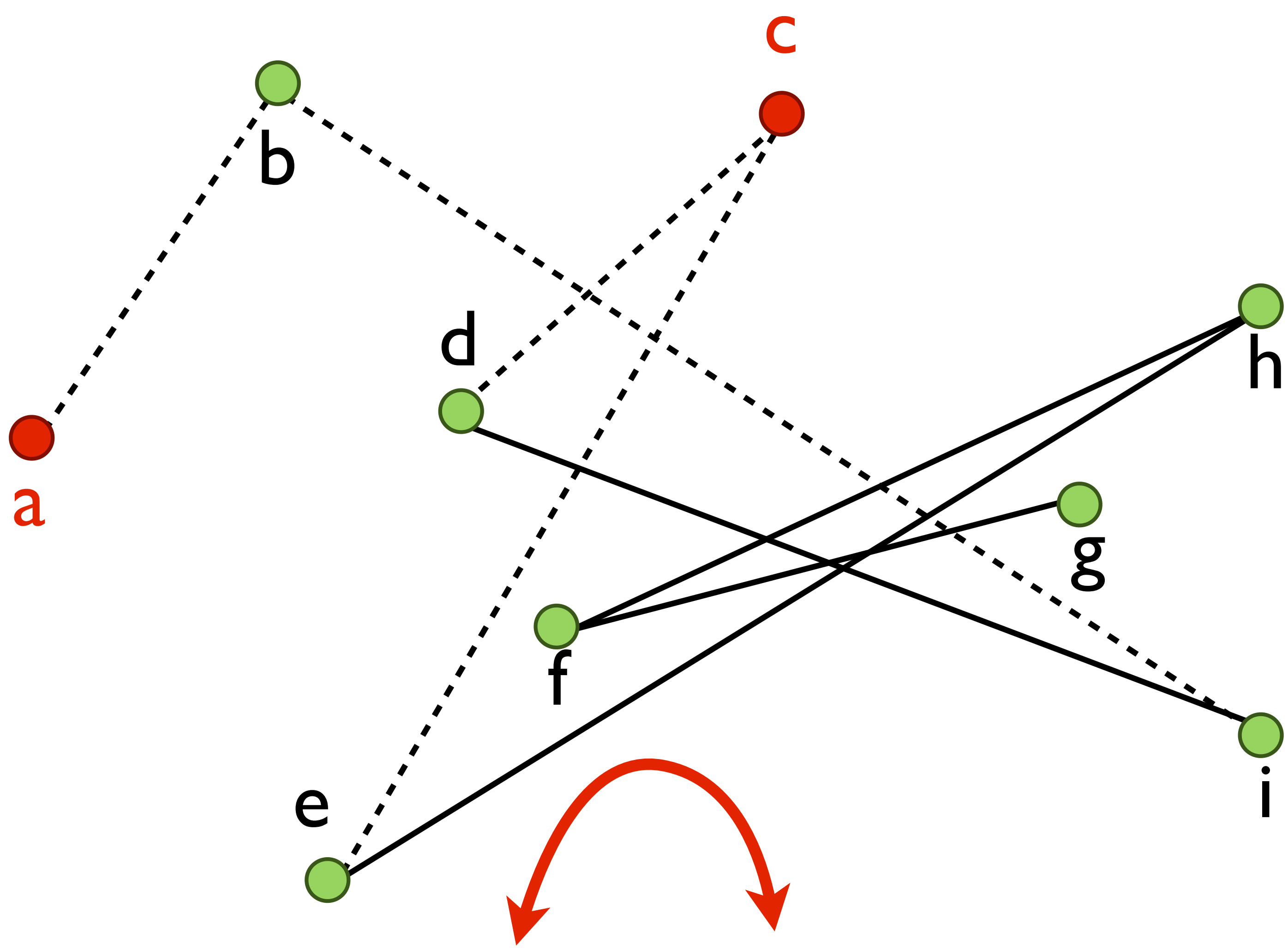
travelling salesman problem



$X^{(i)} = (a, b, i, d, c, e, h, f, g)$

what is "neighbourhood" of $X^{(i)}$?

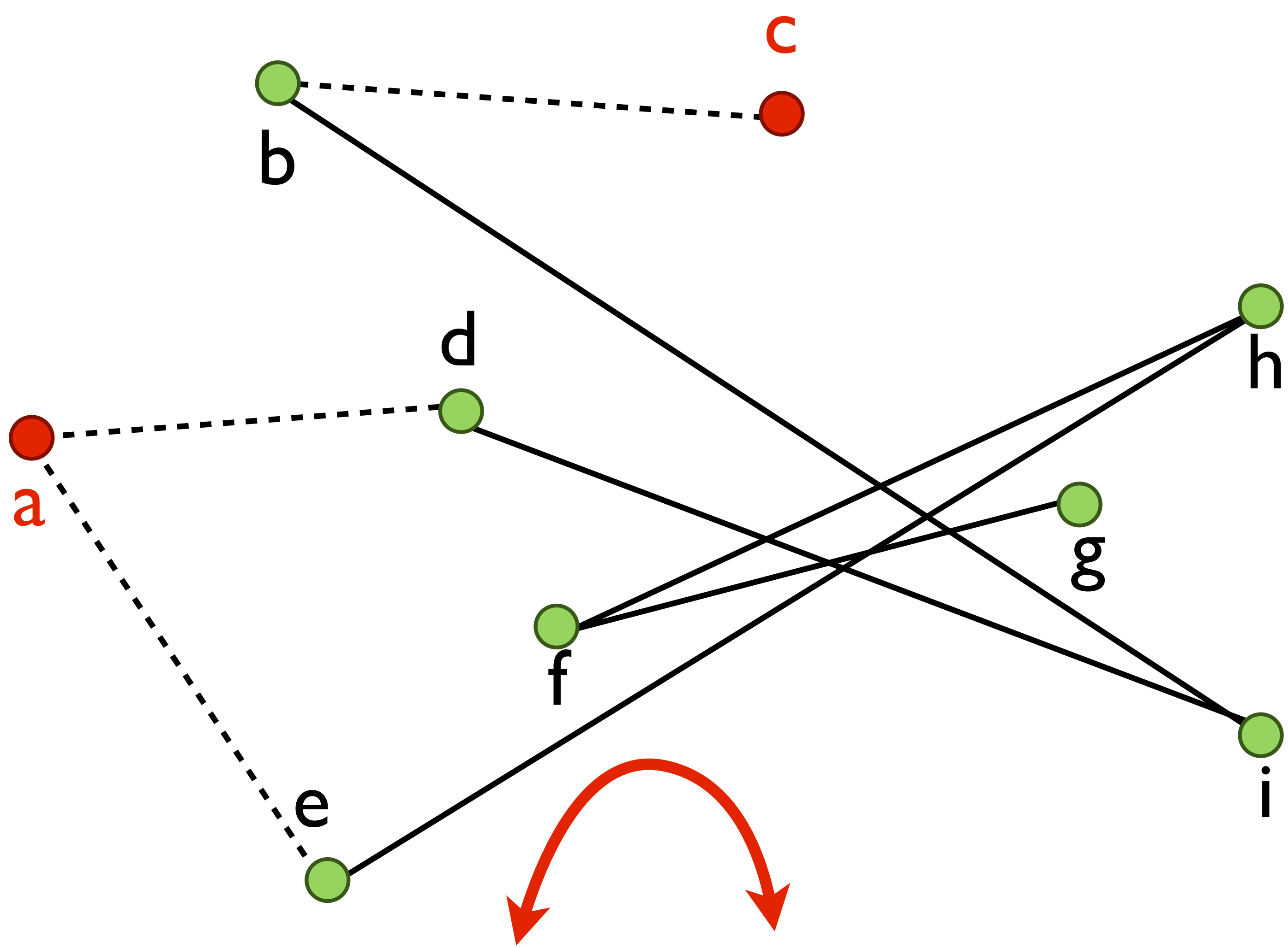
travelling salesman problem



$X^{(i)} = (a, b, i, d, c, e, h, f, g)$

example. pick two cities and “swap” them

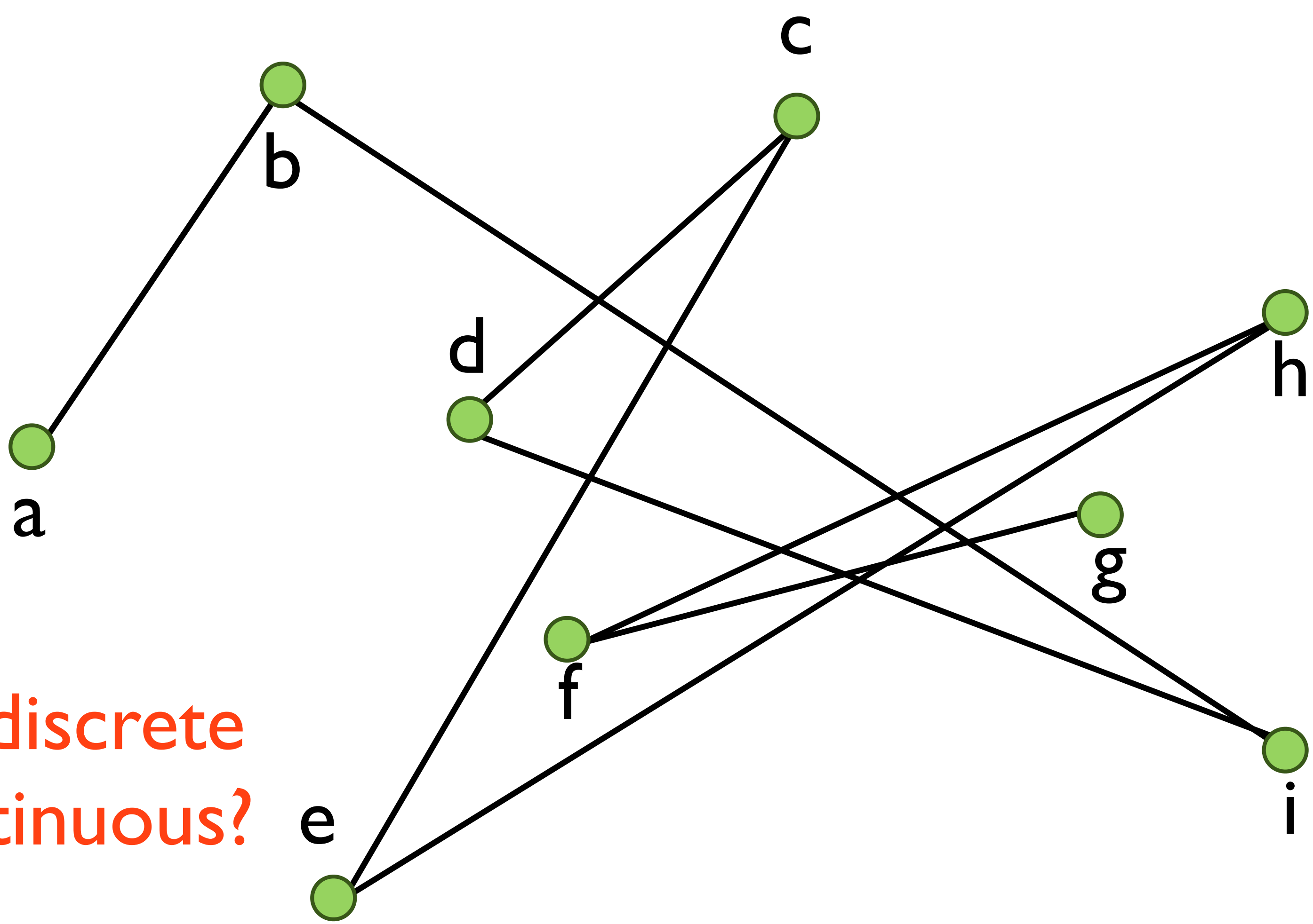
travelling salesman problem



$X^{(i)} = (c, b, i, d, a, e, h, f, g)$

example. pick two cities and “swap” them

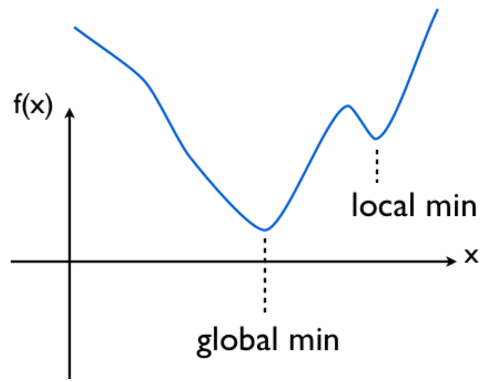
travelling salesman problem



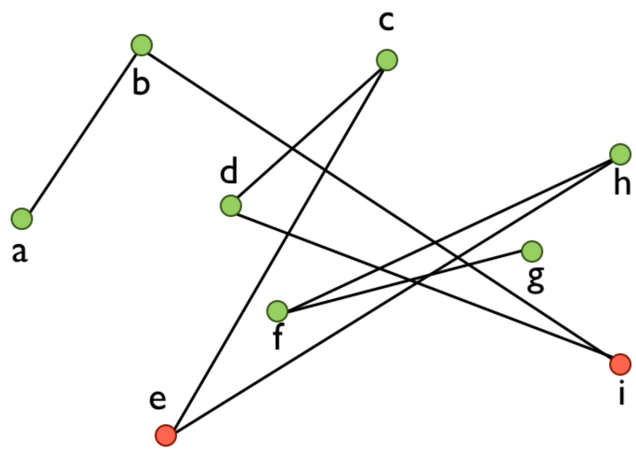
is this discrete or continuous?

$$X^{(i)} = (a, b, i, d, c, e, h, f, g)$$

SUMMARY Part 1. preliminaries



- global vs local optimisation
- local search
- deterministic vs stochastic
- discrete vs continuous optimisation

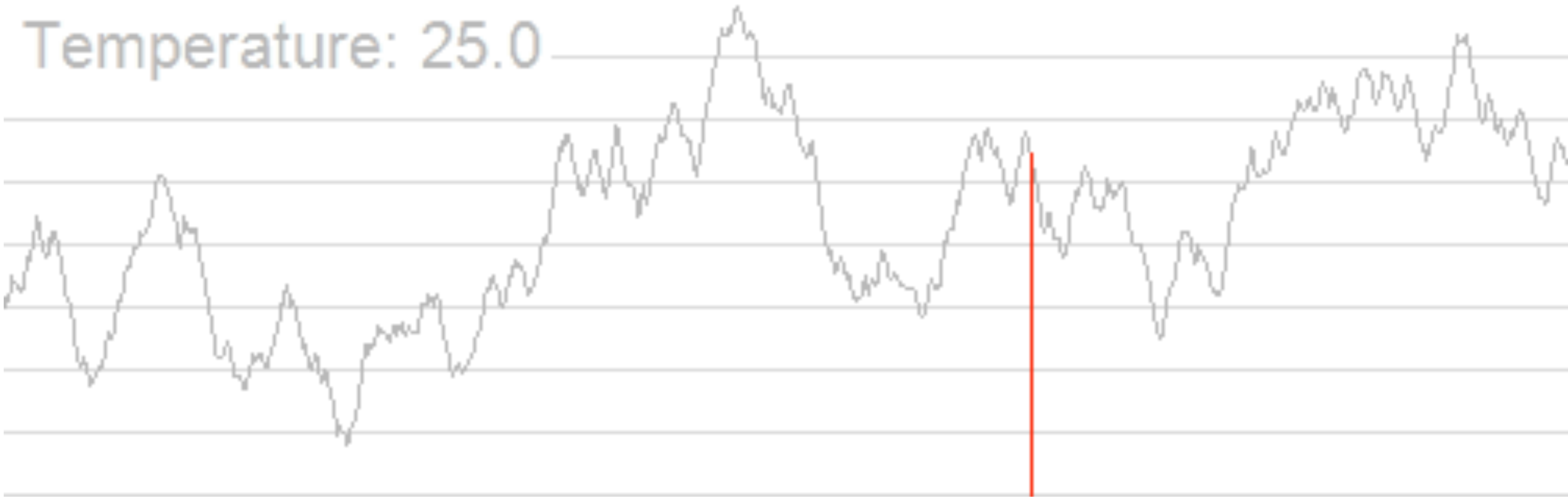


Part 2

simulated annealing

- metallurgy metaphor
- **annealing**: heat a material up, then slowly cool it down
- result: material becomes “less hard”, more ductile, more easy to mold and shape (lower energy)

	*optimisation*
"energy" in system	"cost" of current candidate soluion
during annealing, energy may decrease	during search, cost of best solution may decrease
during annealing, energy may increase with some probability	during search, we may accept a worse candidate solution with some probability

