Evolúciós algoritmusok 4. előadás

Constrain handling, leállási feltételek

2017.10.20.

A problémák típusai

Tegyük fel, hogy az optimalizálási feladat megoldásainak leírásához az $x_1, x_2, \ldots, x_n$ változókat használjuk, és ezek értéke a $D_1, D_2, \ldots, D_n$ tartományból kerülhet ki.

Ekkor a szabad keresési tér a $D_1 \times D_2 \times \cdots \times D_n$. Ha ennek egy valódi részhalmazán keresünk optimális megoldást, akkor beszélünk megkötésről ('constrain').

A feladatok osztályozása:

- Nincsenek megkötések, egy célfüggvényt maximalizálunk (FOP: free optimalisaton problem)
- Vannak megkötések, egy ezeket kielégítő megoldást keresünk (CSP: constrain satisfaction problem)
- Vannak megkötések, ezek mellett keressük a célfüggvény maximumát (COP: constrained optimalisation problem)

Példák COP-re

Hátizsák-pakolás

A változók száma megegyezik a tárgyak számával. A változók értéke 0 vagy 1 lehet. A szabad keresési tér a $\{0,1\}^n$, de a probléma tartalmaz egy megkötést is: nem választhatunk a hátizsák kapacitásánál nagyobb összsúlyú pakolást. Mivel a legnagyobb összértékű pakolást keressük, így ez egy COP.

Utazó ügynök probléma

Legyen a városok száma n. Ha potenciális megoldásnak az n hosszú, 1 és n közötti egészeket tartalmazó listát vesszük, akkor még szükség van egy megkötésre is: minden szám csak egyszer szerepelhet. Mivel ezen feltétel mellett keressük a legrövidebb utat, így az utazó ügynök probléma egy COP.

Példák CSP-re

Gráfszínezés k színnel

Adott egy gráf a szomszédsági mátrixával, kérdés, hogy ki lehet-e színezni k színnel a gráfot. Ez egy CSP, hiszen a megkötés az, hogy két szomszédos csúcs nem lehet azonos színű. De nem kell optimalizálni, bármelyik jó színezés eldönti a kérdést.

n-királynő

Adott egy $n \times n$-es sakktábla, el szeretnénk helyezni n darab királynőt úgy, hogy semelyik kettő ne üsse egymást. A feladat, hogy keressünk egy jó lerakást. Itt sincs optimalizálás, így ez egy CSP.

Genetikus operátorok szerepe a feltételek kielégítésében

Direkt megközelítésnek nevezzük, ha a megkötések kielégítését az evolúciós algoritmus futása során kikényszerítjük.

Erre egy lehetséges megoldás, ha az operátorok a megkötéseket kielégítő (feasible) megoldásból ilyen tulajdonságú utódot állítanak elő. Ekkor egy megkötéseket teljesítő kezdeti populációból kiindulva végig ilyen megoldásokat kapunk.

Példa. Az utazó ügynök-feladatban a megkötést úgy lehet kezelni, hogy permutációkból indulunk ki, és olyan operátorokat használunk, amelyek permutáció szülőkből permutáció utódokat készítenek.

