Zebra típusú logikai rejtvények megoldása evolúciós algoritmussal

Szili Dániel, Schöffer Fruzsina, Tóth Sándor Balázs, Varga Máté

Témavezető: Dr. Hegyháti Máté

2018. február 23.

Tartalomjegyzék

1. fejezet

Bevezetés

TODO: Absztrakt bovebben, szoveges tartalomjegyzek

2. fejezet

Zebra rejtvények

TODO: Egy bevezeto mondat, + hogy melyik alfejezetben mi lesz

2.1. Történet és szerkezet

TODO: Miert zebra, mikbol all a rejtveny, Einstein peldajabol reszlet akar

2.2. Megoldhatóság, egyértelműség

TODO: Pici peldakon bemutatni, hogy ha rosszak a szabalyok, akkor lehet nincs megoldas, vagy ha keves a szabaly, akkor lehet tobb megoldas is van. Egy nagyon apro (3 szek, 2 tulajdonsag mondjuk) pelda kitalalasa es megoldasa par lepesben.

2.3. Megoldó módszerek

TODO: Irodalomban talalhato modszerek, 2 mondat roluk, hivatkozasok

3. fejezet

Evolúciós algoritmusok

TODO: Tortenetuk, hivatkozasok TODO: Altalanos felepitesuk TODO: Akar par szo arrol, mi mindenre alkalmaztak oket, hivatkozasok

4. fejezet

Evolúciós algoritmus Zebra rejtvények megfejtésére

TODO: Par mondat az alapveto elgondolasrol, hogy melyik fejezetben mirol lesz szo

4.1. Kódszerkezet

TODO: Hogy van szervezve a kod, milyen fuggvenyek vannak,azok mikert fognak felelni roviden.

4.2. Egyedreprezentáció és segédfüggvények

TODO: Hogy reprezentaljuk az egyedet TODO: egyedkiir, sorbarendez, ...

4.3. Evolúciós mechanizmusok

TODO: esetleg par felvezeto szo, a kapcsolodo makrok megemlitese (popmeret, megtart)

4.3.1. Random új egyed generálás

TODO: Milyen volt az elso valtozat, hogy lett fejlesztve

4.3.2. Mutálás

TODO: Ugyanez. Milyen valtozatok voltak, vannak, részletesen bemutatva

4.3.3. Keresztezés

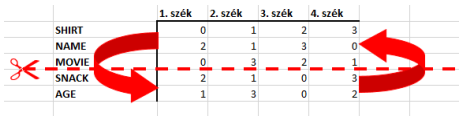
Keresztezésből alapvetően két fajtát különböztetünk meg. Az egypontos és a kétpontos keresztezést. Az egypontos keresztezés esetében a kromoszómákat véletlenszerűen választott helyen kettévágjuk, majd a felcserélt fél-kromoszómákból újakat hozunk létre. A 4.1 ábra szemlélteti a függvény működésének a lényegét. A kétpontos keresztezés hasonlóan működik, csak ebben az esetben 2 ponton vágjuk el az allélt és a keletkezett 3 darabot fűzzük össze tetszőleges sorrendben.

A mi esetünkben az egyedek allélja egy 2 dimenziós tömb, amelyen egypontos keresztezés került alkalmazásra. Itt 2 fajta választási lehetőség fordul elő. Vagy soronként vágunk vagy a tömböt szedjük ketté. Ebben az esetben az utóbbi eljárás került megírásra. A programok mindegyike egypontos keresztezést használ, ami a sorokat cseréli meg egy bizonyos ponton elvágva a tömböt. A program során 2 fajta keresztezés került

kidolgozásra. Az első verzió egy fix ponton vágta el a tömböt és a 2 felét cserélte meg. A fix pont a számtani közepe a tulajdonságok számainak. Ezzel a módszerrel az a probléma, hogy a folytonos közepén való vágás nem illeszkedik bele a genetikus algoritmus randomitásába. Későbbiekben ez a módszer egy változó segítségével javítva lett azt biztosítva, hogy minden egyedpár különböző helyen legyen elválasztva.

A program írása során két fajta keresztezés került kipróbálásra. A 4.2 ábrán látható crossover 1.0 is úgy lett kitalálva, hogy minden egyeden végezzen keresztezést, viszont az a hibája, hogy mindig 2 egymás mellett lévőn végzi el. Ez viszont nem bizonyult előnyösnek, mivel a populáció tömb rendezve volt így a legjobb egyedek egymás között keresztezve igen nagy eséllyel rosszabb egyedet adtak eredményképpen. A 4.2 ábrára visszautalva, ezt igen egyszerűen ki lehetett javítani azzal, hogy a populáció tömbből véletlenszerűen választunk ki 2 egyedet és azokon végez a program keresztezést. Ez a módszer lehetővé teszi a programnak azt is, hogy 2 egyforma egyedet válasszon ki, ezzel fenntartva a lehetőséget, hogy egy-egy egyed keresztezés nélkül kerüljön be a temp tömbbe. Az 1.0-ás változathoz képest jelentős eltérés, hogy ez a módszer nagy eséllyel hagy olyan egyedeket amiken nem végez keresztezést, mivel nem kerül kiválasztásra. Azonban ez a genetikus algoritmus jegyeit jobban mutatja, mivel 2 véletlenszerűen kiválasztott egyeden végez keresztezést egy véletlenszerűen választott pontban.