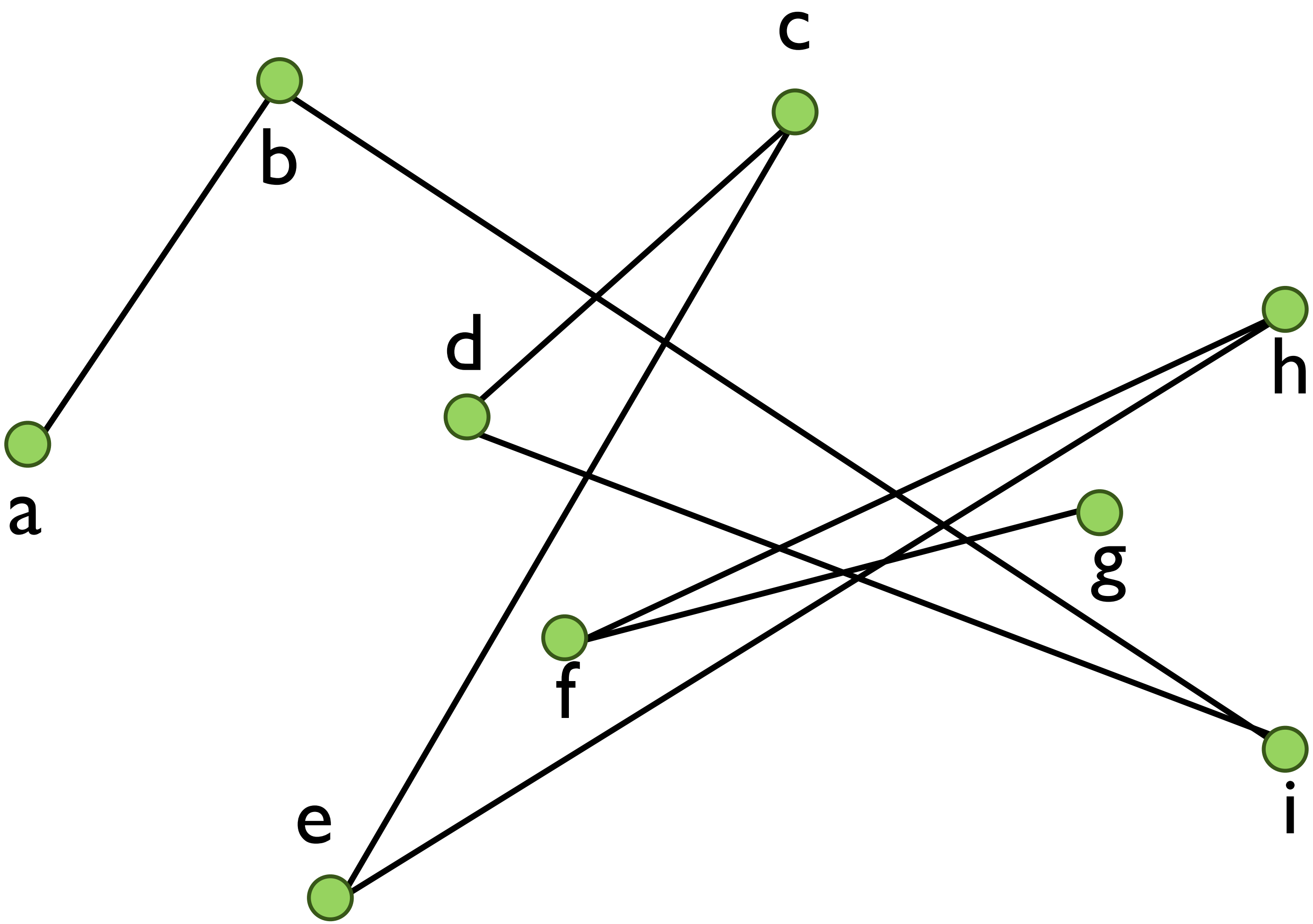


https://en.wikipedia.org/wiki/Simulated_annealing

first let's consider
"naive random search" where we
never take worse candidate

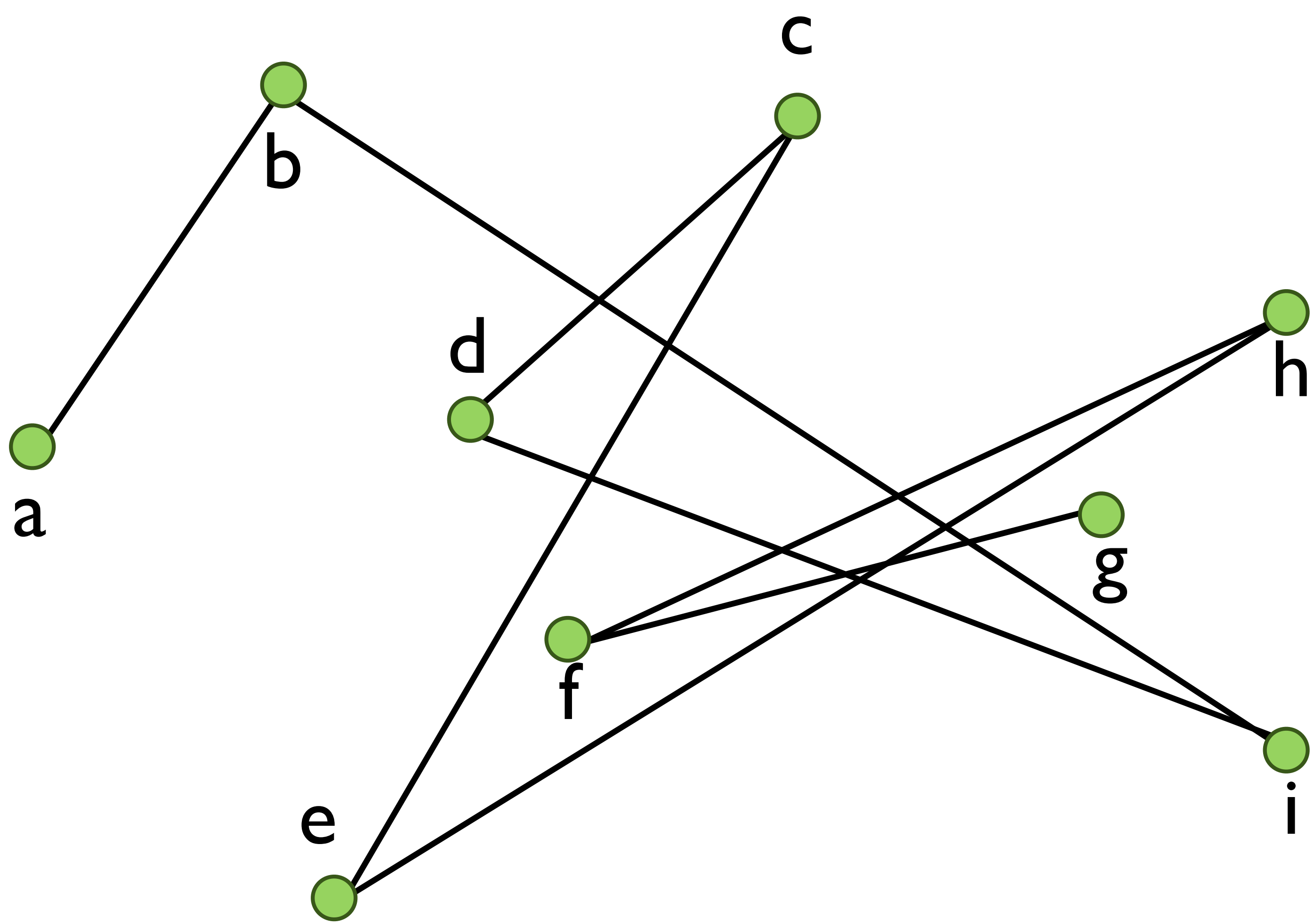
"energy" in system	"cost" of current candidate soluion
during annealing, energy may decrease	during search, cost of best solution may decrease
~~during annealing, energy may increase with some probability~~	~~during search, we may accept a worse candidate solution with some probability~~

1. pick random candidate solution $X^{(0)}$



$$X^{(0)} = (a, b, i, d, c, e, h, f, g)$$

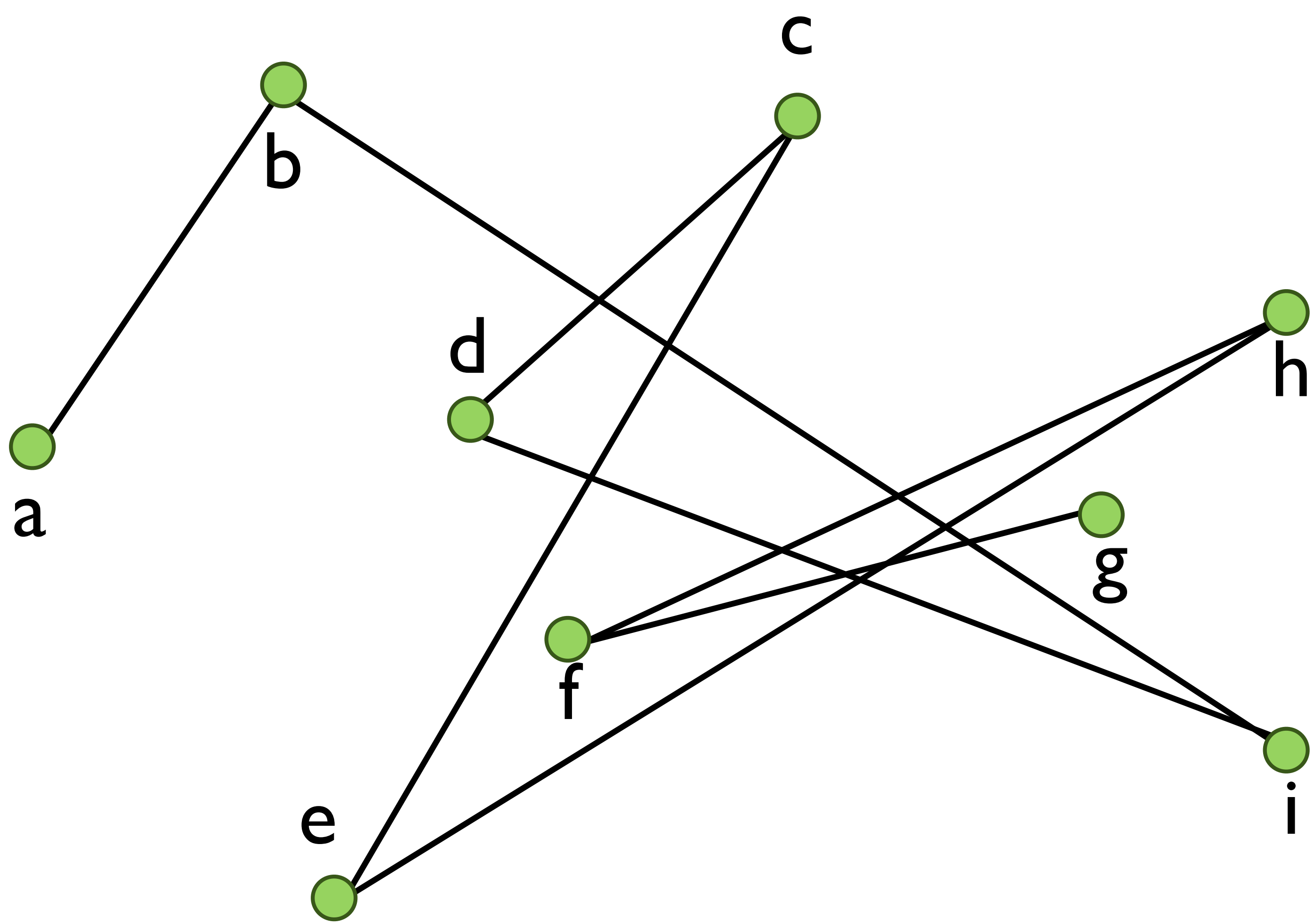
1. pick random candidate solution $X^{(0)}$



what is an example of a **neighbour** candidate solution?

$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

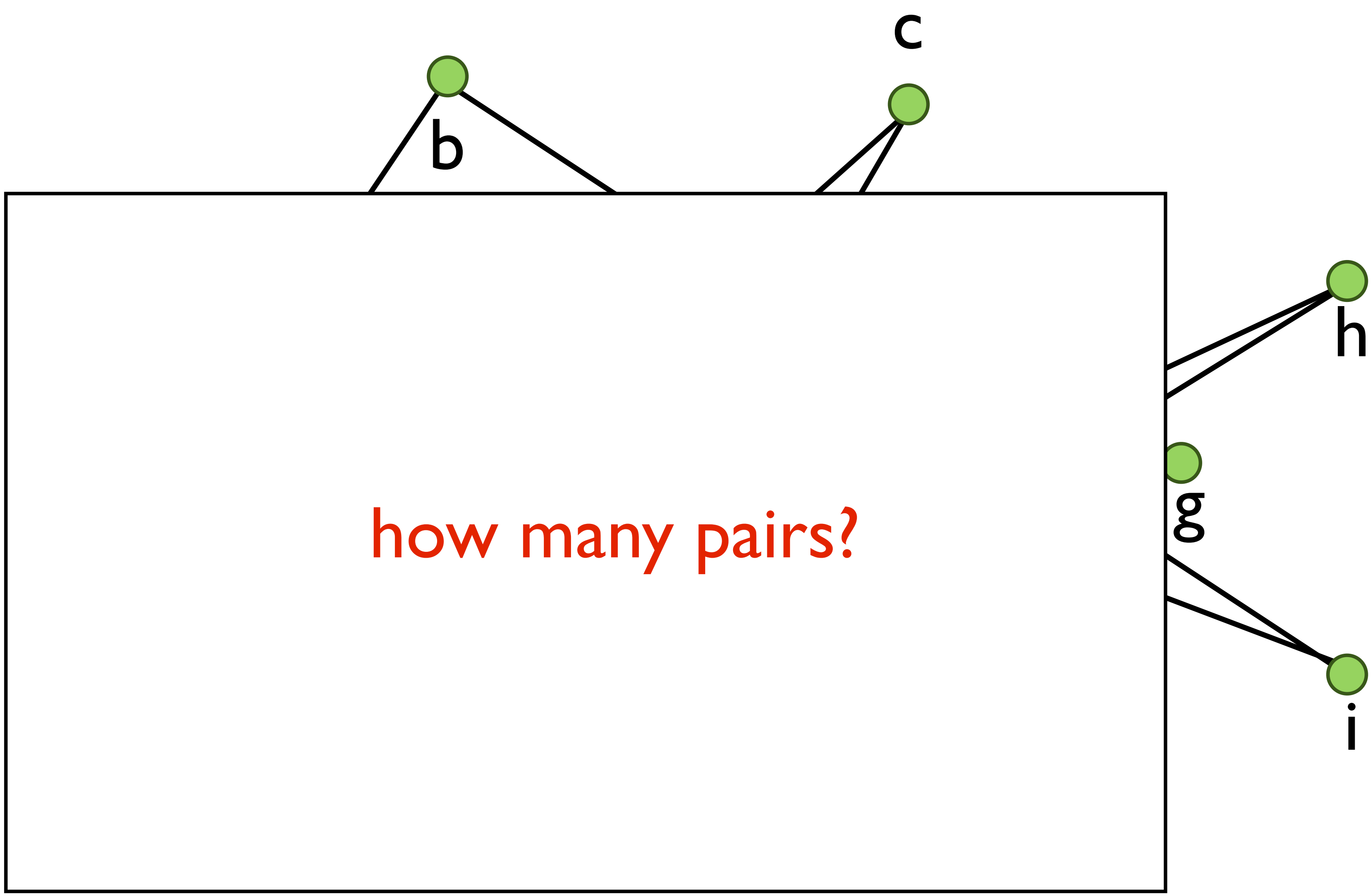
1. pick random candidate solution $X^{(0)}$



how many neighbours are there?

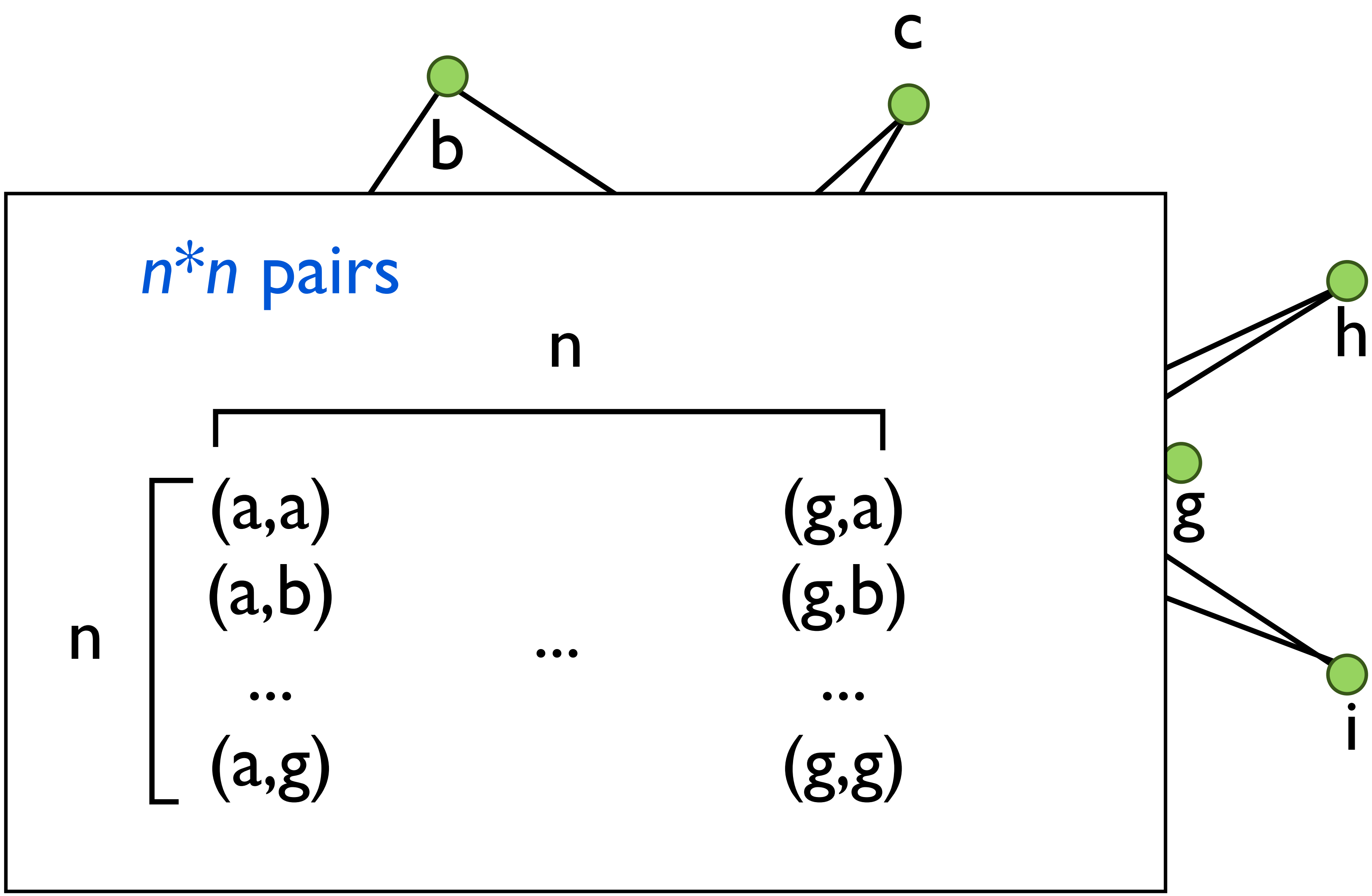
$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

1. pick random candidate solution $X^{(0)}$



$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

1. pick random candidate solution $X^{(0)}$



$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

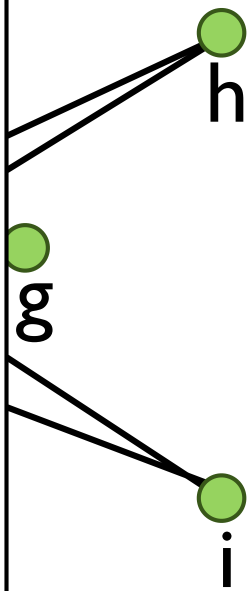
1. pick random candidate solution $X^{(0)}$

but don't want to swap same with same

...each column has one unwanted pair

$n*n$ - ?

$$\begin{array}{cccc} & \multicolumn{3}{c}{n} \\ n & \cancel{(a,a)} & & (g,a) \\ & (a,b) & \cdots & (g,b) \\ & \cdots & & \cdots \\ & (a,g) & & \cancel{(g,g)} \end{array}$$



$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

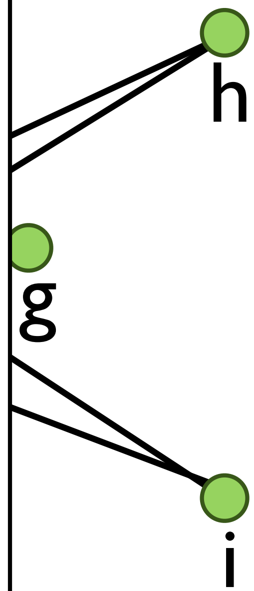
1. pick random candidate solution $X^{(0)}$

but don't want to swap same with same

...each column has one unwanted pair

$n*n - n$

$$
n \left[\overbrace{\begin{array}{ccc} \cancel{(a,a)} & & (g,a) \\ (a,b) & & (g,b) \\ \dots & \dots & \dots \\ (a,g) & & \cancel{(g,g)} \end{array}}^{n} \right.
$$



$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

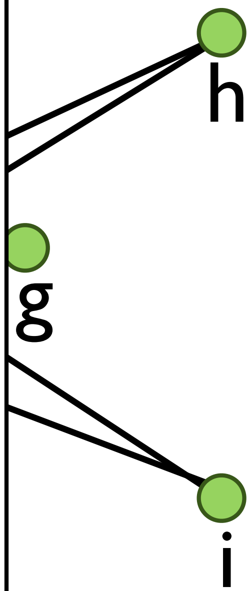
1. pick random candidate solution $X^{(0)}$

but don't want to swap same with same

...each column has one unwanted pair

$n(n - 1)$

$$
\begin{array}{cc}
 & \overbrace{\hphantom{(a,a) \quad \cdots \quad (g,a)}}^{n} \\
n & \left[\begin{array}{ccc}
\cancel{(a,a)} & & (g,a) \\
(a,b) & & (g,b) \\
\cdots & \cdots & \cdots \\
(a,g) & & \cancel{(g,g)}
\end{array}\right.
\end{array}
$$