Dekódolás

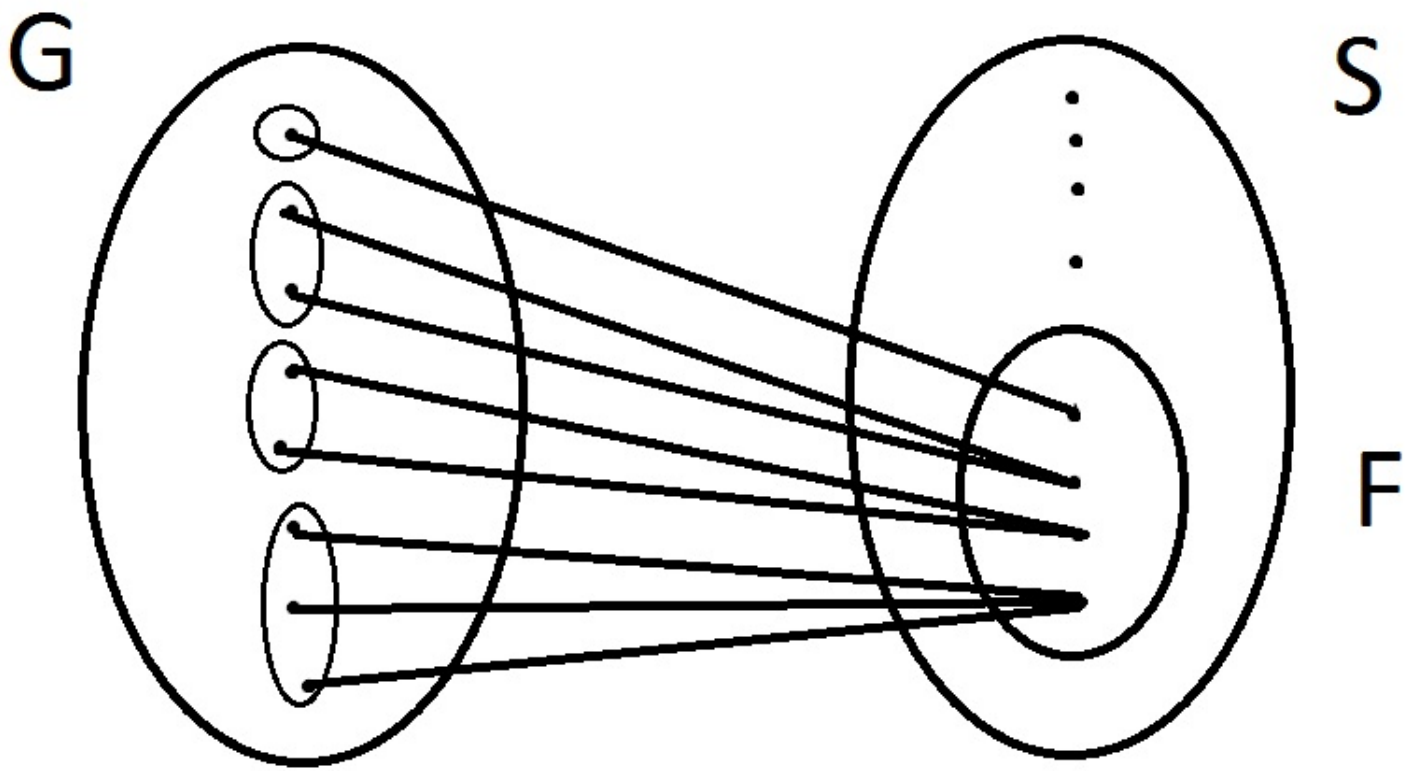
Egy másik lehetőség, hogy a megkötéseket sértő megoldásokat átalakítjuk egy dekódoló leképezés segítségével a megkötéseket kielégítő megoldássá.

Legyen S a keresési tér, $F \subset S$ a megkötéseket kielégítő megoldások halmaza, G a genotípusok halmaza.

Egy $G \to F$ leképezés dekódolás, ha

- Minden $g \in G$ képe egyértelmű és F-beli
- Minden $s \in F$-nek van legalább egy G-beli őse

Dekódolás



Példa: hátizsák pakolás

Most S az összes lehetséges pakolás, míg F a legfeljebb kapacitás összsúlyú pakolás, G a szokásos összes n-hosszú $0-1$ sorozat lesz.

Haladjunk végig az egyeden balról jobbra, és definiáljuk a dekódolást a következő módon: a 0 jelentse azt, hogy nem vesszük be az adott tárgyat, az 1 pedig azt, hogy bevesszük, feltéve, hogy még nem léptük át a kapacitást.

Így minden 0-1 sorozathoz egyértelműen hozzárendelünk egy F-beli megoldást. Továbbá minden F-beli megoldásnak megfelel legalább egy sorozat, hiszen a dekódoló a kapacitást nem túllépő pakolásokat változatlanul hagyja.

Példa folyt.

Tipikus esetben egy F-beli megoldásnak több őse van, ezek száma sem egyforma. Legyen a tárgyak száma 8, és legyen belőlük két fajta (mindkettőből 4-4 darab, az első 4 az 10 súlyú): az egyik értéke 100, a súlya 10, a másik értéke és súlya is 1.

Vegyük a kapacitást 12-nek. Ekkor azon f_1 pakolás, melyben bevesszük a második, ötödik és hatodik tárgyat, kielégíti a megkötést. Hasonlóképpen azon f_2 pakolás is, melyben a harmadik, ötödik és hatodik tárgyat veszem be.

Viszont f_1-nek 16 őse van, a $01**11**$ sorozatok, melyben a $*$ helyén tetszőleges bit áll, míg f_2-nek csak 8, a $001*11**$ típusú sorozatok.

