

II. VEZÉRLÉSI STRATÉGIÁK

1. Nem-módosítható
2. Visszalépéses
3. Gráfkereső

1

1. Nemmódosítható stratégia

- A nemmódosítható keresés olyan kereső rendszer, ahol
 - globális munkaterület: a reprezentációs gráf egy csúcsa
 - kiinduló érték: startcsúcs,
 - terminálási feltétel: az aktuális csúcs egy célcúcs vagy a keresés megakad.
 - keresés szabálya: az aktuális csúcsot cseréli ki
 - vezérlési stratégia: kiválasztja az új csúcsot,

2

Alkalmazás

- Amikor rendelkezünk a problémátérn olyan célfüggvénnyel (heurisztikával), amely a helyes megoldásokban veszi fel az optimumát
 - Lokális keresés, amikor egy csúcs szimbolizál
 - egy lehetséges választ (általános útkeresés)
 - egy választ leíró útnak egy pontját (speciális útkeresés)
 - Evolúciós algoritmus, amikor egy csúcs egyszerre több lehetséges választ szimbolizál.
- Olyan reprezentációs gráfok esetén, ahol egy rossz döntés miatt nem juthat a keresés zsákutcaba
 - Kommutatív állapotter reprezentáció

3

Kommutatív állapotter reprezentáció

- Egy adott állapotra alkalmazható műveletek mindazokra az állapotokra is végrehajthatók, amelyek az adott állapotból egy művelettel elérhetők.
- Egy állapotra alkalmazott műveletsorozat és annak tetszőleges permutációja azonos állapotba vezet.
- Céllállapotból elérhető állapot is célállapot.
- Példa: rezolúció

4

1.2. Lokális keresések

- Egy adott pillanatban ismert egyetlen csúcsot (lehetséges választ) annak környezetéből vett lehetőleg jobb csúccsal cseréli le.
- A jobbság eldöntéséhez a célfüggvényt használ
- Alkalmazás:
 - Adott tulajdonságú elem keresése
 - Függvény optimumának keresése

5

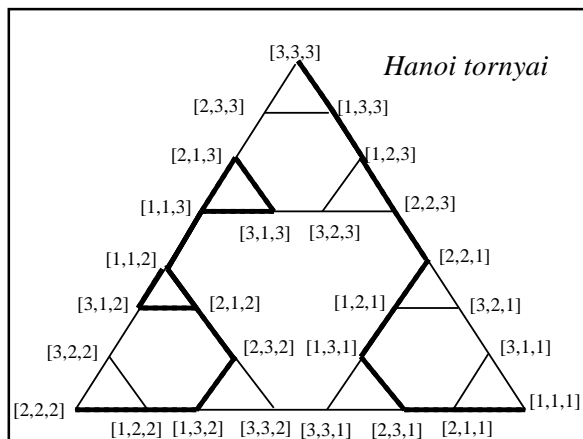
Hegymászó algoritmus

- Lokális optimumban megengedi a legjobb - de az aktuális csúcsnál rosszabb - szomszédra való lépést, és kizárja a szülő csúcsra való visszalépést.

Procedure Hegymászó módszer

1. $n \leftarrow \text{startcsúcs}$
2. while n nem célcúcs loop
3. $n \leftarrow \text{opt}(\Gamma(n), \pi(n))$
4. endloop
- end

6



Megjegyzés

- Hátrányok: Nem végez körfigyelést, ezért
 - lokális optimum hely körül eltévedhet
 - ekvidisztans felületen eltévedhet
- A baj okai:
 - Túl kicsi az algoritmus memóriája
 - Túl erős az alkalmazott mohó stratégia

8

Tabu-keresés

- Az n aktuális csúcson kívül nyilvántartjuk még
 - az eddig legjobbnak bizonyult n^* csúcsot és
 - az utóbbi néhány aktuális csúcsot; ez a tabu halmaz
- Minden lépésben
 - kiválasztjuk a legjobb csúcsot az aktuális csúcs környezetéből (kivéve ebből a tabu csúcsokat)
 - ha ez jobb, mint az n^* , akkor azt lecseréljük
 - fentiek alapján módosítjuk a tabu halmazt
- Terminálási feltételek:
 - ha a célfüggvény az n^* -ban optimális
 - ha az n nem vagy az n^* sokáig nem változik.

