Evolúciós algoritmusok 7. előadás

Genetikus programozás és alkalmazások

Genetikus programozás

Az evolúciós algoritmusok azon válfaját, amikor a megoldásokat fákkal reprezentáljuk, nevezzük genetikus programozásnak.

A faszerkezet természetes reprezentációja például az aritmetikai kifejezéseknek, logikai kifejezéseknek vagy akár programkódoknak.

Ilyen alakú a megoldások reprezentációja néhány görbeillesztési feladatban vagy például a döntési fáknál.

Prefix alak

Egy kifejezés prefix alakjában az operátort írjuk előre, majd utána zárójelek között az operandusokat. Vegyük az

$$\sin\left(\frac{xy}{x+y+1}\right)$$

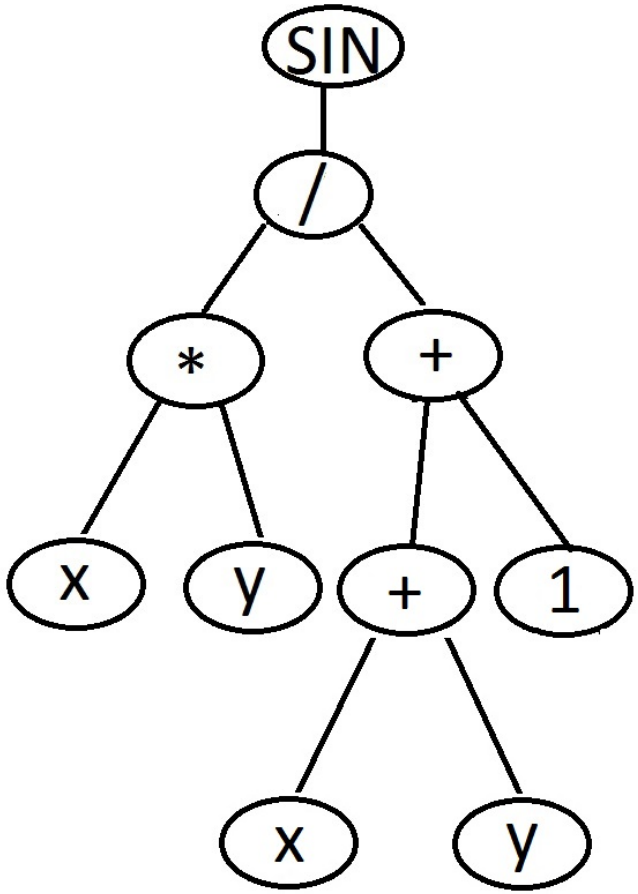
kifejezést. Ennek prefix alakja:

$$\sin(/(*(x,y),+(+(x,y),1)))$$

A prefix alakot könnyen tudjuk átírni egy faszerkezetbe. A csúcsok felelnek meg az operátoroknak, és minden csúcsnak annyi gyereke lesz, ahány operandusa van az operátornak. Viszont ha az operátorok nem kommutatívak, akkor a gyerekek sorrendje is lényeges.

Példa

$$\sin(/(*(x,y),+(+(x,y),1)))$$



Fa reprezentáció

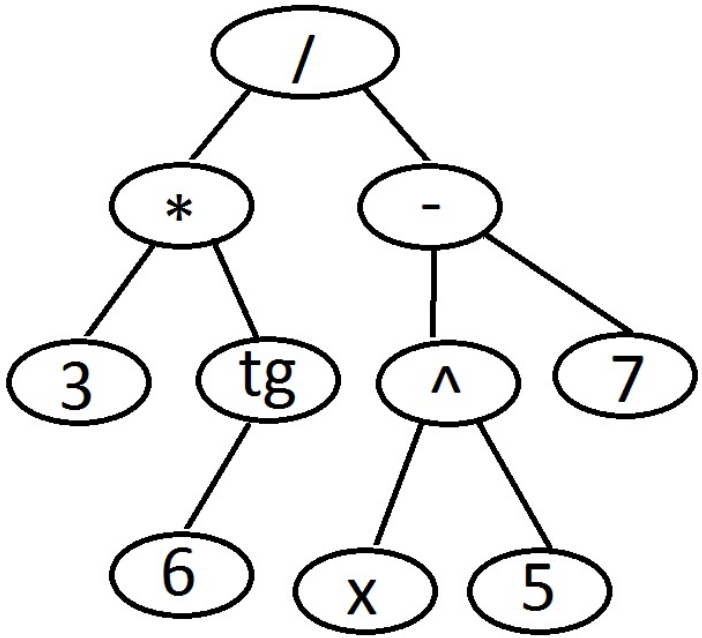
A fák felépítéséhez kétféle csúcsot használhatunk:

