Chapter4: Local Search

feh masa2el ana myhmnesh enta wslt lel 7l da ezay, ana kol el yhmny enk wslt ll 7l.

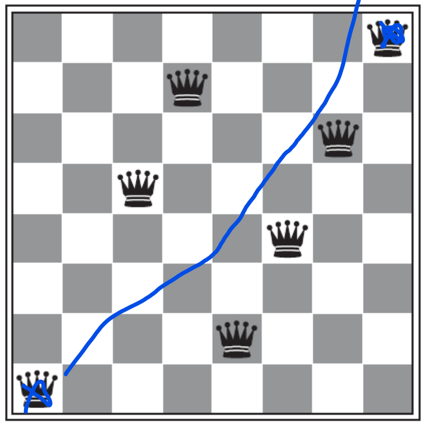
zy el 8-Queens problem, myhmnesh wslt lel rasa ezay, el muhem enk rasthom w khlas.

Local Search

- Local search and optimization problems
- Local search algorithms
  - Hill climbing
  - Simulated annealing
  - Local beam search
  - Genetic algorithms

Local Search

- The search algorithms studied so far keep one or more paths in memory and record which alternatives have been explored at each point along the path.
- When a goal is found, the *path* to that goal also constitutes a *solution* to the problem.
- However, for many problems all that matters is the goal state itself, not the path to reach it.
- For example, for the n-queen problem what matters is the final configuration of queens, not the order in which they are added.



his is not a valid solution btw... :D

Local Search

- Local search algorithms are suitable for problems where the path to the goal does not matter.
- **Local search** algorithms operate using a single **current node** (rather than multiple paths) and generally move only to the neighbors of that node.
- The paths followed by the search are not retained. msh motar eny a7fz el path elly m4et 3leh.

- Local search algorithms are useful for solving pure **optimization problems** (find the best state according to an **objective function)**.
- On the other hand, maximizing/minimizing an objective function cannot fit using Chapter 3 search algorithms as there is no goal test.

read more about genatic programming .

