Tartalomjegyzék

Bevezető

A szerző – s ezáltal e jegyzet – semmilyen szempontból nem törekszik teljességre.

1

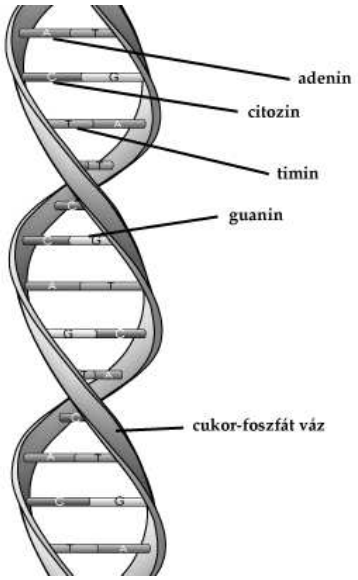
Egy kis biológia

1.1. A DNS

Egy organizmus létrehozásához szükséges teljes utasításhalmazt *genomnak* nevezzük. A genom fő modelljét képezi minden sejtszerkezetnek és sejttevékenységnek a sejt vagy a teljes szervezet egész élettartamára. A genom dezoxiribonukleinsav-szálakból (röviden DNS) és a hozzá tartozó *protein* molekulákból áll. A DNS a sejtmagban található, továbbá az organizmus minden sejtje ugyanazon DNS molekulákat tartalmazza, viszont minden sejttípus más és más proteint generál. A proteinek az előlények elsődleges alkotóielemei, olyan *enzimek*, melyek lehetővé teszik az élethez szükséges kémiai reakciókat. Ezek a proteinek aminosavakból épülnek fel.

Egy egyed genetikai kódját a DNS molekulák tartalmazzák. A DNS alakja egy ,,sodort létra", azaz két, egymással ,,párhuzamos" kötésekkel összekapcsolt szál. Alkotóelemei a nukleotidák, melyek három részből állnak. Ezek fő komponense a bázismolekula, mely kétféle lehet: purin [adenin (A) vagy guanin (G)] vagy piramidin molekula [citozin (C) vagy thimin (T)]. A két szál közötti kötés hidrogén-kötés, azaz gyenge, ami két bázist köt össze: egy purin és egy piramidin molekulát. A lehetséges kötések tehát G–C és T–A, illetve ezek fordítottjai. Ebből látszik, hogy az egyik DNS szál a másik komplementere, ezért ha ismert az egyik, akkor a másik egyértelműen felépíthető. Sejtosztódáskor a spirál kettéválik – egyik szál megy a létrejövő egyik sejtbe, a másik szál a másikba – majd ott, a bázisoknak megfelelően, létrehozzák a maguk kiegészítő szálját.

Az egymást nem átfedő nukleotid-hármasokat *kodonoknak* nevezzük. Minden kodon megfelel egy aminosavnak, azaz minden kodon egy aminosavat kódol (ez fordítva nem igaz, több különböző kodon is kódolhatja egyazon aminosavat). Kétféle kodon létezik. Az ún. *stop-kodon* nem kódol semmit, ez az aminosavszekvencia, azaz a protein végét jelöli. Az összes többi egy-egy aminosavat kódol. A dolgok azonban tovább bonyolódnak. A DNS-ben vannak olyan részek, melyek



1.1. ábra. A DNS leegyszerűsített szerkezete

nem kódolnak semmit; ezeket *intronoknak* nevezzük (a DNS aminosavba való leképezésekor ezek a részek kiesnek). A jelentéssel bíró szekvenciákat *exonoknak* nevezzük (vigyázat, nem összetévesztendő az *axonnal*).
Az aminosav-szekvenciába való leképezés két lépésben történik: az első lépésben a DNS átíródik egy ún. mRNS (m = messenger) molekulába, amibe már csak az exonok kerülnek; a második lépés az mRNS aminosav-szekvenciába való leképezése.

1.2. Kromoszóma, gén, öröklődés

A *kromoszóma* egy DNS molekulát jelent. Azokat az előlényeket, melyek a genomot két példányban tartalmazzák, azaz minden kromoszóma kétszer szerepel, *diploid* egyedeknek, míg a genomot egy példányban tartalmazó egyedeket *haploid* egyedeknek nevezzük. Az ember sejtmagja például 23 pár kromoszómát tartalmaz: 22 pár autoszómát és egy X vagy Y kromoszómapárt. Egy normális nő sejtmagja 22 pár autoszómát és egy pár X kromoszómát, míg egy normális férfié ugyancsak 22 pár autoszómát és egy XY kromoszómapárt tartalmaz.

A *gének* a kromoszómákon, azaz a DNS molekulákon helyezkednek el, és a kromoszóma egy bizonyos részét jelentik. A gének határozzák meg az egyed tulajdonságait (pl. szőke, barna; kékszemű, zöldszemű, stb.). A gén tulajdonképpen csak egy kezdeti és egy végső pozíciót jelent, azaz csak a tulajdonság helyét határozza meg. Egy gén megjelenési formáit *alléloknak* nevezzük, vagy úgy is mondhatjuk, hogy az allél a gén *fenotípusa*.

Öröklődés alatt bizonyos tulajdonságok átvitelét értjük a szülőkről az

utódokra. Az öröklődés egységének a gént tekintjük. Kétféle öröklődésről beszélhetünk, attól függően, hogy a gének milyen típusú kromoszómán helyezkednek el.

Az autoszomális öröklődés felfedezője Gregor Mendel, egy Ágoston-redi szerzetes volt. Mendel Morvaországban született a XIX. században, amely akkor az Osztrák-Magyar Monarchia részét képezte. Az iskola elvégzése után a brünni Szent Tamás Ágoston-rendi kolostorba lépett be. A kolostor vezetői Bécsbe küldték megszerezni a tanítói bizonyítványt, de Mendel megbukott a vizsgákon, és visszatért Brünnbe. Ekkor kezdte el nyövénykeresztezési kísérleteit, amiért ma őt nevezzük a genetika, mint tudomány megalapozójának. Statisztikai adatokból kiindulva törvényszerűségeket fedezett fel a tulajdonságok átörökítésében a szülőtől a leszármazottra. A manapság is használt *domináns* és *recesszív* kifejezéseket Mendel használta először.

A nemi öröklődést Thomas Hunt Morgan kutatta, kinek eredményeit 1934-ben Nobel-díjjal jutalmaztak. A *crossover* (átkeresztezés) felfedezése is Morgan nevéhez fűződik. (A továbbiakban mindenhol az angol kifejezést fogom használni, mivel magyarul túlságosan hasonlít a keresztezés szóhoz, amely mást jelent, ráadásul 13 betűből áll.) Morgan szerint meióziskor (= csírasejtek osztódása) a homológ kromoszómák párba rendeződnek, valahol megtörnek, és részeket cserélnek ki egymás között úgy, hogy genetikai anyag nem veszlődik el.

A crossover mellett a másik fontos genetikai művelet a mutáció, amely az evolúciós változás alapvető forrását képezi. A mutáció a kromoszóma egy részének megváltozását jelenti. A DNS molekulák nem teljesen stabilak, vagyis minden bázispár rendelkezik egy bizonyos mutációs valószínűséggel. A mutáció kétféle lehet: spontán vagy előidézett. Az előbbi bekövetkezésének sokkal kisebb a valószínűsége, de evolúciós szempontból sokkal fontosabb a másodiknál. A változás milyenségét tekintve, sokféle mutáció létezik. Csak a pont-mutációkat figyelembe véve, azaz melyek egyetlen bázisban változtatják meg a DNS-szekvenciát, máris három típusról beszélhetünk: bázis-helyettesítéses, bázis-hozzáadásos, illetve bázis-törléses, és még ezek sem az osztályozási fa levelei! Azonban ne gondoljuk, hogy olyan egyszerű dolog mutációnak bekövetkeznie a génekben. A sejtek ugyanis bonyolult *repair-mechanizmusokkal* rendelkeznek, melyek megjavítják a sérült DNS-t, megakadályozva ezzel a változást. Míg crossover minden esetben történik, azaz valószínűsége 1, mutáció bekövetkezésének valószínűsége ennél nagyságrendekkel kisebb.

1.3. Az evolúció

A darwini evolúció, vagyis a természetes kiválasztódás vagy szelekció elmélete szerint egy populáció azon egyedei maradnak fenn nagyobb valószínűséggel, melyek

bizonyos környezeti feltételeket tekintve rátermettebbek a többieknél. Darwin nem tudott az öröklődési elvekről, viszont egy olyan öröklődést feltételezett, ahol az öröklött tulajdonságok a folyadékok elegyítéséhez hasonlóan keverednek az utódokban.

A természetes szelekciót Michalewicz a következő példával magyarázza el ([Mic96, p.14]):

> ,,[...] Vegyük például a nyulakat: minden időpillanatban létezik egy nyúlpopuláció. A nyulak közül néhány gyorsabb és okosabb mint a többi. Ezeket a gyors és okos nyulakat sokkal kisebb valószínűséggel kapják el és eszik meg a rókák, ezért sokuk fennmarad, hogy azt csinálhassa, amit a nyulak a legjobban tudnak: még több nyulat. Természetesen néhány lassúbb és butább nyúl is fenn fog maradni, kizárólag azért, mert szerencsések. A rókákat túlélő nyúlpopuláció elkezd szaporodni. Ez a nyulak genetikai anyagának keverékéhez vezet: néhány lassú nyúl gyors nyúllal párosodik, néhány gyors nyúl gyors nyúllal, néhány okos nyúl buta nyúllal, és így tovább. És mindennek a tetejébe a természet néha bedob egy 'vadnyulat' a genetikus anyag mutációja révén. A származó kisnyulak (átlagban) gyorsabbak és okosabbak lesznek, mint az előző populációban lévők, mivel a gyorsabb és okosabb szülők élték túl a rókákat. (Szerencsére a rókák is hasonló folyamaton mennek keresztül – máskülönben a nyulak túl gyorsakká és okosokká válnának ahhoz, hogy a rókák elkapják őket.)"

Az evolúció kezdetekor a Földön csak egysejtű élőlények voltak. Az evolúció következtében minden élőlény alapegysége a sejt. A vírusok persze nem sejtekből állnak, azonban fennmaradásuk függ a sejtektől, mert csak azokat ,,megfertőzve" képesek a reprodukcióra. Mára a létező fajok száma 5 és 50 millió közé tehető. Érdekes megjegyeznünk, hogy a legnépszerűbb gerincesek (halak, hüllők, kétéltűek, madarak, emlősök) ezeknek körülbelül 3%-át teszik ki.

A nemi rekombináció, azaz a párosodás feltehetőleg szükséges előfeltétele volt a többsejtű élőlények megjelenésének. Vagyis a párosodás ,,feltalálása" a genetikai kódok kombinációjának nagyobb sikere miatt történt.

2

A kanonikus genetikus algoritmus

A ,,genetikus algoritmusok" elnevezés elsőre furcsának tűnhet az olvasónak. Az elnevezés abból adódik, hogy ezek az algoritmusok megpróbálják szimulálni azt a folyamatot, ami a természetben játszódik le: tulajdonságok átvitele a szülőről az utódra, ezen tulajdonságok kereszteződése, változása és természetes kiválasztódás. Először az alapalgoritmus kerül bemutatásra, majd később látni fogjuk, hogy ezek valós matematikai alapokra épülnek és valóban hatékonyak (a hatékonyságról bővebben ugyancsak később lesz szó).

Néhány további egyszerű fogalom értelmezése, melyekkel a továbbiakban sokszor fogunk találkozni: Egy populációt egyedek alkotnak, az egyedeket pedig a kromoszómájuk határozza meg. A kromoszómák lineáris szekvenciába rendezett génekből épülnek fel. A keresztezés egysége a rekon, a változékonyság, mutáció egysége a muton.

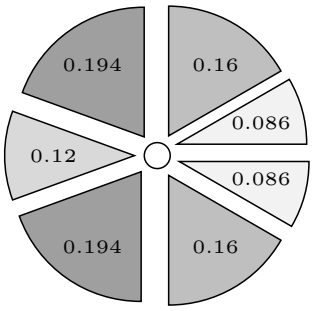
A genetikus algoritmusokkal való foglalkozás kezdetei az 1950-es évek elejére tehető, mikor néhány biológus a számítógép segítségével biológiai rendszereket próbált szimulálni. A tulajdonképpeni genetikus algoritmus John Holland nevéhez fűzodik, aki munkásságával az egész világ számára ismertté tette azt. Ezek az algoritmusok a megoldás bináris stringként való reprezentációjára és a Holland által felállított sémaelmélet (Schema Teorem) alapjaira épülnek.

A genetikus algoritmusok a probabilisztikus algoritmusok osztályába tartoznak, a determinisztikus és sztochasztikus keresés elemeit egyesítik, ezért sokkal hatékonyabbak, mint a determinisztikus keresési metódusok. Olyan sztochasztikus algoritmusok, melyek keresési metódusai egy biológiai jelenséget modelleznek. Globális optimizáló algoritmusok, melyek egy kezdeti populációból kiindulva – mely potenciális megoldásokat tartalmaz – eljutnak egy végső populációhoz, melyben az egyedek értéke, melyet a rátermettségi függvény fog megadni, az optimumhoz fog konvergálni. A megoldást kódolni kell annak érdékében, hogy majd a genetikus operátorok alkalmazásával (keresztezés, mutáció) újabb egyedek "születhessenek". Minden megoldáshoz hozzárendelünk egy sorozatot, mely az illető megoldást fogja kódolni. Ez a sorozat lesz a megoldás genotípusa, míg maga

a megoldás a fenotípus. A sorozat lesz az egyed kromoszómája és a benne levő betűk, számok a genéket reprezentálják. (A genetikus algoritmusok csak haploid egyedekkel foglalkoznak, vagyis minden egyed egy kromoszómával rendelkezik, és az egyedeket egyedül a kromoszómájuk határozza meg, vagyis az egyed ekvivalens a kromoszómával. Ezért sok esetben az egyed helyett kromoszómát mondunk vagy fordítva.)