$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

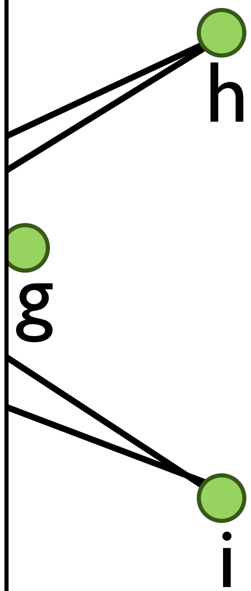
1. pick random candidate solution $X^{(0)}$

swapping (a,b) same as (b,a)

i.e. swapping is a **symmetric** relation

how many symmetric pairs?

$$
\begin{array}{cc}
 & \overbrace{\hphantom{(a,a) \quad \cdots \quad (g,a)}}^{n} \\
n & \left[\begin{array}{ccc}
\cancel{(a,a)} & & (g,a) \\
(a,b) & \cdots & (g,b) \\
\cdots & & \cdots \\
(a,g) & & \cancel{(g,g)}
\end{array}\right.
\end{array}
$$



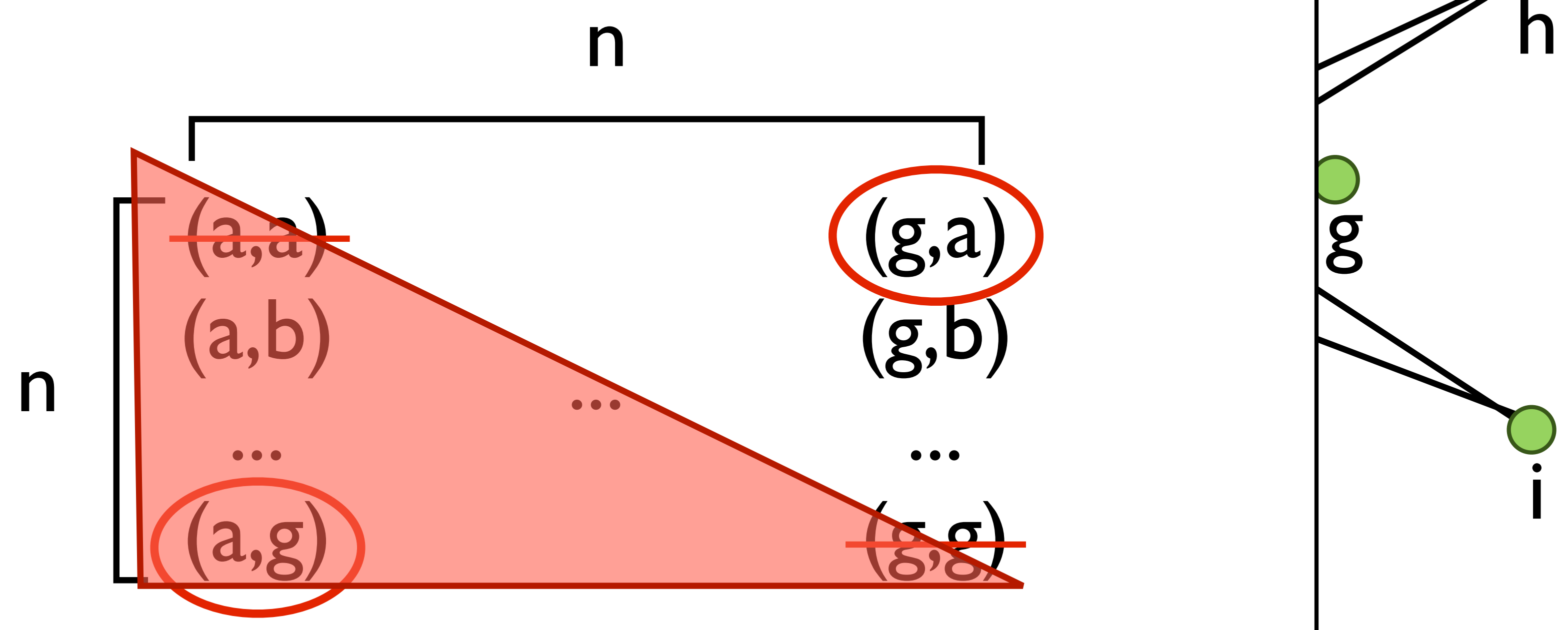
$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

1. pick random candidate solution $X^{(0)}$

swapping (a,b) same as (b,a)

i.e. swapping is a **symmetric** relation

how many symmetric pairs?

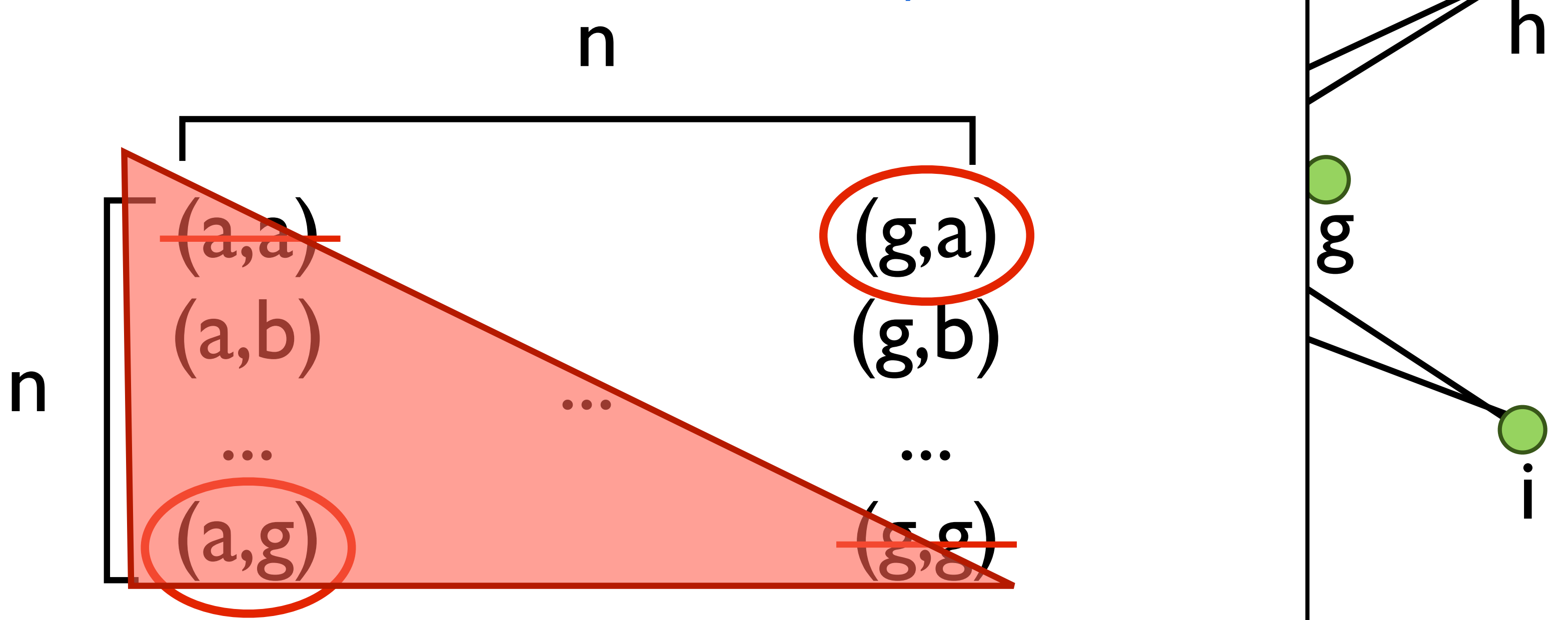


$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

1. pick random candidate solution $X^{(0)}$

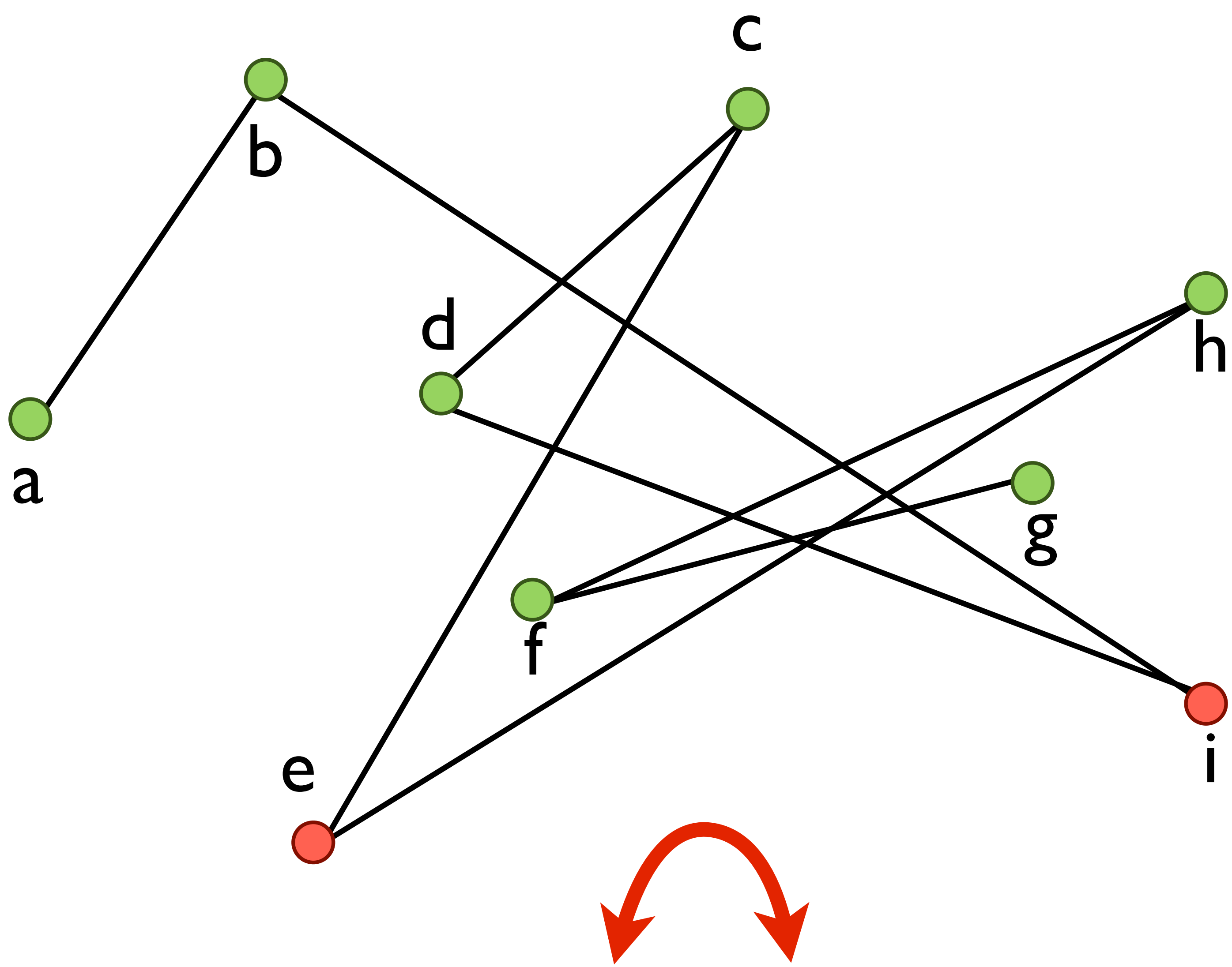
$\frac{n(n-1)}{2}$ pairs

(and each pair represents a neighbour candidate solution)



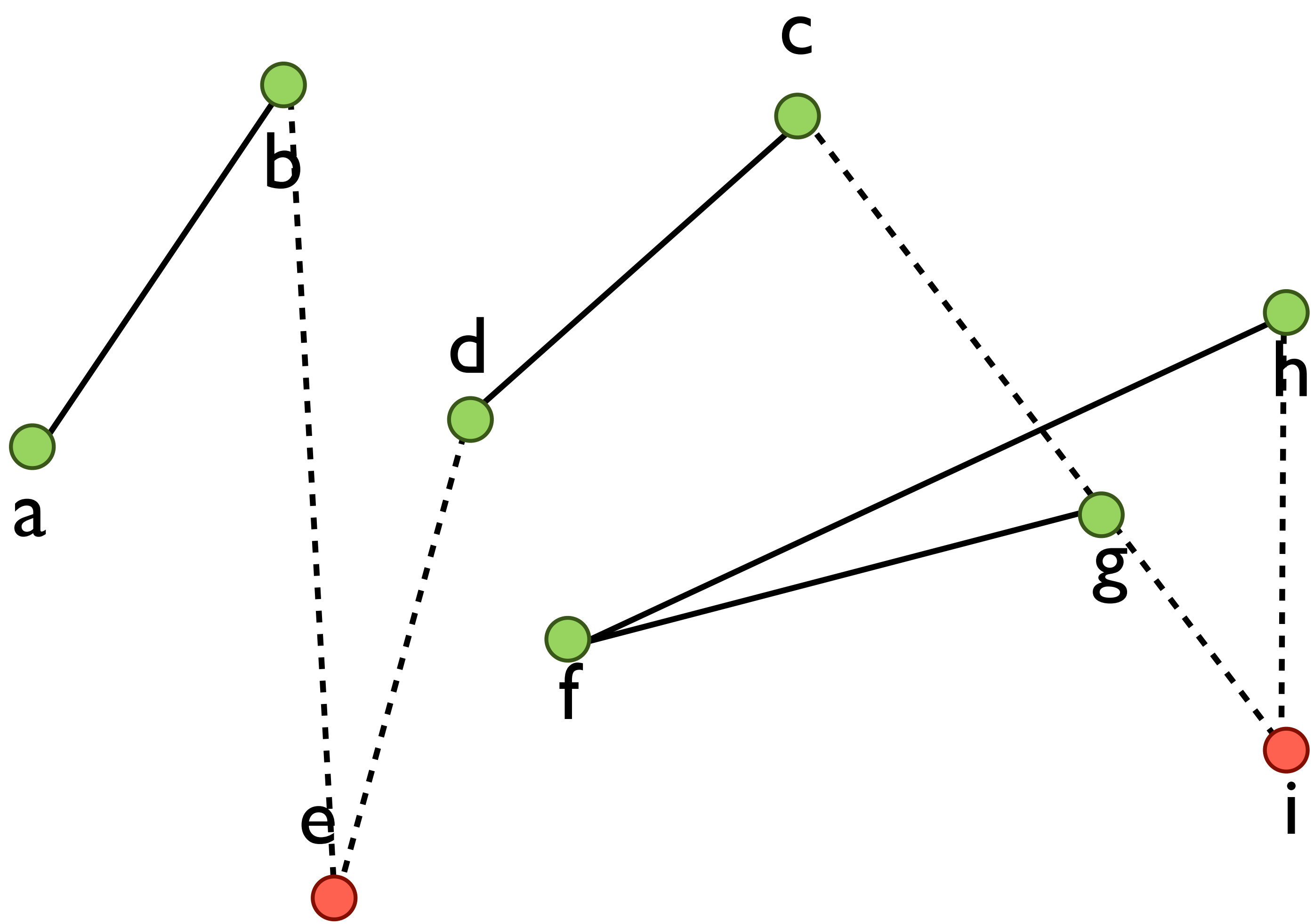
$X^{(0)} = (a, b, i, d, c, e, h, f, g)$

2. randomly choose a neighbour $Z^{(i)}$



$X^{(i)} = (a, b, i, d, c, e, h, f, g)$

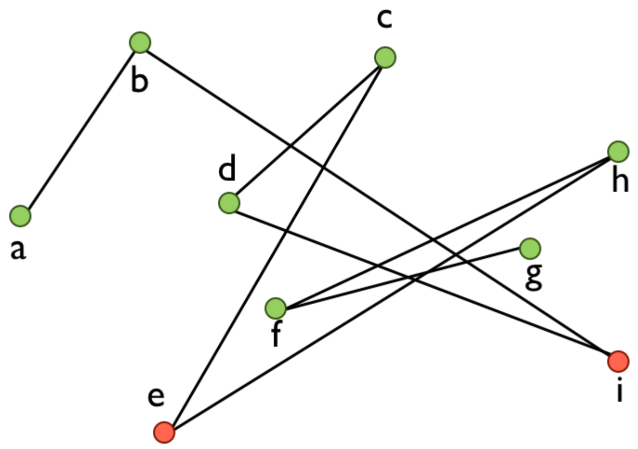
2. randomly choose a neighbour $Z^{(i)}$



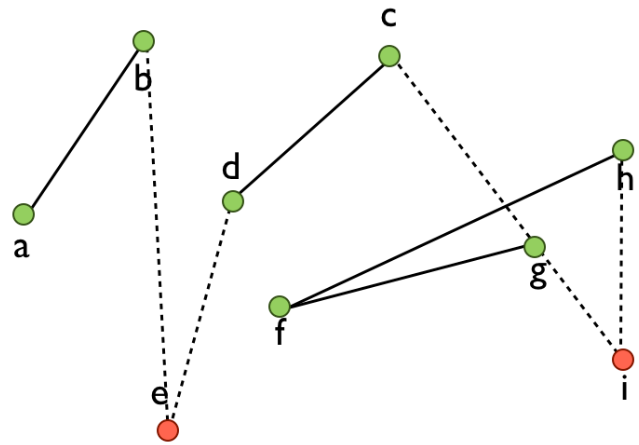
$$X^{(i)} = (a, b, i, d, c, e, h, f, g)$$

$$Z^{(i)} = (a, b, e, d, c, i, h, f, g)$$

3. compare cost of $X^{(i)}$ and $Z^{(i)}$



$f(X^{(i)}) = 5263$

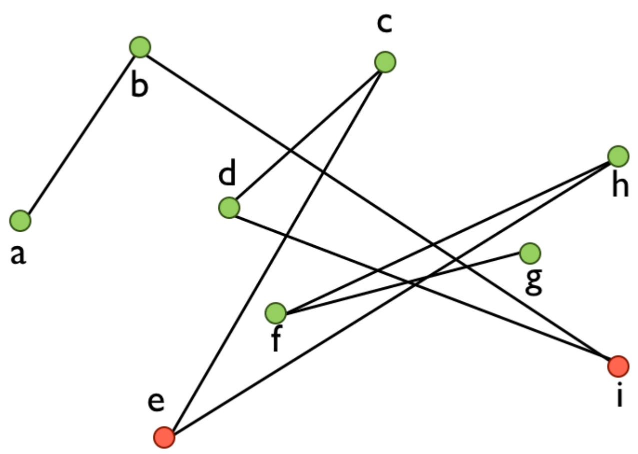


$f(Z^{(i)}) = 1263$

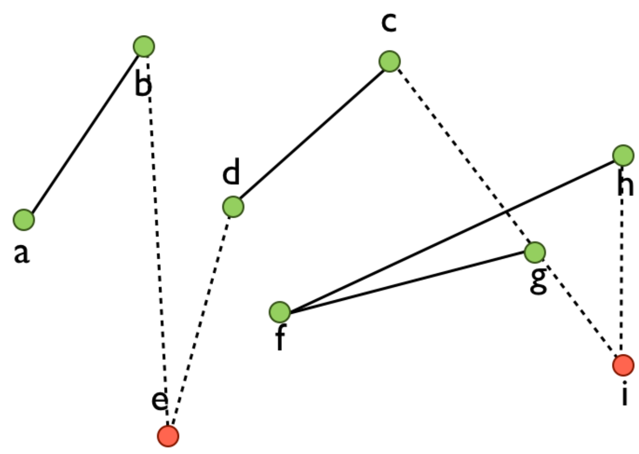
if $f(Z^{(i)}) < f(X^{(i)})$ then set $X^{(i+1)} = Z^{(i)}$

else set $X^{(i+1)} = X^{(i)}$

4. if reached stopping criteria then STOP



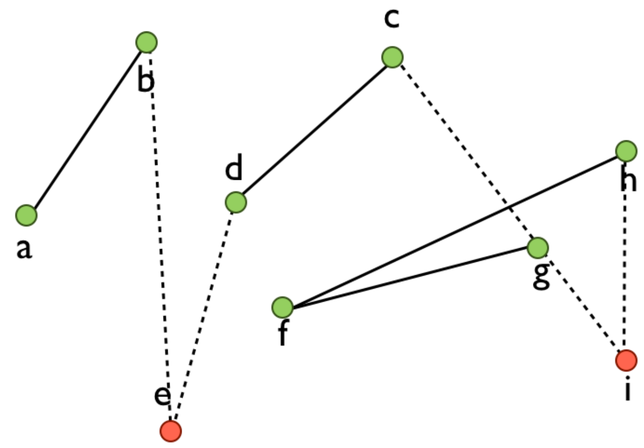
$f(X^{(i)}) = 5263$



$f(Z^{(i)}) = 1263$

...else repeat from step 2

naive random search



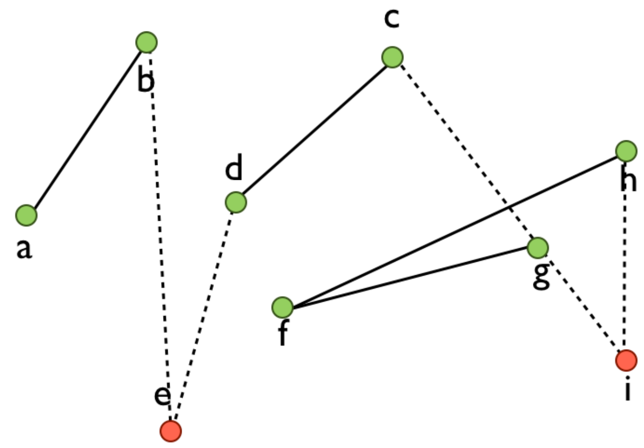
1. pick random candidate solution $X^{(0)}$