- operátorokat, amelyek csak belső pontokban helyezkedhetnek el
- operandusokat, amelyek csak a leveleken

A fa ismeretében a prefix kifejezést egy mélységi bejárással kapjuk vissza. A fa gyökerétől indulva, amíg lehet a baloldali utódokon haladunk, amíg elérünk egy levélig. Itt visszafordulunk, és az első lehetőségnél jobba fordulunk. A csúcsok bejárásának sorrendjében tudjuk a kifejezést visszaépíteni.

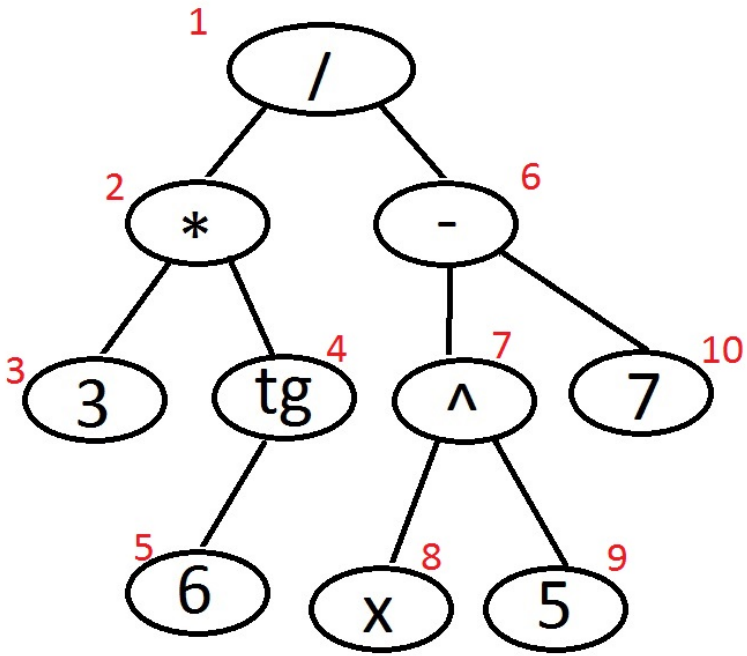
Példa

Írjuk fel, hogy az alábbi fa milyen aritmetikai kifejezést kódol!



Példa

Először elkészítjük a fa mélységi bejárását:



Ez alapján a prefix alak: $/(*(3, \mathrm{tg}(6)), -(\wedge(x, 5), 7))$, így:

$$\frac{3 \cdot \mathrm{tg}6}{x^5 - 7}$$

Logikai kifejezések

Hasonlóan reprezentálhatóak a tiszta logikai kifejezések is, például próbáljuk meg felírni a kódoló fát erre a kifejezésre:

$$(x \wedge \mathit{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$

Gondot jelenthet azonban, ha a megoldások reprezentációjához szükség van logikai és aritmetikai kifejezésre is. Bár

$$(N > 30) \vee (D < 20 \wedge M \geq 100)$$

egy helyes kifejezés, de például $N \wedge 100$ értelmetlen. Ekkor a szabályos fák építéséhez további kikötéseket kell tennünk.