Az említett kezdeti populációból kiválogatunk bizonyos egyedeket valamilyen feltétel szerint, majd ezekből egy új populációt hozunk létre. Ebben az új populációban egyes egyedpárokat keresztezünk, egyes egyedeket pedig mutációnak vetünk alá. A keresztezés a következő módon történik: legyen $(a_1, a_2, a_3, a_4, a_5)$ és $(b_1, b_2, b_3, b_4, b_5)$ két egyed, melyet kiválasztottunk keresztezésre, a keresztezés egysége (rekon) pedig legyen 2. Ekkor a keletkezett két leszármazott $(b_1, b_2, a_3, a_4, a_5)$ és $(a_1, a_2, b_3, b_4, b_5)$ lesz.
A mutáció a kromoszómán belül egy vagy több gént megváltoztat, vagyis mutál. Legyen $(a_1, a_2, a_3, a_4, a_5)$ egy egyed a populációból, melyet kiválasztottunk mutációra, a mutáció egysége (muton) legyen 1, a mutáció poziciója 3. Ekkor az $(a_1, a_2, a_3^*, a_4, a_5)$ leszármazottat kapjuk.
A keresztezés és mutáció befejeztével megkaptuk az új populációt. Most az új populáció lesz a kezdeti populáció, és újból elvégezzük a fent leírt algoritmust. Ezt addig ismételjük, amíg a megállasi feltétel nem teljesül, vagy előre meghatározott k számszor hajtjuk végre. (Az utóbbi esetben k-t a generációk számának nevezzük.)
A keresztezés alkalmazásának célja az, hogy egyes egyedek tulajdonságait ötvözve újabb, esetleg rátermettebb egyedeket hozzunk létre, míg a mutáció célja az egyedek, a populáció frissítése, vagyis változatosság bevitele a populációba. A keresztezés és a mutáció végrehajtásához szükségünk van két valószínűségi értékre: a keresztezés és a mutáció valószínűségére, azért, hogy az egyedeknek csak bizonyos százalékát vessük alá ezen genetikai "műveleteknek".

A legismertebb algoritmus a *rulettkerék* módszer. Nevét a szerencsejátékról kapta, és a módszert úgy kell elképzelnünk mint egy rulettkereket, ahol a szeletek mérete arányos az egyedek rátermettségével. Ezért az ebben az algoritmusban használt szelekciós metódust rátermettség-arányos szelekciónak nevezzük. Az algoritmus leírásához szükségünk van egy rátermettségi függvényre és a már említett két valószínűségi értékre. Legyen $P = \{v_1, v_2, \ldots, v_n\}$ populáció, ahol n a populáció mérete, valamint v_i, $i = 1, 2, \ldots, n$ az egyedek. Legyen továbbá $f(v_i)$ a v_i egyed rátermettségi értéke, ahol $f()$ a rátermettségi függvény.



2.1. ábra. **Egy képzeletbeli rulettkerék.** A szeletek mérete (körívek hossza, körcikkek területe) változó, vagyis annak valószínűsége, hogy a golyó egy adott szeletben álljon meg, arányos a szelet területével. A szeletekben a valószínűségek láthatók.

A kanonikus genetikus algoritmus (CGA):

1. Határozzuk meg a P populáció rátermettségét:

$$F = \sum_{i=1}^{n} f(v_i)$$

2. Minden egyedre számoljuk ki a kiválasztási valószínűséget:

$$p_i = f(v_i)/F, \quad i = 1, 2, \ldots, n$$

3. Minden egyedre kiszámoljuk a kumulatív kiválasztási valószínűséget:

$$q_i = \sum_{j=1}^{i} p_j, \quad i = 1, 2, \ldots, n$$

4. Minden egyedre a P populációból:

   (a) Generálunk egy valós véletlen számot: $r \in [0, 1]$

   (b) Ha $r < q_1$, akkor kiválasztjuk v_1-et az új P' populációba;
   Különben, ha $q_{i-1} < r \leq q_i$, akkor kiválasztjuk v_i-t az új P' populációba

5. Minden egyedre az új P' populációból:

   (a) Generálunk egy valós véletlen számot: $r \in [0, 1]$

   (b) Ha $r < p_c$, akkor kiválasztjuk az aktuális egyedet keresztezésre (p_c a keresztezési valószínűség)

6. Legyen c a keresztezésre kiválasztott egyedek száma. Ha c páratlan, akkor a kiválasztottak közül véletlenszerűen kiejtünk egy egyedet.

7. Minden keresztezésre kiválasztott egyedpárra:

   (a) Generálunk egy pozitív egész véletlen számot: $r \in \{1, 2, \ldots, m-1\}$, ahol m a kromoszóma hossza, azaz a gének száma.

   (b) Az r szám lesz a rekon, és elvégezzük a keresztezést.

   (c) A két kiválasztott szülő helyére a két leszármazott kerül az új P' populációban.

8. Minden egyed (kromoszóma) minden génjére az új P' populációban:

   (a) Generálunk egy valós véletlen számot: $r \in [0, 1]$

   (b) Ha $r < p_m$, a gént mutáljuk (p_m a mutáció valószínűsége; muton=1)

9. Ha a megállási feltétel teljesül, akkor STOP;
   Különben $P := P'$ és visszaugrunk az 1. pontra.

Algoritmusunk röviden a következőképpen vázolható:

```
Létrehozzuk a kezdeti P populációt
Ismételd:
   Szelekció: P → P'
   Keresztezés, mutáció P' populációban
   P := P'
Amíg a megállási feltétel teljesül.
```

A megállás után a végső populáció legrátermettebb egyede jó megközelítését fogja adni az általunk keresett optimumnak. Ezzel a módszerrel az algoritmus véges számú iteráció után a globális optimumhoz fog konvergálni függetlenül a kezdeti populációtól. A kísérletek azt igazolták, hogy hogy a kezdeti populáció különböző megválasztása csak a konvergencia sebességét befolyásolja.
A tulajdonképpeni genetikus algoritmus fogalmához szervesen kapcsolódott a bináris string reprezentáció, azért, hogy a kromoszómák és gének modellezése szemléletes, valósághű legyen. De ezen reprezentációnak hátulütői is vannak: nagy keresési terület és nagy pontosság esetén óriásira nőhet egy kromoszóma hossza, ami próbára teszi a számítógép memóriakészletét. Az oda-vissza

kódolás következtében az algoritmus veszít gyorsaságából. Ezért bevezették a lebegőpontos ábrázolási módot. A kísérletek során a lebegőpontos ábrázolással jobb eredményeket értek el, és nagy keresési terület illetve pontosság esetén is jelentős gyorsaságbeli különbség volt tapasztalható a két reprezentáció között.

A genetikus algoritmusok tehát problémafüggetlen globális optimizáló algoritmusok, melyek biológiai, evolúciós folyamatokat modelleznek ugyan, viszont nem jellemző rájuk a "természetes" szó.

3

A genetikus algoritmusok konvergenciája

3.1. Sémaelmélet

A sémaelmélet a genetikus algoritmusok működésére próbál magyarázattal szolgálni. Ebben a részben ezzel az elméleti háttérrel ismerkedünk meg.

A sémaelméletben az ábrázolás bináris, a szelekció arányos, a keresztezés pedig egypontos.

3.1. definíció (Séma). *Egy olyan speciális kromoszámát, amely a 0 és 1 szimbólumokon (értékeken) kívül még a "*" szimbólumot is tartalmazhatja, sémának nevezzük, a "*" szimbólumot pedig* nemtörődöm *(angolul* don't-care*) szimbólumnak nevezzük.*

Tehát egy nemtörődöm szimbólum helyére akár 0, akár 1 is kerülhet. Egy séma több kromoszómára is illeszkedhet, mégpedig egy k nemtörődöm szimbólumot tartalmazó kromoszóma 2^k kromoszómára illeszkedik. A definícióból az is következik, hogy minden kromoszóma egyben séma is, mely nem tartalmaz "*"-ot, és minden ilyen séma egyetlen kromoszómára illeszkedik, mégpedig saját magára.
Például az $S_1 = (101{*}{*}101{*}{*})$ séma többek között az (1011110111) kromoszómára is illeszkedik.

3.2. definíció (Séma rendje). *Egy séma rögzített bitjeinek számát a séma rendjének nevezzük és S séma esetén $o(S)$ szimbólummal jelöljük.*

A fennebb látható példa esetében $o(S_1) = 6$.

3.3. definíció (Séma definíciós hossza). *Egy séma első és utolsó rögzített bitje közötti távlságot a séma definíciós hosszának nevezzük, és S séma esetén $\delta(S)$ szimbólummal jelöljük.*

Ugyancsak a fenti példa esetében $\delta(S_1) = 7$.

3.4. definíció (Séma rátermettségi függvényértéke). *Egy séma rátermettségi függvényértéke azon kromoszómák rátermettségi függvényértékenek átlaga, amelyekre a séma illeszkedik:*

$$f(S,t) = \sum_{i=1}^{p} f(\nu_i)/p,$$

hogyha S *séma* p *darab kromoszómára illeszkedik (*t *időpontban, illetve a* t*-edik generációban).*

A szelekció hatása a sémára

Ebben a részben a szelekció hatását vizsgáljuk meg a sémákra, vagyis, hogyan változik a szelekció kövekeztében azon kromoszómák száma, melyekre egy bizonyos séma illeszkedik.

Legyen $\xi(S,t)$ azon kromoszómák száma, melyekre az S séma t időpillanatban illeszkedik. Egy általános kromoszóma kiválasztási valószínűsége, melyre az S séma illeszkedik ($F(t)$ a populáció (össz)rátermettsége):

$$\frac{f(S,t)}{F(t)}$$

Mivel n kiválasztás történik (n a populáció mérete), felírhatjuk a következő képletet:

$$\xi(S,t+1) = \xi(S,t) \cdot n \cdot f(S,t)/F(t)$$

Jelöljük $\overline{F}(t)$-vel a populáció átlagrátermettségét, azaz

$$\overline{F}(t) = F(t)/n$$

Fenti képletünk így a következőképpen módosul,

$$\boxed{\xi(S,t+1) = \xi(S,t) \cdot f(S,t)/\overline{F}(t)}$$

Ezt az összefüggést reproduktív séma-növekedési egyenletnek nevezzük.

Azt a sémát, melynek rátermettségi függvényértéke nagyobb mint a populáció átlagrátermettsége, *átlagon felüli* sémának nevezzük. Ugyanígy, ha egy séma rátermettségi függvénye kisebb (vagy egyenlő) mint a populáció átlagrátermettsége, akkor a sémát *átlagon aluli* sémának nevezzük. Egy séma rátermettségi függvényértéke felírható a következőképpen:

$$f(S,t) = \overline{F}(t) + \varepsilon \cdot \overline{F}(t),$$

ahol $\varepsilon \geq -1$ valós szám. Átlagon felüli séma esetén $\varepsilon > 0$, átlagon aluli séma esetén pedig $\varepsilon \in [-1, 0]$. Az előbbi képletet a séma-növekedési egyenletbe helyettesítve kapjuk, hogy

$$\xi(S, t+1) = \xi(S, t) \cdot (1 + \varepsilon),$$

ahonnan

$$\xi(S, t) = \xi(S, 0) \cdot (1 + \varepsilon)^t$$

Ez a képlet azt jelenti, hogy ha az S séma átlagon felüli, akkor azon kromoszómák száma, melyekre S illeszkedik, exponenciálisan növekszik az egymást követő generációkban.

A crossover hatása a sémára

Ha egy kromoszóma hossza m, akkor $m - 1$ egypontos keresztezési pozíció lehetséges. Így felírhatjuk, hogy az S séma elhalálozási valószínűsége (angolul *probability of destruction*)

$$p_d(S) = \frac{\delta(S)}{m-1},$$

vagyis a túlélés valószínűsége (*probability of survival*)

$$p_s(S) = 1 - \frac{\delta(S)}{m-1}$$

Képletünk így még nem igazán helyes, mivel nem vettük figyelembe a p_c keresztezési valószínűséget. Ezt figyelembe véve végső képletünk a következő lesz,

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1}$$

Azért szerepel "$\geq$" jel az egyenlőség helyett, mert ha rögzített pozíciók között is történik a keresztezés, van esély a túlélésre. Tekintsük például a következő esetet: Legyen $S = (**1*0*)$, amely a $\nu_1 = (011100)$ kromoszómára illeszkedik, a keresztezés ν_1 és $\nu_2 = (101011)$ kromoszóma között megy végbe, továbbá legyen a keresztezési pozíció 4. Ekkor $\nu_1' = (101000)$, $\nu_2' = (011110)$. Láthatjuk, hogy S ugyancsak illeszkedik ν_1'-re, mivel S első fele illeszkedett ν_2 első felére.