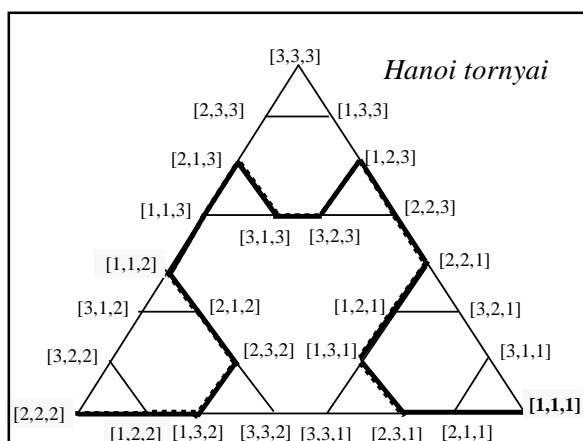
9

Tabu keresés algoritmus

Procedure *Tabu keresés*

1. $n, n^*, Tabu \leftarrow startcsúcs, startcsúcs, \emptyset$
 2. **while** not terminálási feltétel (n^* nem célcsúcs) **loop**
 3. $n \leftarrow \text{opt}_{\Gamma(n) \setminus Tabu}$
 4. $Tabu \leftarrow \text{Módosít}(n, Tabu)$
 5. **if** $f(n) > f(n^*)$ **then** $n^* \leftarrow n$
 6. **endloop**
- end**

10



Megjegyzés

- Hátrányok:
 - A tabu méretét kísérletezéssel kell belőni
 - beszorulhat a keresés

12

Szimulált hűtés

- A következő csúcs választása véletlenszerű.
- Ha a kiválasztott r csúcs célfüggvény értéke rosszabb (kisebb), mint az aktuális n csúcsé, akkor újcúcsként való elfogadásának valószínűsége fordítottan arányos $f(r)$ és $f(n)$ különbséggel.
- Egy rosszabb csúcs elfogadásának valószínűsége az idő függvényében csökken ($T > 0$).

ha $f(r) \geq f(n)$ **vagy**

$$f(r) < f(n) \text{ és } e^{\frac{f(r) - f(n)}{T}} > \text{random}$$

akkor $n \leftarrow r$

13

Az új csúcs elfogadásának valószínűsége

$$T=10, f(n)=120$$

$f(r)$	$\exp((f(r)-f(n))/10)$
147	OK (14.88)
127	OK (2.01)
120	OK (1.00)
107	0.27
77	0.01

14

A T hányados szerepe

- Az új csúcs elfogadásának valószínűsége a T függvényében:

$$f(n)=120, f(r)=107$$

T	$\exp(-13/T)$
1	0.000002
5	0.0743
10	0.2725
20	0.52
50	0.77
10^{10}	0.9999...

15

Szimulált hűtés algoritmus

Procedure Szimulált hűtés

1. $n \leftarrow \text{startcsúcs}; k \leftarrow 1$
2. **while** not terminálási feltétel (n nem célcsúcs) **loop**
3. **for** $i = 1 \dots L_k$ **loop**
4. $r \leftarrow \text{select}(\Gamma(n), \pi(n))$
5. **if** $f(r) \geq f(n)$ or $f(r) < f(n)$ and $e^{\frac{f(r) - f(n)}{T_k}} > \text{random}$
6. **then** $n \leftarrow r$
7. **endloop**
8. **endloop**

16

Hűtési ütemterv

- A T csökkentésével csökken egy új csúcs elfogadásának valószínűsége.
- Adjunk ütemtervet a T változására
- Az ütemterv elemei:
 - Kezdeti hőmérséklet: T_0
 - Hőmérséklet csökkentésének menete és egy hőmérséklet melletti szakasz hossza:

$$(T_k, L_k) \quad k=1, 2, \dots$$
 ahol minden T_k érték L_k lépésen keresztül van érvényben.

17

Szimulált hűtés ereje

- A szimulált hűtés algoritmus (aszimptotikusan) egy optimális megoldáshoz konvergál, ha
 - az algoritmussal bármely csúcsból bármely csúcs elvileg véges lépésen belül elérhető (erősen összefüggés, csúcs környezet)
- Ahhoz azonban, hogy véges lépésen belül is egy elég jó megoldást találjunk, megfelelő hűtési ütemtervet kell találni.

18

1.3. Evolúciós (genetikus) algoritmus

- ❑ Egy adott pillanatban nem egyetlen lehetséges választ, hanem lehetséges válaszok (egyedek) halmazát, populációját tartjuk nyilván.
- ❑ A populációt lépésenként próbáljuk meg jobbra cserélni. Egy populáció annál jobb, minél inkább olyan egyedekkel rendelkezik, amelyek a kitűzött probléma helyes válaszai vagy azokhoz közeli lehetséges válaszok.
- ❑ A populáció megváltozása visszavonhatatlan.