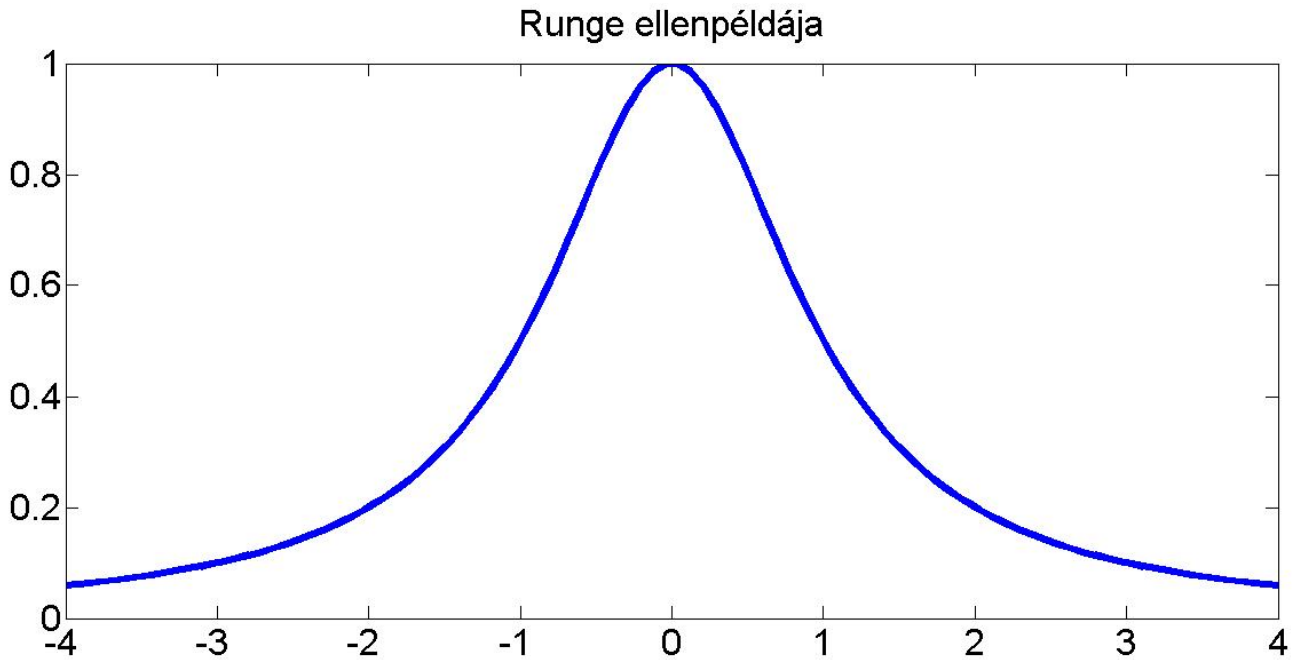
Példa: szimbolikus regresszió

Adott függvények egy halmaza: az alapműveletek, és mondjuk a $\sin, \cos, \exp, \mathrm{abs}$ függvények. Célunk, hogy egy ismeretlen célfüggvényből származó mennyiséget közelítsünk ezek kompozíciójával, ehhez adott néhány (például 20) pontban a célfüggvény értéke. Ekkor az előbbi műveletek az operátorok, és az $x \cup \mathbb{R}$ az operandusok.

A feladat még nem jól definiált. Ismert, hogy n ponton bármely függvényt interpolálni lehet egy $(n-1)$-ed fokú polinommal. Azaz van egy 19-ed fokú polinom, ami ezen 20 pontban az adott függvényértékeket veszi fel. Mi viszont nem ezt keressük, ugyanis ezen közelítés igen rossz tulajdonságokkal rendelkezik.