Figyelembe véve a séma túlélési valószínűségét, séma-növekedési egyenletünk a következőképpen módosul,

$$\xi(S, t+1) \geq \xi(S, t) \cdot \left[f(S, t)/\overline{F}(t)\right] \cdot \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right]$$

A mutáció hatása a sémára

Egy séma akkor fogja túlélni a mutációt, azaz akkor fog illeszkedni a kromoszómákra a mutáció után, ha a rögzített pozíciókon lévő elemek változatlanok maradnak. Egy gén túlélési valószínűsége $1 - p_m$. Minden egyes génre el kell döntsük, hogy megváltoztatjuk vagy változatlanul hagyjuk. Ezek tehát független valószínűségek, így ezek szorzatára van szükségünk, azaz egy séma túlélési valószínűsége

$$p_s(S) = (1 - p_m)^{o(S)}$$

lesz. Newton binomiális képletét alkalmazva felírhatjuk, hogy

$$(1 - p_m)^{o(S)} = 1 + C^1_{o(S)} \cdot 1 \cdot (-p_m) + C^2_{o(S)} \cdot 1 \cdot (-p_m)^2 + \dots$$

Mivel $p_m \in (0, 1)$ és $p_m \ll 1$, ha egy megközelítő összefüggést szeretnénk kapni p_m-re, elég figyelembe vennünk az összeg első két tagját, úgy érvelve, hogy egy nagyon kicsi szám négyzete már tényleg nagyon kicsi szám lesz, stb. Így

$$p_m \approx 1 - o(S) \cdot p_m$$

Mostmár felírhatjuk végső séma-növekedési egyenletünket,

$$\xi(S, t+1) \approx \xi(S, t) \cdot \left[f(S, t)/\bar{F}(t)\right] \cdot \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right] \cdot [1 - o(S) \cdot p_m]$$

3.1. tétel (Séma tétel). *A rövid, kisrendű, átlagon felüli sémák exponenciálisan növekvő részt kapnak a genetikus algoritmusok egymást követő generációiban.*

Rövid sémának azt a sémát nevezzük, melynek definíciós hossza rövid, míg kisrendűnek azt, melyben a rögzített bitek száma kicsi.

$$\xi(S, t) \approx \xi(S, 0) \cdot (1 + \varepsilon)^t \cdot \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right] \cdot [1 - o(S) \cdot p_m]$$

1. Hipotézis (Építőkocka (*Building Block*) hipotézis). *A genetikus algoritmusok rövid kisrendű, átlagon felüli (=építő) sémákon keresztül kvázi-optimális (megközelítően optimális) eredményt szolgáltatnak.*

3.2. A genetikus algoritmusok mint Markov-láncok

A kanonikus genetikus algoritmus (bináris ábrázolás, arányos szelekció, egypontos crossover) (*Canonical Genetic Algorithm – CGA*) algoritmusa a következőképpen is felírható:

kezdeti populáció kiválasztása

szelekció

ismételd

crossover

mutáció

szelekció

amíg a leállási feltétel nem teljesül

Jelöljük n-nel a populáció méretét, valamint legyen l egy kromoszóma hossza. Tekintsünk egy populációt egy állapotnak. Mivel egy populációban n darab l hosszúságú bináris sztring szerepel, egy populáció hossza $n \cdot l$. Tehát az állapottér, $S = \{0,1\}^{n \cdot l}$ mérete $2^{n \cdot l}$, mivel $2^{n \cdot l}$ $n \cdot l$ hosszúságú különböző bináris sztring létezik. A műveleteket (crossover, mutáció, szelekció) pedig tekinthetjük egy sztochasztikus átmenetnek egy másik állapotba, ezért a CGA értelmezhető Markov-láncként, mely átmenetmátrixának mérete $2^{n \cdot l} \times 2^{n \cdot l}$.
Legyen $\pi_k(i)$ egy olyan függvény, amely visszaadja az i-edik állapot k-adik egyedét. A továbbiakban a műveletek átmenetmátrixait vizsgáljuk.

Crossover

Vizsgáljuk meg azt az esetet, amikor a crossover nem változtatja meg a populációt, azaz a crossover előtti és utáni populáció egyedei azonosak. Ez három esetben következhet be. Legyen c a rekombinációra kiválasztott egyedek száma.

(a) $c = 0$

(b) $c = 1$

(c) Bizonyos egyedek rekombinálódnak, de a populáció ennek ellenére változatlan marad.
Például:

$$\begin{array}{cc} \mathbf{P} & \mathbf{P'} \\ \left.\begin{array}{l} 01|01 \\ 10|11 \end{array}\right\} & \rightarrow \left\{\begin{array}{l} 1001 \\ 0111 \end{array}\right. \\ \left.\begin{array}{l} 01|11 \\ 10|01 \end{array}\right\} & \rightarrow \left\{\begin{array}{l} 1011 \\ 0101 \end{array}\right. \end{array}$$

Felírhatjuk a következő valószínűségeket:

$$\begin{aligned}
P\{c = 0\} &= p_c^0 \cdot (1 - p_c)^n = (1 - p_c)^n \\
P\{c = 1\} &= p_c \cdot (1 - p_c)^{n-1} \\
P\{c = 0 \text{ vagy } c = 1\} &= (1 - p_c)^n + p_c \cdot (1 - p_c)^{n-1} = \\
&= (1 - p_c)^{n-1} \cdot (1 - p_c + p_c) = (1 - p_c)^{n-1}
\end{aligned}$$

Figyelembe véve, hogy a (c) eset is bekövetkezhet, felírhatjuk, hogy

$$P\{pop_i \to pop_i\} \geq (1 - p_c)^{n-1} > 0$$

Mutáció

Annak valószínűsége, hogy a mutáció egy egyedet egy meghatározott módon megváltoztat a következőképpen írható fel:

$$P\{\pi_k(i) \to \pi_k(j)\} = p_m^{H(\pi_k(i),\pi_k(j))} \cdot (1 - p_m)^{l - H(\pi_k(i),\pi_k(j))},$$

ahol a $H(\cdot, \cdot)$ függvény két bináris sztring közötti Hamming-távolságot adja meg. Például

$$P\{1011 \to 0001\} = p_m \cdot (1 - p_m) \cdot p_m \cdot (1 - p_m) = p_m^2 \cdot (1 - p_m)^2$$

Ezt ugyanilyen módon felírhatjuk két populáció között:

$$P\{pop_i \to pop_j\} = p_m^{H(pop_i,pop_j)} \cdot (1 - p_m)^{n \cdot l - H(pop_i,pop_j)} > 0$$

Ez azt jelenti, hogy a mutáció átmenetmátrixa szigorúan pozitív.

Szelekció

Rátermettségi függvényünk

$$f : \{0, 1\}^l \to \mathbf{R}_+$$

Annak valószínűsége, hogy egy egyed átkerüljön a következő generációba (szelekciós valószínűség)

$$P\{\pi_k(i) \to \pi_k(i)\} = \frac{f(\pi_k(i))}{\sum_{k=1}^{n} f(\pi_k(i))}$$

Annak valószínűsége, hogy minden egyed átkerüljön a következő generációba, azaz kiválasztódjon ezen valószínűségek szorozata minden egyedre, vagyis

$$P\{pop_i \to pop_i\} = \frac{\prod_{k=1}^{n} f(\pi_k(i))}{[\sum_{k=1}^{n} f(\pi_k(i))]^n} > 0$$

Az együttes átmenetmátrix felírható úgy mint

$$\mathbf{P} = \mathbf{C} \cdot \mathbf{M} \cdot \mathbf{S},$$

ahol $\mathbf{C}$, $\mathbf{M}$ és $\mathbf{S}$ rendre a crossover, mutáció, illetve szelekció átmenetmátrixai.

3.2. tétel. *A CGA* $\mathbf{P} = \mathbf{C} \cdot \mathbf{M} \cdot \mathbf{S}$ *átmenetmátrixa pozitív sztochasztikus mátrix, vagyis* $p_{ij} > 0, \ \forall i, j$.

BIZONYÍTÁS. Legyen $\mathbf{A} = \mathbf{C} \cdot \mathbf{M}$, $a_{i,j} = \sum_k c_{ik} \cdot m_{kj} > 0$, mert $c_{ii} > 0$ és $m_{ij} > 0$. Továbbá $\mathbf{P} = A \cdot S$, $p_{ij} = \sum_k a_{ik} \cdot s_{kj} > 0$, mert $s_{jj} > 0$ és $a_{ij} > 0$. ■

3.3. tétel. *Legyen* $\mathbf{P}$ *egy ML átmenetmátrixa,* $\mathbf{u}$ *pedig a kezdeti eloszlást reprezentáló valószínűségi vektor. Ekkor*

$$\mathbf{u}^{(n)} = \mathbf{u} \cdot \mathbf{P}^n \tag{a}$$

a lánc valószínűségét adja az n*-edik lépésben, vagyis egy valószínűségi vektort, amely megmondja, hogy a ML az* n*-edik lépésben melyik állapotban mekkora valószínűséggel található.*

3.5. definíció. *Egy ML* ergodikus*, ha bármelyik állapotból bármelyik állapotba eljuthatunk véges számú lépésben. (Nem feltétlenül egy lépésben.)*

3.6. definíció. *Egy ML* reguláris *ha átmenetmátrixának valamely véges számú hatványa szigorúan pozitív.*

3.4. tétel. *Legyen* $\mathbf{P}$ *egy reguláris ML átmenetmátrixa. Ha* $n \to \infty$*, akkor* $\mathbf{P}^n \to \mathbf{W}$*, ahol* $\mathbf{W}$ *minden sora ugyanazon* $\mathbf{w}$ *szigorúan pozitív valószínűségi vektorral egyenlő, azaz*

$$\mathbf{W} = \lim_{n \to \infty} \mathbf{P}^n \tag{b}$$

Az (a) és (b) összefüggésekből azonnali módon következik, hogy

$$\mathbf{u}^{(\infty)} = \lim_{n \to \infty} \mathbf{u} \cdot \mathbf{P}^n = \mathbf{u} \cdot \lim_{n \to \infty} \mathbf{P}^n = \mathbf{u} \cdot \mathbf{W}$$

Mivel $\mathbf{u} \neq 0$ – mert abban az esetben semmilyen állapotba sem juthatnánk el –, ezért $\mathbf{u}^{(\infty)} = \mathbf{u} \cdot \mathbf{W} > 0$, azaz $u_i^{(\infty)} > 0, \ \forall i$.

3.7. definíció. *Legyen* $Z_t = \max\left\{f(\pi_k^{(t)}(i))|k = 1,\dots,n\right\}$ *az* i*-edik állapot legjobb egyedének rátermettsége, illetve jelölje* $f^* = \max\{f(b)|b \in \{0,1\}^l\}$ *a függvény maximumát. Azt mondjuk, hogy egy genetikus algoritmus a globális optimumhoz konvergál ha*

$$\lim_{n\to\infty} P\{Z_t = f^*\} = 1$$

3.5. tétel. *A CGA nem konvergál a globális optimumhoz.*

BIZONYÍTÁS. Tekintsünk egy

$$Z_t = \max\left\{f(\pi_k^{(t)}(i))|k = 1,\dots,n\right\} < f^*$$

tulajdonságú i állapotot. Továbbá legyen $p_i^{(t)}$ annak a valószínűsége, hogy a genetikus algoritmus a t-edik időpillanatban ebben az állapotban van. Így

$$P\{Z_t < f^*\} = P\{Z_t \neq f^*\} = p_i^{(t)},$$

vagyis

$$P\{Z_t = f^*\} = 1 - p_i^{(t)},$$

ahonnan

$$\lim_{t\to\infty} P\{Z_t = f^*\} = 1 - p_i^{(\infty)} < 1$$

■

Annak ellenére, hogy a CGA nem konvergál a globális optimumhoz egyszerű változtatással konvergenssé tehetjük algoritmusunkat. Az eljáráson annyit változtatunk, hogy kibővítjük a populációt egy ún. szuper egyeddel (*super individual*), amely nem vesz részt az evolúciós folyamatban. Ez azt jelenti, hogy az állapottér egy egyeddel kibővül, vagyis számossága $2^{n\cdot l}$-ről $2^{(n+1)\cdot l}$-re nő.
Az i-edik állapot szuper egyedét a $\pi_0(i)$ függvény segítségével érhetjük el. A crossover, mutáció és szelekció átmenetmátrixát a következőképpen szerkesszük meg: egymás alá helyezzük azon állapotok átmeneti valószínűségeit, melyek ugyanazt a szuper egyedet tartalmazzák, a szuper egyedek pedig balról jobbra és fentről lefelé rátermettségük szerint csökkenő sorrendben szerepelnek. Így a crossover, mutáció és szelekció kibővített átmenetmátrixát a következőképpen írhatjuk fel:

$$\mathbf{C}^+ = \begin{pmatrix} \mathbf{C} & & & \\ & \mathbf{C} & & \\ & & \ddots & \\ & & & \mathbf{C} \end{pmatrix}; \quad \mathbf{M}^+ = \begin{pmatrix} \mathbf{M} & & & \\ & \mathbf{M} & & \\ & & \ddots & \\ & & & \mathbf{M} \end{pmatrix};$$

$$\mathbf{S}^{+} = \begin{pmatrix} \mathbf{S} & & & \\ & \mathbf{S} & & \\ & & \ddots & \\ & & & \mathbf{S} \end{pmatrix},$$

ahol $\mathbf{C}$, $\mathbf{M}$ és $\mathbf{S}$ $2^{n \cdot l} \times 2^{n \cdot l}$ méretű négyzetes mátrixok, melyek mindegyikéből 2^l darab található a megfelelő fenti mátrixokban.