Példa: Gráfszínezés

Legyen adott egy n csúcsú G gráf a szomszédsági mátrixával.
Keressünk egy jó 3 színezését.
Ezen feladatban természetes választás, hogy a megoldásokat egy n hosszú 1,2,3 sorozattal reprezentáljuk, aminek i.-dik eleme azt mondja meg, hogy az i.-dik csúcs milyen színű. Ez egy CSP, mert egy olyan sorozatot keresek, ahol azonos színű csúcsok között nincsen él.

Példa (folyt.): Gráfszínezés

A megkötés kezeléséhez a potenciális megoldásokat reprezentáljuk egy permutációval. Egy permutációból a színezést úgy nyerjük, hogy sorban haladok a csúcsokon a permutáció által megadott adott sorrendben, és megpróbálom a soron következő csúcsot 1-es színűre festeni. Ha nem sikerül, akkor 2-re, ez ez sem lehet, akkor 3-ra. Ha egyikkel sem járok sikerrel, akkor kiszínezetlenül hagyom. A genetikus algoritmus ezek után a színtelen csúcsok számára minimalizál, így egy COP-problémává alakítottuk az eredeti feladatot.

Példa: Utazó ügynök

Jelöljük a városok számát n-el. Most a potenciális megoldásokat olyan n hosszú számsorozatok fogják reprezentálni, melyekre teljesül, hogy az i. elemük $\leq (n + 1 - i)$. Más megkötés viszont nincs rájuk, tartalmazhatnak egy számot többször is. Ekkor használhatom az egész számokon ható egy- és többpontos keresztezés operátorokat.

Rögzítsük előre a városok egy sorrendjét (mondjuk A, B, C, D, E). Egy ilyen számsorozatból úgy készítjük el az utat, hogy sorban haladunk a listán. Az i. elem azt jelenti, hogy a következő meglátogatandó város hányadik a városok listájából. Ez leírjuk az útba, majd kihúzzuk a városok listájáról. Például a $\{4, 4, 1, 2, 1\}$ lista a DEACB sorrendet kódolja.

A fitnesz függvény szerepe a megkötések kielégítésében

A megkötések teljesítéséhez egy másik lehetőség, hogy átalakítjuk őket 'lágy feltétellé'. A megkötéseket nem kielégítő megoldást büntetjük azzal, hogy a fitneszét növeljük (ha az eredeti feladatban minimalizáltunk), majd erre a módosított fitneszre optimalizálunk.

Ekkor a megkötés kielégítésért az algoritmus optimalizációja felel, ezt nevezzük indirekt megközelítésnek.

A büntetés lehet

- statikus (minden generációban állandó)
- dinamikus (attól függ az értéke, hogy hányadik generációnál tartunk)
- adaptív (értéke a populáció egyedeinek minőségétől függ)

Statikus büntetőfüggvény

Olyan büntető függvényt kell alkalmaznunk, melynek értéke könnyen kiszámítható. Legyen összesen m darab megkötés, ekkor egy lehetséges választás a

$$P(x) = \sum_{i=1}^{m} w_i \cdot d_i(x),$$

ahol a $d_i(x)$ egy metrika, és az x megoldás távolságát jelenti F-től. A legegyszerűbb választás a d_i-ra, hogy értéke legyen 1, ha nem vagyok F-ben, 0 egyébként. A w_i-k a büntetősúlyok, melyek optimális meghatározása nehéz feladat.

Dinamikus büntető függvény

Ha a büntetés mértéke időben (generációnként) változik, akkor dinamikus büntető függvényről beszélünk, ezek általános alakja

$$P(x) = \sum_{i=1}^{m} w_i(t) \cdot d_i(x).$$

Mivel azt szeretnénk elérni, hogy a megoldások kezdeti felderítése után az eljárás vége felé már kielégítsék a megoldások a megkötéseket, ezért a büntetéseket időben növelni kell. A standard választás a $w_i(t) = (w_i \cdot t)^\alpha$, ahol $\alpha = 1$, vagy $\alpha = 2$.