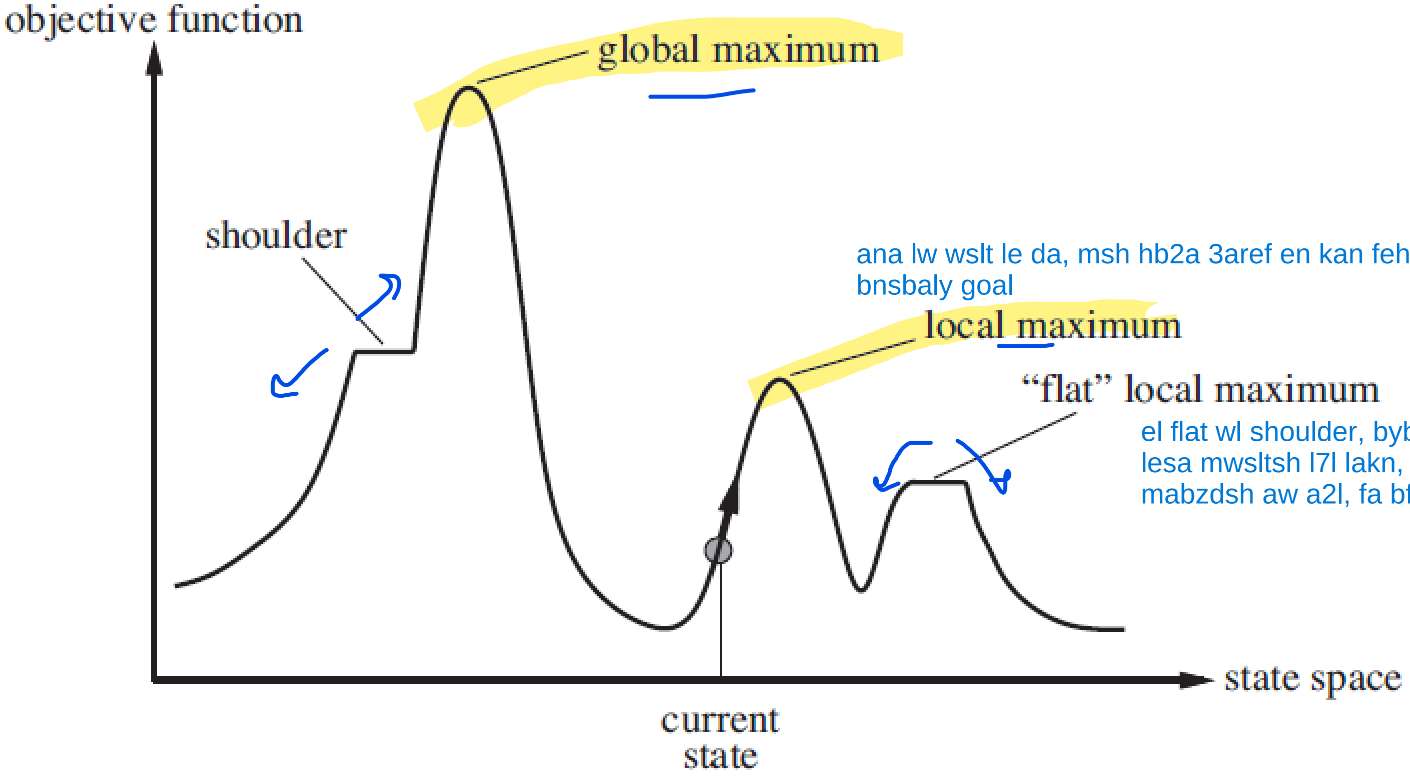
Advantages of Local Search Algorithms

- They use very little memory—usually a constant amount
- They can often find reasonable solutions in large or infinite (continuous) state spaces for which systematic algorithms are unsuitable.

el fr2 ben el flat wl shoulder
flat: lw ra7 ymen aw shmal, el etnen bygebo 7agat a2al meno

el shoulder: feh etgah momken ywdek le 7aga a7sn, w feh etgah tany momken ywdek l 7aga asw2.

State Space Landscape



ana lw wslt le da, msh hb2a 3aref en kan feh haga a7sn meny, lakn ana ha2f hena btw 34an da bnsbaly goal

el flat wl shoulder, byb2o mshakl moz3ga gedan blnesballna 34an ana bab2a lesa mwsltsh l7l lakn, 34an hwa flat, fa ana bgrb ymene w shmaly, bala2y eny mabzdsh aw a2l, fa bftrd eny khalas wslt l7l,, lakn de msh 72e2a, fa bysw7ny.

1. Hill-climbing Search

aghba greedy solution momkrn, bs a7yanan bytl3 omash.