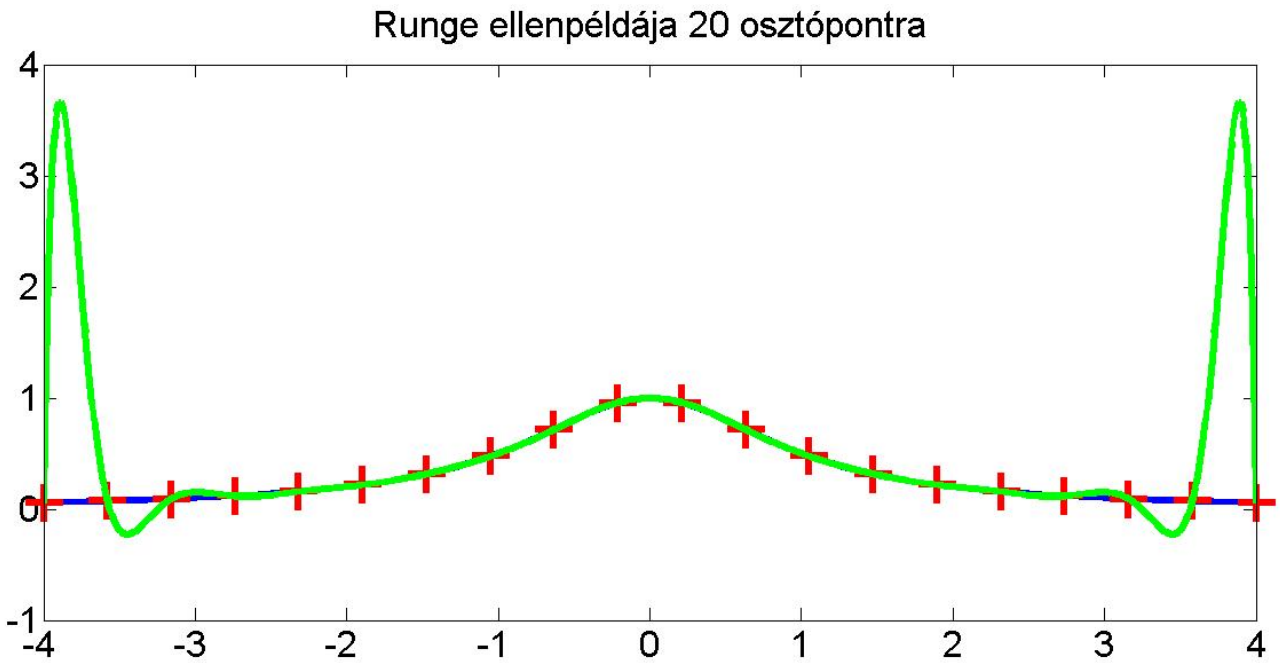
Runge ellenpéldája

Vegyük az $f(x) = \frac{1}{1+x^2}$ függvényt. Szeretnénk közelíteni a $[-4, 4]$ intervallumon. Vegyük az ekvidisztans (=egyenlő osztásközű) alappontokat, majd számítsuk ki a közelítő egyértelmű (Lagrange) polinomot.