Adaptív büntető függvény (SAW)

Itt a büntetés mértéke időben változik. A

$$P(x) = \sum_{i=1}^{m} w_i \cdot d_i(x)$$

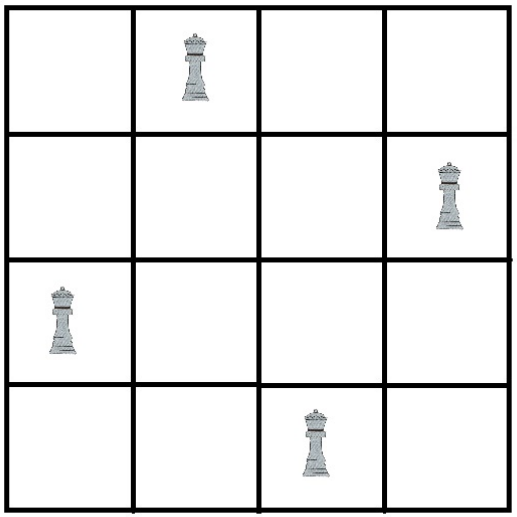
büntetésben a w_i súlyokat növeljük. Rögzítsünk előre egy T_p és egy δw mennyiséget (ezek az eljárás paraméterei).

A büntetőfüggvényt T_p fitneszkiértékelés után megváltoztatjuk. A w_i súlyokat megnöveljük δw mennyiséggel, de csak azokra a feltételekre, amelyet az aktuális legjobb egyed megsért.

Ezzel elérjük, hogy a legnehezebben kielégíthető megkötésekre (amit a legjobb megoldás sem tudott még kielégíteni) nagyobb hangsúlyt fektessen az algoritmus.

n-királynő probléma

Adott egy $n \times n$-es sakktábla, amelyre lerakunk n darab királynőt. Hány olyan elrendezés van, ahol nem ütik egymást?



n-királynő probléma brute force-szal

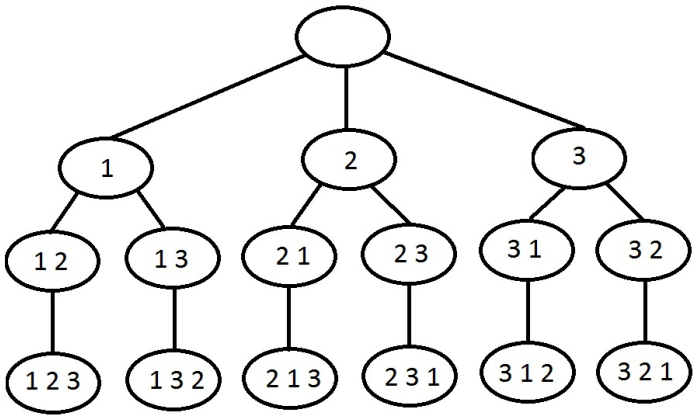
Első ötlet: nézzük meg az „összes" elrendezést, majd ahol nincsenek ütésben, azt megtartjuk. Elég csak a permutációkat vizsgálni, így $n!$ darab permutációt kell ellenőrizni.
Permutációként $\binom{n^2}{2}$ pár kell megvizsgálni, hogy ütik-e egymást átlósan. Ez összesen

$$n!\frac{n^2(n^2-1)}{2} \sim \left(\frac{n}{\mathrm{e}}\right)^n \sqrt{2\pi n}\frac{n^2(n^2-1)}{2}$$

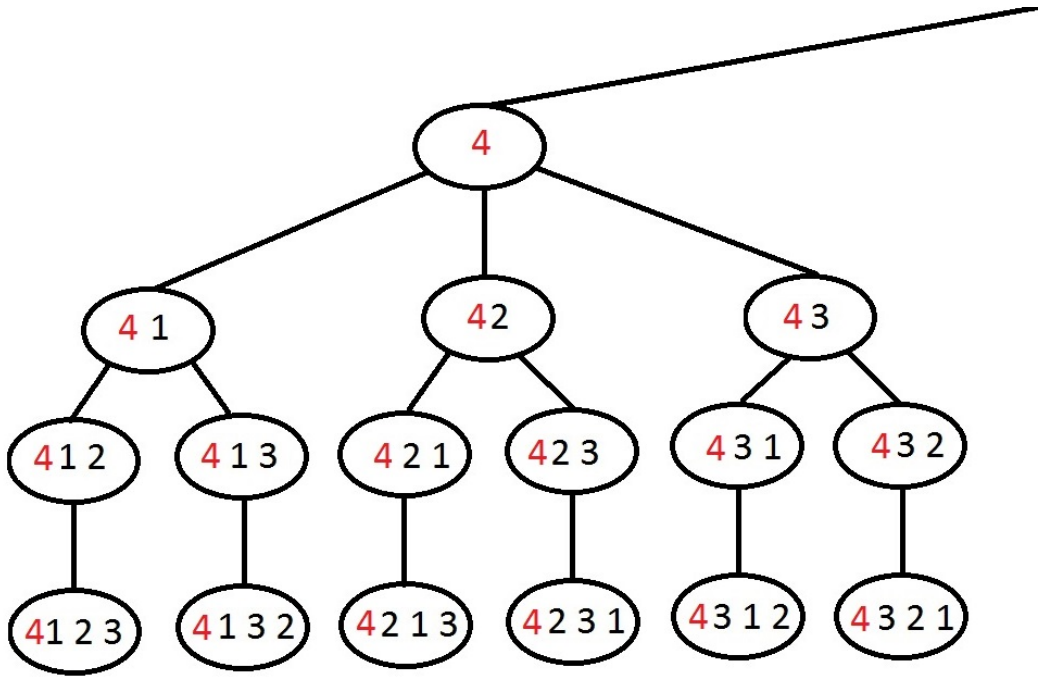
Például $n = 10$-re nagyjából $1.8 \cdot 10^{10}$ ellenőrzés, míg $n = 20$-ra $1.9 \cdot 10^{23}$.