A másolási műveletet egy $\mathbf{U}$ feljavító (*upgrade*) mátrix segítségével valósítjuk meg, amely egy állapotot, melyben a legrátermetteb egyed rátermettsége nagyobb mint a szuper egyedé, átváltoztat olyan állapottá, melyben a szuper egyed azonos a legrátermettebb egyeddel. Formálisan legyen $b = \operatorname{argmax}\{f(\pi_k(i)) | k = 1, \ldots, n\} \in \{0,1\}^l$ az i állapot legrátermettebb egyede figyelmen kívül hagyva a szuper egyedet. Ekkor $u_{ij} = 1$ ha $f(\pi_0(i)) < f(b)$, ahol $j = (b, \pi_1(i), \pi_2(i), \ldots, \pi_n(i)) \in S$, különben $u_{ii} = 1$. Ezért $\mathbf{U}$ felírható a következőképpen,

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_{1,1} & & & \\ \mathbf{U}_{2,1} & \mathbf{U}_{2,2} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{U}_{2^l,1} & \mathbf{U}_{2^l,2} & \cdots & \mathbf{U}_{2^l,2^l} \end{pmatrix},$$

ahol $\mathbf{U}_{\cdot,\cdot}$ $2^{n \cdot l} \times 2^{n \cdot l}$ méretű minormátrixok. Az $\mathbf{U}$ mátrix főátlója feletti elemek mind zérusok, mivel $\mathbf{U}$ csak rátermettebb szuper egyeddel rendelkező állapotba módosíthat egy állapotot, és az aktuálistól rátermettebbek attól balra találhatók. A könnyebb megértés érdekében tekintsük a következő példát: $f(x) = x$, $l = 1$, $n = 2$

szuper egyed		1				0			
	populációk	00	01	10	11	00	01	10	11
1	00	1							
	01		1						
	10			1					
	11				1				
0	00					1			
	01		1						
	10			1					
	11				1				

Nyilvánvaló, hogy $\mathbf{U}_{1,1} = \mathbf{I}$, mivel ezek az állapotok a legrátermettebb szuper egyedet, azaz a globális optimumhoz tartozó értéket tartalmazzák. Továbbá minden $\mathbf{U}_{i,j}$, $i > 1$, $j > 1$ minormátrix olyan egységmátrix, amely néhány zérust is tartalmaz a főátlón, természetesen csak abban az esetben ha a függvény egyetlen

globális optimummal rendelkezik. Ha többel, tegyük fel k globális optimummal, akkor

$$\mathbf{U}_{ii} = \mathbf{I},\ i = 1, \ldots, k$$
$$\mathbf{U}_{ij} = \mathbf{0},\ i = 2, \ldots, k,\ j = 1, \ldots, k-1,$$

minden más $\mathbf{U}_{\cdot,\cdot}$ mátrix legalább egy darab 1-est tartalmaz a főátlón úgy, hogy $\mathbf{U}$ minden sorában csak egy darab 1-es szerepel.
Az egyszerűség kedvéért tételezzük fel, hogy f-nek csak egy globális optimuma van. Az $\mathbf{U}$ mátrixszal történő módosítást a szelekció előtt kell, hogy elvégezzük, ezért $\mathbf{P}^+ = \mathbf{T}^+ \cdot \mathbf{U} \cdot \mathbf{S}^+$, ahol $\mathbf{T}^+ = \mathbf{C}^+ \cdot \mathbf{M}^+$. Így

$$\begin{aligned}
\mathbf{P}^+ &= \begin{pmatrix} \mathbf{T} & & & \\ & \mathbf{T} & & \\ & & \ddots & \\ & & & \mathbf{T} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{U}_{1,1} & & & \\ \mathbf{U}_{2,1} & \mathbf{U}_{2,2} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{U}_{2^l,1} & \mathbf{U}_{2^l,2} & \cdots & \mathbf{U}_{2^l,2^l} \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} \mathbf{S} & & & \\ & \mathbf{S} & & \\ & & \ddots & \\ & & & \mathbf{S} \end{pmatrix} = \\
&= \begin{pmatrix} \mathbf{T}\mathbf{U}_{1,1}\mathbf{S} & & & \\ \mathbf{T}\mathbf{U}_{2,1}\mathbf{S} & \mathbf{T}\mathbf{U}_{2,2}\mathbf{S} & & \\ \vdots & \vdots & \ddots & \\ \mathbf{T}\mathbf{U}_{2^l,1}\mathbf{S} & \mathbf{T}\mathbf{U}_{2^l,2}\mathbf{S} & \cdots & \mathbf{T}\mathbf{U}_{2^l,2^l}\mathbf{S} \end{pmatrix}
\end{aligned} \tag{3.1}$$

ahol $\mathbf{T} \cdot \mathbf{U}_{1,1} \cdot \mathbf{S} > 0$, mert $\mathbf{T} = \mathbf{C} \cdot \mathbf{M}$, $\mathbf{U}_{1,1} = \mathbf{I} \Rightarrow \mathbf{T} \cdot \mathbf{U}_{1,1} \cdot \mathbf{S} = \mathbf{C} \cdot \mathbf{M} \cdot \mathbf{S} > 0$. Legyen

$$\mathbf{R} = \begin{pmatrix} \mathbf{T}\mathbf{U}_{2,1}\mathbf{S} \\ \vdots \\ \mathbf{T}\mathbf{U}_{2^l,1}\mathbf{S} \end{pmatrix}$$

Láthatjuk, hogy $\mathbf{R} \neq \mathbf{0}$, mivel $\mathbf{T} > 0$ és $\mathbf{U}_{2,1} \cdot \mathbf{S} \neq \mathbf{0}$, mert $\mathbf{S}$ főátlóján csak zérótól különböző elemek vannak, $\mathbf{U}_{2,1}$ főátlóján pedig legalább egy darab 1-es található. Ugyanígy, a fennmaradó minormátrixokat $\mathbf{V}$-vel jelölve nyilvánvaló, hogy $\mathbf{V} \neq \mathbf{0}$.

3.8. definíció. *Egy Markov-láncot* reducibilis *Markov-láncnak nevezünk ha* $\mathbf{P}$ *átmenetmátrixa a következő alakra hozható,*

$$\mathbf{P} = \begin{pmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{R} & \mathbf{T} \end{pmatrix},$$

ahol $\mathbf{C}$ *és* $\mathbf{T}$ *négyzetes mátrixok.*

3.6. tétel. *Ha* $\mathbf{P}$ *egy reducibilis sztochasztikus mátrix,* $\mathbf{C}$ *egy* $m \times m$ *méretű primitív sztochasztikus mátrix, illetve* $\mathbf{R}, \mathbf{T} \neq \mathbf{0}$*, akkor*

$$\mathbf{P}^{\infty} = \lim_{k \to \infty} \mathbf{P}^{k} = \begin{pmatrix} \mathbf{C}^{k} & \mathbf{0} \\ \sum_{i=0}^{k-1} \mathbf{T}^{i}\mathbf{R}\mathbf{C}^{k-i} & \mathbf{T}^{k} \end{pmatrix} = \begin{pmatrix} \mathbf{C}^{\infty} & \mathbf{0} \\ \mathbf{R}^{\infty} & \mathbf{0} \end{pmatrix}$$

egy stabil sztochasztikus mátrix. Ha $\mathbf{u}$ *egy kezdeti eloszlásvektor, akkor*

$$\begin{aligned} &\mathbf{u}^{(\infty)} = \mathbf{u} \cdot \mathbf{P}^{\infty}, \\ &u_i^{(\infty)} > 0, \ 1 \leq i \leq m, \\ &u_i^{(\infty)} = 0, \ i > m \end{aligned}$$

Vagyis a mi esetünkben annak a valószínűsége, hogy az elitista kanonikus genetikus algoritmus $t \to \infty$ lépésben olyan állapotban legyen, ahol a szuper egyed nem a globális optimum egyenlő zéróval.
Mivel a bizonyítást előre elvégeztük nincs más hátra mint kijelentenünk tételünket:

3.7. tétel. *Az elitista CGA a globális optimumhoz konvergál.*

4

A genetikus algoritmusok konvergencia-ideje

5

Szelekció, keresztezés, mutáció

6

Osztályozás genetikus algoritmusokkal

7

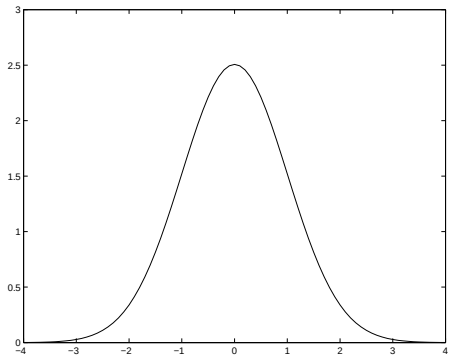
Párhuzamos genetikus algoritmusok

Magától értetődik, hogy a kezdeti populáció megválasztása nagyban befolyásolja a konvergencia sebességét. Ezért ha párhuzamosan több populációt tartunk fenn és bizonyos időközönként, hogy ne lassuljon lényegesen a folyamat, kicserélünk egyedeket a populációk között, a konvergencia idejének csökkenését várjuk el.

8

Más evolutív algoritmusok

8.1. Evolúciós stratégiák



8.1. ábra. Gauss-görbe, $y = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2)$, $\mu = 0$, $\sigma = 1$

8.1.1. Az (1,1)-ES konvergenciája

Az (1,1)-ES-kat a következő feltételek mellett tehetjük konvergenssé. Legyen $f : E \to \mathbb{R}$ függvény, $E \subset \mathbb{R}^N$. A feladat megtalálni f minimumát az E halmazon a következő feltételek mellett:

(i) az E keresési tér Lebesgue-mérhető

(ii) f mérhető függvény

(iii) f-nek létezik minimuma az E halmazon

(iv) ha $m(S)$ egy S halmaz Lebesgue-mértéke, akkor bármely $a > 0$ szám esetén

$$m(\{x|f(x) \leq \min_{y \in E} f(y) + a\}) > 0$$

8.1. definíció. *Egy $\widetilde{x}$ pontot az optimizálási (minimizálási) feladat megoldásának nevezünk, ha*

$$f(\widetilde{x}) \leq \min_{x \in E} f(x) + \epsilon,$$

ahol ϵ a hiba mértéke vagy hibaküszöb.

8.1. tétel (Konvergencia-tétel). *Az (1,1)-ES – az (i)–(iv) feltételeknek megfelelően – a globális optimumhoz konvergál.*

BIZONYÍTÁS. (A bizonyítás a [GZ01]-ből származik.)
Tételezzük fel, hogy a hibaküszöb ϵ, vagyis $\widetilde{x}$ a minimizálási probléma megoldása ha

$$f(\widetilde{x}) \leq \min_{x \in E} f(x) + \epsilon$$

■

8.2. Evolúciós programozás

8.3. Genetikus programozás

8.3.1. Lineáris genetikus programozás

9

Alkalmazások

9.1. Függvényoptimizálás kanonikus genetikus algoritmussal

Legyen az optimizálandó függvényünk

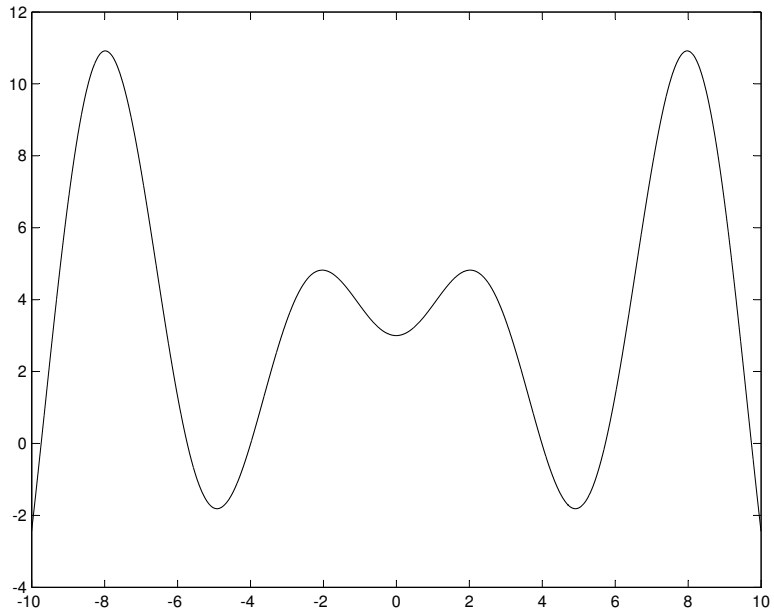
$$f(x) = x \cdot \sin x + 3 \tag{a}$$

melynek meg szeretnénk határozni a maximumát például a $[-10, 10]$ intervallumon. Azért szükséges egy *véges* intervallum megadása (a ,,véges" alatt itt mindössze azt értjük, hogy az intervallumnak van eleje és vége, egyszóval az intervallum határai véges számok), mert a végtelen nem ábrázolható a számítógép véges memóriájában. Természetesen a fenti intervallum is végtelen, viszont könnyen végessé alakítható, ha csak egy bizonyos pontossággal vizsgáljuk (például 4-tizedes pontossággal).

