

Evolúciós algoritmusok

Sáfár Orsolya

2017/2018 ősz

Tárgykövetelmények

- ▶ Páratlan héten előadás, páros héten labor, jelenlét kötelező
- ▶ Jegyet az otthon elkészített házi feladatokra kapnak
- ▶ Hetente van házi, a félév alatt három nagyobb témából
- ▶ Összesen 12 házi, egy alkalommal legfeljebb 10 pontért
- ▶ Témánként 12 pontot kell elérni
- ▶ A félévvégi ponthatárok 40, 50, 60, 70.

Motiváció

Optimalizálási feladat: adott potenciális megoldások halmazából válasszuk ki „belátható időn belül” a legjobbat vagy legjobbakat.
Példák:

1. Keressük optimális útvonalat egy gyalogtúrához
2. Készítsünk órarendet
3. Találjunk legjobb útvonalat egy robotkarnak
4. Keressünk jó stratégiát amőbában

Futásidő jellemzése

Legyen a bemenet hossza $n \in \mathbb{N}$. Az mondjuk, hogy az algoritmus $O(f(n))$ futásidejű, ha $f(n)$ -nek valamely (n -től független) konstansszorosával felülről becsülhető a futás közben végzett elemi műveletek száma.

Egy algoritmus **polinomidejű**, ha $O(n^k)$ futásidejű valamely $k \in \mathbb{N}$ -re, **exponenciális** futásidejű ha $O(a^n)$ valamely $a \in \mathbb{R}^+$ -ra.

Példák futásidőre

Egy n elemszámú, valós számokból álló tömböt szeretnék növekvő sorrendbe rendezni.

Kiválasztásos rendezés:

Minden körben kiválasztom a még rendezetlen tömb legkisebb elemét, majd kicserélem az első elemmel. A következő körben a maradék, eggyel rövidebb tömböt rendezem, ...

Ekkor az $n - i$. körben $i - 1$ összehasonlítást végzek, és legfeljebb egy cserét, így a lépésszám $\sum_{i=1}^n i = \frac{n(n+1)}{2}$, azaz ez rendezés $O(n^2)$ futásidejű.

Példák futásidőre

Összefésüléssel rendezés:

Egy lépésben kettéválasztjuk a listát, a két rövidebbet rendezzük valahogy (pl rekurzívan), majd a két rendezett listát összefésüljük.

A futásidő így n_1, n_2 elemszámú bontás esetén, ha egy n elemszámú listára $T(n)$ -el jelöljük a szükséges lépésszámot :

- ▶ az összefésülés $n_1 + n_2$ összehasonlítás,
- ▶ résztömb rendezése $T(n_1) + T(n_2)$ lépés

Mivel $T(1)=0$ és $T(n) \leq n + 2T\left(\frac{n}{2}\right)$, így

$$T(n) \leq n + 2\frac{n}{2} + \dots + 2^{k-1}T\left(\frac{n}{2^{k-1}}\right) \leq n[\log_2 n]$$

Így az összefésülés $O(n \log n)$ futásidőjű rendezés.

Polinomidejű algoritmusok futásideje

Tegyük fel, hogy az algoritmus futásideje $O(n^k)$ valamilyen ismeretlen k kitevővel. Célunk, hogy a futásidők ismeretében adjunk jó becslést a kitevőre. Sajnos a probléma túlhatározott, tipikusan nincs a klasszikus értelemben vett megoldása.

Jelöljük $T(n)$ -el az n hosszú bemenetre az algoritmus lépésszámát. Ekkor $T(n) = cn^k$, emiatt $\log(T(n)) = \log c + k \log(n)$. Ha a futásidő logaritmusát $\log(n)$ függvényeként ábrázoljuk, akkor megközelítőleg egy egyenest kapunk, amelynek meredeksége az ismeretlen kitevő.

A brute force

A brute force az összes eset ellenőrzését jelenti véges sok potenciális megoldás esetén.

Általában egyszerű megvalósítani az ilyen algoritmust, de bizonyos esetekben a futásidő a probléma méretével elfogadhatatlanul megnő.

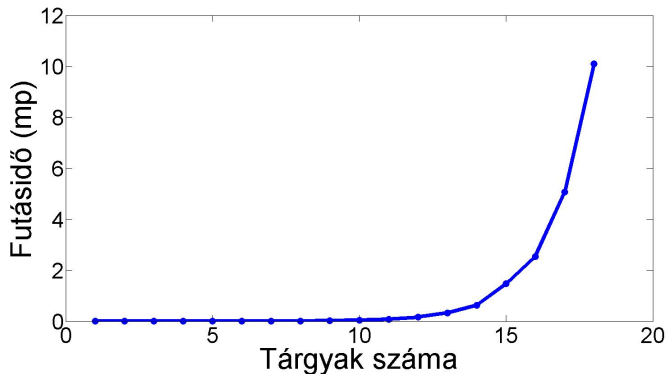
Kis méretű problémáknál nem biztos, hogy megéri mást megvalósítani.