n-királynő probléma branching-gel

Jobb ötlet: ha egy rész-lerakásban már van ütés, akkor az összes ezt tartalmazó lerakásban is lesz, ezeket már nem kell megvizsgálni. Egy fát építünk a potenciális megoldásokból (permutációkból), melynek levelein vannak a kész lerakások, a csúcsokon pedig a lerakások közös része.

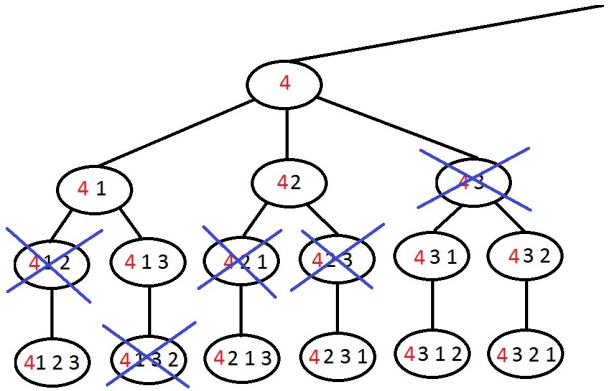


n-királynő probléma branching-gel



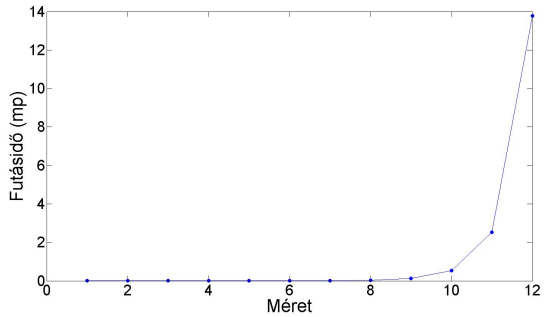
n-királynő probléma branching-gel

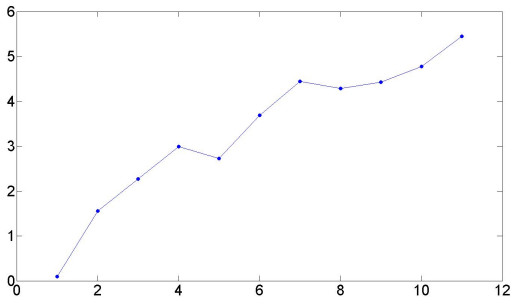
Ha egy rész lerakásban ütés van, akkor azt az ágat eldobhatjuk, az összes levéllel együtt.



n-királynő probléma branching-gel

Kérdés, hogy mennyit segít ez a futásidőn?





Még mindig exponenciálisnál rosszabb, így nagy n-re használhatatlan.
Megoldás genetikus algoritmussal?

Megállási feltételek

A megállási feltételekként a fix számú generáció futtatása helyett érdemes az alábbi lehetőséget is megvizsgálni:

- Addig futtatjuk, míg a fitnesz értékeke egy előre meghatározott értéket elér. Ilyenkor kell gondoskodnunk egy határról a generációk számában, hogy elkerüljük a végtelen ciklust. Ezt használhatjuk például a CSP típusú feladatoknál, ahol a 0 fitnesz azt jelenti, hogy minden feltételt kielégítettünk.
- Addig futtatjuk, míg a fitnesz értéke már nem változik (itt nem két egymást követő generációt érdemes figyelni, hanem pár tucatot). Ekkor nem pazarolunk több időt a lokális optimumok környékének felderítésével. Ez a feltétel alkalmas akkor, ha nincs előre információnk arról, hogy mi az elérhető legjobb fitnesz, azaz a FOP és COP típusú feladatoknál.