A kanonikus genetikus algoritmus rátermettség-arányos szelekciót használ, a rátermettség pedig, ha egy függvény maximumát szeretnénk meghatározni, akkor egyenlő a függvényértékkel, függvényminimum meghatározásakor pedig a függvényérték ellentettjével vagy reciprokával. Mivel a szelekciós valószínűségeket a rátermettségből számítjuk ki, a rátermettség, azaz a függvényérték pozitív kell legyen ahhoz, hogy érvényes valószínűségeket kapjunk. Ha függvényünk nem pozitív, akkor a pozitivitás biztosításának legegyszerűbb módja egy elégségesen nagy pozitív konstans hozzáadása a függvényértékhez (feltéve, hogy rendelkezünk ilyen információval), azaz

$$g(x) = f(x) + K,\ K > 0 \text{ úgy, hogy } g(x) > 0\ \forall x$$

(Itt a $g(x) > 0$ feltételben elvileg állhatna ,,$\geq$" is, viszont ez azt eredményezné, hogy a zérus rátermettségű egyedek sohasem kerülnek kiválasztásra, ami viszont befolyásolhatja az algoritmus konvergenciájának sebességét.)



9.1. ábra. Az $x \cdot \sin x + 3$ függvény grafikus képe a $[-10, 10]$ intervallumban

Ezután pedig $g(x)$-et tekintjük az optimizálandó függvénynek. Ha a keresett maximumot megtaláltuk – jelölje ezt x_{opt} –, akkor az $f(x_{\text{opt}}) = g(x_{\text{opt}}) - K$ kiszámítása által megkapjuk az *eredeti* függvény maximumát.

Tekintsük tehát az (a) optimizálandó függvényt. Tudjuk, hogy x minimuma a $[-10, 10]$ intervallumban -10, míg $\sin x$-é -1. Némi logikával kiszámíthatjuk a függvény minimumának egy alsó értékét úgy, hogy tekintjük (a) komponensenkénti minimumát, majd ezeket összevonjuk. Nézzük a jelenlegi függvényt. Tudjuk, hogy x minimuma a $[-10, 10]$ intervallumon -10, míg $\sin x$-é -1. Viszont e két komponens között szorzás van, ezért az összevont minimumot akkor kapjuk, ha az előjelek különböznek, vagyis $x \cdot \sin x$ minimuma -10. A 3 „függvény” minimuma természetesen 3, ezért, a két értéket összeadva kapjuk, hogy $-K = -7$, azaz $K = 7$. Így a $g(x) = x \cdot \sin x + 10$ függvény most feltételezhetően pozitív, viszont nem vagyunk biztosak a szigorú pozitivitásban (természetesen szigorúan pozitív, de feltételezzük, hogy ezt nem tudjuk). Ezért ha $g(x)$-hez egyszerűen hozzáadunk egy tetszőleges pozitív értéket, akkor ily módon az eredő függvény szigorúan pozitív lesz az adott intervallumon. Ennélfogva optimizálandó függvényünk *végső* alakja:

$$h(x) = x \cdot \sin x + 11$$

Miért szükséges a szigorú pozitivitás, miért nem elegendő a pozitivitás biztosítása? Ha rátermettség-arányos szelekcióval dolgozunk, akkor a rátermettségi függvényünk $f(\cdot)/F$ alakú, ahol f az optimizálandó függvény, vagy annak a fentiekhez hasonló transzformációja, míg F egy normalizáló tényező, azaz a populáció

rátermettsége (lásd az x. fejezetet). Ha $f(\cdot) \geq 0$, akkor lesz egy (vagy több) olyan pont, ahol a függvény, illetve a rátermettségének értéke egyenlő lesz zérussal. Ha ezt engedélyezzük, akkor ezek az egyedek sohasem fognak részt venni a populáció fejlődésében, és mi ezt nem akarjuk, mivel lehet, hogy egy zérus és egy zérusnál nagyobb rátermettségű egyed rekombinációjából a szülőknél rátermettebb egyedet vagy egyedeket kapunk. Tehát ez befolyásolhatja az algoritmus konvergenciáját vagy konvergenciájának sebességét, ezért mindig biztosítsuk a szigorú pozitivitást.

A következő lépésben tegyük végessé a $[-10, 10]$ intervallumot, azaz válasszunk egy elégséges pontosságot. Legyen ez most 4. Ez tulajdoképpen azt jelenti, hogy az egységnyi $[k, k+1]$ intervallumot, $k \in \mathbb{Z}$, $4 \cdot 10$-részre kell osszuk, vagyis a $[-10, 10]$-intervallumot $(10-(-10)) \cdot 40 = 800$ részre. Egy $[-10, 10]$ intervallumbeli számot 4-tizedes pontossággal b biten tudunk ábrázolni,

$$2^{b-1} < 800 \leq 2^b - 1,$$

ahonnan $b = 10$, mivel $2^9 = 512 < 800 \leq 2^{10} - 1 = 1023$. Azaz algoritmusunkban a kromoszómák hossza 10 lesz. Viszont ahhoz, hogy egy egyed rátermettségét ki tudjuk számítani, szükségünk lesz egy olyan függvényre, mely egy 10 hosszúságú bitsztringet a $[-10, 10]$ intervallumba képez. Az átalakító függvényhez a következő lépéseken keresztül juthatunk. Általánosan, ha adott egy b hosszúságú x bitsztring, melyet az $[i_1, i_2]$ intervallumba akarunk leképezni, akkor

$$0 \leq \mathrm{dec}(x) \leq 2^b - 1,$$

ahol $\mathrm{dec}(x)$ az x bitsztring 10-es számrendszerbeli értékét adja. Osszuk el az összefüggést $(2^b - 1)$-gyel, majd szorozzuk be $(i_2 - i_1)$-gyel. Így

$$0 \leq \frac{\mathrm{dec}(x)}{2^b - 1} \cdot (i_2 - i_1) \leq i_2 - i_1$$

Most adjuk az összefüggéshez i_1-et, ahonnan

$$i_1 \leq \frac{\mathrm{dec}(x)}{2^b - 1} \cdot (i_2 - i_1) + i_1 \leq i_2$$

Így a középső összefüggés pontosan a keresett átalakító függvény lesz, vagyis

$$\mathrm{alakít}(x) = \frac{\mathrm{dec}(x)}{2^b - 1} \cdot (i_2 - i_1) + i_1$$

Legyen például $e_1 = 1010001111$, ahonnan $\mathrm{dec}(e_1) = 655$, $\mathrm{alakít}(e_1) = \frac{655}{1023} \cdot 20 - 10 = 2.8054 \in [-10, 10]$. Ha még most is akad aki kételkedik az alakít függvény helyességében, vizsgáljuk meg a legkisebb illetve a legnagyobb 10-es hosszúságú bináris számot (sztringet):

$$e_2 = 00\ldots0$$
$$e_3 = 11\ldots1$$

Elvégezve a műveleteket kapjuk, hogy

$$\text{alakít}(e_2) = \frac{0}{1023} \cdot 20 - 10 = -10$$

$$\text{alakít}(e_3) = \frac{1023}{1023} \cdot 20 - 10 = 10$$

Egy egyed rátermettségét a következőképpen számíthatjuk ki:

$$\text{rátermettség}(e) = h(\text{alakít}(e))$$

A szelekció típusának megválasztását az olvasóra bízzuk. A mutációt és a crossovert úgy kell definiálnunk, hogy csakis érvényes egyedeket eredményezzenek (ebben az esetben minden 10 hosszúságú bináris sztring érvényes egyed). Használhatjuk a kanonikus operátorokat, azaz az állandó valószínűségű, független génmutációt és az egypontos crossovert, azonban ekkor a konvergencia biztosításához elitista szelekciót kell használnunk.

9.1.1. A Gray-kód

A Gray-kódokat a Bell Labsnél dolgozó Frank Gray-ről nevezték el, aki analóg jelek bináris kódba való átalakítását valósította meg ezek által. Az ilyen típusú kódokat nem ő fedezte fel, ő csak magát az eljárást találta ki, majd védte le.

A Gray-kódok azzal az érdekes tulajdonsággal rendelkeznek, hogy két egymás melletti természetes szám Gray-kódjainak Hamming-távolsága 1, vagyis az egymás melletti számok 1 bitben különböznek. A genetikus algoritmusokban való használatuk azért vált elterjedtté, mert gyorsabb kovergenciát biztosítanak. A mutáció azt jelenti, hogy egy (bináris) egyed egyik bitjét megváltoztatjuk, azaz az aktuális bitet értékének komplementere lesz az új bit. Tehát a keresési térben az előző pozíciótól való eltávolodás arányos a mutációk számával. Ha a mutációt a bináris sztringeken végezzük, akkor ez csak ritkán fog arányos elmozdulást eredményezni. A keresztezés is előnyösebb Gray-kódokkal, ezt az alábbi példa szemlélteti (ref. Haupt, section 5.4). Tekintsük az alábbi két egyedet:

$$p_1 = 100|00000 \quad (= 128)$$

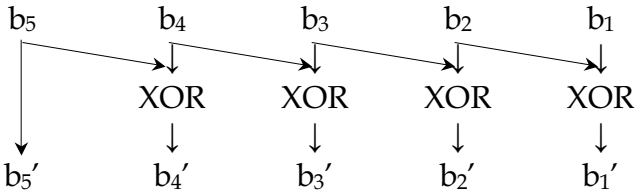
$$p_2 = 011|11111 \quad (= 127)$$

A függőleges vonal a keresztezési pontot jelöli. Az egyedek rátermettsége legyen egyszerűen bináris kódjuk decimális értéke (zárójelben ezek láthatók). Elvégezve a keresztezést az alábbi leszármazottakat kapjuk:

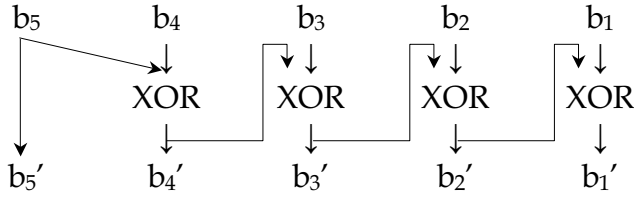
$$o_1 = 100|11111 \quad (= 159)$$

$$o_2 = 011|00000 \quad (= 96)$$

Láthatjuk, hogy a szülők rátermettsége rendre 128 és 127, míg a leszármazottaké 159 és 96. A keresztezés – definíció szerint – a szülők tulajdonságait egyesíti és a leszármazottak nagy valószínűséggel rátermettebbek lesznek szüleiknél. A leszármazottak rátermettsége nagyon eltér a szülők rátermettségétől és két irányban is változik, és nem is kis mértékben. Erre a problémára ugyancsak megoldást szolgáltatnak a Gray-kódok. A bináris sztringek Gray-kódba való alakítása, illetve ezek visszaalakítása bináris sztringgé a 9.2. és 9.3. ábrákon látható.



9.2. ábra. Bináris szám Gray-kódba való átalakítása



9.3. ábra. A Gray-kód visszaalakítása bináris számra

Tekintsük p_1 és p_2 Gray-kódját:

$$G(p_1) = 110|00000$$

$$G(p_2) = 010|00000$$

Láthatjuk, hogy a két bináris sztring közötti Hamming-távolság valóban 1. Ha most a keresztezésnek megfelelően felcseréljük a szülők között a „—”-tól jobbra eső részeket, akkor a szülőkkel ekvivalens leszármazottakat kapunk.

A binárisból Gray-kódba alakítás művelete a következőképpen is leírható:

$$(b_n, b_{n-1}, \ldots, b_2, b_1) \otimes (0, b_n, \ldots, b_3, b_2) = (b_n \otimes 0, b_{n-1} \otimes b_n, \ldots, b_2 \otimes b_3, b_1 \otimes b_2),$$

ahol $\otimes$ az XOR (eXclusive OR, 'kizáró vagy') műveletet jelöli. Lássuk most annak bizonyítását, hogy két egymásutáni bináris szám Gray-kódjainak Hamming-távolsága 1. Legyen $(b_n, b_{n-1} \otimes b_n, \ldots, b_2 \otimes b3, b_1 \otimes b_2)$ a $b = (b_n, \ldots, b_1)$

Gray-kódja, s ugyanígy $(d_n, d_{n-1} \otimes d_n, \ldots, d_2 \otimes d_3, d_1 \times d_2)$ a $d = (d_n, \ldots, d_1)$ Gray-kódja, a közöttük levő összefüggés pedig $b + 1 = d$. Észrevehetjük, hogy a Hamming-távolság tulajdonképpen a bitenkénti XOR műveletek összege, ugyanis ez akkor fog 1-et adni egy bitre, ha azok megfelelő értékei különböznek. Vagyis azt kell kimutatnunk, hogy

$$b_n \otimes d_n + b_{n-1} \otimes b_n \otimes d_{n-1} \otimes d_n + \ldots + b_1 \otimes b_2 \otimes d_1 \otimes d_2 = 1.$$

Ha $b_1 = 0$ volt akkor $d_1 = 1$, és b és d az összes többi bitpozícióban megegyezik. Ekkor a fenti összegnek csakis az utolsó tagja fog 1-et szolgáltatni, mivel $b_1 \otimes b_2 \otimes d_1 \otimes d_2 = b_1 \otimes d_1 \otimes b_2 \otimes d_2 = 1 \otimes 0$=1, vagyis a két sztring Hamming-távosága 1. Ha viszont $b_1 = 1$ volt, akkor $d_1 = 0$, és a következő két eset áll fenn: **(i)** $b_2 = 0$, ahonnan következik, hogy $d_2 = 1$, így $b_1 \otimes b_2 \otimes d_1 \otimes d_2 = 1 \otimes 1 = 0$, és ekkor csak a fenti összeg következő tagjának értéke lesz 1, az összes többi tag 0-t eredményez, mert $b_2 \otimes d_2 = 1$ és b_3-tól (d_3-tól) felfelé minden bit megegyezik b-ben és d-ben; **(ii)** $b_2 = 1$, ahonnan következik, hogy $d_2 = 0$, így újfent zérus lesz e tag értéke. Ebben az esetben addig mind zérusértékű tagokat kapunk míg b-ben egy 0-hoz nem érünk. Ezt az alábbi példa szemlélteti:

$$\begin{aligned} b &= 110111 + \\ & \qquad\quad 1 \\ d &= \overline{111000} \end{aligned}$$

A felső szám a b, míg az eredmény, azaz a lenti szám a d. Látható, hogy mikor b-ben egy zérushoz érünk, akkor $G(b)_i$ 1 lesz, és hasonlóan $G(d)_i$ is 1-et szolgáltat, így $1 \otimes 1 = 0$. A következő tag esetében viszont $G(b)_{i+1}$ és $G(d)_{i+1}$ különbözik, ezért ezek értékei XOR-ral összekötve 1-et fognak szolgáltatni. Innentől felfelé viszont megintcsak minden tag értéke 0 lesz, mivel a bitek páronként megegyeznek.