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

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of current
    if neighbor.VALUE ≤ current.VALUE then return current.STATE
    current ← neighbor
```

hnlf 3la kol el neighbours bto3na, lw 7d fehom a3la meny haro7lo, lakn lw kolohom zayey, fa sa3tha h5rog w arg3 el kema bt3ty w khlas.

Figure 4.2 The hill-climbing search algorithm, which is the most basic local search technique. At each step the current node is replaced by the best neighbor; in this version, that means the neighbor with the highest VALUE, but if a heuristic cost estimate h is used, we would find the neighbor with the lowest h.

34an el hurestic by2olak enta b3ed ad a 3n el goal, fa a7sn wa7d hwa sa7b a2al heuristic.

moshkelto el flat & shoulders, w el local minima, la2no by-stuck. da zy el gradient descient shwya.

Hill-climbing Search

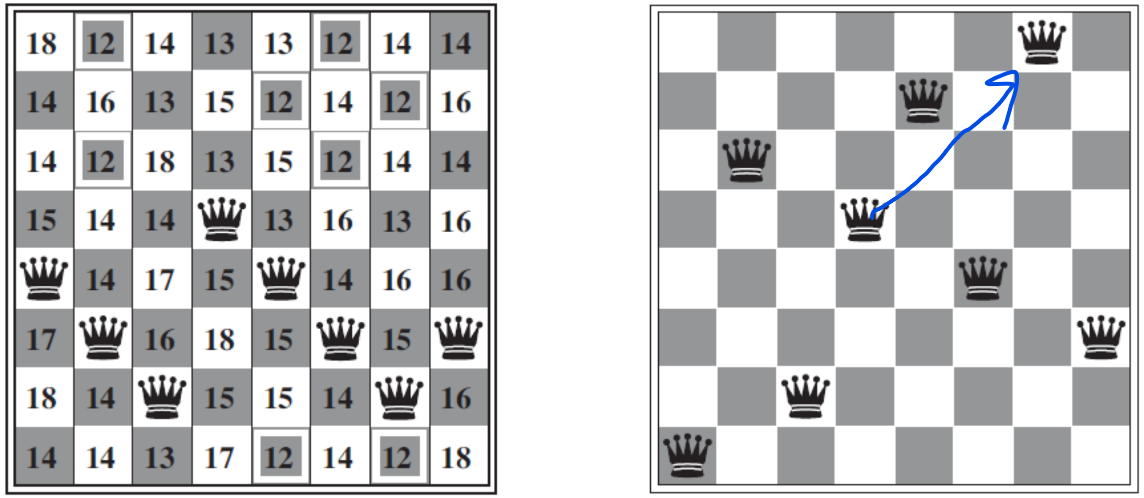
- It continually moves in the direction of **increasing value** until it reaches a "peak" where no **neighbor** has a higher value.

- The algorithm does not maintain a search tree, for the current node it only saves the state and the value of the objective function.

- Hill climbing is sometimes called **greedy local search** because it grabs a good neighbor state without thinking ahead about where to go next.

Example: n-queen problem

- The heuristic cost function h is the number of pairs of queens that are attacking each other, either directly or indirectly.



(a) h=17 (b) h=1

Hill-climbing Search

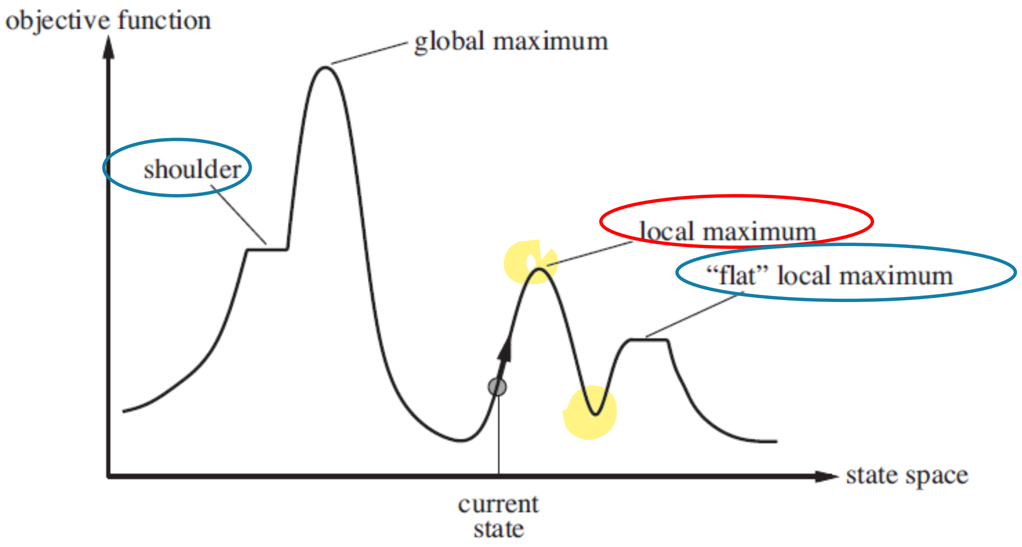
- Hill climbing gets stuck due to:

– Local maxima/minima

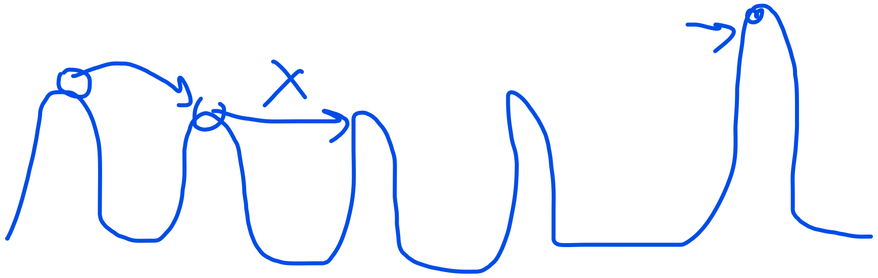
A peak that is higher than each of its neighboring states, but lower than the global maximum.

– Plateau (shoulder or flat local maxima/minima)

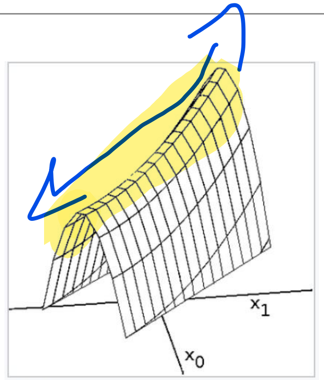
A plateau is a flat area of the state-space landscape.



Hill-climbing Search



Ridges



A ridge

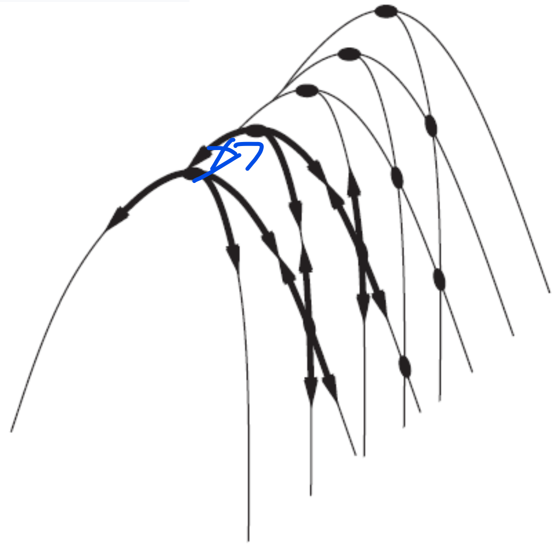
- Ridges result in a sequence of local maxima