A hátizsákpakolás

Adott n darab tárgy, mindegyiknek ismert a súlya $\mathbf{s} = (s_1, s_2, \dots, s_n)$ és az értéke $\mathbf{v} = (v_1, v_2, \dots, v_n)$. Célunk, hogy egy ismert $C \in \mathbb{R}^+$ teherbírású hátizsákba minél nagyobb összértékű tárgyakat pakoljunk be.

Mivel minden egyes tárgy vagy bekerül vagy nem, ezért összesen 2^n féle pakolás lehetséges. A brute force megoldás az összeset végignézi, ez exponenciálisnál is rosszabb futásidőt jelent.

Futásidő brute force megvalósításra



Biológiai párhuzam

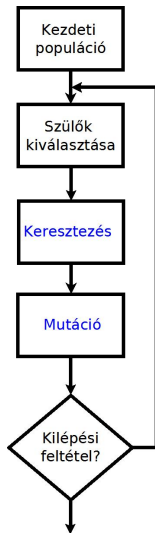
Az evolúciós algoritmus egy olyan, véletlent is felhasználó optimumkereső eljárás, amely a darwini evolúció működését utánozza. A potenciális megoldások egy multihalmazából (populáció) újabb megoldásokat készítünk (szaporodás), ahol az új egyedek (utódok) hasonlítanak a régiekre (szülők). Az evolúcióra pedig azért hasonlít, mert annak az egyednek lesz több utóda, aki maga jó minőségű megoldás.

Környezet	Probléma
Egyed	Potenciális megoldás
Populáció	Megoldások egy multihalmaz
Fitnessz	A megoldás minősége
Genom	A megoldások reprezentációja

Hátizsák pakolás genetikus algoritmussal

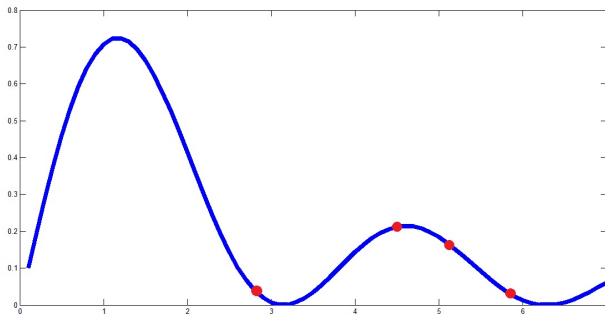
- ▶ Reprezentáció: egy n hosszú 0 – 1 sorozat. Ha az i . pozícióba 0-t írunk, akkor azon tárgy nem kerül be a hátizsákba, ha 1-et, akkor bekerül
- ▶ Fitnessz: a bekerült tárgyak összértéke, ha a súlykorlát alatt vagyok, nulla egyébként (jobb ötlet?). Cél ennek a maximalizálása
- ▶ Egyed: egy lehetséges pakolás
- ▶ Populáció: lehetséges pakolások egy részhalmaza, ismétlődés előfordulhat köztük
- ▶ Genotípus: az egyedet leíró 0 – 1 sorozat
- ▶ Fenotípus: egy konkrét pakolás

Az általános séma



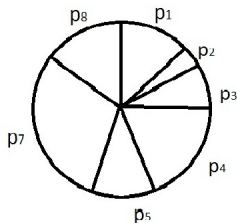
Szülők kiválasztása

A jó minőségű, azaz magas fitnessű megoldásokat választjuk ki a következő generáció számára. Fontos, hogy a rossz minőségű megoldások is kapjanak egy kicsi, de pozitív esélyt, hogy megakadályozzuk a lokális optimumba beleragadást. Azaz fenntartjuk a populáció diverzitását.



Rulettkerék módszer (Fitnessz-arányos kiválasztás)

Tegyük fel, hogy az adott populáció egyedeinek fitnessze sorra $f_1, f_2, \dots, f_k \geq 0$. Ebből készítünk egy valószínűségeloszlást, úgy hogy a fitnessz vektort normáljuk, hogy az elemeinek összege 1 legyen, így kapjuk a p_1, p_2, \dots, p_k valószínűségeket.



Majd kiválasztunk egymástól függetlenül k darab egyedet (szülőt), úgy, hogy a i . egyed p_i eséllyel lesz kiválasztva minden körben.

Problémák a rulettkerék kiválasztással

- ▶ Egy kimagasló megoldás túl gyorsan elterjedhet, így az eljárás beleragad a lokális optimumba
- ▶ Az eljárás végén kicsi a különbség az egyedek között, így nagyon lelassul a konvergencia
- ▶ A kiválasztott egyedek elfordulása nagyon eltérhet az elméleti valószínűségtől
- ▶ Nagy hatással van az eljárásra a fitnessfüggvény alakja

Az első két probléma megoldására skálázást alkalmazhatunk.