A keresztezés POPMÉRET db alkalommal fut le, azonban a többi evolúciós algoritmushoz hasonlóan innen sem maradt ki a keresztezés a mutált egyedeken , ami szintén POPMÉRET db-szor fut le. Mivel a temp tömb 4*POPMÉRET méretű így a keresztezett egyedek pont a temp felét teszik ki végül, így ebben az algoritmusban is a keresztezés dominál.



4.1. ábra. A keresztezés mechanizmusa

```
372
373         // Crossover 1.0
374         temp[k]=Keresztez(populacio[POPMERET-1],populacio[0]);
375         k++;
376         for (j=0;j<POPMERET-1;j++){
377             temp[k]=Keresztez(populacio[j],populacio[j+1]);
378             k++;
379         }
380
381         // Crossover 2.0
382         for (j=0;j<POPMERET;j++){
383            int x=rand()%POPMERET;
384            int y=rand()%POPMERET;
385            temp[k]=Keresztez(populacio[x],populacio[y]);
386            k++;
387         }
388
```

4.2. ábra. Keresztezési módok kódja

4.3.4. Megold függvény

TODO: Ugyanez. Milyen valtozatok voltak, vannak, részletesen bemutatva

4.4. Egyedek kiértékelése

TODO: Itt is szepen be lehet mutatni, hogy hogyan fejlodott, meg meg lehet mutatni mind az ot-hat tipusra egy peldat

5. fejezet

Tesztek

TODO: Futtatasi eredmenyek, megoldasok megmutatasa, stb.

6. fejezet

Kód automatikus generálása

TODO: Miert akarjuk TODO: Hogy csinaltuk TODO: pelda

7. fejezet

Összefoglalás

TODO: Mit csinaltunk roviden

Hivatkozások

A. függelék

Mintafejezet

A.1. Kep betoltese

Kep betoltese a `Abra` makroval az alabbiak szerint. Az elso parameter a fajl neve, ebbol lesz egy fig_fajlnev cimke, amit ref-ekhez lehet hasznalni. MAsodik parameter a kepalairas, a harmadik a meret.

```
\Aref{fig_zebra} abran bla bla.
\Abra{zebra}{Talalo kepalairas}{width=6cm}
```

Az A.1 abran bla bla.

A.1. ábra. Talalo kepalairas

A.2. Tablazatok

Tablazatot a `Tablazat` makroval lehet csinalni az alabbiak szerint. Az elso parameter a cimke, ebbol lesz egy tab_cimke cimke, amit ref-ekhez lehet hasznalni. MAsodik parameter a tablazat cime, a harmadik az oszlopok szerkezete (lasd tabular tutorial), az utolso maga a tablazat tartalma.

```
Ahogy az \aref{tab_cimke} tablazatban lathato, bla bla bla.
\Tablazat{cimke}{Tablazat cime}{r||cc|l}
{
    Elso sor elso cella jobra igazitva & kozepre & kozepre & balra \\
```

```
    \hline
    Masodik sor & & yay & much wow\\
    ize & bize & mize & meh... \\
}
```

Ahogy az az A.1 tablazatban lathato, bla bla bla.

Elso sor elso cella jobra igazitva	kozepre	kozepre	balra
Masodik sor		yay	much wow
ize	bize	mize	meh...

A.1. táblázat. Tablazat cime

A.3. Forraskodok beemelese

Forraskodot vagy a `Forraskod` makroval lehet betolteni a fajlbol, ahol az elso parameter a fajl neve, a masodik tovabbi opciok, pl hogy melyik sortol melyik sorig, stb. A masik lehetoseg, hogy a kodba keruljon be az alabbiak szerint:

```
\Forraskod{ize.c}{frame=single,lastline=4}
```

```c
int fuggvenynev(int tralala){
    int a;
    int b=3; // egy nagyon hasznos komment
    return a+3*b-masikfuggveny(tralala);
```

```
\begin{lstlisting}[frame=single, language=C]
    printf("Hello world\n");
    if(whatever){
        return 1;
    } else do {
        tanuljmegindentalni();
    } while (i<5);
\end{lstlisting}
```

```c
    printf("Hello world\n");
    if(whatever){
        return 1;
    } else do {
        tanuljmegindentalni();
    } while (i<5);
```