that is very difficult for greedy algorithms to navigate.

- The grid of states is a sequence of local maxima that are not directly

connected to each other.

- From each local maximum, all the available actions point downhill.



t5yl en m3ak graph, w enta wa2f 3nd node, kol el neighbours bto3k a2al mnk, lakn lw ro7t l wa7d menhom, sa3tha te2dr tkml tro7 le wa7d a3la w b3dha ywslk lel tre2 el s7, sa3tha el situation da bnsmeh ridge, tb eh el fr2 keda beno w bet en flat?

Hill climbing search-example

- Starting from a randomly generated 8-queens state, hill climbing gets stuck 86% of the time, solving only 14% of problem instances.

- It works quickly, taking just 4 steps on average when it succeeds and 3 when it gets stuck.

- These results are not bad for a state space with $8^8 \approx 17$ million states.

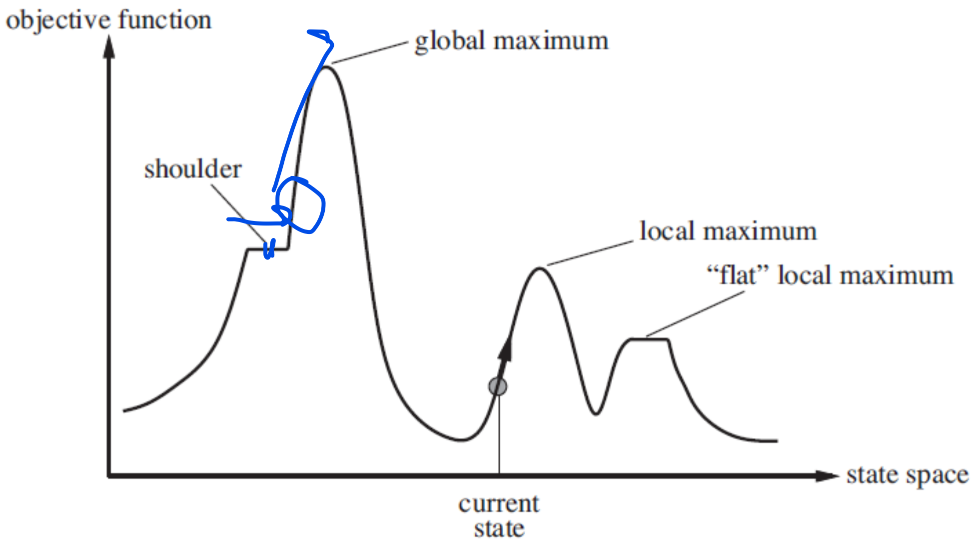
hena el bn3mlo, en badal ma nw2f b3d awl khatwa, la2, e7na ndelo forsa y3ml expansion shwya kman fa ygrb kaza neighbour, bs bshrt enk lazm t7otlo limit yo2aaf b3do.

Hill Climbing Search

- May allow sideway move, this could help if plateau is really a shoulder.

- But what if there is no uphill movements?
  - Put a limit on the number of consecutive sideways moves allowed.

- This raises the percentage of n-queen problem instances solved by hill climbing from 14% to 94%.



- Success comes at a cost: After allowing sideway moves, the algorithm has on average roughly 21 steps for each successful instance and 64 for each failure.

Variants of Hill Climbing Search

not complete

- Stochastic hill climbing

hena enta dwrt fl 7walek, l2et shwya aw7sh w shwya a7sn, el a7sn dol enta msh hta5ud a7sn wa7d w tkml m3ah, laa enta ht5tar menhom in random, w elly hytl3 m3ak, htemshy wrah, 34an keda hwa stochastic.

– chooses at random from among uphill moves

3 helwen, 3 we7shen, khud wahd mn el 3 el helween 34wa2y.

– converges more slowly, but finds better solutions in some landscapes

not complete

- First-choice hill climbing

bnmshy 34wa2y, awl mnla2y wahed kwys, emshy 3leh.

– generate successors randomly until one is better than the current