A dekódolás, vagyis a Gray-kód visszaalakítása bináris kódba az XOR művelet tulajdonságát használja ki, azaz, hogy $(a \otimes b) \otimes b = a$.

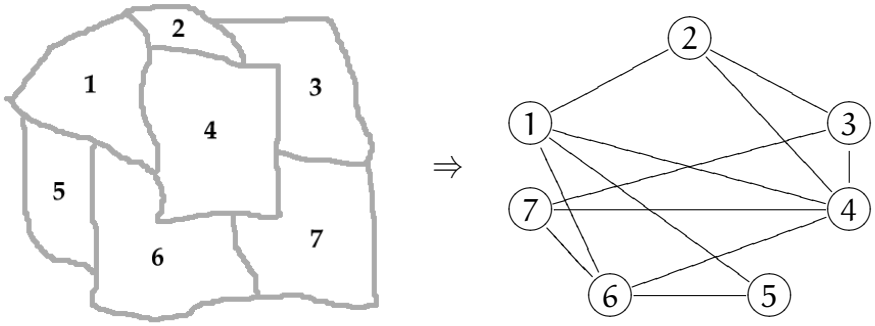
9.2. Függvényoptimizálás valós ábrázolással

9.3. Nemlineáris programozás

9.4. Térkép- vagy gráfszínezési probléma

Tekintsük a következő problémát. Adott egy térkép, melyet úgy szeretnénk kiszínezni, hogy az egymással szomszédos területek (országok, megyék, stb.) különböző színüek legyenek. Legyen k a rendelkezésünkre álló színek, N pedig a kiszínezendő területek száma. E problémát gráfszínezési problémának is szokták

nevezni, mivel egy térkép gráfként is ábrázolható úgy, hogy a gráf csúcsai jelölik a területeket, az egymással szomszédos területeknek megfelelő csomópontokat pedig irányítatlan élek kötik össze.



Ebben az esetben a gráf csúcsainak egy olyan színezését keressük, ahol a szomszédos csúcsok különböző színűek.

A probléma legegyszerűbb kódolása egy olyan N hosszúságú $(s_1, s_2, \ldots, s_N)$ lista, ahol $s_i \in \{1, 2, \ldots, k\}$, $i = 1, 2, \ldots, N$. Természetesen más kódolás is alkalmazható, például *sorrendi kódolás* (ref. lásd Futó, pp. 554–556), ezekre viszont itt nem térünk ki. Például ha a rendelkezésre álló színek száma 3, míg az országoké $N = 7$ akkor egy lehetséges egyedet a következőképpen kódolunk:

$$s = 1\,1\,2\,3\,3\,2\,1.$$

Az országok, területek szomszédosságát egy $\mathbf{W}$ szomszédossági mátrixban tárolhatjuk, ahol $W_{ij} = 1$ ha az i-edik és j-edik terület egymással szomszédos, és $W_{ij} = 0$ különben. Ekkor a feladat értelmezhető úgy, mint a következő függvény minimalizálása:

$$E(s) = \sum_{i,j} W_{ij} \cdot d(s_i, s_j),$$

ahol

$$d(s_i, s_j) = \begin{cases} 1, & \text{ha } s_i = s_j \\ 0, & \text{különben} \end{cases}$$

A fenti minimalizálandó függvév azon területpárok számát adja meg, melyek színe megegyezik. A problemát egyszerűen átalakíthatjuk maximalizálási feladatra. Néhány példa rátermettségi függvényre:

$$\text{rátermettség}(s) = N \cdot (N - 1) - E(s)$$

vagy

$$\text{rátermettség}(s) = e^{-\beta \cdot E(s)},$$

ahol a β paraméter a színegyezéseket súlyozza. A probléma megtalálható a [PKS01] 154. oldalán is.

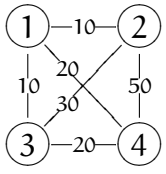
9.5. Az utazó ügynök problémája

Az utazó ügynök problémája (*travelling salesman problem*, TSP) az egyik legtöbbet tárgyalt NP-teljes probléma. A feladat a következőképpen hangzik. Adott egy ügynök, akinek adott n városon keresztül kell utaznia úgy, hogy végül visszaérjen a kezdeti városba, ahonnan elindult, és úgy, hogy az összességében megtett út minimális legyen. A kezdeti város tetszőleges (bár ez nem igazán fedi a valóságot), továbbá minden két város között van út. A feladatot úgy is értelmezhetnénk, mint egy n-edrendű teljes gráf minimális Hamilton-körének megtalálása. (A Hamilton-út értelmezés szerint a gráf összes csúcsát egyszer érintő út, míg a Hamilton-kör zárt Hamilton-utat jelent, vagyis az első és utolsó csúcs megegyezik. Kérdés: melyiket tekintjük ,,első"-nek?)

Legyen $V = \{1, 2, \ldots, n\}$ a városok halmaza, és jelölje $d(i, j)$, $i, j \in V$ a városok közötti távolságot. Formálisan a feladat a következő: keressük meg azt a v_i, $i = 1, \ldots, n + 1$ sorozatot ($v_{n+1} = v_1$), hogy

$$\sum_{i=1}^{n} d(v_i, v_{i+1})$$

minimális legyen. A keresési terület mérete $n!$ függvény, pontosabban $(n-1)!/2$. Ez azt jelenti, hogy a megoldás megtalálásához ennyi lehetséges konfigurációt kell megvizsgálnunk, ami már 20 város esetén 10^{11} nagyságrenndel nagyobb, mint ahány hajszál van fejünkön, azaz pontosan 60 822 550 204 416 000. Természetesen léteznek közelítő megoldások (tulajdonképpen a genetikus algoritmusos megoldás is csak közelítésnek nevezhető, mert sajnos a végtelen nem rekonstruálható a számítógépen, legalábbis eleddig), az egyik ilyen egyszerűen Kruskal algoritmusa alapján működik (minimális feszítőfa megkeresése gráfokban, vagyis listázzuk az éleket növekvő sorrendben, majd sorra kiválasztjuk azokat, melyek nem okoznak konfliktust. Konfliktus akkor adódik, ha egy olyan élt akarunk bevinni, amely egy csúcs fokszámát 2-ről háromra növeli, illetve ha n-nél rövidebb kör alakul így ki (egy kör hosszát az őt alkotó csúcsok számával definiáljuk). Tekintsük a következő példát:



Az élek növekvő sorrendben:		Út:
1 2	⋆	1 2
1 3	⋆	3 1 2
1 4		
3 4	⋆	4 3 1 2
2 3		
2 4	⋆	4 3 1 2 4

Bal oldalon a városok és a közöttük levő távolságok láthatók, míg jobb oldalon az előbb említett algoritmus működése követhető nyomon. A csillagok azokat az

éleket jelölik, melyeket bevettünk a végső útba, az út alakulása pedig a csillagoktól jobbra látható. Ebben az esetben csak 3 lehetőségünk van, mégpedig 1 2 3 4 1, 1 3 2 4 1 vagy 2 1 3 4 2, melyek hossza rendre 80, 110 és 90. A fenti algoritmus által kapott út viszont innen a másodiknak felel meg, amely pedig a leghosszabb, így láthatjuk, hogy ez az algoritmus nem mindig működik úgy ahogy kellene.

...

Egy másik lehetséges reprezentációt alkothatunk, ha megfigyeljük, hogy az összes eddigi reprezentációnk által leírt keresési tér $n!$ és nem $(n-1)!/2$ nagyságú. Először is nézzük meg, n város esetén hogyan lesz $(n-1)!/2$ lehetséges megoldásunk. Első lépésben rögzítsünk egy csúcsot (várost), mert ellenkező esetben a cirkuláris permutációk ugyanazt az utat írják le, így n-szer több elrendezést kapunk, mint ahány valójában van. Így eljutunk az $(n-1)!$-hoz. Ez így még mindig nincs rendjén, mert ha így felsorojuk az összes lehetőséget, észrevesszük, hogy minden út kétszer szerepel, mivel mindegy, hogy balra vagy jobbra indulunk el az így kialakul kör egy pontjából. Ennélfogva a lehetséges elrendezések száma pontosan $(n-1)!/2$.

Az új reprezentáció lényege az lenne, hogy csak $(n-1)!/2$ utat írjon le, vagyis ne tegyen különbséget az $1\,2\,3$ és $2\,3\,1$, illetve a $2\,1\,3$ és $3\,1\,2$ utak között. Az fenti ,,rögzítést" úgy oldhatjuk meg, ha egyszerűen kihagyunk egy várost, amely legyen a maximális számmal ellátott város. Így csak $n-1$ város permutációit fogjuk vizsgálni. Mostmár csak a második pontot kell valahogyan megoldanunk, azaz, hogy egy út és annak tükörképe ugyanazzal a reprezentációval rendelkezzen. Egy lehetséges megoldás a korábban már bemutatott mátrix-reprezentációhoz hasonlít, azzal az eltéréssel, hogy most a szomszédossági mátrixban mind i és j szomszédossága esetén mind az (i,j), mind a (j,i) pozícióra 1-et írunk. Mivel a mátrix szimmetrikus a főátlóra nézve, elegendő csak a főátló alatti vagy a főátló feletti résszel foglalkoznunk. Válasszuk a főátló alatti részt. Az elemek száma ebben a háromszögmátrixban $(n-1)\cdot(n-2)/2$. Ez a mátrix akkor fog egy lehetséges konfigurációt ábrázolni, ha benne az 1-esek száma $n-2$, a többi elem pedig zérus. Így például – 5 város esetén – az

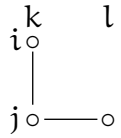
2	1		
3	1	0	
4	0	1	0
	1	**2**	**3**

a $4\,2\,1\,3\,(5)$ bejárást ábrázolja, vagy úgy is mondhatjuk, hogy a $4\,2\,1\,3\,(5)$ egyedüli reprezentációja a fenti háromszögmátrix (a félkövér számok a sorok, illetve oszlopok indexét jelölik). Viszont valamit elhallgattunk a fenti ábrázolásmóddal kapcsolatoban. Mivel a mátrix elemeinek száma n város esetén $(n-1)(n-2)/2$, amiből $n-2$ elemnek 1-nek kell lennie, ezért a fenti reprezentációval leírható elrendezések száma nem $(n-1)!/2$, hanem $\binom{(n-1)(n-2)/2}{n-2}$ (ami egy bizonyos n-től kezdve

nagyobb, mint az első összefüggés). Vagyis bizonyos megkötéseket kell tennünk a zérusokkal és egyesekkel kapcsolatban, mégpedig a következő kettőt:

1. az 1-esek száma bármely sorban és oszlopban maximum kettő lehet,
2. a leírt gráfban nem lehet kör.

Az első feltétel világos és könnyen ellenőrizhető. Nézzük a másodikat. Egy szomszédossági mátrix által leírt gráf akkor tartalmaz kört, ha egy `i` csúcsból eljuthatunk egy `j` csúcsba, majd a `j` csúcsból visszajuthatunk az `i`-be egy másik úton. Vagyis



konfiguráció esetén, amikor $\mathtt{i} = \mathtt{l}$. Mert ekkor egy `ijki` alakú úthoz jutunk, ami természetesen egy kört ír le.

Tehát mindig ellenőriznünk kell, hogy a populáció inicializálásakor, mutáció, illetve keresztezéskor érvényes egyedet kapjunk. A lehetséges mutációkat és keresztezéseket az olvasó képzeletére bízzuk.

