Mesterséges Intelligencia MI

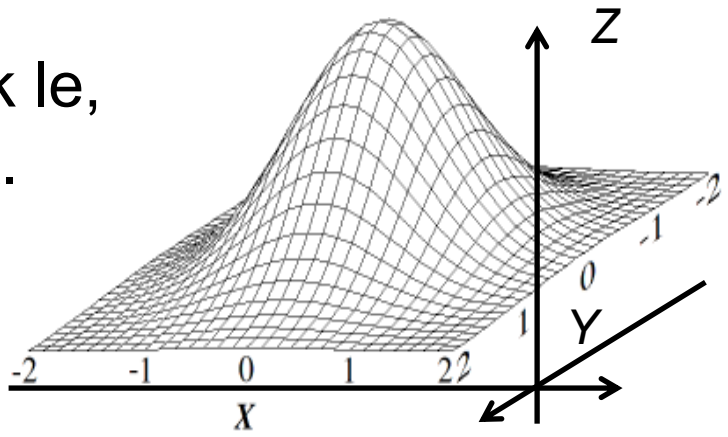
Problémamegoldás kereséssel - csak lokális információra alapozva

Pataki Béla

BME I.E. 414, 463-26-79
pataki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/pataki

Lokálisan kereső algoritmusok

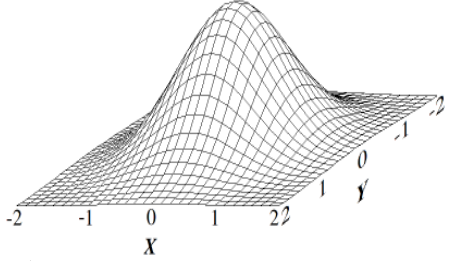
Ha a probléma állapotát *N* változóval írjuk le, akkor alkotunk egy (*N*+1)-dimenziós teret. *N* tengelyen egy-egy változó értékeit mérjük fel, az *N*+1-diken az állapotok jóságát vagy „rosszaságát”.



(Ábra: *X* és *Y* változók írják le a problémát, *Z* az adott X-Y-hoz tartozó jóság).

Állapotok jósága: egy hiperfelület az állapotok *N*-dim. paramétertérében. A felület minden pontjának a „magassága” = az adott állapot optimalitása, jósága.

Lokálisan kereső algoritmusok



Hegymászó algoritmus:

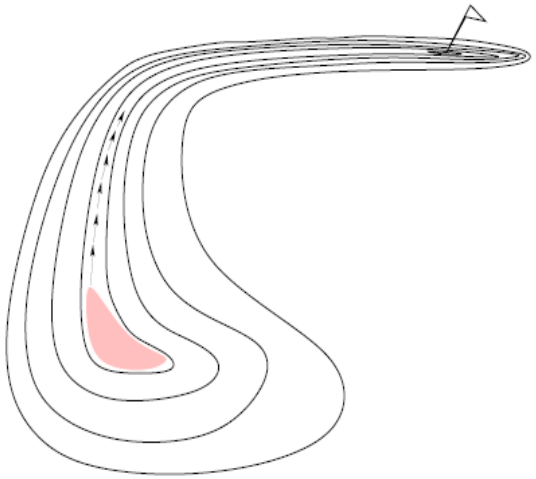
mindig olyan változtatás, ami javít az aktuális állapoton

A felületen mozogva a legmagasabb pontot keressük, ami egyben az legjobb megoldásnak felel meg. A csúcs helyén a paraméterértékek adják a probléma megoldását.

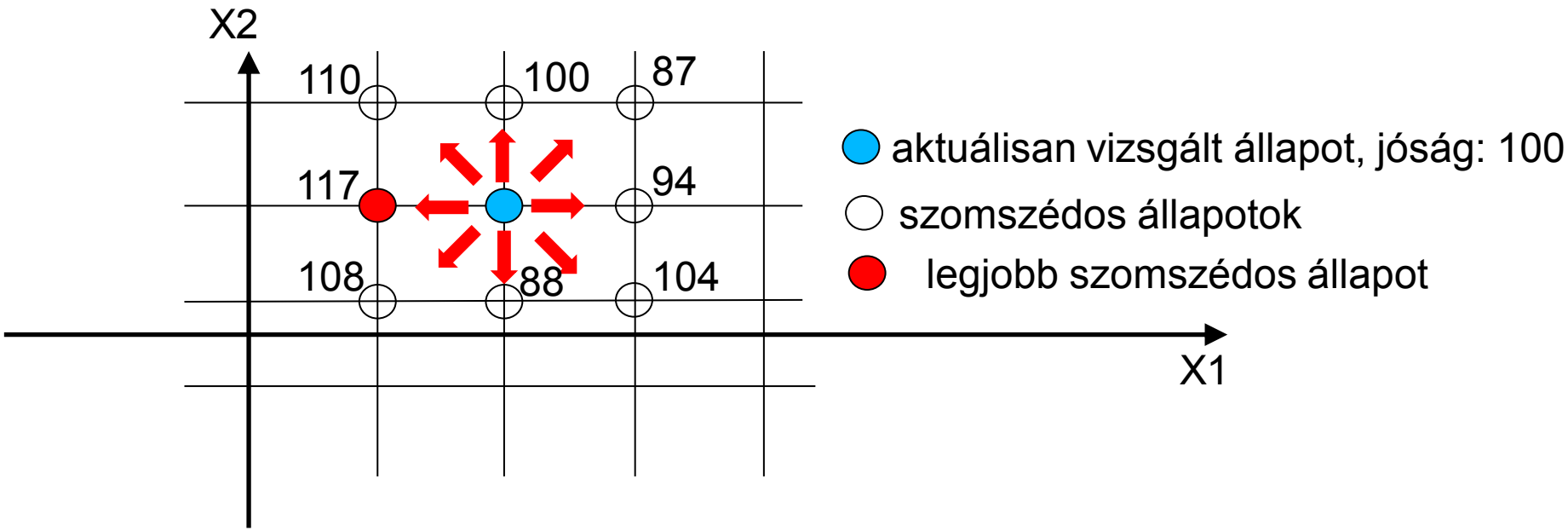
Általában csak az aktuális állapotot tartjuk nyilván, és csak a közvetlen szomszédságában nézünk előre.