hena msh hngeb el 6 kolhom, laa enta msh ht5zn el 6, la enta bt-generate time-fly ashuf el node, we7sha ermeha khush 3la el b3dha, helwa, emshy 3leha.

el fr2 benhaa w ben el fo2ha hwa el implementation bs.

– good when a state has many successors

- Random-restart hill climbing

complete algo, lama te3tl, e5tar node gdeda random w ebd2 mn 3ndha, el probability bt3tk btzeed awy. w complete b3tbar enk hatedelo w2t atwal shwya ydwr.

– conducts a series of hill climbing searches from randomly generated initial states, stops when a goal is found

– It’s complete with probability approaching 1 edelo option eno yekhtar 7aga asw2 mn el enta wa2f 3ndha, yemkn tfedak fe um mn el ayam.

2 Simulated annealing

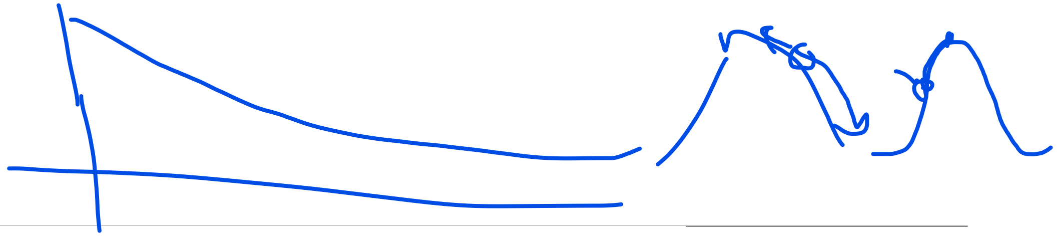
- A hill-climbing algorithm that *never* makes “downhill” moves toward states with lower value (or higher cost) is incomplete, because it can get stuck on a local maximum.

- In contrast, a purely random walk that is, moving to a successor chosen uniformly at random is complete but extremely inefficient.

What about combining both completeness and efficiency?

Simulated annealing algorithm combines both.

Simulated annealing



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** ∞ **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 $e^{\Delta E/T}$

time stamp

bybd2 b very high temp, w gradually ben2ll el temp 34an n2ll forset enna nakhud bad walk

T is temp.

T=0 -> Stop

Picks a random move

Always accept the move if is better.

delta E > 0 m3naha en el next a7sn meny.

this else makes it annealing, fa enta @ T = inf e = 1.

Figure 4.5 The simulated annealing algorithm, a version of stochastic hill climbing where some downhill moves are allowed. Downhill moves are accepted readily early in the annealing schedule and then less often as time goes on. The *schedule* input determines the value of the temperature T as a function of time.

Simulated annealing

- **Annealing** is the process used to temper or harden metals and glass by heating them to a