9.6. SAT

A rövidítés SAT az angol *Boolean Satisfiability Problem* elnevezésből ered, melyet magyarul Boole-kielégíthetőségi problémának nevezünk, azonban a rövidség miatt az angol SAT elnevezést részesítjük előnyben, és a továbbiakban így fogunk rá hivatkozni. A SAT probléma – egyszerűen megfogalmazva – abban áll, hogy egy ítéletkalkulusbeli formulának keressük meg egy olyan interpretációját (azaz a *hamis* és *igaz* [vagy 0 és 1] olyan hozzárendeléseit), amely a formulát igazként értékeli ki. Az ítéletkalkulusban (vagy nulladrendű predikátumkalkulusban) nincsenek kvantorok, vagyis csak ítéletszimbólumok, zárójelek és logikai műveleti jelek (negáció, konjunkció, diszjunkció, implikáció és ekvivalencia) léteznek. Például az

$$A \vee \overline{A} \leftrightarrow 1$$

egy ítéletkalkulusbeli formula, egyben logikai törvény (az ekvivalencia jeltől balra eső részt tautológiának nevezzük, mert annak minden interpretációja igaz).

A SAT ugyancsak NP-teljes probléma, ezért egy lehetséges megoldás megkereséséhez itt is gyakran genetikus algoritmusokat alkalmaznak. Ennek a feladatnak a genetikus algoritmusok szempontjából nézve az az érdekessége, hogy míg a reprezentáció szinte magától adódik, elég nehéz jó és hatékony rátermettségi

függvényt találni a megoldáshoz. Ha adott egy n darab ítéletváltozót tartalmazó formula, akkor annak evidens reprezentációja azon n hosszúságú kromoszóma, ahol minden gén egy változónak felel meg a formulából. Egy gén értéke (allélja) csak 0 és 1 lehet, és az adott változóhoz rendelt igazságértéket tartalmazza. Például ha adott a következő formula:

$$(\overline{A} \vee B) \wedge A \wedge C,$$

akkor a kromoszómánk (b_1, b_2, b_3) alakú lesz, ahol b_1 A, b_2 B, b_3 pedig C értékét reprezentálja (ennek a formulának az egyetlen igaz értékű interpretációja $(1, 1, 1)$).

Rátermettségi függvényként tekinthetnénk egyszerűen a formula igazságértékét, viszont ezzel az a probléma, hogy csak 0 és 1 értéket vehet fel, és az esetek nagy többségében ez sajnos 0 lesz, vagyis a populációnkat fejlődését nem tudjuk a helyes irányba terelni, nem tudnunk különbséget tenni két hamis formula között. Egy másik – immár használható – lehetőség az lenne, hogy először a formulát konjunktív normál formára (= diszjunkciók konjunkciója) hozzuk, majd rátermettségként összeadjuk az elemi diszjunkciók igazságértékét, megszámlálva ezzel a konjunktív normál forma 1 igazságértékű elemeit. Ehhez azonban a formulát először konjunktív normál formára kell hoznunk. Ezt a következőképpen tehetjük meg (ref. in Futó Iván, 132 old):

1. Küszöböljük ki az implikáció és ekvivalencia jeleket az alábbi szabályok segítségével:

$$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

$$A \rightarrow B \equiv \overline{A} \vee B$$

2. Redukáljuk a logikai tagadások hatáskörét, azaz vigyük őket közvetlenül a logikai változók fölé (vagy mellé, ha a másik szokásos jelölést használjuk). Ezt a következő szabályokat felhasználva tehetjük meg:

$$\overline{A \wedge B} \equiv \overline{A} \vee \overline{B}$$

$$\overline{A \vee B} \equiv \overline{A} \wedge \overline{B}$$

$$\overline{\overline{A}} \equiv A$$

3. Alkalmazzuk a disztributivitás törvényeit:

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$

A fenti módszerrel ezután könnyen ki tudjuk számolni egy egyed rátermettségét, csupán annyi a probléma, hogy az előbb ismertetett algoritmus nem lineáris. Ezért egy másik rátermettségi függvény után nézünk. Tekintsük a következő kiértékelési szabályokat:

1. $val(\overline{e}) = 1 - val(e)$
2. $val(e_1 \wedge e_2 \wedge \ldots \wedge e_n) = \min(val(e_1), val(e_2), \ldots, val(e_n))$
3. $val(e_1 \vee e_2 \; vee \ldots \vee e_n) = \max(val(e_1), val(e_2), \ldots, val(e_n))$,

ahol e, e_1 és e_2 formulákat jelölnek. Mivel minden formula értéke csak 0 vagy 1 lehet, ezért könnyen belátható, hogy a fenti rekurzív módon értelmezett val függvény értékkészlete ugyancsak a $\{0, 1\}$ halmaz lesz. Viszont egy kis változtatással a rátermettségi függvény értékkészlete intervallummá alakítható ([DS89]):

$$val(e_1 \wedge e_2 \wedge \ldots \wedge e_n) = \text{avg}(val(e_1), val(e_2), \ldots, val(e_n))$$

Így a fenti példánk nehány interpretációjához a következő rátermettségi értékeket rendelhetjük:

A	B	C	$(\overline{A} \vee B) \wedge A \wedge C$
1	0	0	$\frac{1}{3}$
0	0	1	$\frac{2}{3}$
0	1	1	$\frac{2}{3}$
1	1	1	1

Viszont van néhány probléma ezzel a rátermettség-kiszámítással, mégpedig az, hogy az avg (*average*, 'átlag') függvény nem invariáns az ítéletkalkulusbeli ekvivalenciákra nézve. Például

$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C,$$

viszont

$$\frac{A + \frac{B+C}{2}}{2} \neq \frac{\frac{A+B}{2} + C}{2}.$$

Ugyanígy a De Morgan-törvények sem érvényesek, vagyis

$$A \vee B \equiv \overline{(\overline{A} \wedge \overline{B})},$$

viszont

$$\max(A, B) \neq \frac{A + B}{2}.$$

Így például a $(0, 0)$-n kívül a $\overline{(\overline{A} \wedge \overline{B})}$ formula összes többi interpretációja igaz, viszont a $(0, 1)$ és $(1, 0)$ interpretációkra 0.5-öt kapunk, pedig ezek is ugyanúgy helyesek, mint az $(1, 1)$. Általánosan kimutatható, hogy az így megépített rátermettségi függvény nem rendel 1-et hamis interpretációkhoz, de

gyakran rendel 1-nél kisebb értéket bizonyos megoldásokhoz, és ez kisebb is lehet, mint a hamis interpretációkhoz rendelt érték. De Jong és Spears ([DS89]) rájöttek arra, hogy ilyen jellegű probléma csak akkor adódik, ha a formulában $(e_1 \wedge e_2 \wedge \ldots \wedge e_n)$ alakú részformulák szerepelnek. Ezeket átalakítva, azaz alkalmazva a De Morgan szabályt, a probléma kiküszöbölhető. Azt is észrevették, hogy az $(e_1 \vee e_2 \vee \ldots \vee e_n)$ alakú részformulák, ahol $e_1, \ldots, e_n$ csak 0 és 1 értéket vesz fel, 0 vagy 1 rátermettséget eredményez. Ha viszont itt is alkalmazzuk a megfelelő De Morgan törvényt, akkor a rátermettséget az avg irányítja, amely nagyobb értéket rendel a ,,jobb" interpretációkhoz.

Egy másik módosítást is tehetünk, mégpedig hogy az avg helyett az avg^p függvényt használjuk, vagyis az avg p-edik hatványát. Ha p végtelenbe tart, akkor az avg^p a min függvényhez hasonlít. De Jong és Spears a legjobb eredményeket $p = 2$ esetén kapta.

9.7. A 8 királynő problémája

Feladatunk a következő: helyezzünk el egy tetszőleges $N \times N$ méretű sakktáblán N darab királynőt úgy, hogy azok ne ,,üssék" egymást, azaz ne legyen konfliktus közöttük. Konfliktus akkor adódik, ha két királynő ugyanazon sorban, ugyanazon oszlopban, vagy ugyanazon átlón helyezkedik el. Az eredeti feladat egy 8×8-as sakktáblára vonatkozott, viszont ez kiterjeszthető tetszőleges méretre. A problémát először Gauss vetette fel 1850-ben. A feladatot sokszor úgy fogalmazzák meg, hogy keressük meg az az N királynő összes lehetséges ilyen elrendezését. Habár ennek (gyors) megoldására sok hatékony heurisztikus algoritmus született az évek során, a 8 királynő problámáját szinte mindenki a backtracking programozási technikával asszociálja. Láthatjuk, hogy a keresési terület nagyon nagy, mérete N^N vagyis exponenciális, és így a megoldások száma is exponenciálisan növekszik N függvényében. Míg 8 királynő esetén 92 megoldás létezik (12 egyedi megoldás, vagyis figyelmen kívül hagyva azokat, melyek egy már feljegyzett megoldás forgatással vagy tükrözéssel kapott másai), 16 királynő esetén 14 772 512 (melyből 1 846 955 egyedi megoldás), 20 királynő esetén 39 029 188 884 (melyből 4 878 666 808 egyedi) megoldásunk van[1].

Genetikus algoritmussal egyetlen megoldást keresünk. Reprezentációnk legyen a következő alakú:

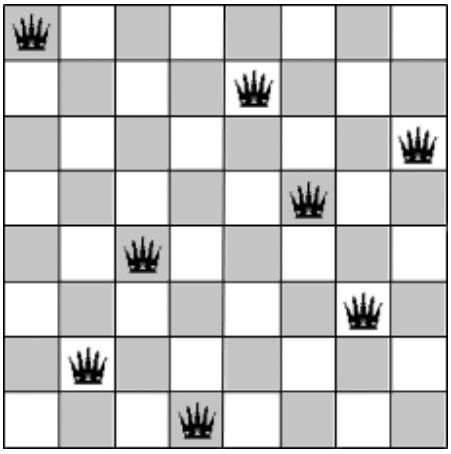
$$(q_1, q_2, \ldots, q_N),$$

ahol q_i azt jelöli, hogy a sakktábla i-edik sorának q_i-edik oszlopában egy királynő található. Például a

$$q = (1, 5, 8, 6, 3, 7, 2, 4)$$

[1] Az adatok a `http://www.durangobill.com/N_Queens.html` oldalról származnak

egy lehetséges megoldás 8×8-as sakktábla esetén (a sorokat a balfelső saroktól lefelé, az oszlopokat pedig balról jobbra indexeljük; lásd a 9.4. ábrát). Ezt a reprezentációt használva már eleve megoldottuk azt a problémát, hogy egy sorba, illetve egy oszlopba csak egyetlen királynő kerülhet.



9.4. ábra. Egy lehetséges megoldás 8 királynő esetén

Könnyen észrevehető, hogy a főátlón vagy a főátlóval párhuzamos átlón akkor következik be ütközés, ha a pozíciók koordinátáinak összege megegyezik, azaz $\exists i, j$ úgy, hogy $i + q_i = j + q_j$, vagyis, ha $q_i - q_j = j - i$. A mellékátlón vagy a mellékátlóval párhuzamos átlón pedig akkor, hogyha $\exists i, j$ úgy, hogy $i - q_i = j - q_j$, vagyis ha $q_i - q_j = i - j$. Ezeket összevonva felírhatjuk, hogy akkor és csakis akkor van ütközés, ha $q_i - q_j = \pm(i - j)$, vagyis ha $|q_i - q_j| = |i - j|$. Ez alapján könnyen felírhatjuk a rátermettségi függvényünket, ami

$$\text{rátermettség}(q) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \delta(|q_i - q_j| - |i - j|),$$

ahol

$$\delta(x) = \begin{cases} 1, & \text{ha } x = 0 \\ 0, & \text{különben} \end{cases}$$

Keresztezésként használhatjuk a bináris ábrázolási módnál is használt egypontos keresztést, míg mutációkor egyszerűen generálhatunk egy véletlen számot az $\{1, 2, \ldots, N\}$ halmazból minden mutációra kiválasztott gén esetén, s a gén új értéke ez a szám lesz.

Egy másik példát is adunk a rátermettségi függvényre. Egy $q = (q_1, q_2, \ldots, q_N)$ sorozat akkor megoldás, hogyha a következő feltételek teljesülnek:

$$\sum_{\substack{i=1 \\ i+q_i=k}}^{N} 1 \leq 1, \; k = 2, 3, \ldots, 2N,$$

$$\sum_{\substack{i=1 \\ i-q_i=k}}^{N} 1 \leq 1,\ k = -(N-1), -(N-2), \ldots, N-1.$$

Az első feltétel azt fejezi ki, hogy a főátlón vagy a főátlóval paárhuzamos átlókon egy vagy zérus darab királynő állhat, a második feltétel pedig ugyanezt írja le a mellékátlóra és az azzal párhuzamos átlókra. Így felírhatjuk a következő két büntetőfüggvényt:

$$g_1(q) = \sum_{k=2}^{2N} \max\left\{0, \left(\sum_{\substack{i=1 \\ i+q_i=k}}^{N} 1\right) - 1\right\},$$

$$g_2(q) = \sum_{k=-(N-1)}^{N-1} \max\left\{0, \left(\sum_{\substack{i=1 \\ i-q_i=k}}^{N} 1\right) - 1\right\}.$$

Mivel e függvények maximális értéke páronként $N-1$, ezért rátermettségi függvényként választhatjuk például a következő kifejezést:

$$\text{rátermettség}(q) = 2(N-1) - (g_1(q) + g_2(q)).$$

9.8. Stratégiák keresése genetikus algoritmusokkal

9.8.1. A fogoly-dilemma

9.8.2. Tic-tac-toe

9.8.3. A ragadozó és zsákmány problémája

9.9. Klaszterezés genetikus algoritmusokkal

A klaszterezés (az angol *clustering* szóból) nagy fontosságú feladat a matematikában és az informatikában. A gépi tanulásban nem-felügyelt tanulásnak is nevezik, ami azt jelenti, hogy a rendszer előzőleg nem lát egyetlen tanulási példát sem, vagyis a rendszert nem tanítjuk, nem-felügyelt módon magától tanul. A feladat úgy szétválasztani az adatpontokat, hogy a felismert különböző halmazokban (klaszterekben) a pontok minél jobban hasonlítsanak egymáshoz, míg a különböző halmazokban levő pontok minél jobban eltérjenek egymástól. A klaszterezési

problémát sok irányból meg lehet közelíteni, ezért hatalmas irodalommal rendelkezik. Mivel ez a könyv nem klaszterezéssel és gépi tanulással foglalkozik, nem vázolom a különböző irányokat és a klaszterezési metódusokat. Ilyen szempontból nagyon jó áttekintést nyújt Belkhin cikke (belkhin) vagy Jain és Dubes ,,Algorithms for Clustering Data" (jain...) könyve.