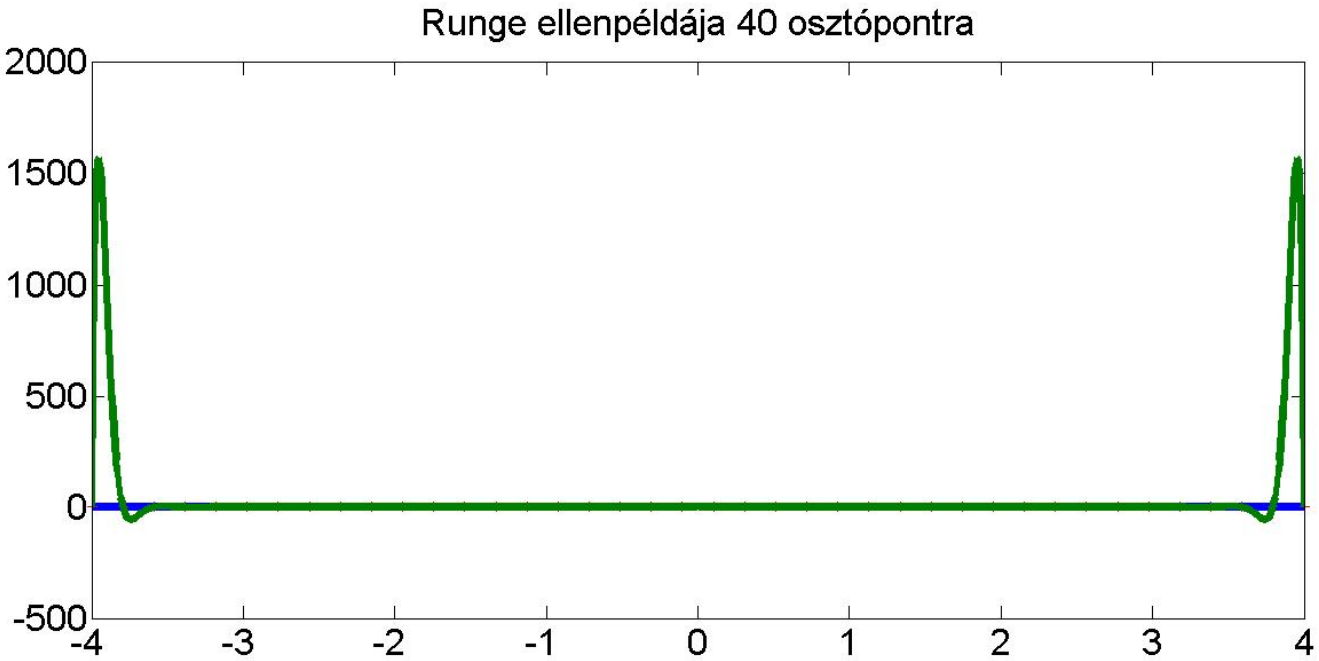
Runge ellenpéldája

A közelítő 19-ed fokú polinom:



Runge ellenpéldája

A közelítő 39-ed fokú polinom:



A magas fokú polinommal közelítés az intervallum szélei felé közeledve erősen oszcillál, ezért nem szokás használni. Helyette egy ,,rövid" közelítő kifejezést keresünk. A rövidséget az algoritmus futása alatt mi kényszerítjük majd ki.

Kezdeti populáció létrehozása

Meghatározunk egy maximum szintmélységet. Majd az alábbi két módszer egyikével hozzuk létre a fát:

- Minden ág maximum szintmélységű, és a csúcsokat egyenletes eloszlással választjuk az operátorok (belső pontok esetén), illetve az operandusok (levelek esetén) közül
- A gyökértől haladva indulunk. Minden csúcsot egyenletes eloszlással választunk az operátorok és operandusok uniójából. Ha elértük a maximális szintmélységet, akkor mindenképp operandust választunk a maradék csúcsokra.

Keresztezés és mutáció fa operátorokra

Eltérően a genetikus algoritmusok többségétől, a genetikus programozás nem használja mindkét operátort, hanem véletlenszerűen választja az egyiket.

A mutáció egy új véletlen részfa hozzákapcsolását jelenti valamely csúcshoz. Ha ez belső csúcs volt, akkor az onnan kiinduló eredeti részfát eldobjuk.

Keresztezéskor pedig kiválasztunk egy-egy véletlen csúcsot a két szülőn, és az onnan induló részfákat kicseréljük.

Populációméret

A fitnesz maximalizálása és a mutáció méretnövelő hatása is a fák növekedését okozza (bloat). Ezt nekünk kell megakadályozni: vagy úgy, hogy veszünk egy maximális megengedett szintmélységet, vagy úgy, hogy büntető függvényt alkalmazunk a túl nagy méretű fákra.

Mivel tipikusan nagy méretű populációkkal dolgozunk, ezért a szelekciós nyomás nagyon erős. Hogy mégis biztosítsuk a kisebb fitneszű egyedeknek mérhető esélyt a túlélésre, ezért alkalmazzuk a 80/20-as szabályt. Azaz a szülők 80%-át választom a magas fitneszű egyedek közül, 20%-át az alacsony fitneszűek közül. Azt, hogy hol van az alacsony és magas fitnesz közötti határ, a populációméret dönti el.

Párhuzamosítás

Célunk, hogy a futásidőt lerövidítsük a genetikus algoritmus párhuzamosításával. Ehhez több különálló szigetre osztjuk a kezdeti populációt, és megtervezzünk a szigeteket, mint csúcsokat tartalmazó gráfot (lehet ez egy egyszerű kör is).