19

Evolúciós algoritmus működése

- ❑ Kezdetben egy véletlen populációt választunk.
- ❑ Minden lépésben
 - *Szelekció*: Kiválasztja a szülő egyedeket.
 - *Rekombináció (keresztelés)*: A szülőkből utódokat állít elő.
 - *Mutáció*: az utódok kismértékű változtatása.
 - *Visszahelyezés*: utódokat is tartalmazó új populáció kialakítása
- ❑ A cél lehet egy keresett célegyed előállítása, vagy a populáció globális értékének változatlansága.

20

Alapalgoritmus

```
Procedure EA
   $p \leftarrow$  kezdeti populáció
  while terminálási feltétel nem igaz loop
     $p' \leftarrow$  szelekció( $p$ )
     $p'' \leftarrow$  rekombináció( $p'$ )
     $p''' \leftarrow$  mutáció( $p''$ )
     $p \leftarrow$  visszahelyezés( $p, p'''$ )
  endloop
```

21

Algoritmus elemei

- ❑ Kódolás (egyed reprezentáció)
- ❑ Rátermettségi (fitness) függvény
 - Kapcsolat a célfüggvénnyel
- ❑ Evolúciós operátorok
 - szelekció, rekombináció (keresztelés), mutáció, visszahelyezés
- ❑ Kezdő populáció, Megállási feltétel
- ❑ Stratégiai paraméterek
 - populáció mérete, mutáció valószínűsége, utódképzési ráta, visszahelyezési ráta, stb.

22


Kódolás

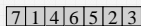
- ❑ Egy egyedet jelsorozattal (kromoszóma) kódolunk.
- ❑ A jelek (gén) vagy azok csoportjai írják le az egyed tulajdonságait: attribútum és érték.
- ❑ Sokszor egy jelnek a kódsorozatban elfoglalt pozíciója (lókuszt) adja meg az attribútumot, a jel pedig az értéket (allél). Ilyenkor a kód szerkezete tulajdonságunkénti feldarabolhatóságot mutat.
- ❑ Gyakori megoldás:
 - valós (egész) számok tömbje, bináris tömb,
 - permutáció, fa-ábrázolás

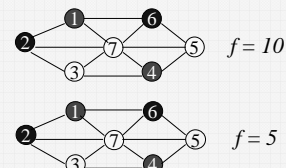
23

Gráfszínezési példa kódolása

- ❑ Adott egy véges egyszerű gráf, amelynek a csúcsait a lehető legkevesebb szín felhasználásával úgy kell kiszínezni, hogy a szomszédos csúcsok eltérő színűek legyenek.

1. 2. 3. 4. 5. 6. 7.

direkt


indirekt



24

Szelekció

- **Célja:** a rátermett egyedek kiválasztása úgy, hogy a rosszabbak kiválasztása is kapjon esélyt.
- Néhány módszer:
 - **Rátermettség arányos:** a rátermettségi függvényre vagy annak skálázására épülő rulett kerék algoritmus
 - **Rangsorolásos:** rátermetség alapján sorba rendezett egyedek közül a kisebb sorszámúakat nagyobb valószínűséggel választja ki
 - **Versengő:** véletlenül kiválasztott egyedszoportok (pl. párok) legjobb egyedét választja ki.
 - **Csonkolásos:** a rátermetség szerint legjobb valahány egyedből véletlenszerűen választ néhányat.

25

Rekombináció

- A feladata az, hogy adott szülő-egyedekből olyan utódokat hozzon létre, amelyek a szülei tulajdonságait hordozzák.
- Tulajdonságai
 - A rekombináció a kód szerkezetéhez illeszkedő átalakítás, amely kihasználja, hogy a kód szerkezete általában tulajdonságonkénti darabolhatóságot mutat.
 - A csupán szerkezeti átrendezést keresztezésnek hívják
 - Ügyelni kell a kód-invariáns megtartására (permutáció)
 - Sztochasztikus és heurisztikus módszerek