- **Visszalépés nincs, emiatt a memóriaigény kicsi, fix.**
- **Az időigény lineáris – az optimumtól való távolságunktól függ.**
- **Garancia a megoldásra? (teljesség?, optimalitás?)**

Hegymászó keresés



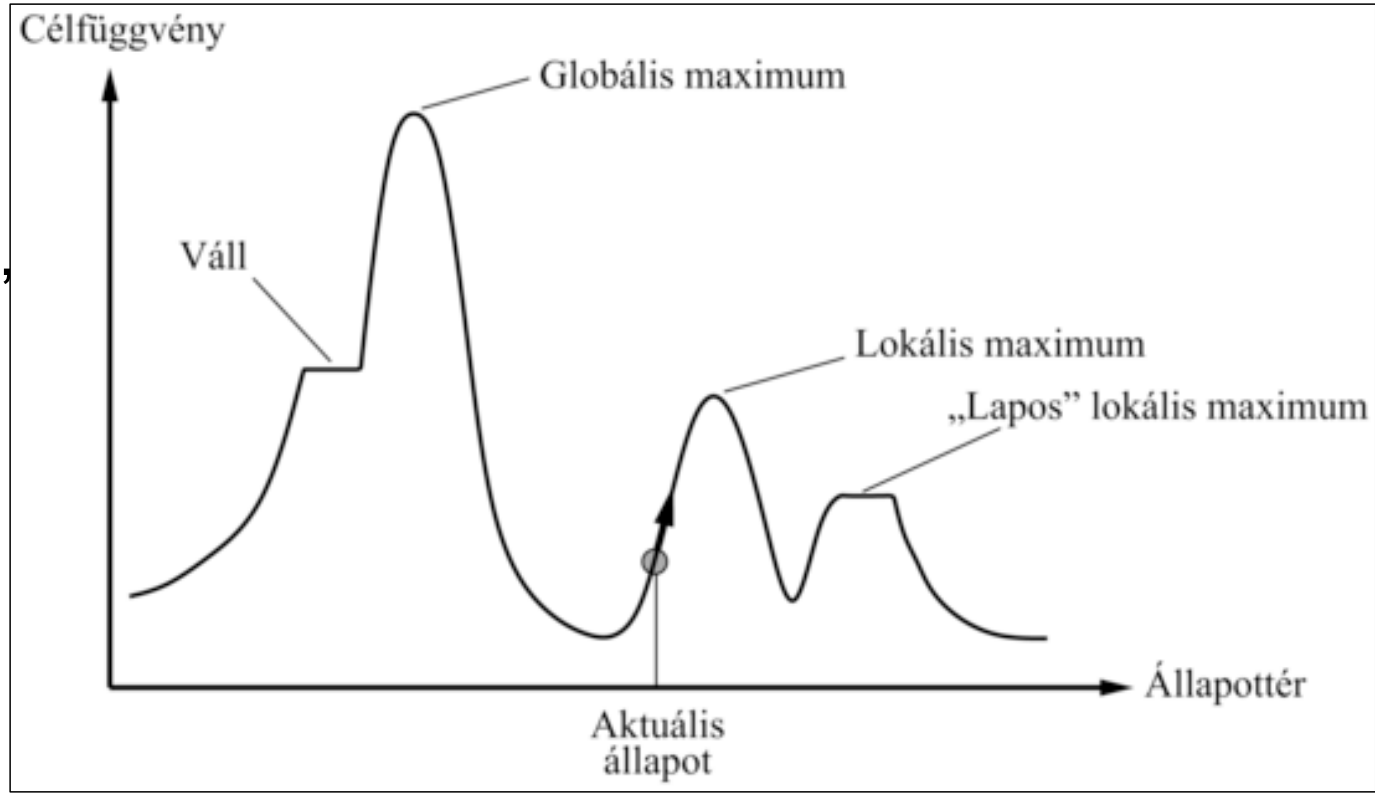
- ciklikusan lép mindig a javuló értékek felé tart (diszkrét gradiens módszer)
- nem tart nyilván keresési fát
- ha egynél több egyformán legjobb követő csomópont merül fel, véletlenszerűen bármelyiket kiválaszthatja.