Ebben a részben két ismert megközelítést fogunk megvizsgálni genetikus szempontból. Az egyik egy particionáló algoritmus, a k-központú klaszterezés (angolul *k-means*), a másik pedig a normalizált vágáson alapuló gráf-klaszterezés.

A klaszterezni kívánt adatpontok $\mathbf{x}_i \in \mathbb{R}^n, i = 1, \ldots, m$, és feltételezzük, hogy a klaszterek száma ismert, éspedig K.

9.9.1. K-központú klaszterezés

K-központú klaszterezéskor a következő függvényt akarjuk minimizálni,

$$F = \sum_{i=1}^{K} F_i = \sum_{i=1}^{K} \sum_{j=1}^{m} U_{ij} \cdot \|\mathbf{x}_j - \mathbf{c}_i\|^2,$$

ahol K a klaszterek számát, $\mathbf{x}_j$ az adatpontokat és $\mathbf{c}_i$ a klaszterközéppontokat jelöli, $\mathbf{U}$ pedig az úgynevezett hozzárendelési mátrix, azaz $U_{ij} = 1$ ha $\mathbf{x}_j$ az i-edik klaszterhez tartozik, különben zérus, $i = 1, \ldots, K$, $j = 1, \ldots, m$. Az $\mathbf{U}$ hozzárendelési mátrixra az alábbi megszorításokat tesszük,

$$\sum_{i=1}^{K} U_{ij} = 1 \quad \forall j; \qquad \sum_{j=1}^{m} U_{ij} \geq 1 \quad \forall i, \tag{$*$}$$

vagyis egy pont csak egy klaszterhez tartozhat és nem lehetnek üres klaszterek, vagyis minden klaszterhez legalább egy pontnak tartoznia kell. Az F összefüggést egy iteratív módszerrel minimizálhatjuk, amit Lloyd-algoritmusnak nevezünk. Ha F-et $\mathbf{c}_i$ szerint deriváljuk, majd egyenlővé tesszük zérussal, a következő összefüggéshez jutunk,

$$\frac{\partial F}{\partial \mathbf{c}_i} = 0$$

$$2 \sum_{j=1}^{m} U_{ij} \mathbf{c}_i - 2 \sum_{j=1}^{m} U_{ij} \mathbf{x}_j = 0$$

$$\mathbf{c}_i = \frac{\sum_{j=1}^{m} U_{ij} \mathbf{x}_j}{\sum_{j=1}^{m} U_{ij}},$$

vagyis akkor lesz minimális, ha a klaszterközéppontoknak a pontok közepét (átlagát) választjuk. Ekkor az F-et $\mathbf{x}_j$ szerint úgy minimizáljuk, hogy minden

pontot hozzárendelünk a hozzá legközelebb álló klaszterközépponthoz, klaszterhez, mivel $\|\mathbf{x}_j - \mathbf{c}_i\|$ ekkor lesz minimális. Lloyd algoritmusa a következőképpen vázolható:

1. Rendeljük a pontokat véletlenszerűen a K klaszterhez úgy, hogy a hozzárendelési mátrix teljesítse a $(*)$ feltételeket.

2. Számoljuk ki a klaszterek középpontjait.

3. Minden pontot rendeljünk a hozzá legközelebb eső klaszterhez a középpont alapján.

4. Ismételjük az eljárást a 2. lépéstől míg $\mathbf{U}$ már nem változik.

A K-központú klaszterezés legnagyobb hátránya – euklideszi távolságot használva –, hogy csak lineárisan szétválasztható klasztereket képes helyesen megtalálni. Ezeket sem minden esetben helyesen határozza meg, mivel a végső hozzárandelés nagyban függ a kezdeti hozzárendeléstől, ezért átalában többször futtatjuk egymás után az algoritmust különböző kezdeti hozzárendelésekre. A lineárisan nem szétválasztható klaszterek esetén a K-központú klaszterezés *kernelizált* változatát használhatjuk ([DGK04], [VM02]).

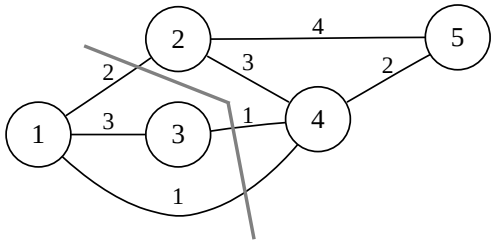
A K-központú klaszterezés egy módosított változata a fuzzy K-központú klaszterezés ([Ped05]), amely annyiban tér el az első változattól, hogy egy pont akár több klaszterhez is tartozhat. A $(*)$ feltételek itt is érvényesek, viszont az $\mathbf{U}$ mátrix már nem csak zérusokat és egyeseket tartalmazhat, hanem $U_{ij} \in [0, 1]$.

9.9.2. Gráf-klaszterezés

A gráf-klaszterezések alapját az adatpontok hasonlósága alapján felépített gráf képezi. Ezen a gráfon dolgozva megpróbáljuk minél jobban szétválasztani a *jól* összefüggő komponenseket.

A legegyszerűbb gráf-klaszterezés Kruskal algoritmusa, amely egy tetszőleges (összefüggő) gráf minimális feszítőfáját határozza meg. Az algoritmus úgy működik, hogy növekvő sorrendbe rendezzük az éleket, majd az így kapott növekvő sorrendbe rendezett lista elejétől a vége felé haladva kiválasztunk éleket, feltéve, hogy azok nem képeznek kört a már kiválasztott élekkel; ha igen, akkor azt az élt nem választjuk ki. Ez úgy használható fel klaszterező algoritmusként, hogy addig folytatjuk a fenti eljárást, míg K összefüggő komponenst kapunk.

A gráf-klaszterezések egy nagy osztályát a minimális vágással történő klaszterezések képezik. Egy vágás alatt egy olyan élhalmazt értünk, melynek eltávolítása után a gráf nem marad összefüggő, hanem két összefüggő komponensre bomlik. Minimális vágás keresésekor mindig meg kell határoznunk a gráf



9.5. ábra.

két pontját, melyek között a minimális vágást keressük, és a vágást úgy keressük, hogy az egyik pont az első, a másik a második komponensben legyen. Például a 9.5. ábrán látható gráf esetén az 1 és 2 csúcsok közötti minimális vágás értéke $2 + 1 + 1 = 4$, amely az $S = \{1, 3\}$, $T = \{2, 3, 4\}$ csúcsokat tartalmazó komponenseket eredményezi. A minimális vágás azt az élhalmazt jelenti, melyek minimális költségűek, és eltávolításuk két összefüggő komponensre bontja a gráfot, vagyis jól használható klaszterezésre. Ha csak két klaszterünk van, akkor egyszerűen megkeressük a minimális vágást minden két pont esetén, és ezek közül a minimális értékűt választjuk. Több klaszter esetén a Gomory-Hu algoritmust használhatjuk. Ezzel az algoritmussal magkapjuk egy gráf minimális vágásait. Az algoritmus a következőképpen működik:

1. Kiválasztunk két csúcsot ugyanabból a komponensből.

2. Megkeressük a minimális vágást ezen csúcsok között.

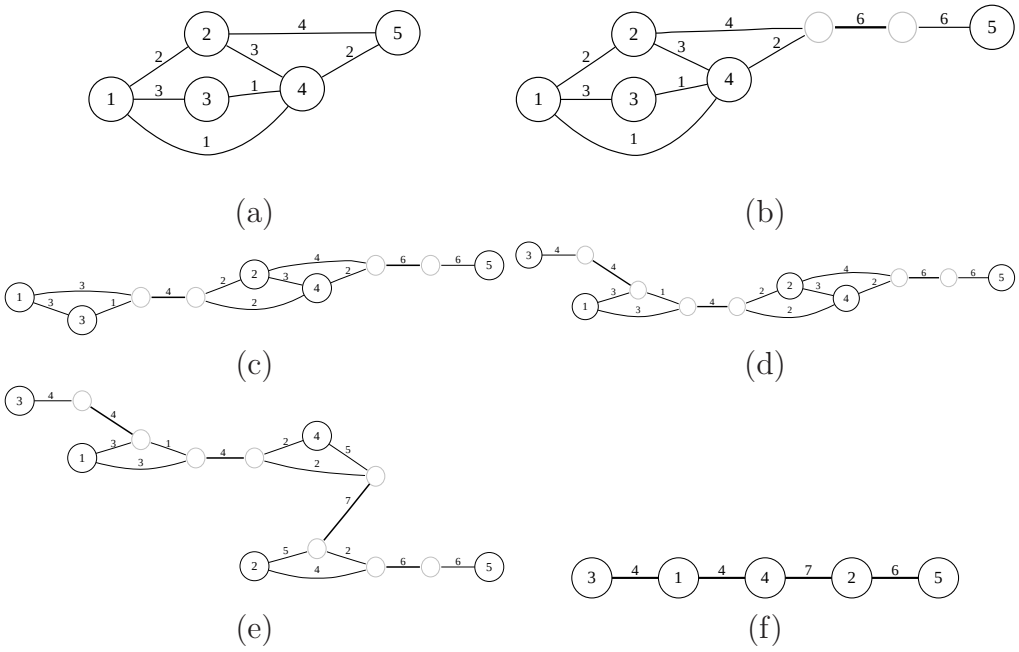
3. Annyi új jelöletlen csúcsot hozunk létre ahány már létező komponenshez a csúcsok kötődtek.

4. A csúcsokat összekötjük a jelöletlen csúcsokkal úgy, hogy a hozzárendelt súly az adott csúcsból a másik komponensbe menő súlyok összege lesz.

5. A jelöletlen csúcsokat összekötjuk, és a minimális vágás értékét rendeljük hozzá súlyként.

6. Vissza a 1-es lépésre.

A 9.6. ábra a Gomory-Hu algoritmus lépéseiben kapott gráfokat mutatja. Az utolsó lépésben, 9.6/(f), megkapjuk a gráfot (fát), melyből kiolvasható bármely két csúcs közötti minimális vágás értéke úgy, hogy a két csúcs közötti úton szereplő élek súlyai közül kiválasztjuk a minimumot. Így például az 1 és 2 csúcsok közötti minimális vágás értéke 4, ami megfelel a 9.5. ábrán kapott értéknek. A klasztereket



9.6. ábra.

most úgy kaphatjuk meg, hogy egyszerűen eltávolítjuk az éleket a fából, növekvő sorrendben. Például 2 klaszter esetén

$$\{3\},\{1,4,2,5\} \text{ vagy } \{3,1\},\{4,2,5\},$$

3 klaszter esetén a

$$\{3\},\{1\},\{4,2,5\}$$

komponenseket kapjuk.

Normalizált vágáson alapuló gráf-klaszterezés

$$G = \sum_{i=1}^{K} G_i = \sum_{i=1}^{K} \frac{\sum_{a\in C_i, b\in \overline{C_i}} w(a,b)}{\sum_{a\in C_i, b\in V} w(a,b)}$$

Irodalomjegyzék

[Aga01] Alexandru Agapie. Theoretical analysis of mutation-adaptive evolutionary algorithms. *Evolutionary Computation*, 9(2):127–146, 2001.

[DGK04] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *KDD*, pages 551–556. ACM, 2004.

[DS89] Kenneth A. De Jong and William M. Spears. Using genetic algorithm to solve NP-complete problems. In James D. Schaffer, editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 124–132, San Mateo, CA, 1989. Morgan Kaufmann.

[GM00] David Greenhalgh and Stephen Marshall. Convergence criteria for genetic algorithms. *SIAM J. Comput*, 30(1):269–282, 2000.

[GZ01] Garrison W. Greenwood and Qiji Zhu. Convergence in evolutionary programs with self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):147–157, 2001.

[HH98] Randy I Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley and Sons, New York, 1998.

[Mic96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.

[Mit96] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[Ped05] Witold Pedrycz. *Knowledge-Based Clustering: From Data to Information Granules*. Wiley, 2005.

[PKS01] Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos, editors. *Machine Learning and Its Applications, Advanced Lectures*, volume 2049 of *Lecture Notes in Computer Science*. Springer, 2001.

[Rud94] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[Rud98] Günter Rudolph. Finite Markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35:67–89, 1998.

[VM02] S.V.N. Vishwanathan and Narasimha M. Murty. Kernel enabled K-means algorithm. Technical report, January 01 2002.