Fix számú generáció (epoch) után a szomszédos szigeteken lévő egyedek egy részét kicseréljük (migráció).

Ha az epoch túl rövid, akkor ugyanoda konvergálnak a megoldások, ha túl hosszú, akkor sok számítási idő megy el lokális optimumok környezetének felderítésére. Itt célszerű adaptívan választani a hosszt: egy szigeten megállítjuk az evolúciót, ha egy előre meghatározott idő (mondjuk 25 generáció) óta nem tapasztaltunk javulást.

Párhuzamosítás

Kérdés, hogy mennyi egyedet cseréljünk ki, és hogyan válasszuk ki őket: véletlenszerűen vagy a legjobb egyedeket.

A véletlenszerű választás esetén kisebb az esély arra, hogy egy magas fitneszű migráns az egész szigetet dominálja. Hosszabb epoch esetén viszont már homogén a szigetek populációja, így a választás mikéntjének kevesebb jelentősége van.

Lehetséges a különböző szigeteken különböző paramétereket használni, akár a mutáció és keresztezés valószínűségében, de akár a reprezentáció módjában is. Például egy folytonos optimalizálási feladatnál annak megválasztásában, hogy hány jegyet reprezentálunk a megoldásból. Ilyenkor azonban a migrációnál körültekintően kell eljárnunk.

Optimális tömörítés

Legyen adott egy felület a térben egy kétváltozós függvény segítségével egy rácson. A célunk az, hogy a rácspontok egy részét kiválasztva és ezeken interpolálva jó közelítését kapjuk az eredeti függvénynek.

A kiválasztandó rácspontok aránya előre adott (egy és tíz százalék között), kérdés, hogy a pontok melyik részhalmazát tartsuk meg. Erre a problémára adunk genetikus algoritmust.

Az egyik felmerülő kérdés, hogy miként mérjük az adott rácspont-részhalmaz jóságát. A fitnesz-függvényt az adott részhalmaz szög optimális háromszögelésének segítségével definiáljuk.

Az optimális háromszögelés

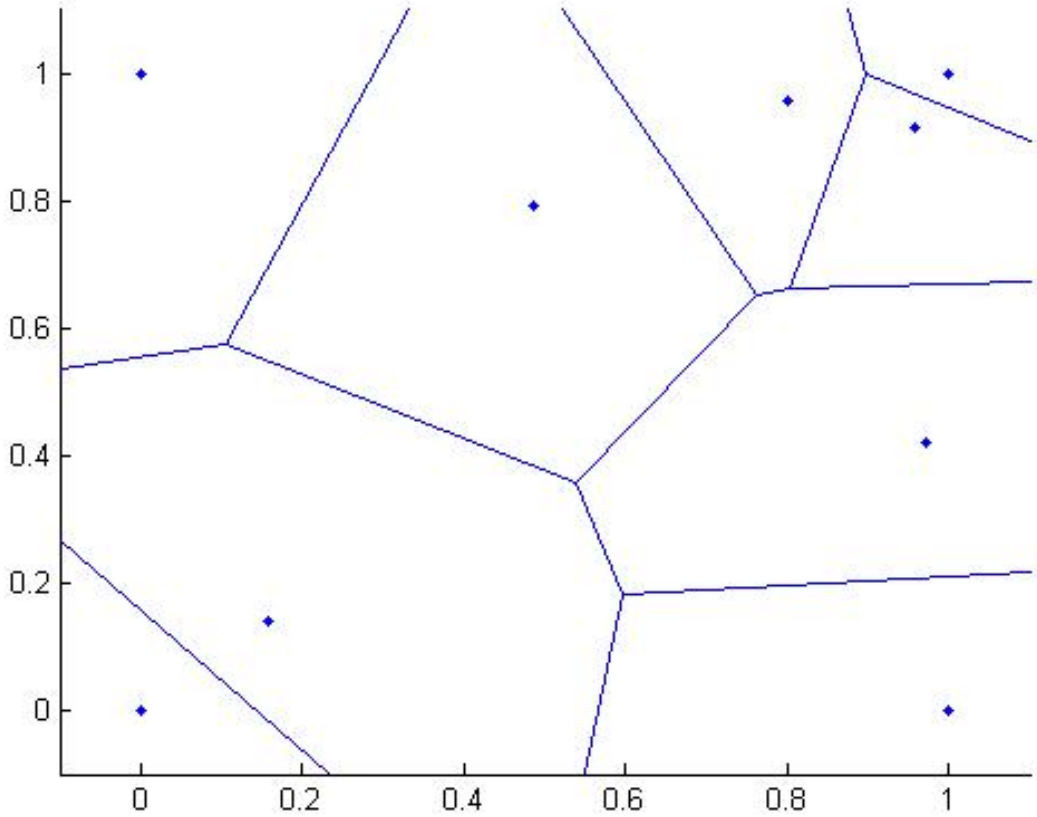
A függvény közelítéséhez olyan tartományonként lineáris függvényt használunk, ahol a tartományokat a rácspontok egy háromszögelése adja.