Skálázás

A fitnessarányos kiválasztásnál problémákat vet fel az optimális skálázás megtalálása, hogy fenntartsuk a megfelelő szelekciós nyomást a konvergencia lassuló fázisában.

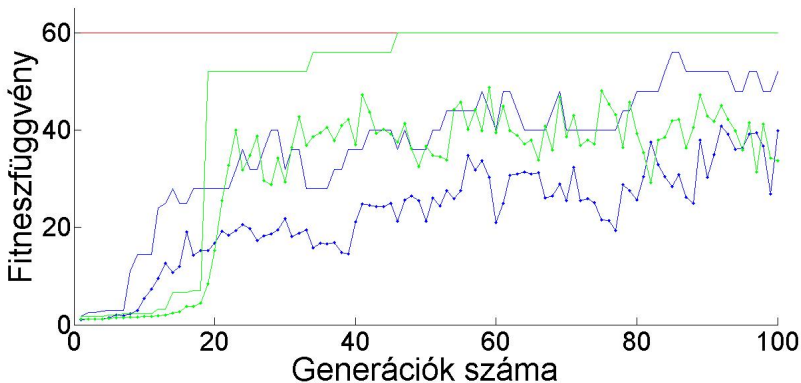
A különbségek növelésére alkalmazhatunk x^n illetve $\exp(x)$ jellegű skálázást, a fitnessértékek különbségét csökkenthetjük $\ln(x)$ illetve \sqrt{x} típusú skálázófüggvénnyel.

Elitizmus

Hiába találja meg akár az optimális megoldást az algoritmus, mégis idővel csökkenhet a megoldások minősége. Ennek a jelenségnek a kiszűsöbölésére alkalmazhatjuk az elitizmust (elitism), ami azt jelenti, hogy az adott generáció legmagasabb fitnessű egyede változtatás nélkül átkerül a következő generációba (egyetlen példányban).

Ez az elit megoldás helyettesíthet egy véletlenszerűen kiválasztott utódot, vagy a legkisebb fitnessűt. Az elitizmus garantálja, hogy a következő generációban a legjobb fitnessű egyed fitnessze sohasem csökken, viszont használata növeleli a korai konvergencia veszélyét.

Elitizmus hatása



Tournament-kiválasztás

Egy másik lehetséges megoldás a szülők kiválasztására a tournament kiválasztás. Ekkor úgy választjuk ki a szülőket, hogy kiveszünk néhány (mondjuk k darab) egyedet a populációból véletlenszerűen, majd közülük a legjobb lesz a szülő. Ezt az eljárást megismétlem annyiszor, ahány szülőre szükség van.

Ha k értéke nagy, akkor kisebb eséllyel kerül a szülők közé alacsony fitnessű egyed, ezért ez a k érték egyben a genetikus algoritmus egy bemenő paramétere, mellyel a szelekciós nyomás szabályozható. Ezen paraméter akár futás közben is változtatható (a lassuló konvergencia gyorsítására csökkentjük k értékét).

Genetikus operátorok

A kiválasztott szülőkből hozzuk létre az utódokat. Első lépésben néhány kiválasztott szülőt keresztezünk. A keresztezés lehet

- ▶ **egyponos** kiválasztunk egy pozíciót, mindkét szülőt félbetörjük ott, majd a darabokat keresztsben összeragasztjuk
pl 11100 és 00011 utódai a 2 pozicóban törve: 11011 és 00100
- ▶ **többpontos** hasonló az egypontoshoz, de több helyen török
- ▶ **uniform** minden pozícióban $\frac{1}{2}$ eséllyel döntök az egyik vagy a másik szülő génje mellett

A keresztezés célja, hogy a magas fitnessű egyedek tovább örökítsék a jó tulajdonságaikat. Ehhez szükséges a problémát jól kódolni.

A mutáció

A mutáció célja a keresési tér felderítése és a megoldások lokális optimumba ragadásának megakadályozása. Viszont, ha túl erős a hatása, akkor tönkreteszi a konvergenciát. Az optimális paraméter megtalálása ezért fontos feladat.

Ökölszabály a mutáció valószínűségére: általában egy gén/generáció és egy egyed/generáció között legyen a mutációk számának várható értéke.

Pédánkban megfelel mutáció operátornak az, hogy az egyed 0 – 1 sorozatának egy véletlen pozíciójában kicserélem az értéket valamekkora eséllyel.

Kilépési feltétel

A legegyszerűbb kilépési feltétel, hogy lefuttatunk előre adott számú generációt, majd az összes addigi generáció legmagasabb fitnessű egyede lesz a tippünk a globális optimumra.

A későbbiekben lesz szó ennél összetettebb feltételekről is, például megfigyelhetjük az eljárás konvergenciáját, majd ez alapján döntünk a kilépésről.