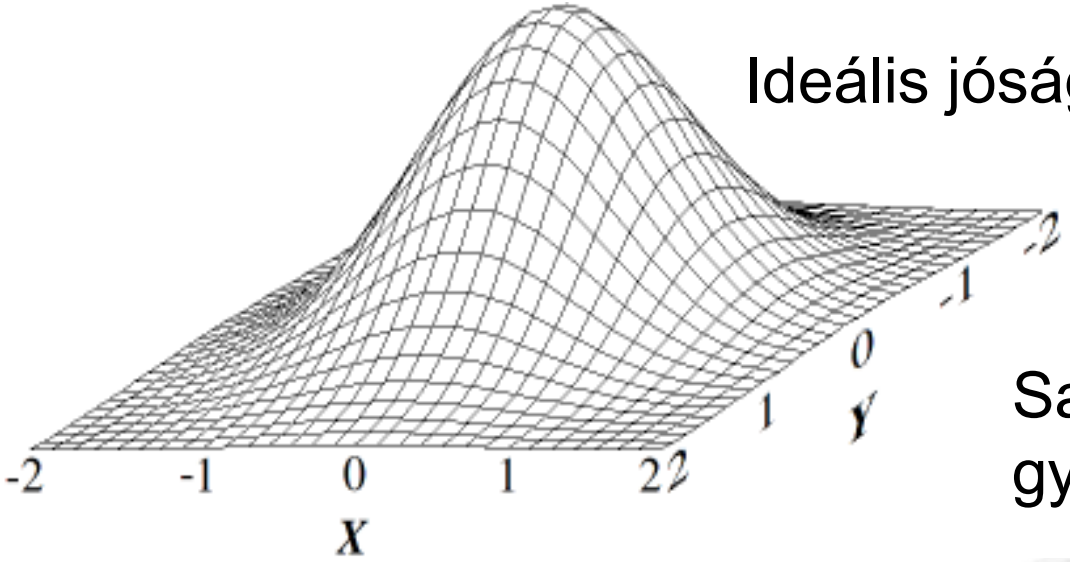
Hegymászó keresés

3 fő probléma:

- **lokális maximum(ok)**: ha egy lokális maximumba ér, akkor ott megáll
- **fennsík**: ott a kiértékelő függvény lapos, véletlen irányválasztás
- **hegygerinc**: egy hegygerinc oldalai meredekek, a keresés könnyen eljut a gerincre, de előfordul, hogy a gerinc csak nagyon lassan emelkedik a csúcs felé

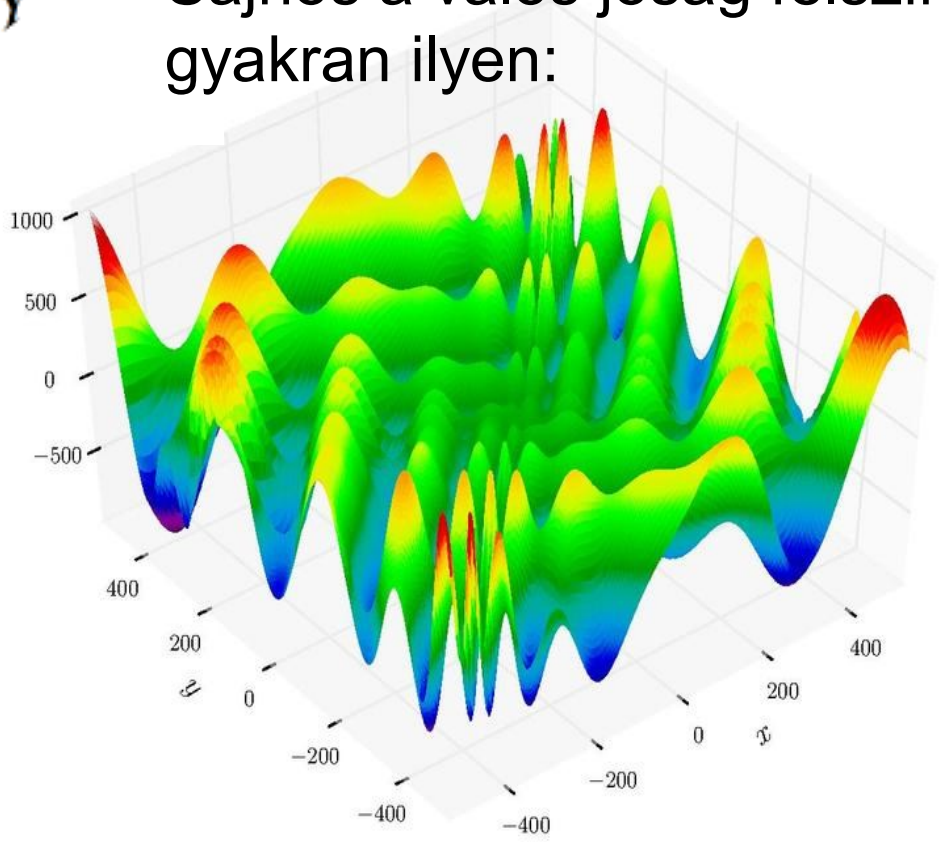


Hegymászó keresés



Ideális jóság felszín

Sajnos a valós jóság felszín gyakran ilyen:



Lokális szélsőérték

$$f'(x) = 0$$

Ha több szélsőérték van, melyik a globális?