2. randomly choose a neighbour $Z^{(i)}$

3. compare cost of $X^{(i)}$ and $Z^{(i)}$, use to set $X^{(i+1)}$

4. if reached stopping criteria then STOP
   else repeat from step 2

naive random search



1. pick random candidate solution $X^{(0)}$

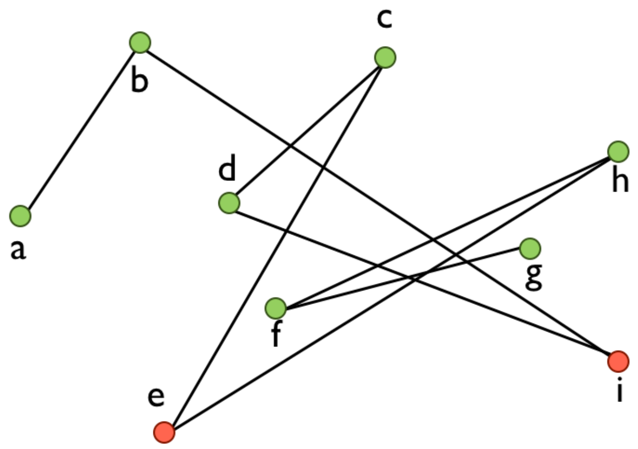
2. randomly choose a neighbour $Z^{(i)}$

3. compare cost of $X^{(i)}$ and $Z^{(i)}$, use to set $X^{(k+1)}$

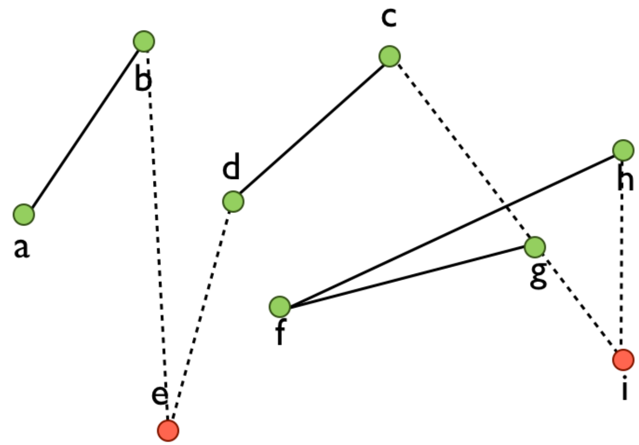
4. if reached stoppin
   else repeat from

simulated annealing changes this step...

when should you accept a new candidate solution, $X^{(i+1)} = Z^{(i)}$



$f(X^{(i)}) = 5263$

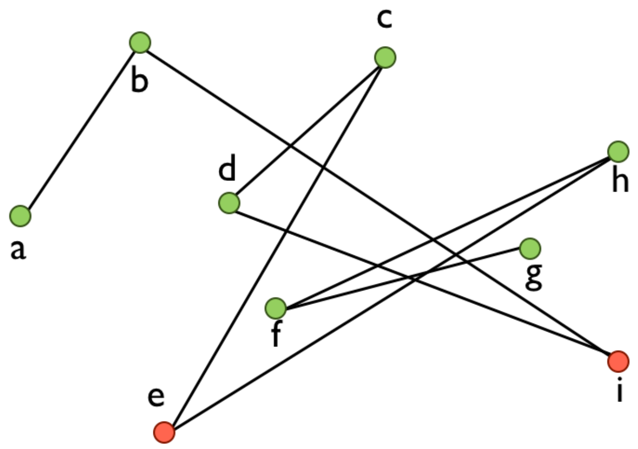


$f(Z^{(i)}) = 1263$

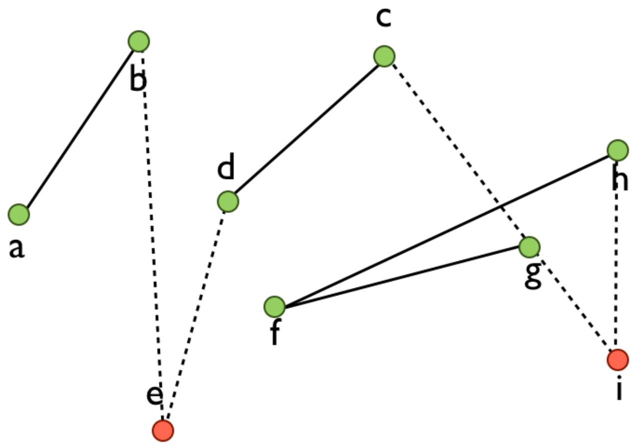
if $Z^{(i)}$ is an improvement, then take it
(same as naive random search)

but what if $Z^{(i)}$ is worse?

when should you accept a new candidate solution, $X^{(i+1)} = Z^{(i)}$



$f(X^{(i)}) = 5263$

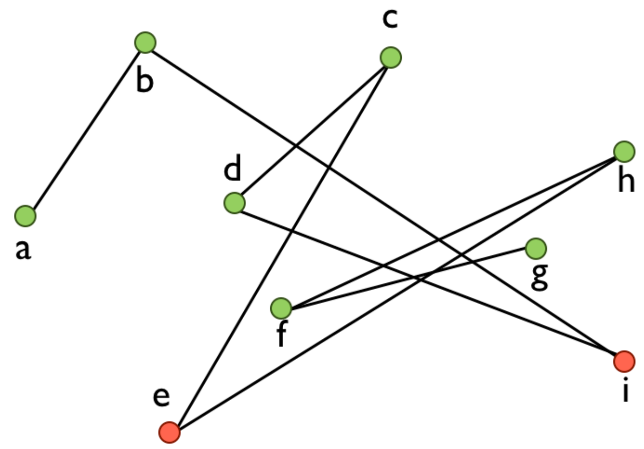


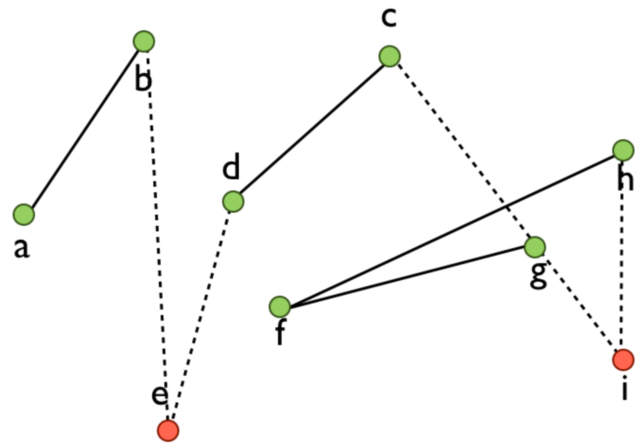
$f(Z^{(i)}) = 1263$

if "temperature" of system is **high** then **more likely** to accept a worse candidate solution

if "temperature" of system is **low** then **less likely** to accept a worse candidate solution

when should you accept a new candidate solution,
$X^{(i+1)} = Z^{(i)}$





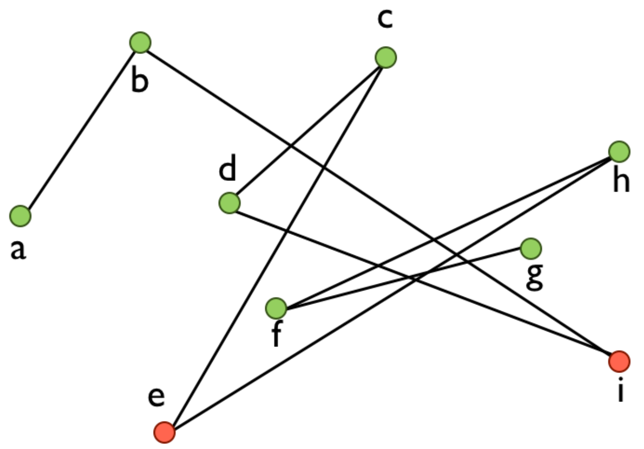
$f(X^{(i)}) = 5263$

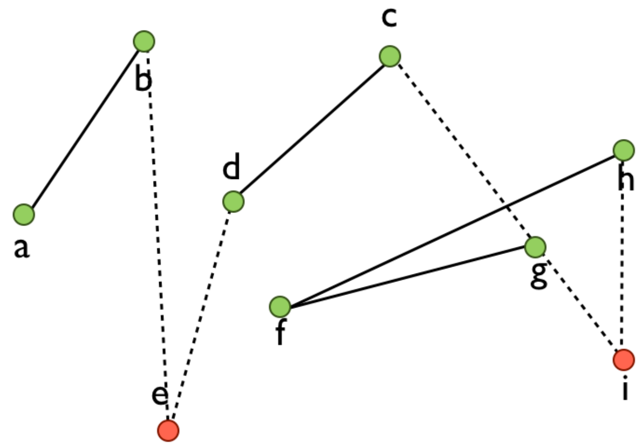
$f(Z^{(i)}) = 1263$

start by making temperature **high**,
as iterations continue "**cooling**" occurs
i.e. temperature decreases with more iterations

what's the effect?

when should you accept a new candidate solution,
$X^{(i+1)} = Z^{(i)}$





$f(X^{(i)}) = 5263$

$f(Z^{(i)}) = 1263$

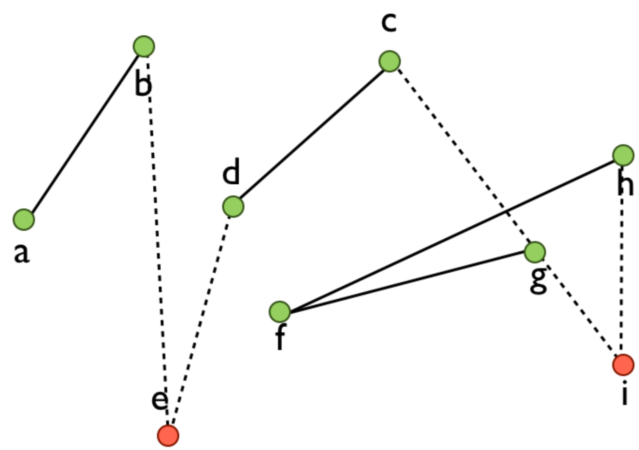
temperature T changed according to a **cooling schedule**

start T,
end T,
decrement T
iterations per T

temperature should always
be non-negative
$T^{(i)} \geq 0$ for all i

should we take a **worse** candidate solution?

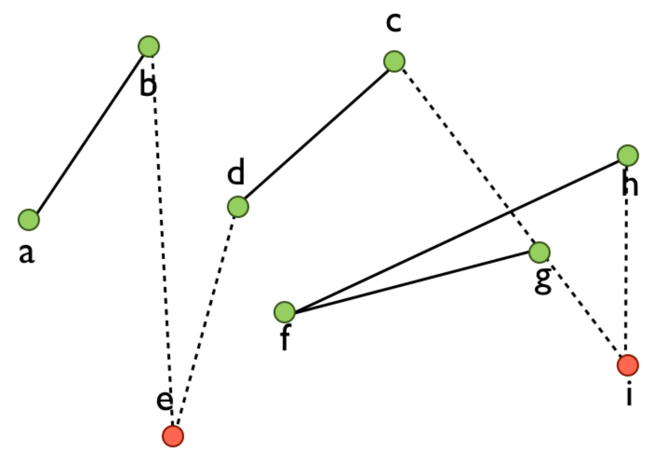
$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



$f(Z^{(i)}) = 1263$

should we take a **worse** candidate solution?

$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



$$f(Z^{(i)}) = 1263$$

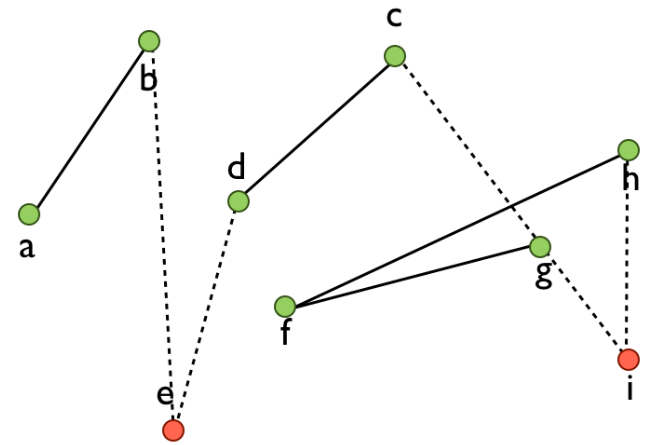
note. analogy of law of thermodynamics

$$P(\delta E) = \exp(-\delta E / kt)$$

...probability of increase in magnitude of energy, δE

should we take a **worse** candidate solution?

$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



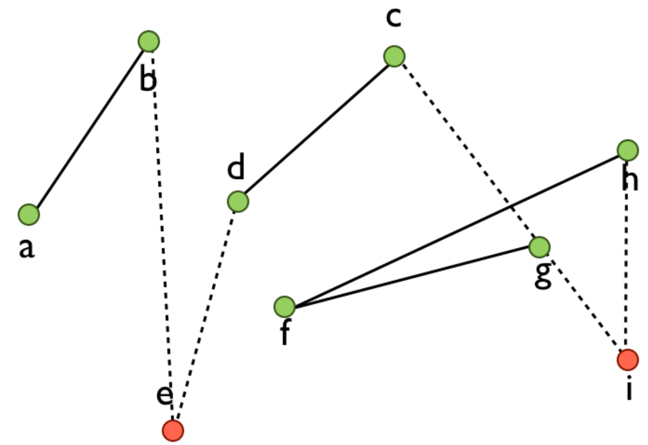
$f(Z^{(i)}) = 1263$

positive or negative?

$(f(Z^{(i)}) - f(X^{(i)}))$

should we take a **worse** candidate solution?

$$\exp\left(\frac{-\left(f(Z^{(i)}) - f(X^{(i)})\right)}{T^{(i)}} \right)$$



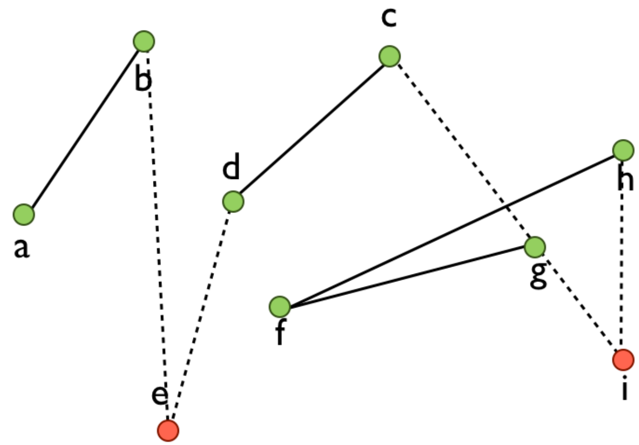
$f(Z^{(i)}) = 1263$

positive or negative?

$-\left(f(Z^{(i)}) - f(X^{(i)})\right)$

should we take a **worse** candidate solution?

$$\exp\left(\frac{-\left(f(Z^{(i)}) - f(X^{(i)})\right)}{T^{(i)}}\right)$$



$f(Z^{(i)}) = 1263$

positive or negative?

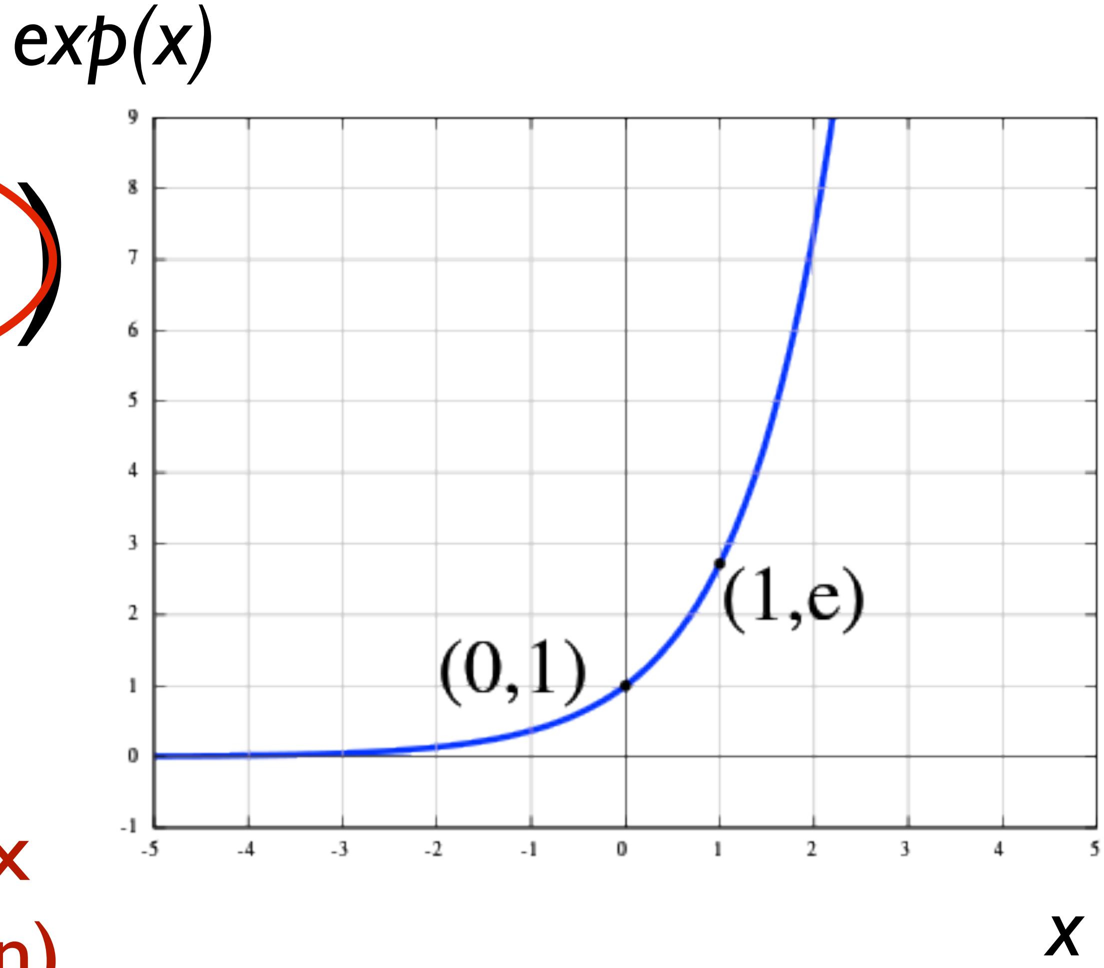
$$\frac{-\left(f(Z^{(i)}) - f(X^{(i)})\right)}{T^{(i)}}$$

(remember $T^{(i)} \geq 0$)

should we take a **worse** candidate solution?