high temperature and then gradually cooling them, thus allowing the material to reach a low energy crystalline state.

- The idea of simulated annealing is to escape local maxima by allowing some “bad" moves but

gradually decrease their size and frequency.

- If the move improves the situation, it is always accepted. Otherwise, the algorithm accepts the move with
- probability $e^{\frac{\Delta E}{T}}$.
- The probability decreases exponentially with the “badness” of the move (ΔE).
- The probability also decreases as the “temperature” T goes down. Why?

  “Bad” moves are more likely to be allowed at the start when T is high, and they become more unlikely as T decreases.

Simulated Annealing

- If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1.

- Widely used in VLSI layout, airline scheduling, etc.

el by7sl kal taly
1. hat k random states fl awl
2. hat kol el neighborhood bto3hom 7otohom fe list.
3. e3ml sort.
4. e5tar best k, w e3ml recursive call.
base case:
awl ma tewsl le goal -> stop.

3. Local beam search

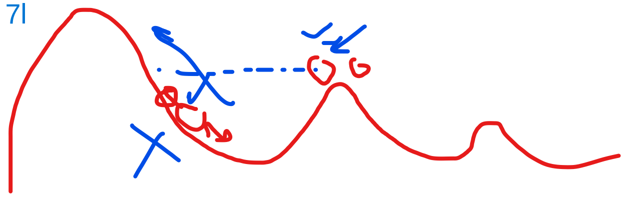
ebd2 mn kaza wa7da fe nafs el w2t.

- Keep track of k states rather than just one
  - Start with k randomly generated states.
  - At each iteration, all the successors of all k states are generated.
  - If any one is a goal state, stop; else select the k best successors from the complete list and repeat.

- Is it the same as running k random-restart searches?
  - No, in a random-restart search, each search process runs independently of the others.
  - In contrast, in local beam search, useful information is passed among the k parallel search threads.

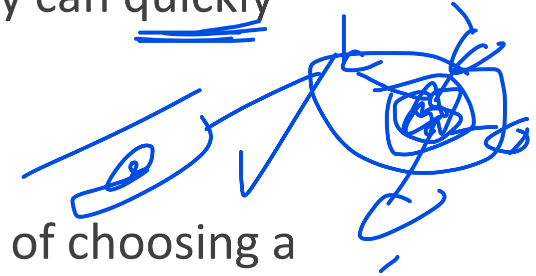
here we use the benfit of the multi-threding.

3ebo baa, en hena msln enta lw mshet m3 elly 3la el shmal dol, hywslok le 7al a7sn, lakn hwa hena lama yemshy bl tre2a el lesa aylen 3leha, htla2y eno hyshof en el 3l ymenk homa elly a7sn fa hayrmy el 3la el shmal w msh hywsl le a7sn 7l

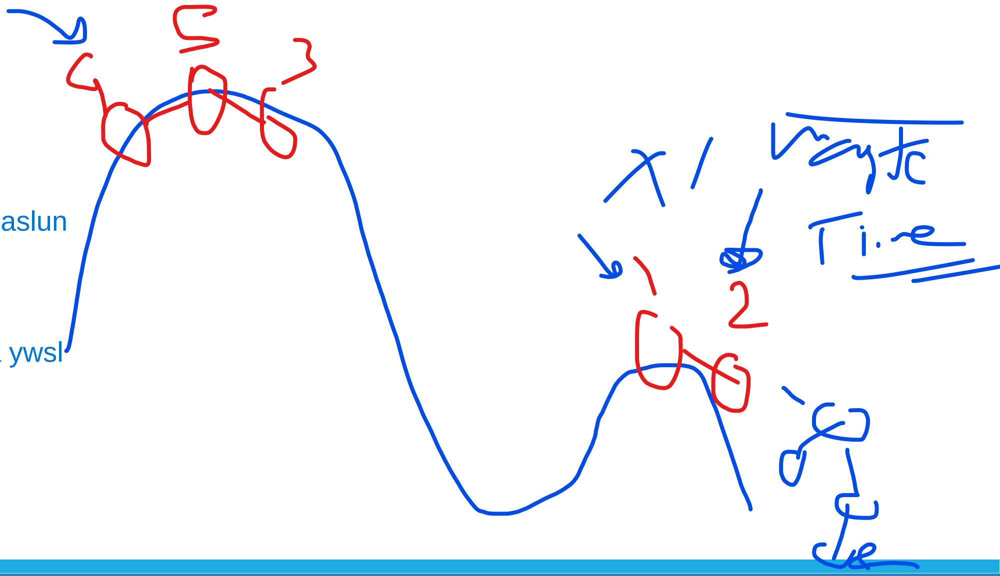


Local beam search