Lokálisan kereső algoritmusok kvíz következik!

Sok különböző változat van, pl.:

- **szimulált lehűtés**: néha megenged olyan lépéseket is, amelyek hatására (legalábbis átmenetileg) rosszabb állapotba kerül

- **lokális nyaláb keresés**: mindig *k* csomópontot tart számon, azok minden gyerekét feltárja és belőlük a *k* legjobbat tartja meg

- **véletlen indítású** … …

- **genetikus algoritmusok**: hasonló a lokális nyaláb keresésre, de speciális (genetikus) lépésoperátorokkal

… … …

Kvíz

Alapvetően miért alkalmazunk szimulált lehűtést?

A. Azt reméljük, hogy nem ragadunk be a lokális szélsőértékekbe

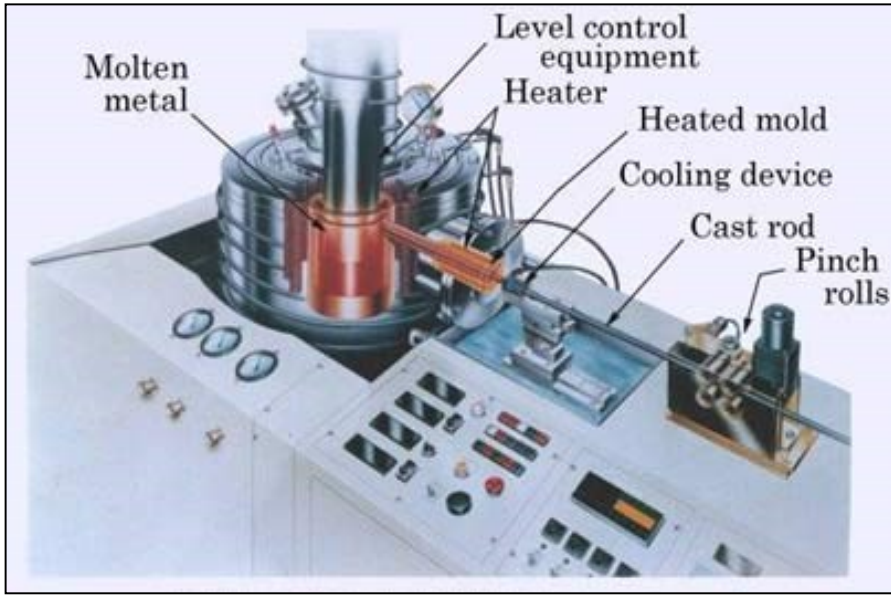
B. Azt reméljük, hogy gyorsabb lesz a konvergencia

C. Azt reméljük, hogy jobb globális optimumhoz jutunk

D. Azt reméljük, hogy közelebbi globális megoldáshoz jutunk

Szimulált lehűtés

Ha a keresés egy lokális maximumban beragadna, megengedhetjük, hogy néhány lefelé (rossz irányba) vezető lépést tegyen, hogy kimeneküljön a lokális maximumból.



A szimulált lehűtés ciklusa:

- a legjobb lépés megtétele helyett egy **véletlen lépést** tesz.
- **ha a lépés javít**, akkor az mindig végrehajtásra kerül.
- ellenkező esetben az algoritmus a lépést csak valamilyen, 1-nél kisebb *P* **valószínűséggel** teszi meg (Boltzmann-eloszlás).

$$P \approx e^{\frac{-\Delta E}{T}}$$

A valószínűség exponenciálisan csökken a lépés "rontó" képességével – azzal a ΔE mennyiséggel, amivel a kiértékelő függvény romlott, és az idővel, a hűtés – $T(t)$ – függvényében.

Szimulált lehűtés

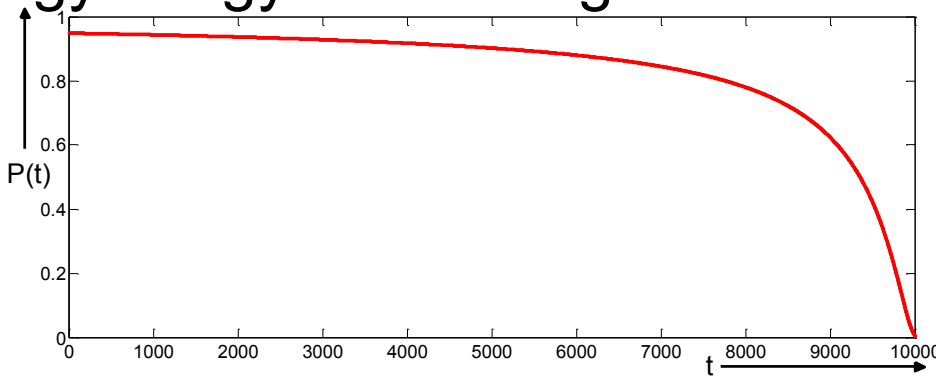
T(idő) - hűtési karakterisztika

- magas T-nél a "rossz" lépések nagy valószínűséggel fordulnak elő $P \approx e^{\frac{-\Delta E}{T}} \approx e^{0} \approx 1$
- ahogy *T* tart nullához, a rossz lépések egyre kevésbé valószínűek $P \approx e^{\frac{-\Delta E}{0}} \approx e^{-\infty} \approx 0$

az algoritmus többé-kevésbé egy hegymászó algoritmusként viselkedik.