Pontok egy véges halmaza exponenciálisan sokféleképpen háromszögelhető, mert minden konvex négyszögbe kétféleképpen húzhatjuk be az átlót, és 5 általános helyzetű pontból minimum 4 konvex helyzetben van, így van legalább $2^{\frac{n}{5}}$ háromszögelés.

Olyan háromszögelést keresünk, amiben nincsenek nagyon kicsi szögek (azaz hosszú lapos háromszögek). Vegyük az adott háromszögelés összes szögét sorrendben tartalmazó $\underline{\alpha}$ vektort. Ezek közül lexikografikus sorrendben a legnagyobbat nevezzük szög optimálisnak (ez nem feltétlen egyértelmű). Ehhez szükség van egy kis előkészületre.

Voronoi–cella

Egy ponthalmaz egy adott P pontjához tartozó Voronoi–cella azon pontok összessége, amelyek P-hez vannak legközelebb.

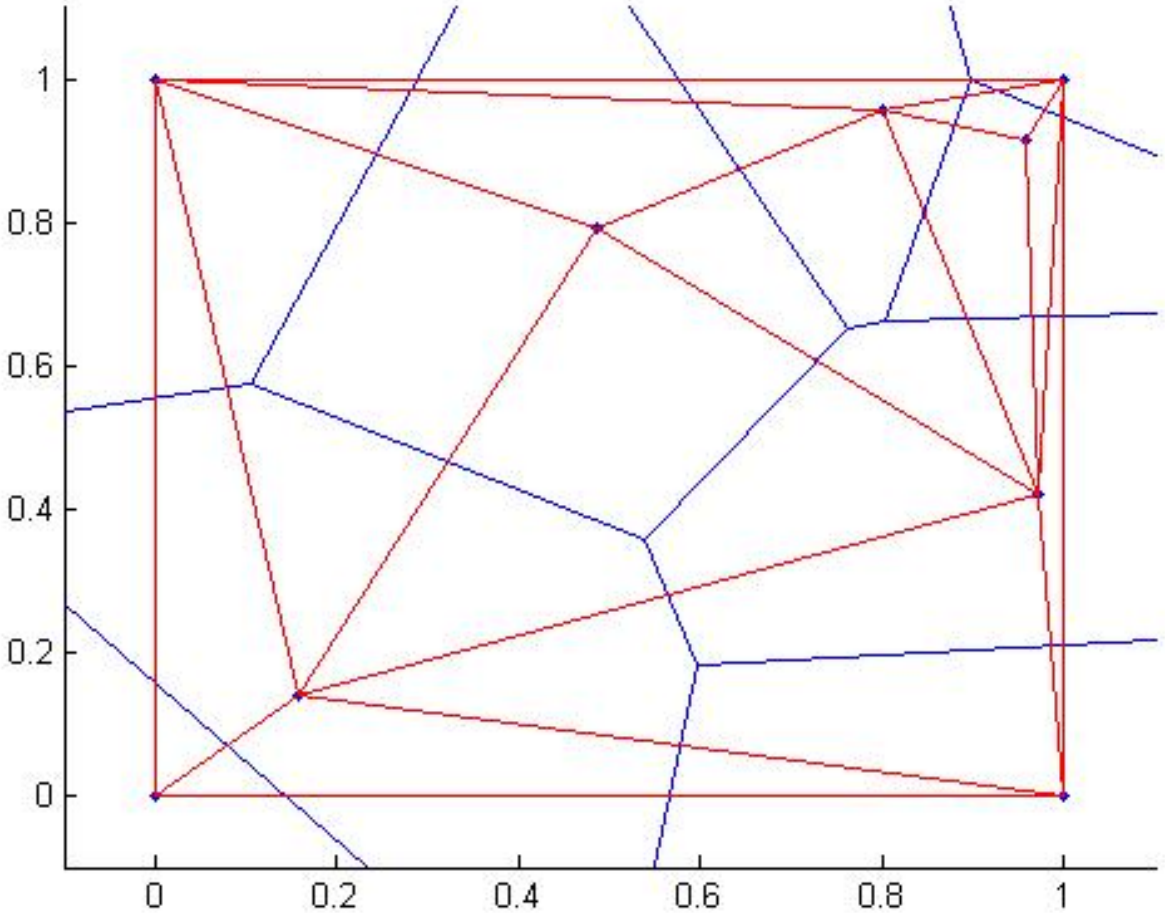


Delaunay–háromszögelés

Az adott ponthalmazra illeszkedő háromszögelések közül szög optimálisat ad a Voronoi-cellák duális gráfja (ezt nem bizonyítjuk).

A duális gráf elkészítéséhez a következőképpen járunk el: az egyes cellák lesznek a duális gráf csúcsai, két csúcs akkor van összekötve a duális gráfban, ha a cellák szomszédosak. Ezt nevezzük Delaunay–háromszögelésnek.

Delanuay–háromszögelés



Delanuay–háromszögelés

Ha a ponthalmaz elemeinek száma n és konvex burkon k pont van, akkor $2n - 2 - k$ darab háromszögből áll a háromszögelés.

Biz Minden él két tartományhoz tartozik, és a belső (korlátos) tartományokat 3 él határolja, a külsőt pedig k. Mivel a háromszögelés egy síkgráf, így az Euler-formulába $c - e + l = 2$ helyettesítve