26

Rekombináció tömbökre

- **Köztes rekombináció**
 - A szülők (x, y) által kifeszített hiperkocka valamelyik eleme lesz az utód (u)
 - $\forall i=1 \dots n : u_i = a_i x_i + (1-a_i)y_i \quad a_i \in [-h, 1+h]$ véletlen
- **Lineáris rekombináció**
 - A szülők (x, y) által kifeszített hiperkocka egy adott hipersíkjának valamelyik eleme lesz az utód (u)
 - $\forall i=1 \dots n : u_i = a x_i + (1-a)y_i \quad a \in [-h, 1+h]$ véletlen

27

Keresztezés

- **Egy- illetve többpontos keresztezés**
 - Kódszakaszokat cserélünk (az allélok nem szerencsés kettévágni)

0	0	1	0	1
1	1	1	1	0

 \Rightarrow

0	1	1	0	1
1	0	1	1	0

- **Egyenletes keresztezés**
 - Jeleket cserélünk

0	0	1	0	1
1	1	1	1	0

 \Rightarrow

1	0	1	0	1
0	1	1	1	0

28

Permutációk keresztezése

- **Parciálisan illesztett keresztezés**
 - Szakasz cseréje után a perm.tul. sértő párokat.
- **Ciklikus keresztezés**
 - $a_i \leftrightarrow b_i; a_j \leftrightarrow b_j$, ahol $a_i = b_j (j \neq i)$; stb.

2 3 1 5 4 6 7 \Rightarrow 2 7 4 2 4 6 7 \Rightarrow 1 7 4 2 5 6 3
 1 7 4 2 5 3 6 1 3 1 5 5 3 6 2 3 1 5 4 7 6

2 3 1 5 4 6 7 \Rightarrow 2 3 1 2 4 6 7 \Rightarrow 1 3 1 2 4 6 7
 1 7 4 2 5 3 6 1 7 4 5 5 3 6 2 7 4 5 5 3 6

\Rightarrow 1 3 4 2 4 6 7 \Rightarrow 1 3 4 2 5 6 7
 2 7 1 5 5 3 6 2 7 1 5 4 6 3

29

Mutáció

- A mutáció egy egyed (utód) kis mértékű véletlen változtatását, finom közelítését végzi.
- Valós tömbbel való kódolásnál kis p valószínűséggel:
 - $\forall i=1 \dots n : z_i = x_i \pm range_i * p$
- Bináris tömbbel való kódolásnál kis p valószínűséggel:
 - $\forall i=1 \dots n : z_i = 1 - x_i$
- Permutáció esetén kis valószínűséggel választott pozíció-párra
 - csere; a pozíciók közötti szakaszon a jelek léptetése, a jelek sorozatának megfordítása, esetleg átrendezése.

30

Visszahelyezés

- A visszahelyezés a populációnak az utódokkal történő frissítése. Kiválasztja (második szelekció) a populációnak a lecserélendő egyedeit, és azok helyett az utódokat veszi fel.

$$\text{utódképzési ráta} = \frac{\text{utódok száma}}{\text{populáció száma}}$$

$$\text{visszahelyezési ráta} = \frac{\text{lecserélendő egyedek száma}}{\text{populáció száma}}$$

- ha $u=v$, akkor feltétlen cseréről van szó
- ha $u < v$, akkor a valóban lecserélendő egyedeket (egy harmadik) szelekcióval válogatjuk ki
- ha $u > v$, akkor a visszahelyezendő utódokat (egy harmadik) szelekcióval válogatjuk ki

31

GYAKORLAT

32

Gráfszínezés szimulált hűtéssel

□ Feladat:

- Adott egy véges egyszerű gráf, amelynek a csúcsait a lehető legkevesebb szín felhasználásával úgy kell kiszínezni, hogy a szomszédos csúcsok eltérő színűek legyenek.

□ Cél:

- A gráf csúcsainak olyan minimális osztályozását (egy osztályba tartozó csúcsok azonos színűek) keressük, ahol egy osztályhoz tartozó csúcsok között nem vezet él.

33

Gráfszínezés állapotér-reprezentációja

□ Állapot: a csúcsok egy olyan osztályozása, ahol

- A gráf maximális fokszámánál több osztály van,
- lehet egy osztály üres is,
- egy osztályon belül lehetnek élek.

□ Művelet: Egy osztályból egy csúcsot egy másik osztályba helyezünk.

□ Állapotgráf: Exponenciális méretű az eredeti gráf csúcsszámahoz mérve.

□ Célláallapot: a legjobb osztályozás

34

Gráfszínezés célfüggvénye

□ Annál jobb egy (O_1, \dots, O_k) osztályozás,

- minél több csúcs van az első néhány osztályában (ezáltal minél kevesebb nem üres osztály van), és
- minél kevesebb egy osztályon belül vezető élek száma.

$$\square f(n) = \sum_j w_j (|O_j| - \lambda |A(O_j)|)$$

- ahol $A(O_j)$ az O_j osztálybeli élek halmaza
- a $w_j > 0$ számok szigorúan fogyó sorozatot alkotnak.

□ Könnyű a „szomszédos” osztályozás célfüggvény-értékét kiszámolni.

35

Példa

- Hol veszi fel az $f: [0 \dots 1024] \rightarrow [-1, 1]$ függvény a egész intervallumon a maximumát? (A f -et nem ismerjük, de az $f(x)$ -t tetszőleges x -re ki tudjuk számolni)

36

x	kód	$f(x)$	$(f(x)+1)/(\Sigma+10)$	rulett
13	0000001101	0.22	0.10	2
53	0000110101	0.79	0.15	0
119	0001110111	0.87	0.15	1
339	0101010011	-0.35	0.05	0
358	0101100110	-0.03	0.08	1
482	0111100010	0.84	0.15	2
602	1001011010	-0.88	0.01	0
778	1100001010	0.84	0.15	2
841	1101001001	0.85	0.15	2
956	1110111100	-0.82	0.01	0

Össz:	2.33
Átl:	0.23
Max:	0.87

rulett	x	szelekció	rekombináció	mutáció
rulett	x	x	kód	kód
2	13	13	0000001101	0000001001
0	53	841	1101001001	1101001101
1	119	13	0000001101	0000001010
0	339	778	1100001010	1100001101
1	358	119	0001110111	0001011100
2	482	778	1100011100	1101110111
0	602	358	0101100110	0101100010
2	778	482	0111100010	0111100110
2	841	482	0111100010	0111001001
0	956	841	1101001001	1101100010

visszahelyezés = lecserélés

x	kód	$f(x)$
9	0000001001	0.15
845	1101001101	0.81
10	0000001010	0.17
781	1100001101	0.87
92	0001011100	0.99
887	1101110111	0.22
354	0101100010	-0.10
486	0111100110	0.80
457	0111001001	0.99
832	1101000000	0.92

Össz:	2.33
Átl:	0.23
Max:	0.87

Össz:	5.82
Átl:	0.58
Max:	0.99

Utazó ügynök probléma kódolásai

- Út kódolás
 - A városok az útvonalnak megfelelő sorrendben helyezkednek el a kódban.
- Szomszédsági kódolás
 - Ha az i -edik város után a j -edik következik, akkor a j a kód i -edik pozícióján áll.
- Közöséges kódolás
 - A városokat a kód alapján egy C listából vesszük ki sorban egymás után. A kód eleme azt mutatja meg, hogy a C lista hányadik városa a soron következő. A kiválasztott várost mindig töröljük a C -ből.

Rekombináció: Permutációk keresztezése 3.

□ Rendezett keresztezés

- illesztési szakaszon mindent cserélünk, a problémát nem okozó elemeket ciklikusan felzárkóztatjuk az illesztési szakasz jobb széléhez, majd a hiányzó elemeket beírjuk azok szülőbeli sorrendjében.

2 3 1 5 4 6 7 1 7 4 2 5 3 6	→	5 7 4 2 6 3 1 2 3 1 5 6 7 4
--------------------------------	---	--------------------------------

. 7 4 2 . 6 .	. 7 4 2 6 . .	< 3, 1, 5 >
. 3 1 5 . . 6	. 3 1 5 6 . .	< 7, 4, 2 >

Rekombináció: Permutációk keresztezése 4.

□ Él rekombináció

- szülőkbeli szomszédságok megőrzésére törekszik

1 2 3 4 5 4 3 5 2 1	⇒	1: 2, 2, 4, 5 2: 1, 1, 3, 5 3: 2, 4, 4, 5 4: 1, 3, 3, 5 5: 1, 2, 3, 4	⇒	1 2 5 3 4
------------------------	---	---	---	-----------

Véletlen választ egyet (1), és azt mindenholon törli
 1 szomszédjai közül a dublát (2) választja, majd törli
 2 szomszédjai közül a kevesebb szomszédút (5) választja, és törli
 5 szomszédjai közül a kevesebb szomszédút (3), majd a 4-et

Egyéb keresztezések

❑ **Heurisztikus keresztezés**

- A célfüggvényt figyelembe véve alakítjuk ki az utódokat

43