Hűtési karakterisztika:

a hőmérséklet csökkenté-sének időbeli lefolyása



Fizika: ha a hőmérsékletet kellően lassan csökkentjük, akkor az anyag a legkisebb energiájú állapotba jut.

Szimulált lehűtés: ha a **hűtési karakterisztika** kellően lassan csökkenti *T*-t, az algoritmus a globális maximumot találja meg. Az egyedi lépések - termikus zaj okozta véletlen fluktuáció.

Véletlen újraindítású hegymászás (és sok más változat):

Véletlenül generált kiinduló állapotokból hegymászó keresés, amíg nem jár le az idő, vagy már nincs észrevehető javulás.

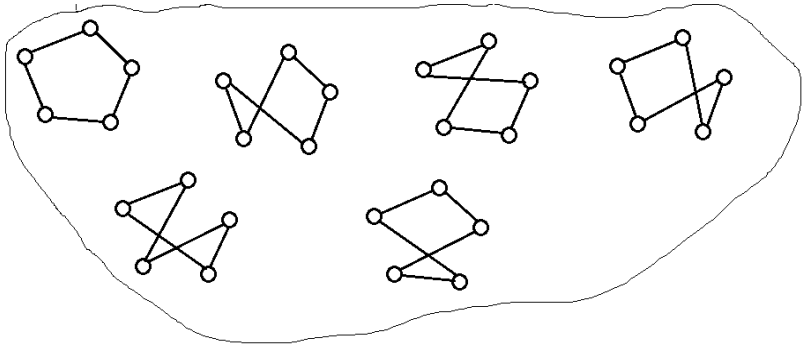
A hegymászás sikere: az állapottér „**felszínének" alakjá**tól függ

- ha csak **néhány** lokális maximum: gyors megoldás
- egy valódi probléma = „sündisznó" jellegű felszín

Ha a probléma NP-teljes, akkor az exponenciális időigénynél jobb nem lesz = exponenciálisan sok lokális maximum.

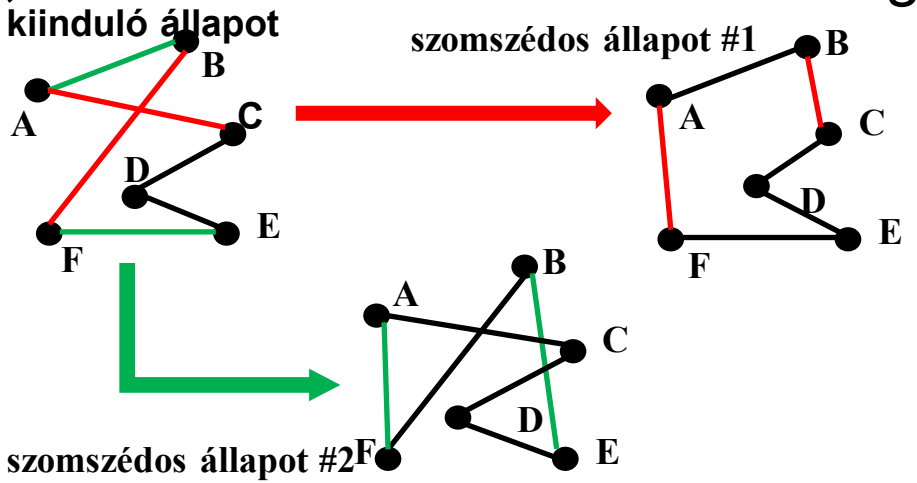
Általában kisszámú iteráció után már **elfogadhatóan jó** megoldást lehet találni. (***Sokszor jobb egy belátható időn belül elérhető szuboptimum, mint végtelen ideig keresgélni a globális optimumot.***)

Demópélda: **Utazó ügynök probléma – Travelling Salesman Problem**



Feladat: A városokat az utazó-ügynök a lehető legrövidebb úton kell végigjárja (NP-teljes probléma!)

pl. 2-opt heurisztika (nem javítható oly módon, hogy két élét el-vágjuk, és a keletkező két utat máshogy kötjük össze) – **O(n^2) idő**



Követő állapotok: a kiinduló állapotban az összes lehetséges módon kiválasztunk 2 élt, és átcseréljük a végpontokat. Megnézzük, hogy az összúthossz csökkent-e. (Költségminimumot keresünk)

Kvíz – utazó ügynök probléma 48 város esetén

- Hány lehetőség van,ha 48 városnál kimerítő vizsgálatot végzünk?
- Hány lehetőséget vizsgálunk meg, ha 100-szor véletlen helyzetből (gráfból) indítva, átlagosan 200 lépésben 2-opt vizsgálatot végzünk?