$$n - \frac{3(l-1)+k}{2} + l = 2$$

innen

$$2n - 3l + 3 - k + 2l = 2$$

átrendezve l-re kapjuk a kívánt állítást.

Fitnesz függvény

Az eredeti felület közelítése azon háromszög lapokkal történik, amelyek csúcsait a Delaunay–háromszögelés határozza meg, a csúcsokon felvett függvényértékek adják a háromszöglap térbeli elhelyezkedését.

Kiszámítjuk, hogy a háromszögelés belsejébe eső rácspontokra mekkora az eltérés a közelítés és a függvényértékek között, ezeket összegezzük az összes rácspontra. A közelítés jellegéből következik, hogy a csúcspontokon a közelítés hibája 0.

Az összeg adja a közelítés teljes hibáját, amely a nyers fitnesz, célunk ennek minimalizálása.

Genetikus operátorok

Kezdeti pontok felvétele

Szeretnénk elkerülni a nagyon szabálytalan alakzatokat, ezért a teljesen véletlenszerű pontválasztás helyett (amennyiben összesen k darab pontra van szükség) a határoló éleken felveszünk $\frac{\sqrt{k}}{2}$ pontot, csak a többit valóban véletlenszerűen.

Keresztezés

Válasszunk véletlenszerűen egy tengelyt, majd ezen tengelyen egy konstans értékeket. Mindkét rácsot félbevágjuk a határoló konstans mentén, majd a két felet összeragasztjuk, hasonlóan az egypontos keresztezéshez. Ekkor még a rácspontok számát valamilyen módszerrel (új pontok felvétele, régiek törlése) rendbe kell raknunk.

Mutáció

Egy pontot áthelyezünk egy szomszédos rácspontba.