- Local beam search can suffer from a lack of diversity among the k state as they can quickly become concentrated in a small region of the state space.



- **Stochastic beam search** chooses k successors at random, with the probability of choosing a given successor being an increasing function of its value.



3auzak tefhm el fr2 ben da w ben el random restart:

local beam, delw2ty hwa nftrd fl awl ekhtar 2 nodes random el homa hena 2 w 4 msln.
el 5twa el b3dha hy7ot kol el neighbours fl array w y3ml sort, hyla2y en el 5 a7sn bkter awy mn el 1 wl 2, fa msh hybos aslun 3la el 1 wl 2 tany w hayrkz fl na7ya el fo2 de baa.

el random restart baa la2, hwa lw msln bd2 el awl mn 3nd el 2 msln, hwa hena 34an el 3adad olayl fmomken mtb2ash wad7a awy, lakn lw 5lenaha rakm akbur shwya zy msln 10 nodes fl na7ya el ymen de, hwa hygrb el 10 kolohom l7d ma ywsl 3nd el 1, w b3d keda yla2y nfso 3etlt, fa yro7 ygrb random node gededa, w yebd2 mn 3nd el 3 msln.

fa keda tab3n el local beam bywfrlk w2t kter awy enk matebzlsh maghood m3 nodes malhash lazma.

hna5ud two states, w n2t3hom bl nos, na5ud nos mn el state el oula, w nos mn el state el tanya, w nekml shughl.

fa el 7elw eno by3ml mixing. fa da bykhleh a7sn kter.

4 Genetic algorithms

- A **genetic algorithm** (or **GA**) is a variant of stochastic beam search in which successor states are generated by combining *two* parent states rather than by modifying a single state.
- Like beam searches, GAs begin with a set of k randomly generated states, called the **population**.
- A state is represented as a string over a finite alphabet (often a string of 0s and 1s or digits).
- Evaluation function (fitness function) has higher values for better states.
- The probability of being chosen for reproducing is directly proportional to the fitness score.
- Produce the next generation of states by selection, crossover, and mutation.

Genetic algorithms: n-queens example

Crossover points

this is a single node

choosing the parts to be crossed

crossing them over.

ghyr node random keda.

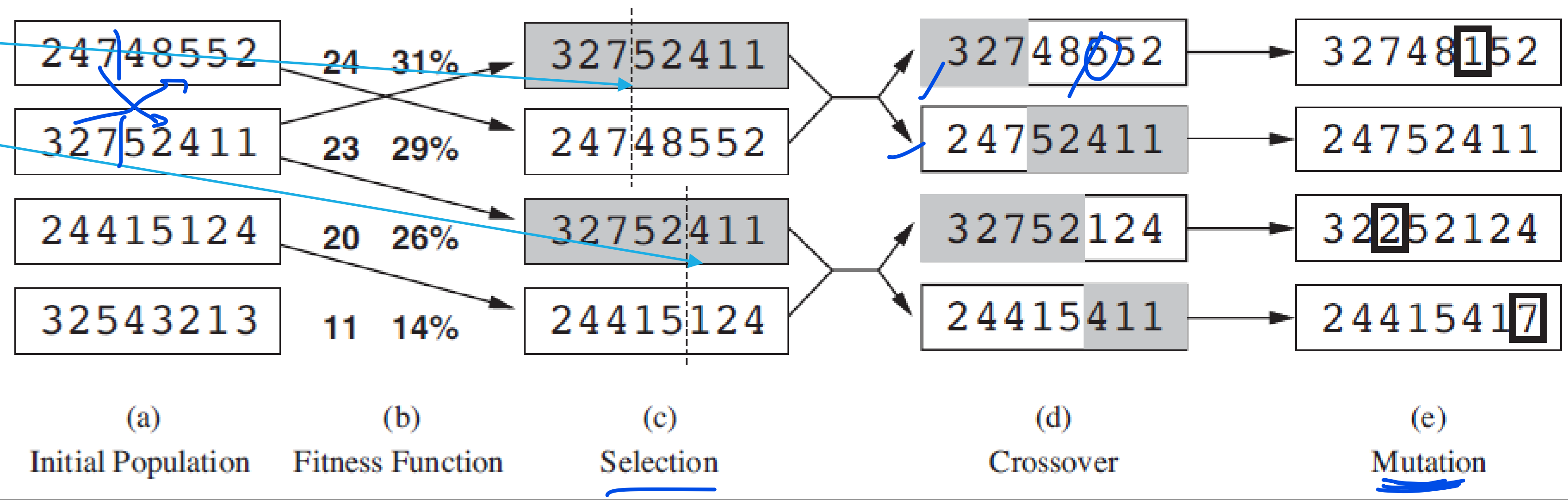


Figure 4.6 The genetic algorithm, illustrated for digit strings representing 8-queens states. The initial population in (a) is ranked by the fitness function in (b), resulting in pairs for mating in (c). They produce offspring in (d), which are subject to mutation in (e).

el mutation by7l moshkelt el lack of diversity.

byt3ml b probability olyla.

Genetic algorithms

function GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  inputs: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

fe tre2a keda esmha metshing

byshof area keda, men a7sn node fehom, w y2tl kol el nodes elly 7wleh 34an yemn3 ay 7d fehom ytm ekhtyaro.

  repeat
    new_population ← empty set
    for i = 1 **to** SIZE(*population*) **do**
      x ← RANDOM-SELECTION(*population*, FITNESS-FN)
      y ← RANDOM-SELECTION(*population*, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    population ← *new_population*
  until some individual is fit enough, or enough time has elapsed
  return the best individual in *population*, according to FITNESS-FN

el probability bona2n 3la el fitness.

e5tar 2 mn el population elly fatet bshkl random.

el node elly bn5trha, msh bnshelha mn el population fmomken nakhudha m3 kaza node.

fe condition by-5rgk.

function REPRODUCE(x, y) **returns** an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING($x, 1, c$), SUBSTRING($y, c + 1, n$))

Figure 4.8 A genetic algorithm. The algorithm is the same as the one diagrammed in Figure 4.6, with one variation: in this more popular version, each mating of two parents produces only one offspring, not two.

Genetic Algorithms

- Like stochastic beam search, genetic algorithms combine an uphill tendency with random exploration.

- The population is generally quite diverse early on in the process, so crossover (like simulated annealing) frequently takes large steps in the state space early in the search process and smaller steps later on when most individuals are quite similar.

- Genetic algorithms are used in optimization problems such as circuit layout and job-shop scheduling.

DONE

Section 4.1 from Chapter 4 of the textbook.