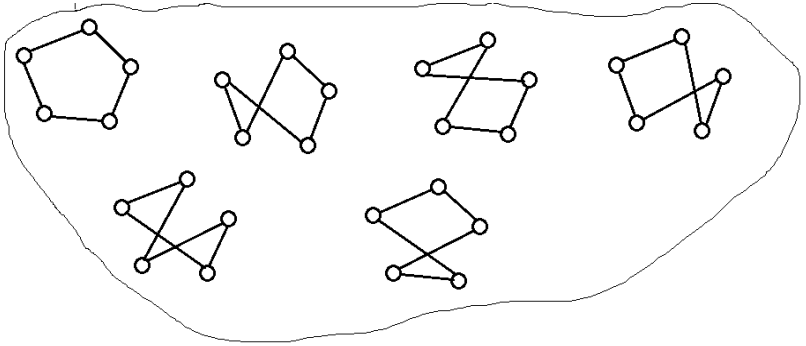
A. Kimerítő: 2^48 (kb. 10^14) ; 100 indítás 200 lépés 2-opt: 3*10^7

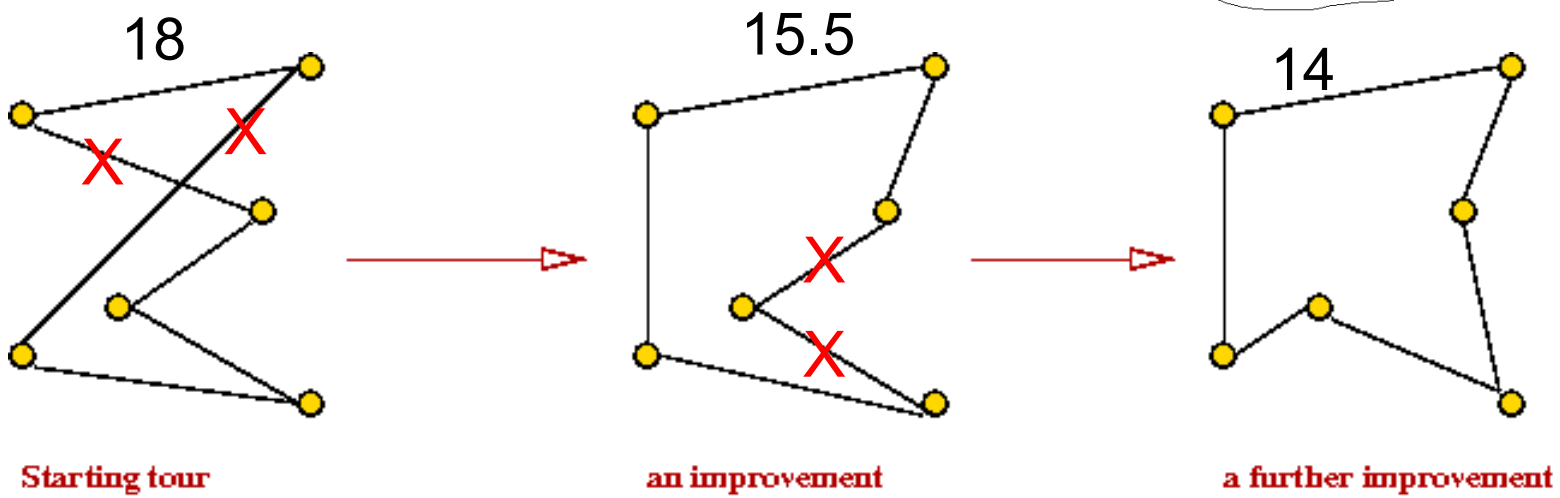
B: Kimerítő: 48! (kb. 10^61) ; 100 indítás 200 lépés 2-opt: 3*10^7

C: Kimerítő: 2^48 (kb. 10^14) ; 100 indítás 200 lépés 2-opt: 10^4

D. Kimerítő: 48! (kb. 10^61) ; 100 indítás 200 lépés 2-opt: 10^4

Demópélda: **Utazó ügynök probléma – Travelling Salesman Problem**





3-opt, …, *k*-opt heurisztika, stb.

3-opt utak, 48 város, 100 véletlen újraindítás, optimális megoldás 0,99 valószínűséggel
(elvben 48! = $1{,}2 \cdot 10^{61}$ lehetőség)

Nyaláb-keresés

Lokális nyaláb keresés - *k* állapotot követ nyomon

Indulás: *k* véletlen módon generált induló állapot

- Minden lépésben a *k* állapot mindegyikének összes követőit kifejti
- Ha ezek valamelyike egy cél, az algoritmus leáll
- Különben a teljes listából kiválasztja a legjobb *k* követőt,
- és ezt az eljárást ismétli.