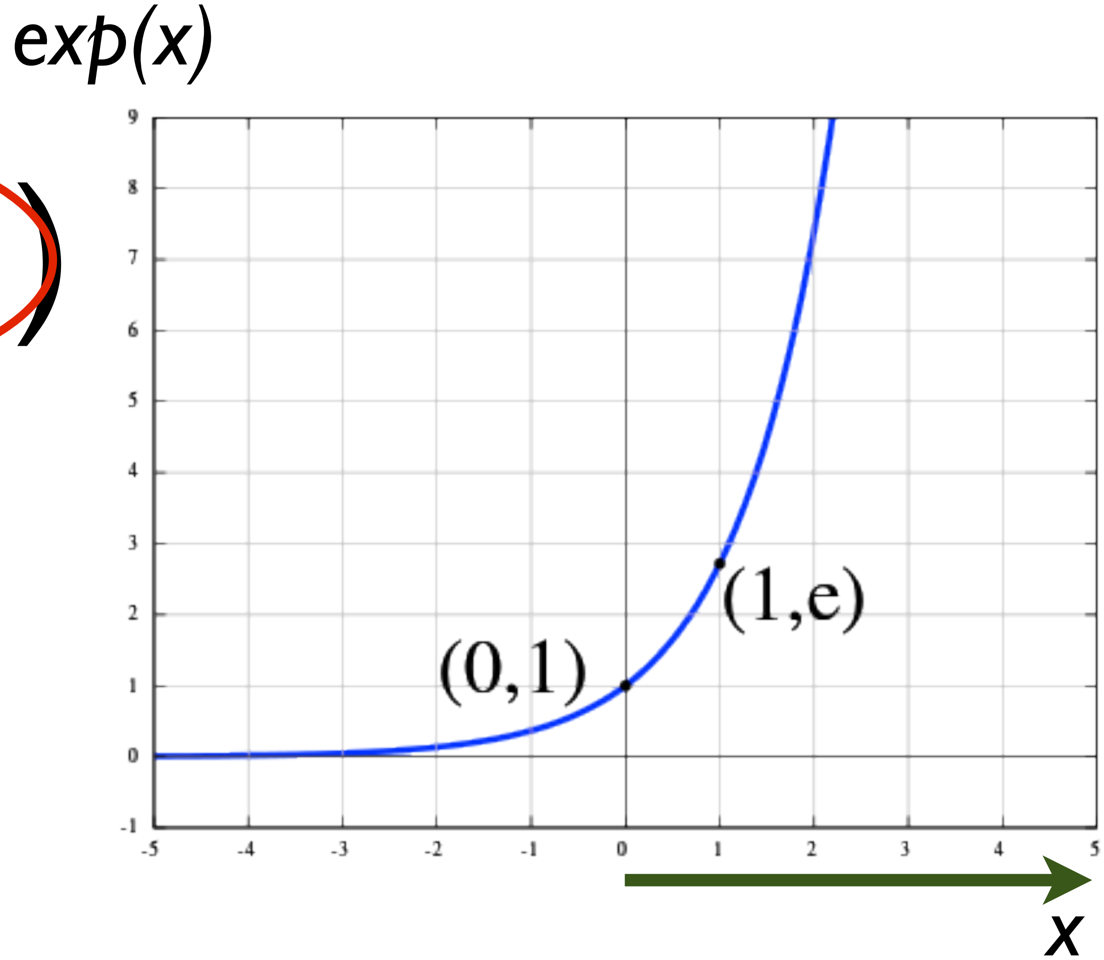
$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$

what is range of this expression? (i.e. min/max values that it could return)



should we take a **worse** candidate solution?

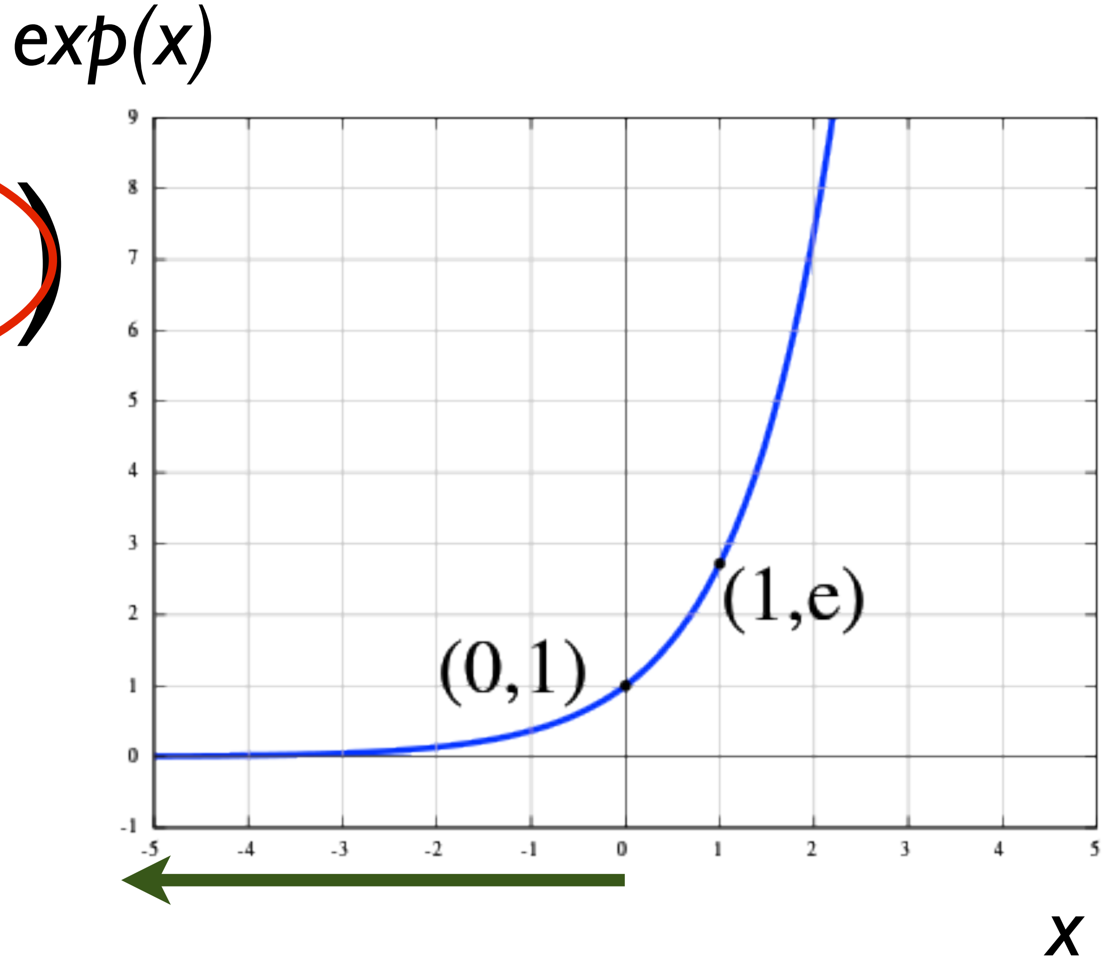
$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



$Z^{(i)}$ was better than $X^{(i)}$

should we take a **worse** candidate solution?

$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



$Z^{(i)}$ was worse than $X^{(i)}$

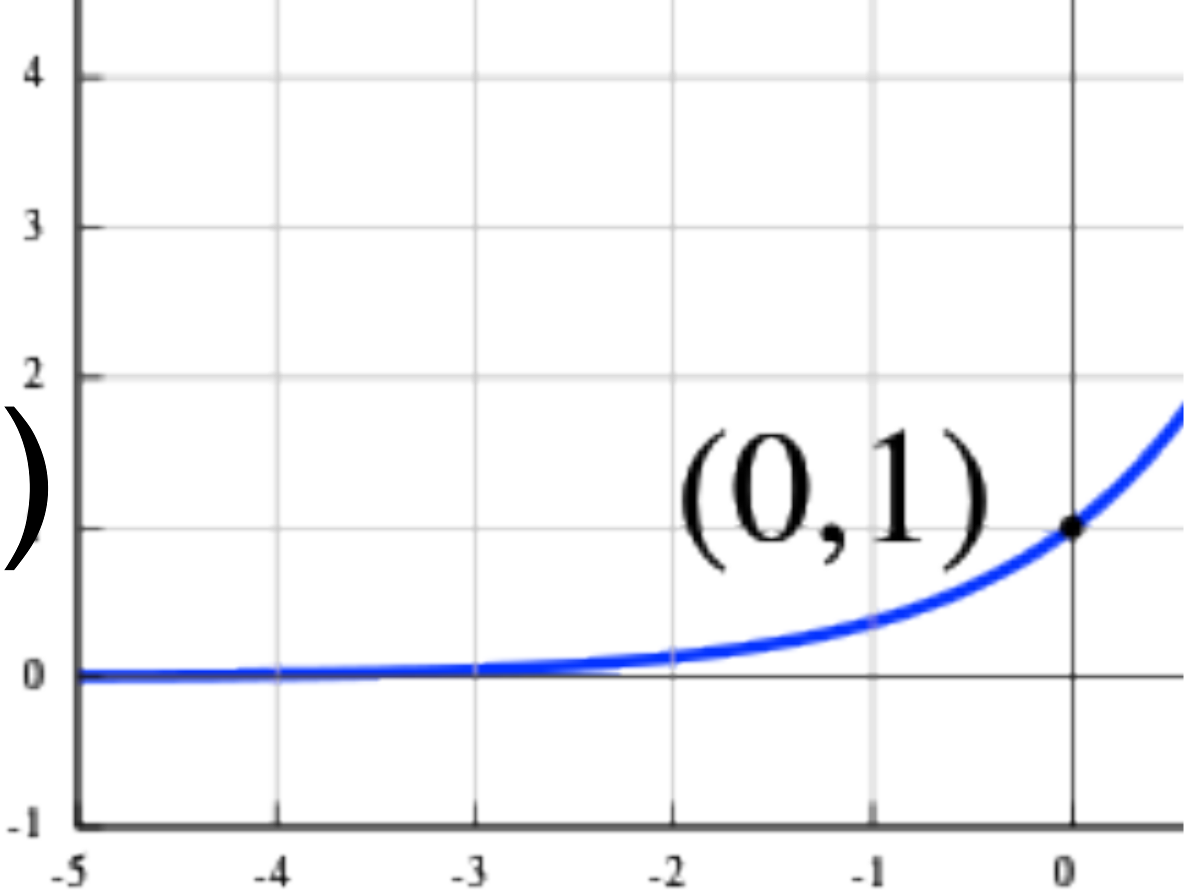
choose a random real r between 0 and 1

take worse candidate solution if:

$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right) > r$$

so, when are we more likely to take a worse solution?

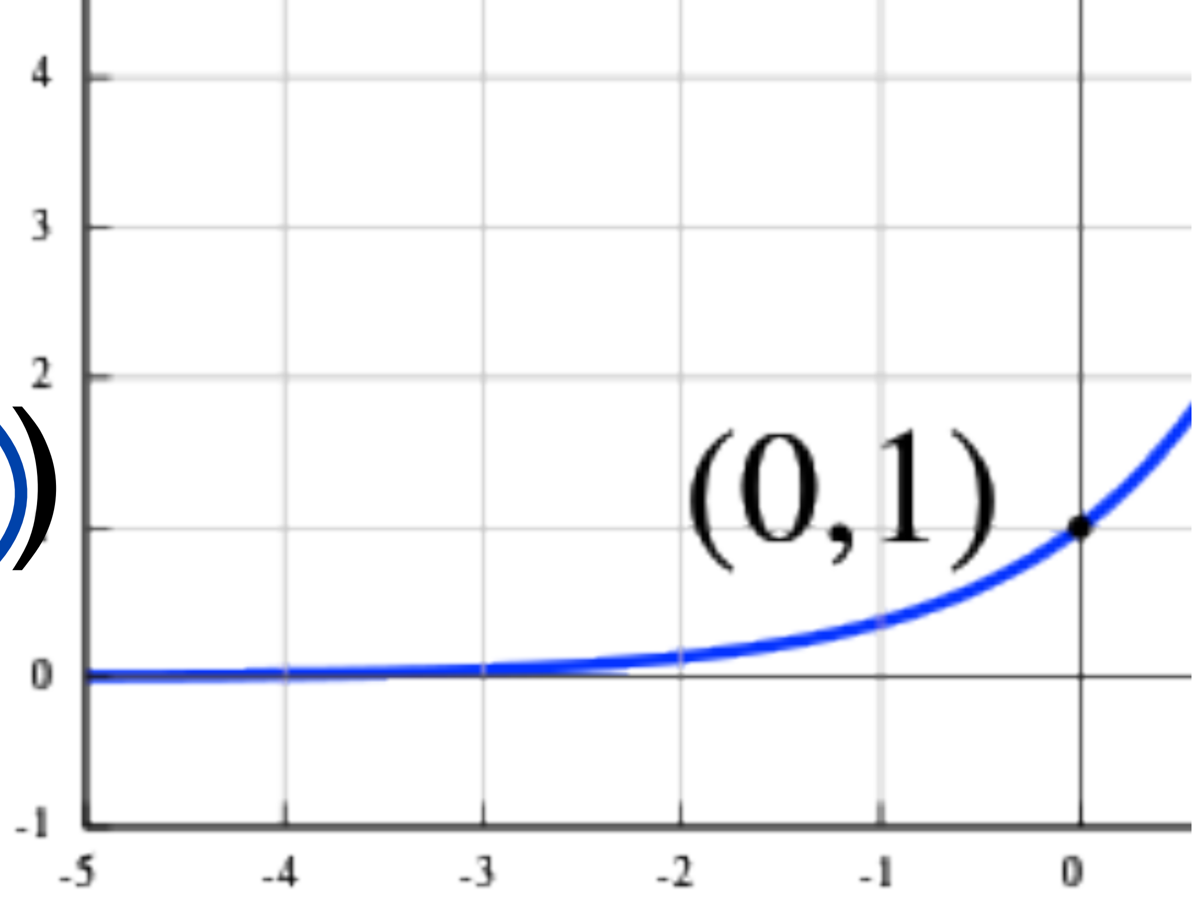
$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



what happens if
temperature is high?

$$\exp\left(\frac{-\left(f(Z^{(i)}) - f(X^{(i)})\right)}{T^{(i)}}\right)$$

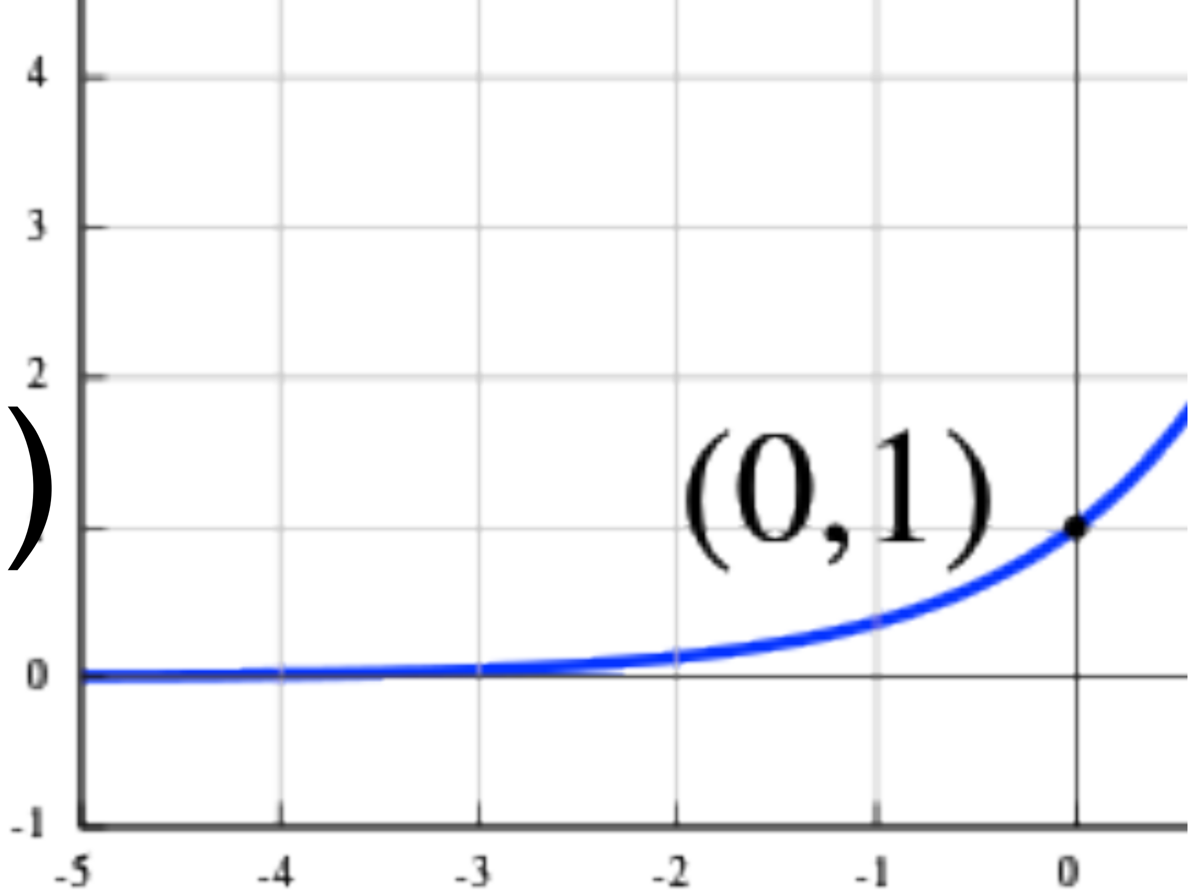
this will move towards zero



what happens if temperature is high?

increasing T moves us this way, increases exp(...)

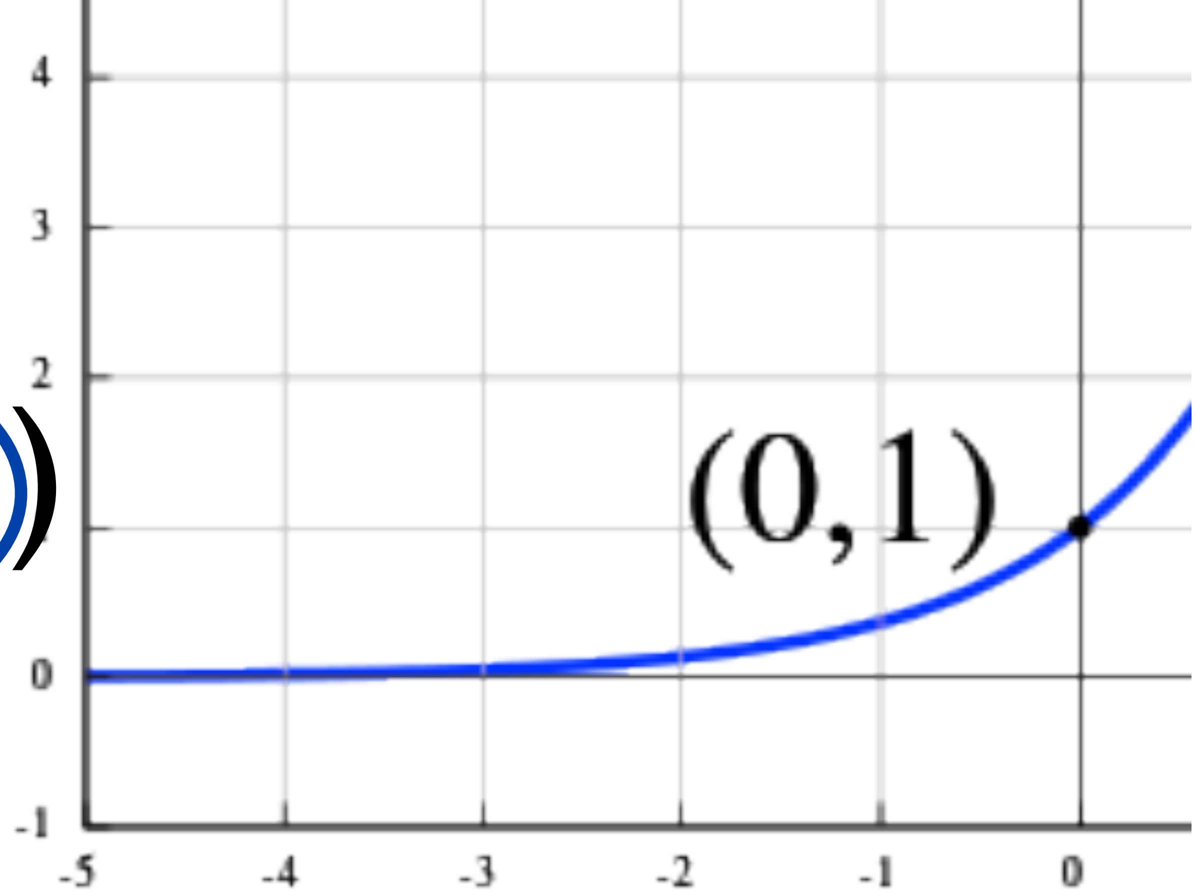
$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



what happens if cost difference is low?

$$\exp\left(-\frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$

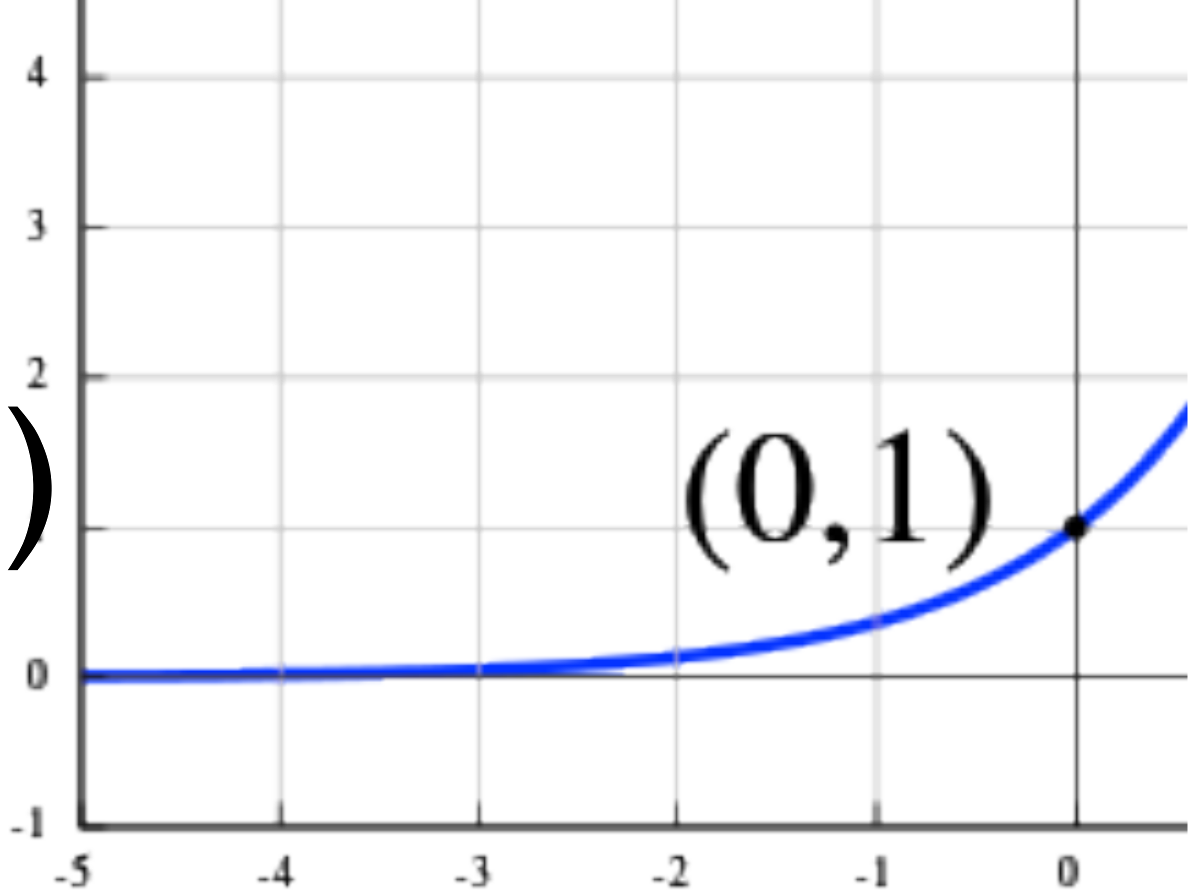
this will move towards zero



what happens if cost difference is low?

decreasing Δf moves us this way, increases exp(...)

$$\exp\left(- \frac{(f(Z^{(i)}) - f(X^{(i)}))}{T^{(i)}} \right)$$



we’re more likely to accept worse solution if:
(1) temperature is high
(2) the increase in cost is not too much

- Let $s = s_0$
- For $k = 0$ through $k_{\max}$ (exclusive):
  - $T \leftarrow \text{temperature}(k / k_{\max})$
  - Pick a random neighbour, $s_{\text{new}} \leftarrow \text{neighbour}(s)$
  - If $P(E(s), E(s_{\text{new}}), T) \geq \text{random}(0, 1)$:
    - $s \leftarrow s_{\text{new}}$
- Output: the final state s

https://en.wikipedia.org/wiki/Simulated_annealing

cooling schedule implemented in here

- Let $s = s_0$
- For $k = 0$ through $k_{\max}$ (exclusive):
    - $T \leftarrow \text{temperature}(k / k_{\max})$
    - Pick a random neighbour, $s_{\text{new}} \leftarrow \text{neighbour}(s)$
    - If $P(E(s), E(s_{\text{new}}), T) \geq \text{random}(0, 1)$:
        - $s \leftarrow s_{\text{new}}$
- Output: the final state s

https://en.wikipedia.org/wiki/Simulated_annealing probability of accepting the new candidate implemented here

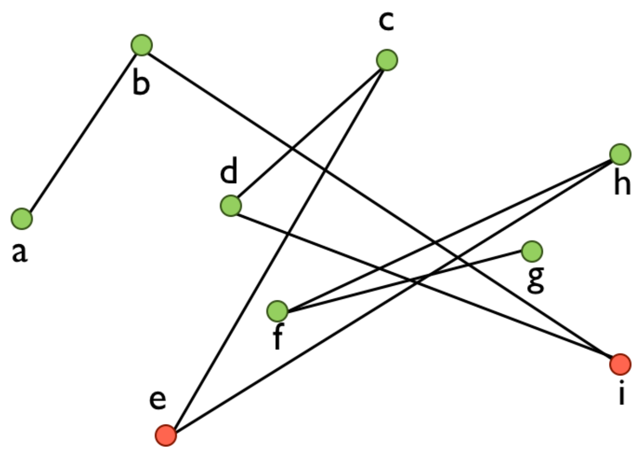
- Let $s = s_0$
- For $k = 0$ through $k_{\max}$ (exclusive):
  - $T \leftarrow \mathrm{temperature}(k / k_{\max})$
  - Pick a random neighbour, $s_{\mathrm{new}} \leftarrow \mathrm{neighbour}(s)$
  - If $P(E(s), E(s_{\mathrm{new}}), T) \geq \mathrm{random}(0, 1)$:
    - $s \leftarrow s_{\mathrm{new}}$
- Output: the final state s

https://en.wikipedia.org/wiki/Simulated_annealing

http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/

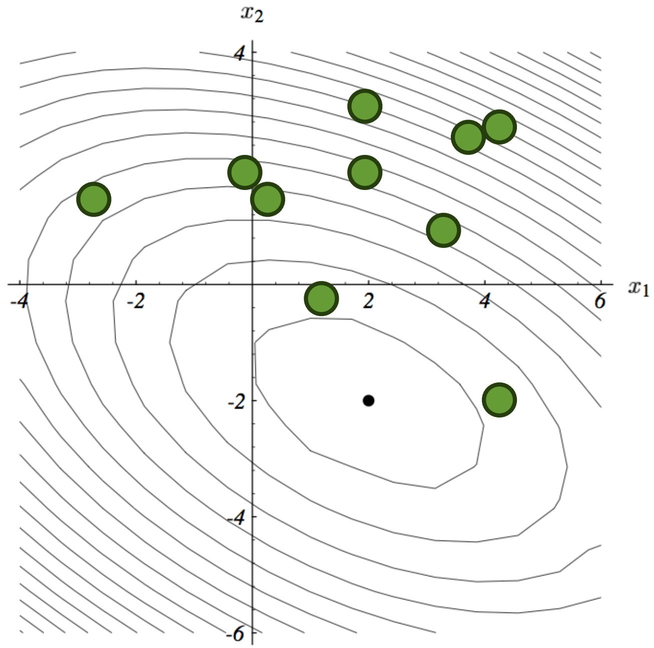
SUMMARY Part 2. simulated annealing

- naive random search
- probabilistically accept worse candidate
- based on "temperature"
- cooling (annealing) schedule

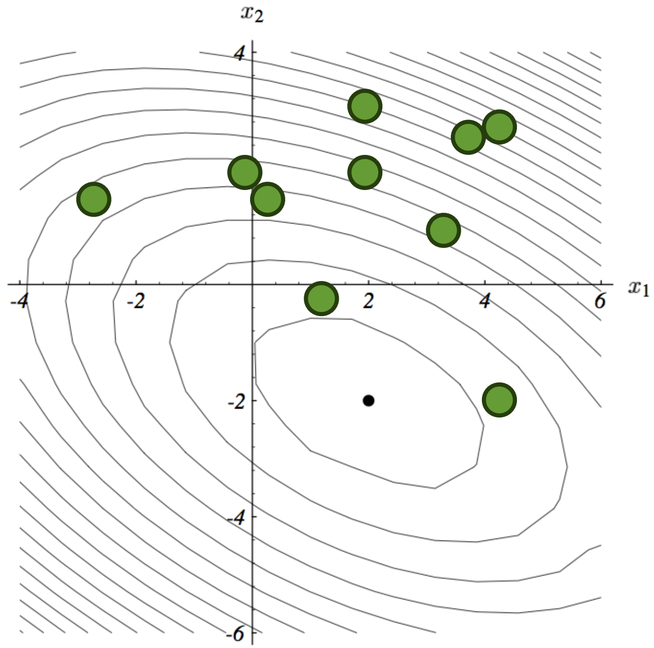


Part 3

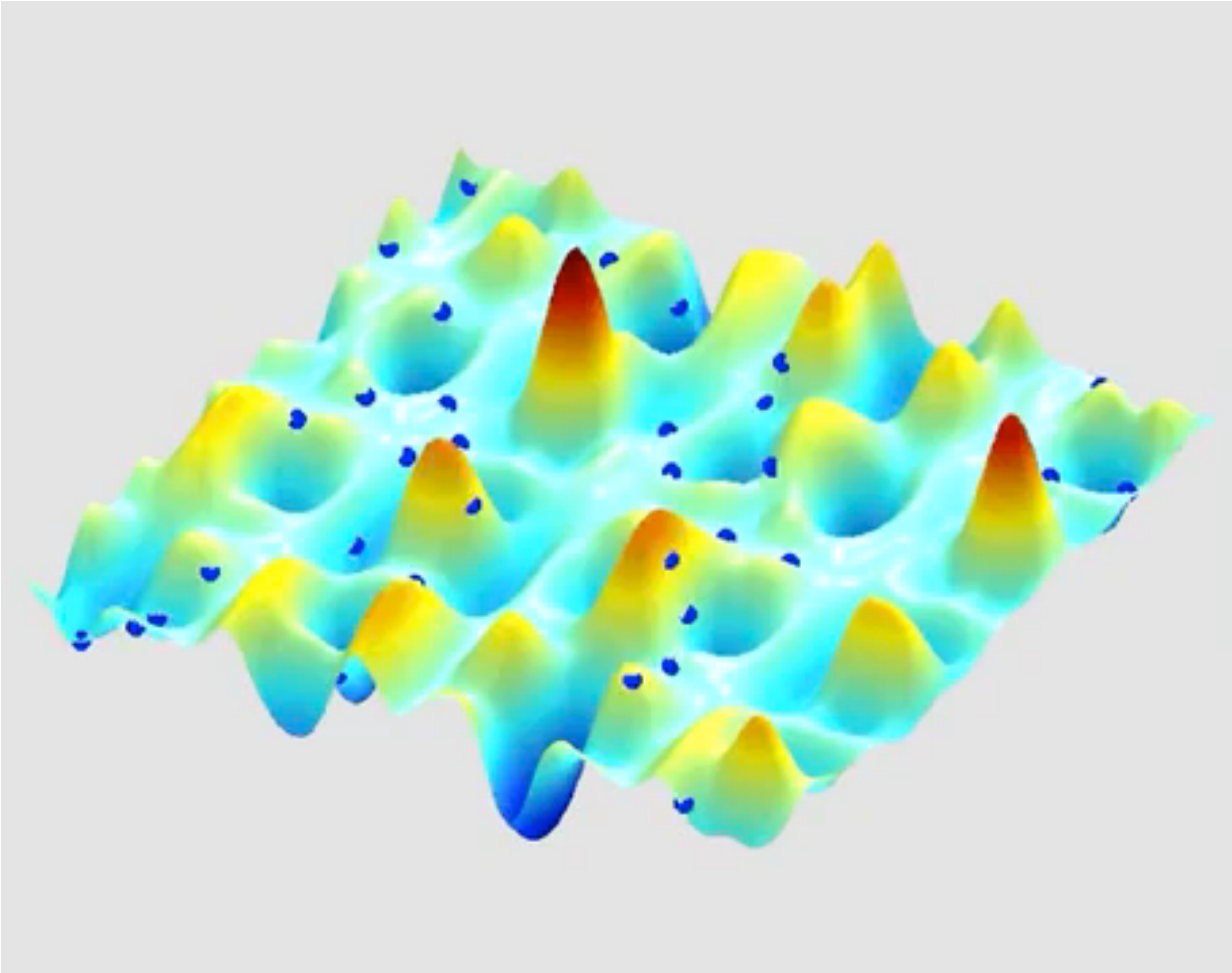
particle swarm optimisation



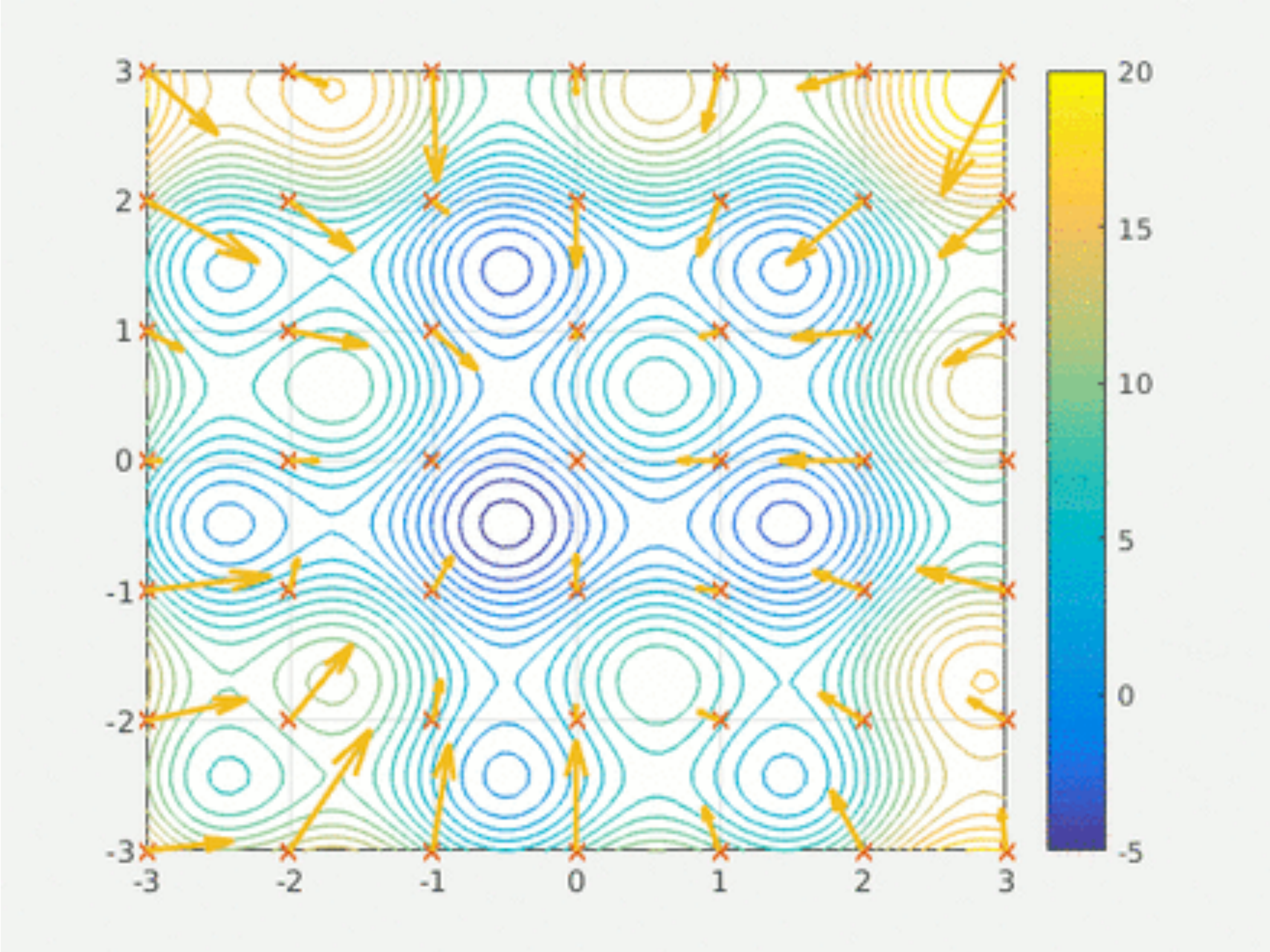
- not just one candidate solution, but a **population** of candidate solutions at each iteration
- each particle's movement based on:
  - it's current position and velocity
  - it's best solution so far (local info)
  - swarm's best solution so far (global info)



- once search is finished, return the best position that any of the particle's found, at any time
- i.e. at each iteration, check the "cost" of each particle position, and check whether it's the lowest cost found so far



https://www.youtube.com/watch?v=VAASmSSsFaY



https://en.wikipedia.org/wiki/Particle_swarm_optimization

http://cs.armstrong.edu/saad/csci8100/pso_tutorial.pdf

Particle Swarm Optimization: A Tutorial

James Blondin

September 4, 2009

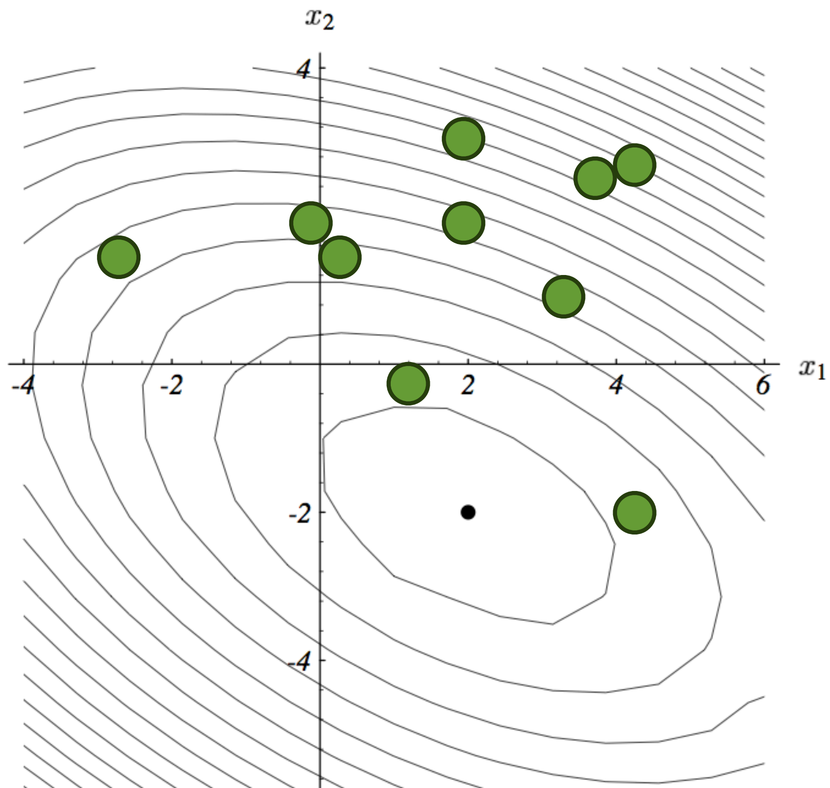
1 Introduction

Particle Swarm Optimization (PSO) is a technique used to explore the search space of a given problem to find the settings or parameters required to maximize a particular objective. This technique, first described by James Kennedy and Russell C. Eberhart in 1995 [1], originates from two separate concepts: the idea of swarm intelligence based off the observation of swarming habits by certain kinds of animals (such as birds and fish); and the field of evolutionary computation.

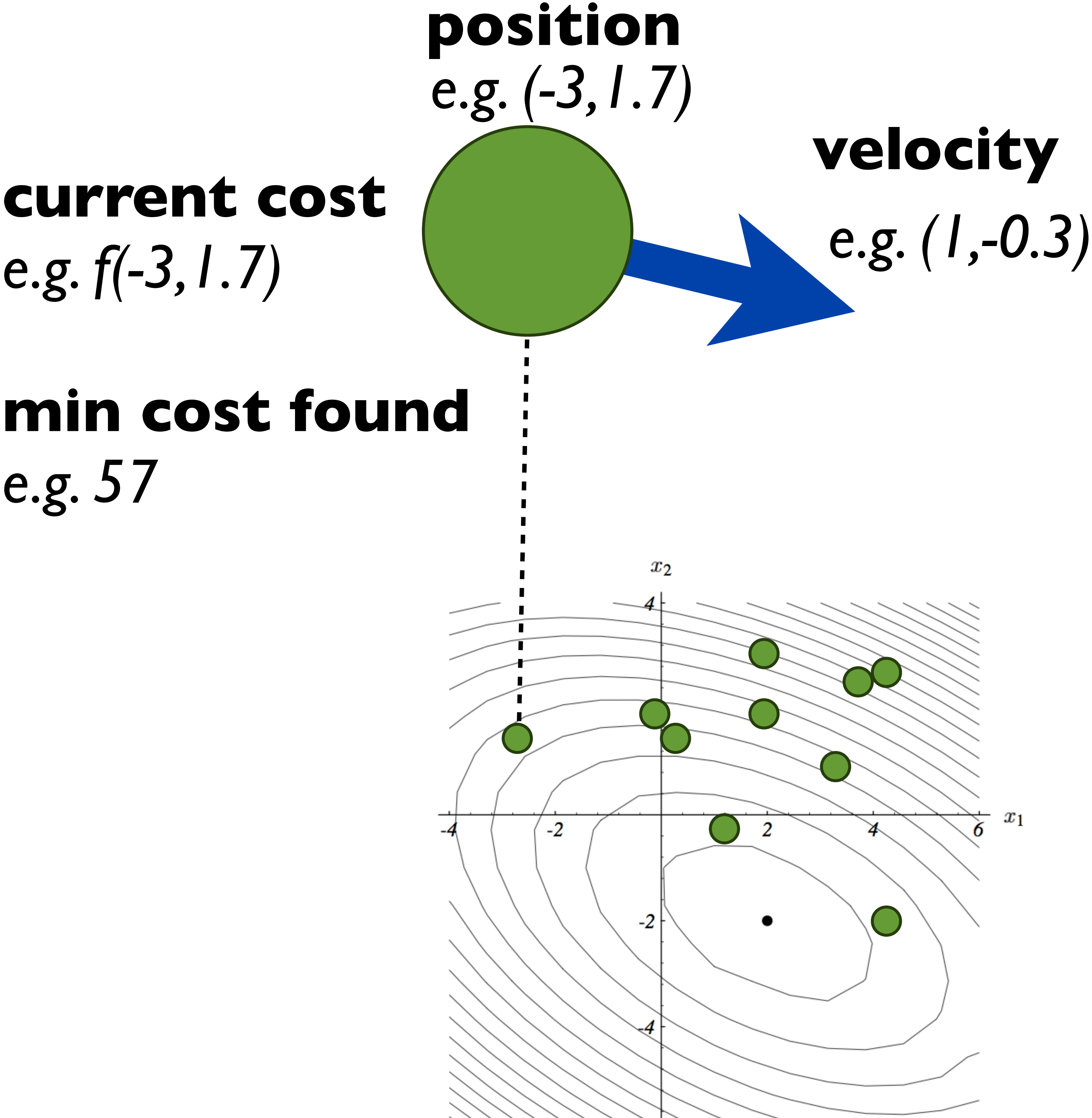
This short tutorial first discusses optimization in general terms, then describes the basics of the particle swarm optimization algorithm.

2 Optimization

Optimization is the mechanism by which one finds the maximum or minimum value of a function or process. This mechanism is used in fields such as physics, chemistry, economics, and engineering where the goal is to maximize efficiency, production, or some other measure. Optimization can refer to either minimiza-



x_2
4
-2
-4
x_1
-4
-2
2
4
6

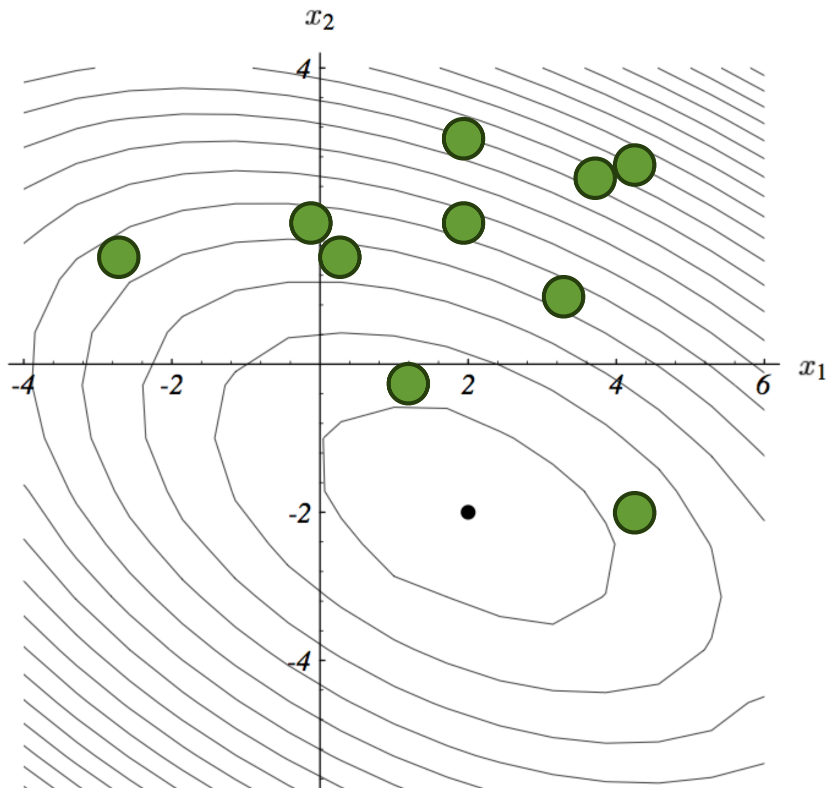


position
e.g. (-3,1.7)
velocity
e.g. (1,-0.3)
current cost
e.g. f(-3,1.7)
min cost found
e.g. 57
x2
x1
4
-2
-4
-4
-2
2
4
6

1. evaluate fitness of each particle

2. update individual and global fitnesses

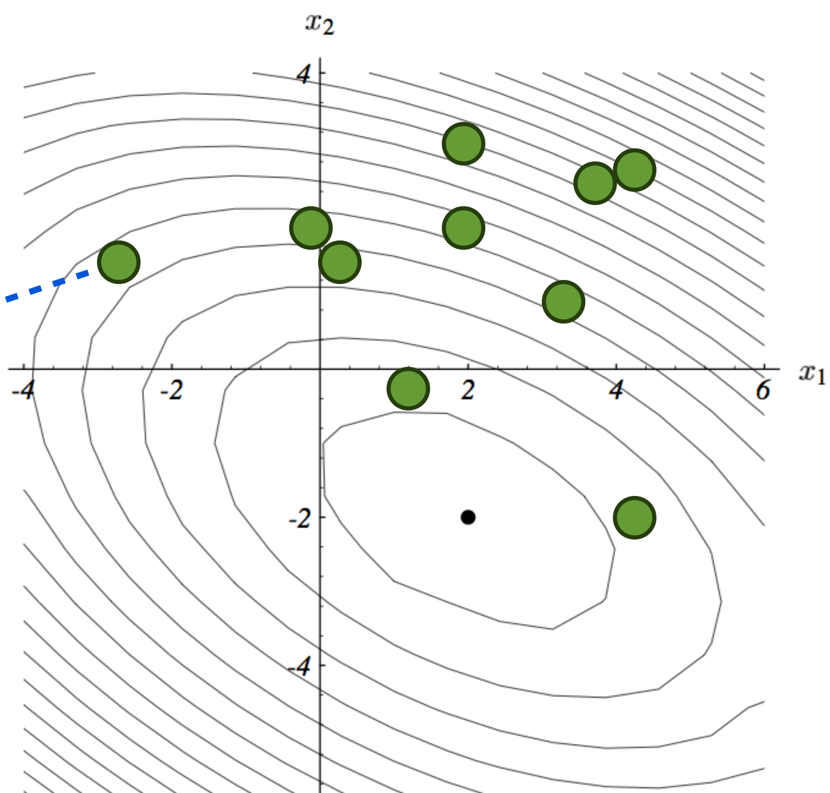
3. update individual velocity and position



1. evaluate fitness of each particle

2. update individual and global fitnesses

3. update individual velocity and position



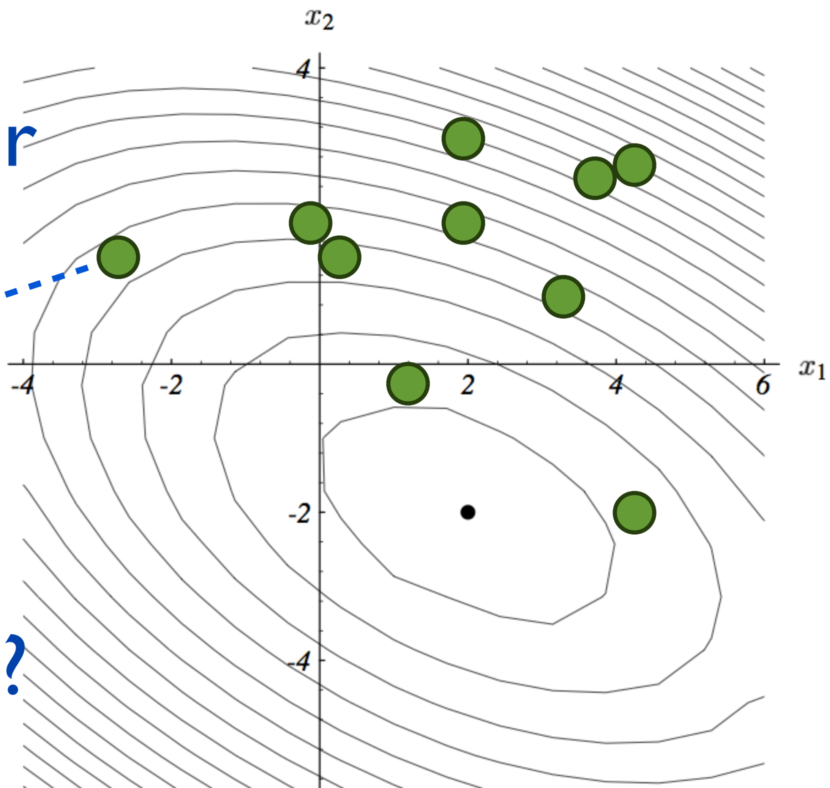
f(-3,1.7) = 42

keep track of each individual's best, and the overall best found so far

2. update individual and global fitnesses

3. update individual velocity and position

min cost so far for this particle?



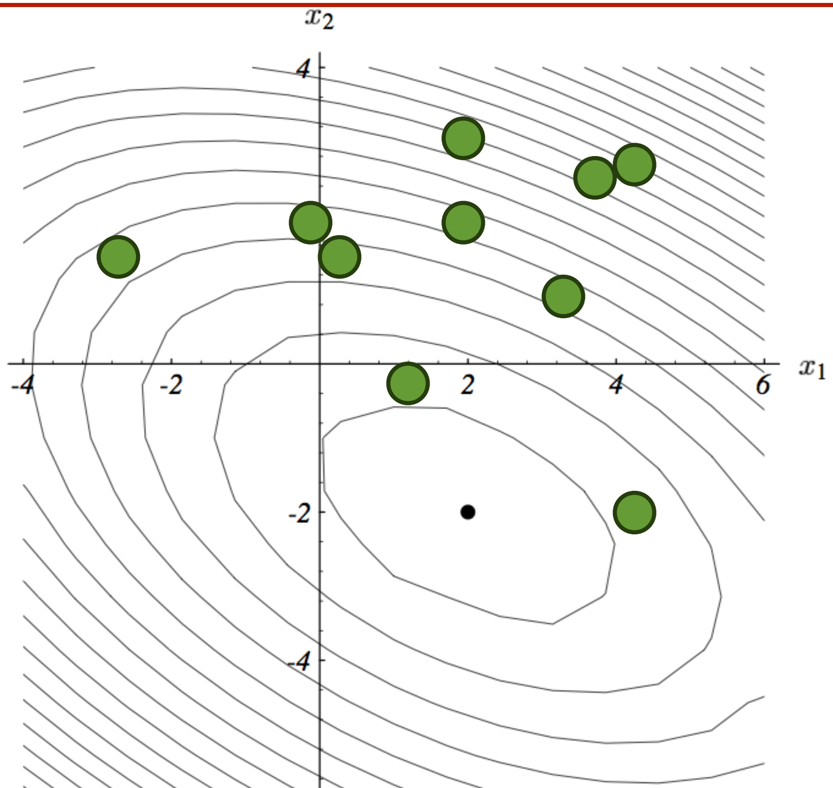
$f(-3, 1.7) = 42$

min cost overall?

1. evaluate fitness of each particle

2. update individual and global fitnesses

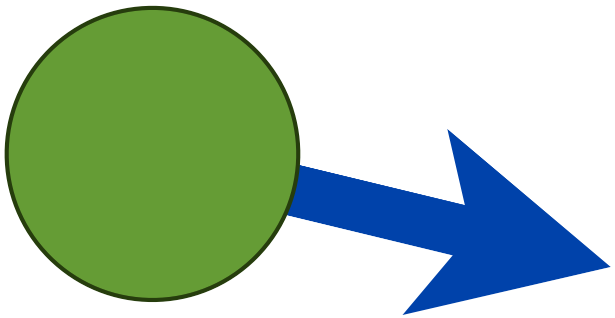
3. update individual velocity and position



update velocity of particle “i”

$$v_i(t+1) = wv_i(t) + c_1 r_1 [\hat{x}_i(t) - x_i(t)] + c_2 r_2 [g(t) - x_i(t)]$$

here the subscript means
“particle i”, not the iteration



update velocity of particle “i”

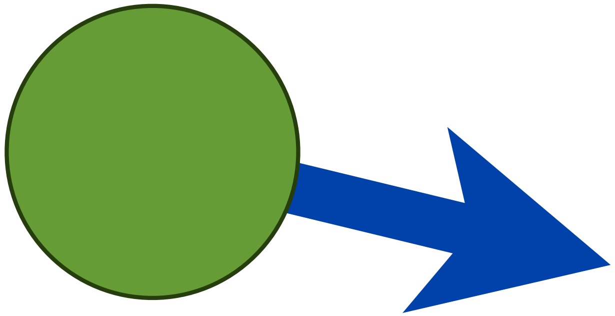
$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

velocity of particle “i” at time *t*

best position of particle “i” up to time *t*

particle “i” position at time *t*

best position from any particle in the swarm at time *t*

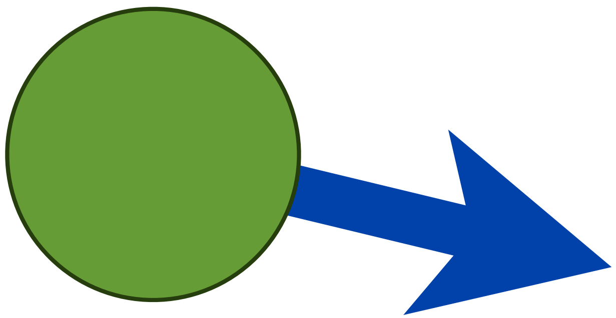


update velocity of particle "i"

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

once we stop the whole search algorithm, this is the value that is returned

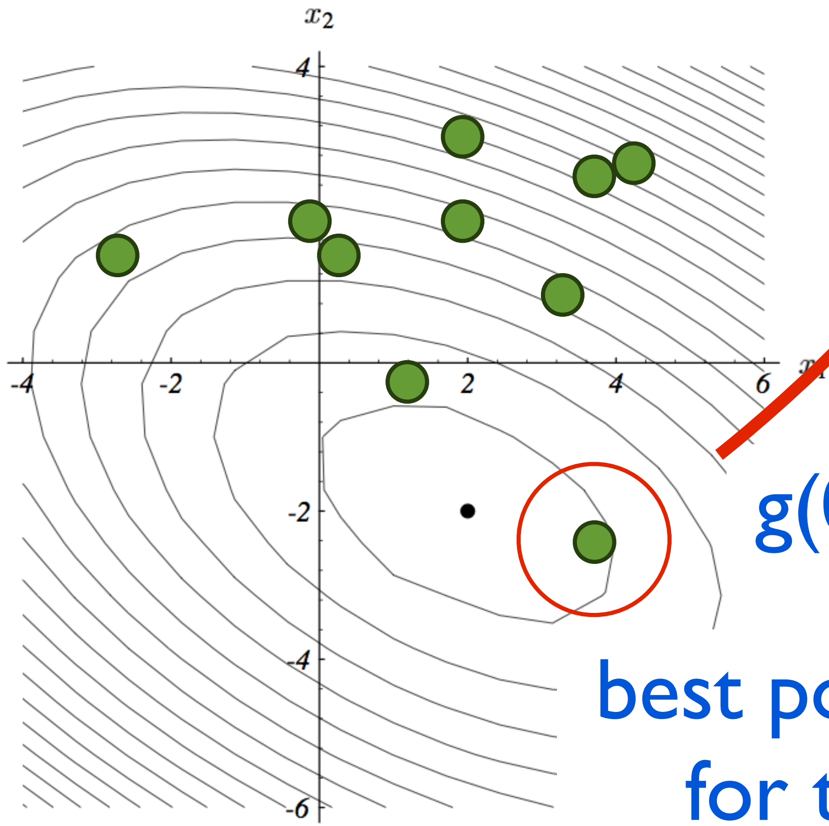
i.e. this is the position (input values for variables in X) that gave the minimum cost during the search (f(X))



update velocity of particle “i”

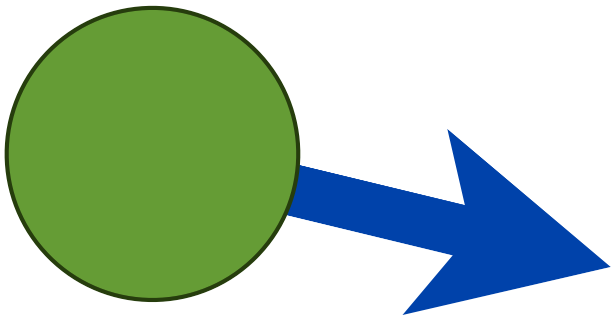
$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

e.g. assume this is the first iteration, t=0



g(0)=(3.7,-2.2)

best position found for the swarm



update velocity of particle "i"

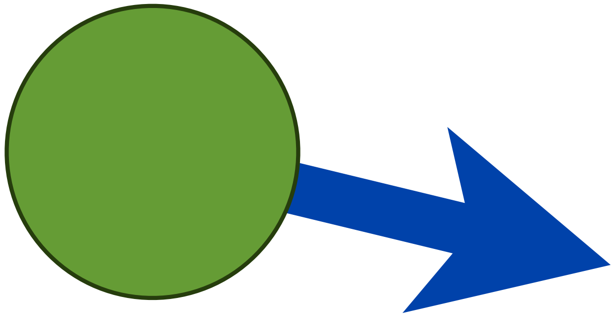
$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

random number

$0 \leq r_2 \leq 1$

random number

$0 \leq r_1 \leq 1$



update velocity of particle “i”

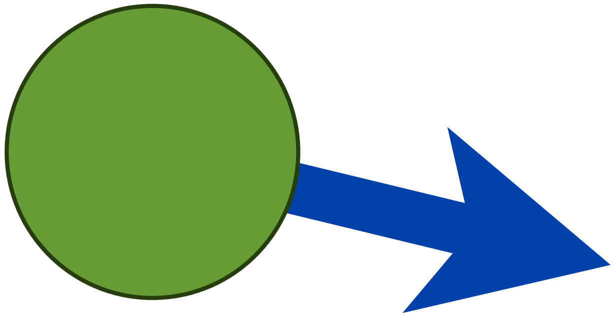
$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

user-supplied coefficients

$0.8 \leq w \leq 1.2$

$0 \leq c_1 \leq 2$

$0 \leq c_2 \leq 2$

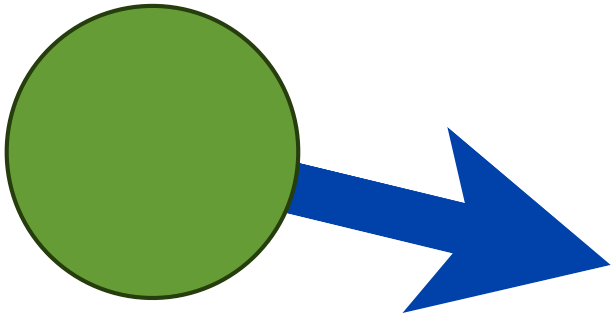


update velocity of particle "i"

$$v_i(t+1) = \underbrace{w v_i(t)} + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

inertia component:
keeps particle moving in similar direction

lower w: speeds up convergence to local optima
higher w: encourages exploration

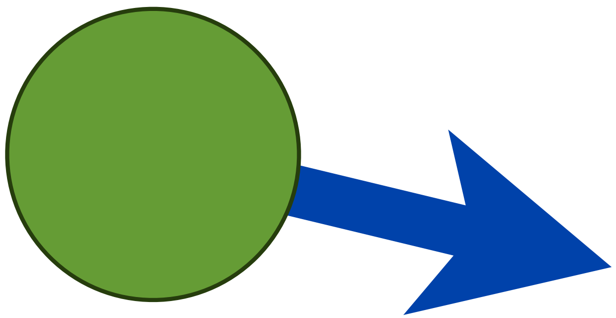


update velocity of particle "i"

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

cognitive component:
particle's memory, encourages particle to
go back to best position

high c_1: take larger step towards best found position

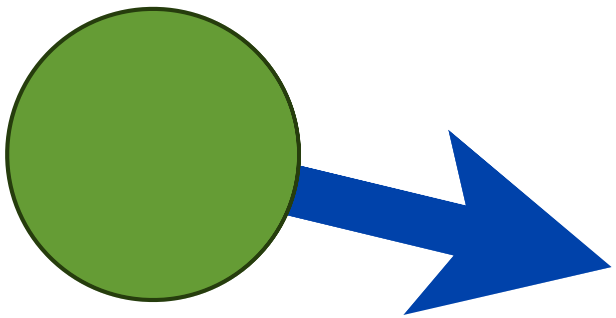


update velocity of particle "i"

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

social component:
encourages particle to move to swarm's
best found position so far

high c_2: take larger step towards swarm best so far

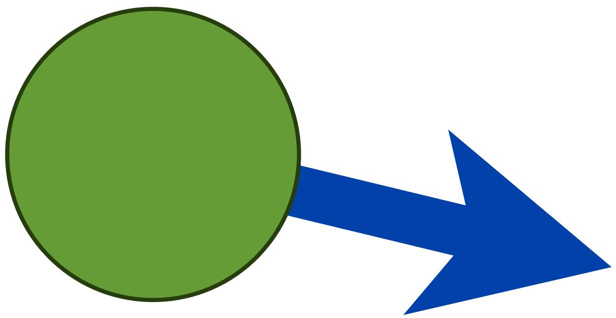


update velocity of particle "i"

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

update position of particle "i"

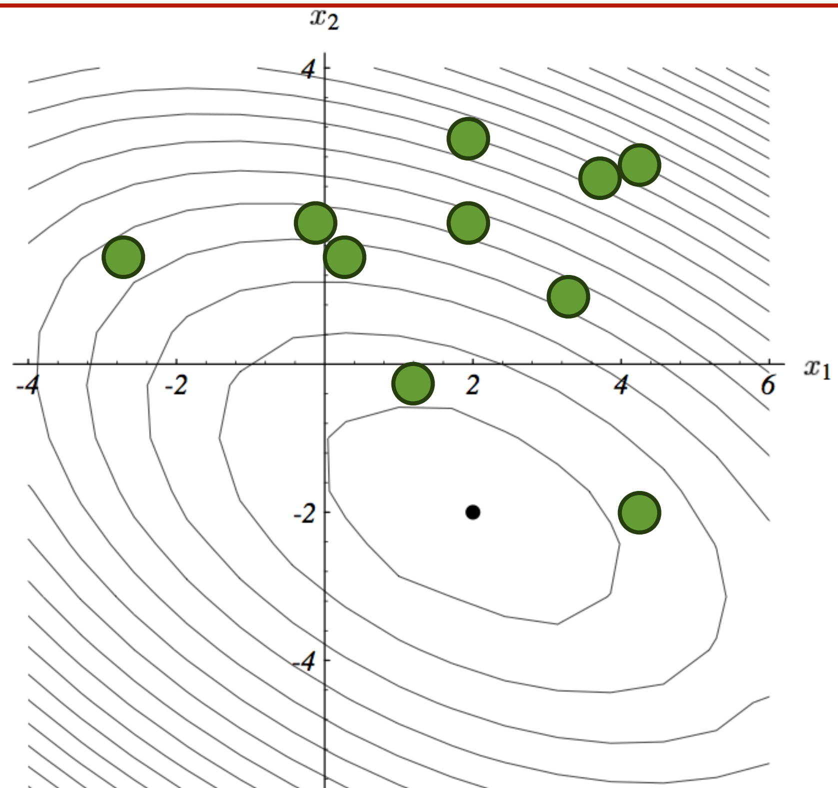
$$x_i(t+1) = x_i(t) + v_i(t+1)$$



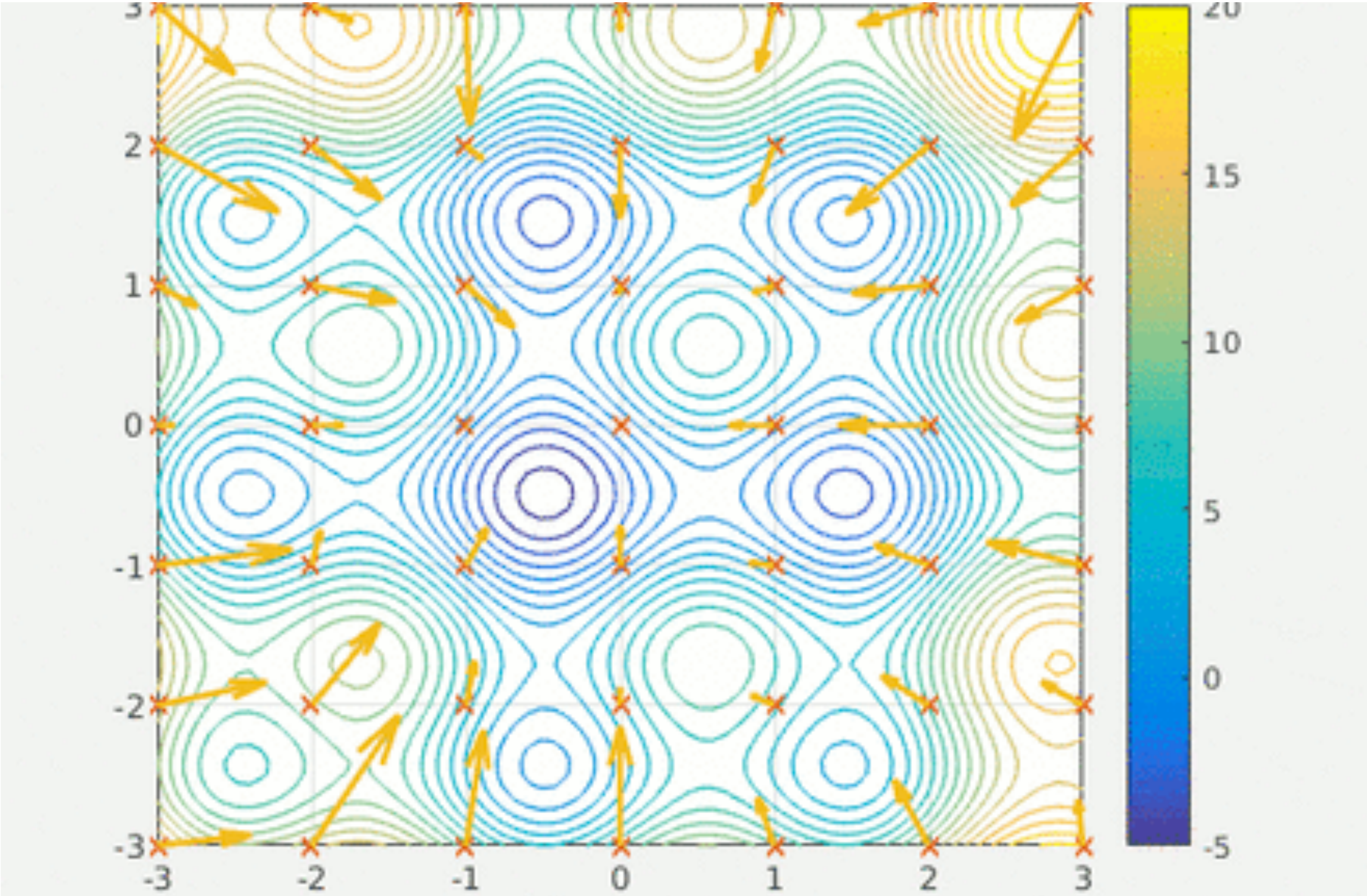
1. evaluate fitness of each particle

2. update individual and global fitnesses

3. update individual velocity and position

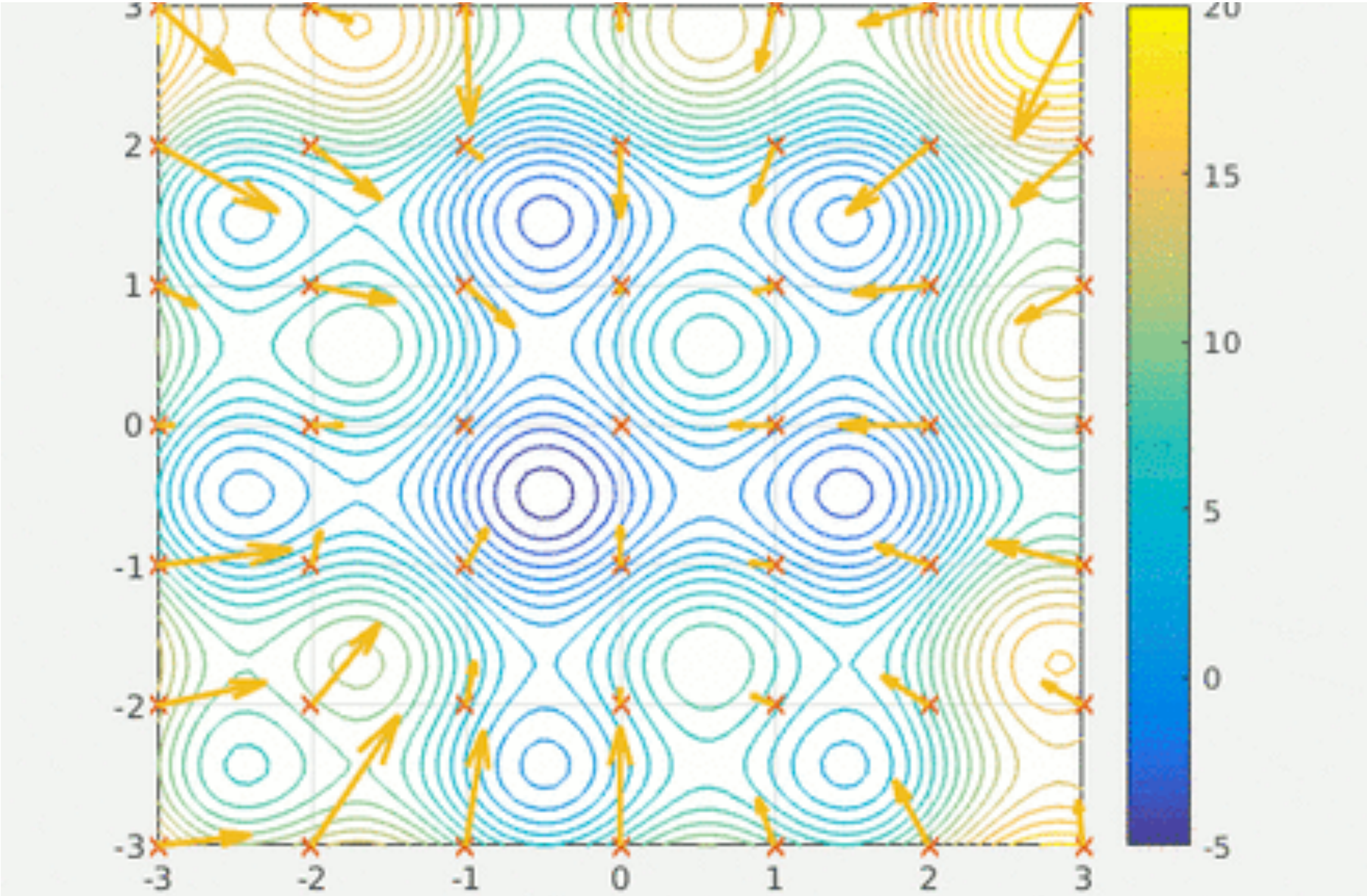


consider this again, thinking about how at each step, each particle velocity is update (inertia, cognitive, social)



https://en.wikipedia.org/wiki/Particle_swarm_optimization

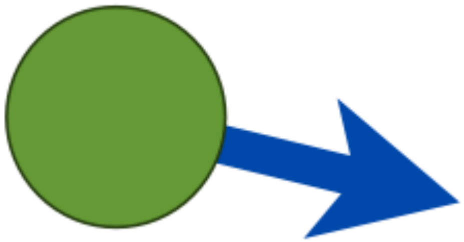
swarm manages to find the global min (i.e. at least one particle went to this position at least once)

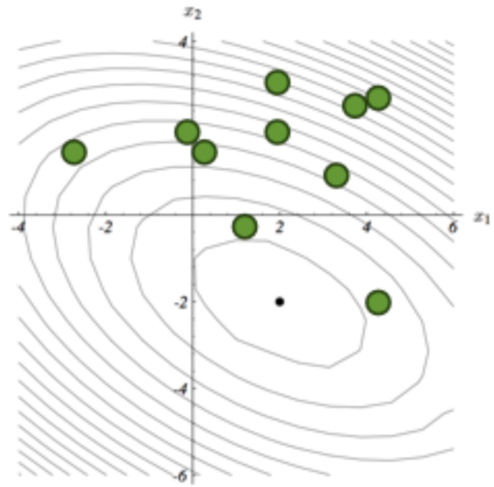


https://en.wikipedia.org/wiki/Particle_swarm_optimization

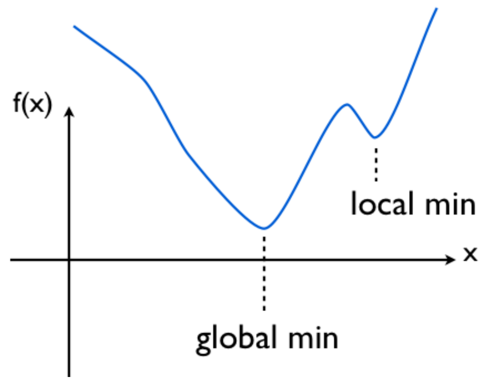
SUMMARY Part 3. particle swarm

- population of candidate solutions
- particle has position and velocity
- velocity update components: inertia, cognitive, social





TODAY'S SUMMARY



1. preliminaries
2. simulated annealing
3. particle swarm optimisation