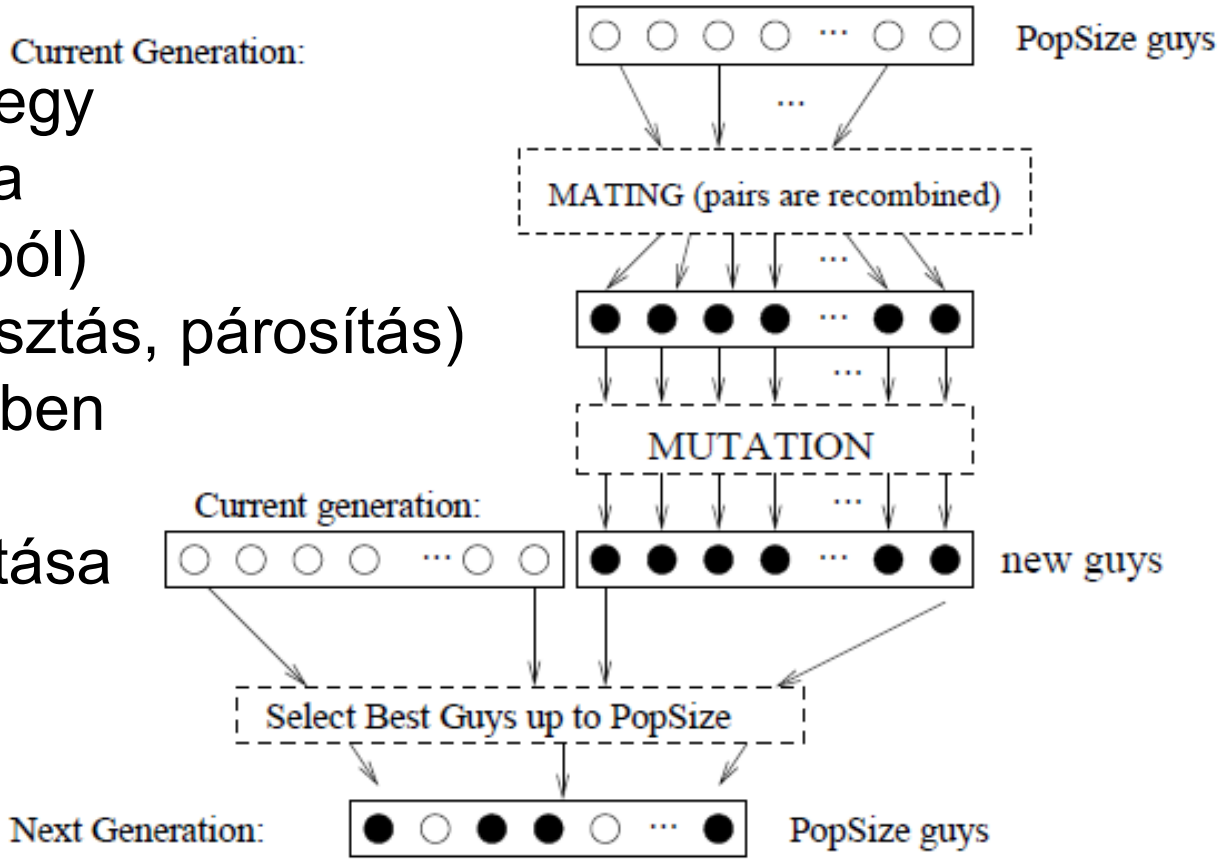
Megnövelt tár **-** … **-** megnövelt esély a „jó” szélsőértékre.

Genetikus algoritmusok

Indulás: *k* véletlen módon generált állapot = populáció

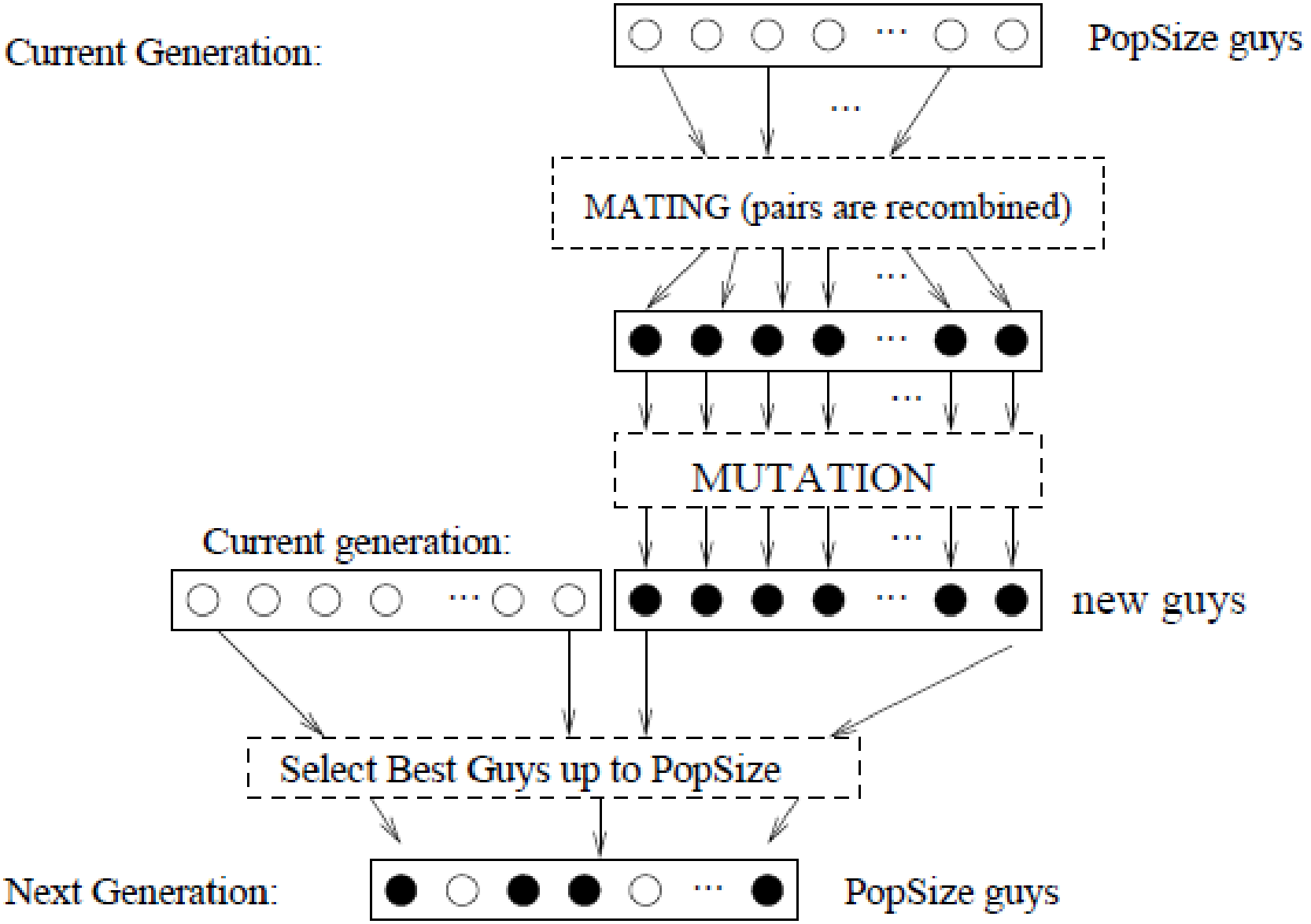
Minden **állapot** vagy **egyed** = egy véges ábécé fölött értelmezett, leggyakrabban egy, a 0-kból és 1-ekből álló füzér (a probléma kódolása)

- kiinduló populáció
- fitnesz-függvény (=egy egyed mennyire jó a célunk szempontjából)
- keresztezés (kiválasztás, párosítás) a fitnesz függvényében
- mutáció
- új populáció kialakítása



Mint a hegymászó, ill. a nyaláb-keresés (ivartalan/ ivaros szaporodás)

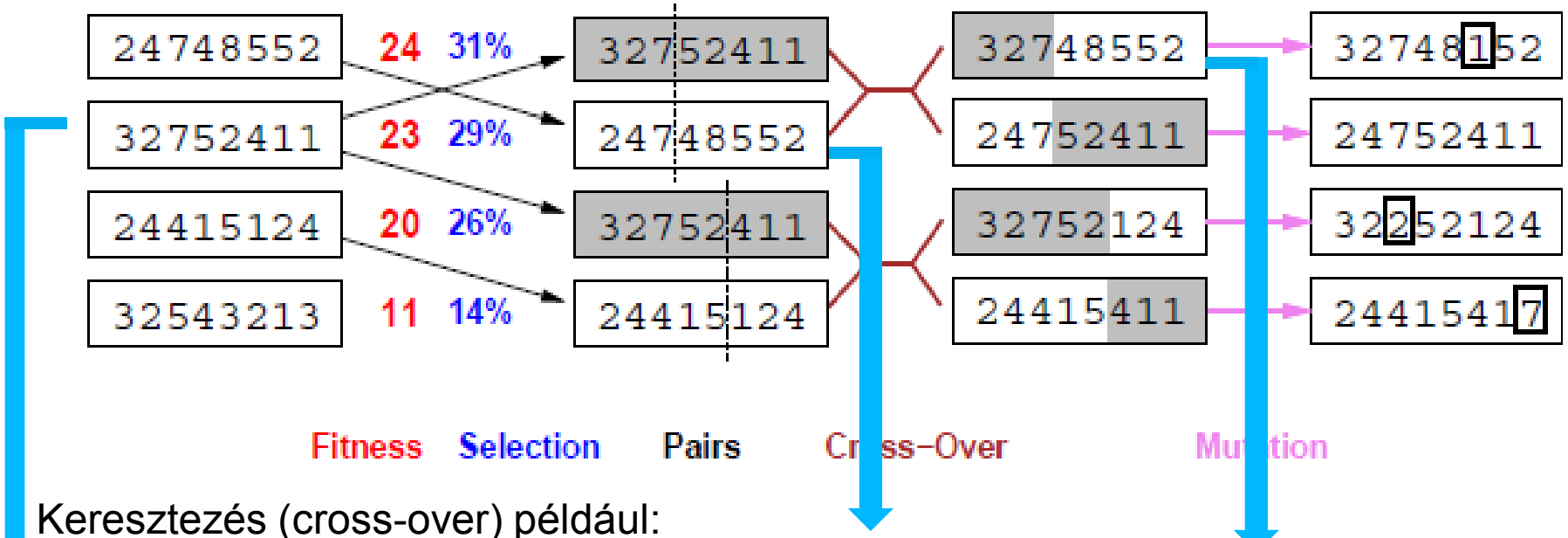
Genetikus algoritmusok



Genetikus algoritmusok

Demópélda: A 8-királynő probléma, **állapot**ai pl. 8 számjegyes füzérek (1.oszlopban királynő poz., 2. oszlopban poz., …, 8. oszlopban poz.)

fitnesz = 28 – *n* (ahol *n* az adott elhelyezésben az ütések száma)



Keresztezés (cross-over) például: