[Külső fekete borítólap formátuma]

Széchenyi István Egyetem
Gépészmérnöki, Informatikai és Villamosmérnöki Kar
Informatika Tanszék

SZAKDOLGOZAT

Herenkovics András
Mérnök Informatikus BSc szak

[2018]

SZÉCHENYI ISTVÁN EGYETEM

SZAKDOLGOZAT

Csomagok kiszállításának optimalizálása

Herenkovics András

Mérnök Informatikus BSc szak

[2018]

NYILATKOZAT

Alulírott, Herenkovics András (AQLRX3), Mérnökinformatika, BSc szakos hallgató kijelentem, hogy a Csomagok kiszállításának optimalizálása című szakdolgozat feladat kidolgozása a saját munkám, abban csak a megjelölt forrásokat, és a megjelölt mértékben használtam fel, az idézés szabályainak megfelelően, a hivatkozások pontos megjelölésével.

Eredményeim saját munkán, számításokon, kutatáson, valós méréseken alapulnak, és a legjobb tudásom szerint hitelesek.

Győr, [2018. december 9.] _____

hallgató

KIVONAT

Csomagok kiszállításának optimalizálása

[1 oldalas, magyar nyelvű tartalmi kivonat]

ABSTRACT

Optimalization of package delivering

[1 oldalas, angol nyelvű kivonat]

TARTALOMJEGYZÉK

BEVEZETÉS

Témaválasztásomnál olyan feladatot próbáltam keresni, mellyel nemcsak a programozási készségeimet tudom fejleszteni, hanem elméleti kihívást is biztosít. Így kerültem szembe a csomagok kiszállításának optimalizálásával, aminek elvégzésével mindkét célomat elérhetem.

Szakdolgozatom egyik fő célja, ha nem a legfontosabb, hogy a futárok számára meghatározza az optimális útvonalat a csomagok kiszállításához. Ez a feladat az utazó ügynök probléma néven lett híres a tudomány világában. Ennek megoldása első hallásra talán egyszerűnek tűnhet (- és kevés célpont esetén az is -), de a célpontok számának növekedésével a megoldás bonyolultsága rendkívül megnőhet, hiszen ez egy NP teljes probléma (- nem determinisztikusan polinomiális -). Az ilyen NP teljes feladatok megoldásához jól használhatók a meglévő jó, és kevésbé jó algoritmusok és heurisztikák mellett a genetikus algoritmusok.

A genetikus algoritmusok kiindulásának alapja az élővilágban jelenlévő DNS-ek változásának, keveredésének reprodukálása számítógépes környezetben különböző számítógépes és matematikai problémák megoldására. A feladat megoldása során én is genetikus algoritmust használok, melyet részletesen be is fogok mutatni a szakdolgozat későbbi részében.

A dolgozat első felében bemutatom a feladat megoldásához használható technológiákat, technikákat, illetve azt, hogy melyiket választottam ezek közül és miért. A második felében ismertetem az elkészült szoftver részeit, működését, használatát.

1. GENETIKUS ALGORITMUSOK

Nagyon sok olyan feladat van, amelyre nem ismert algoritmus, vagy ismert ugyan, de az nem hatékony, nem gyors, vagy éppen nem lehet megkeresni az optimális megoldást. Sok ilyen feladattal találkozhatunk a keresés illetve az optimalizálás témakörében. A módszer egyik fő előnye, hogy a számítástechnikában előforduló problémák egy nagyon széles osztályára alkalmazható, mert általában nem használ területfüggő tudást, így akkor is működik, ha a feladat struktúrája kevéssé ismert. Az evolúciós algoritmusokhoz tartozó számítási modellnek tekinthető genetikus algoritmusok optimalizálási és keresési problémákra alkalmazhatók. A fejezetben felhasznált irodalmak: [3],[5],[8],[9].

1.1. Történeti áttekintés

Az 1950-es években merült fel el az a gondolat több tudósban is, hogy a biológiai evolúció a mérnöki problémákban is felhasználható lehet optimalizációs eszközként. Az alap ötlet az volt, hogy lehetséges megoldások halmazát képezzék különböző, természetes kiválasztódás és genetikus változások által ihletett operátorok használatával.

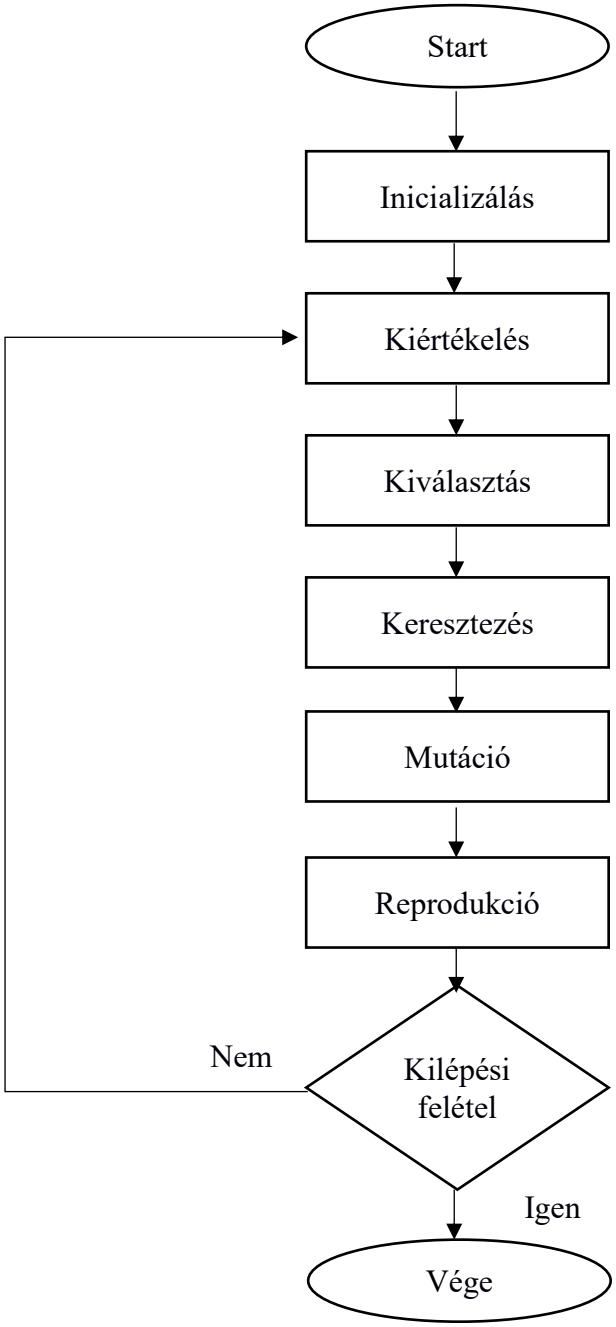
Ingo Rechenberg publikálta 1965-ben az evolúciós stratégiák alapjait [6][7], mely alapvető ma használt fogalmakat nem használt még, mint a keresztezés, de a populációk már megjelentek benne. Fogel, Owens és Walsh 1966-ban kifejlesztették az evolúciós programozás technikáját [2], amiben egy adott feladat lehetséges megoldásait véges állapotú automatákként ábrázolták, melyek fejlődését az állapotátmeneti diagrammjuk véletlenszerű mutációjával és a legjobbak szelektálásával oldották meg. Számos más kutató dolgozott még evolúció inspirálta algoritmusokon (Box, Friedman, Bledsoe, …), de egyikük munkája se kapott annyi figyelmet, mint az evolúciós stratégiák, evolúciós programozás vagy a genetikus algoritmus.

A darwini evolúciós elmélet [1] és a genetika alapjait magába építő szigorúan vett genetikus algoritmusok első változatát, az egyszerű genetikus algoritmusokat John Holland, a Michigani Egyetem professzora javasolta 1975-ben [4]. Ez a publikáció tekinthető az első mérföldkőnek a genetikus algoritmusok történetében és ez az eredmény nevezhető még ma is a genetikus algoritmusok elméleti alapjának. Holland genetikus algoritmusa a populációkban lévő kromoszómákból hozza létre a következő generáció populációit,

mutáció, keresztezés és inverzió műveletekkel. Minden kromoszóma génekből áll, melyek lehetséges értékeit génváltozatoknak vagy alléloknak nevezünk. Az utazó ügynök problémában, például, egy útvonalat reprezentálhat egy kromoszóma, míg egy várost egy gén. Minél jobb egy kromoszóma rátermettsége, annál nagyobb eséllyel választják ki a szelekció művelete során, vagyis annál nagyobb eséllyel adhatja tovább génjeit a következő generáció számára. A mutáció megváltoztatja véletlenszerűen kiválasztott gének értékét, a keresztezés (rekombináció) két kromoszómát vág két- vagy több részre, majd ezeket egymásközt felcserélve új kromoszómát alkot, az inverzió pedig a kromoszómán belül kiválasztott szakaszt fordít meg, ezzel változtatva annak felépítését. Az azóta eltelt évek alatt az evolúciós módszerek komoly fejlődésen mentek keresztül és a kutatók egymásra hatása egyre jobban elmosta a határokat a genetikus algoritmusok, az evolúciós stratégiák, az evolúciós programozás és a többi evolúciós megközelítés között. Mai kutatásokban gyakran használják a „genetikus algoritmus” kifejezést olyan dolgokra, amik messze állnak Holland eredeti leírásától.

1.2. Működése

Az algoritmus működése többféle módon mehet végbe, de az alábbi fő részek szinte mindegyik formában megtalálhatók: inicializálás, kiértékelés, kiválasztás, keresztezés, mutáció, reprodukció. Az 1. ábra az előbb felsorolt lépéseket szemlélteti egy folyamatábrán.



1. ábra: Genetikus algoritmus működésének folyamatábrája

1.2.1. Reprezentáció

A természetes kiválasztódás nagyban függ az egyedek reprezentációjától, hiszen a szülő kromoszómákon végrehajtott műveletek által tovább vitt génekkel jönnek létre az újabb generációk. A kódolásra többféle lehetőség van. A klasszikus módszer szerint egy kromoszómát egy adott hosszúságú bitsztringben ábrázoljuk. Itt a legnagyobb kérdés a

leképezés módszere. Van, ahol ez egyszerűen megoldható, de ahol nem, ott használható a Gray-kódolás, ami a szomszédos egészekhez 1 Hamming-távolságú kódot rendel (vagyis 1 bitben térnek csak el az egymás melletti értékek). Bináris kód mellett használhatók még a lebegőpontos számok vagy a gráf alapú reprezentációk.

1.2.2. Inicializálás

Ahogy az 1.-es ábrán látható, az új generáció egy iteráció során, több lépésen keresztül automatikusan jönnek létre, de a kezdeti populációknál más a helyzet, ezt az inicializálás során hozzuk létre. Mivel a genetikus algoritmusok használatának nagy részénél nem rendelkezünk alapvető információkkal a problémáról, ezért véletlenszerűen válogatunk az értékkészletből, hogy megalkossuk a kiindulási kromoszómákat. Természetesen léteznek olyan esetek, amikor vannak előismereteink, ilyenkor a jól megválasztott kezdeti értékek tudják gyorsítani az algoritmus futását. Azonban ezek a heurisztikák plusz munkát jelentenek és mivel a genetikus algoritmus gyorsan tudja javítani a kezdeti rossz egyedeket ezek használta nem feltétlen éri meg a ráfordított erőforrásokat.

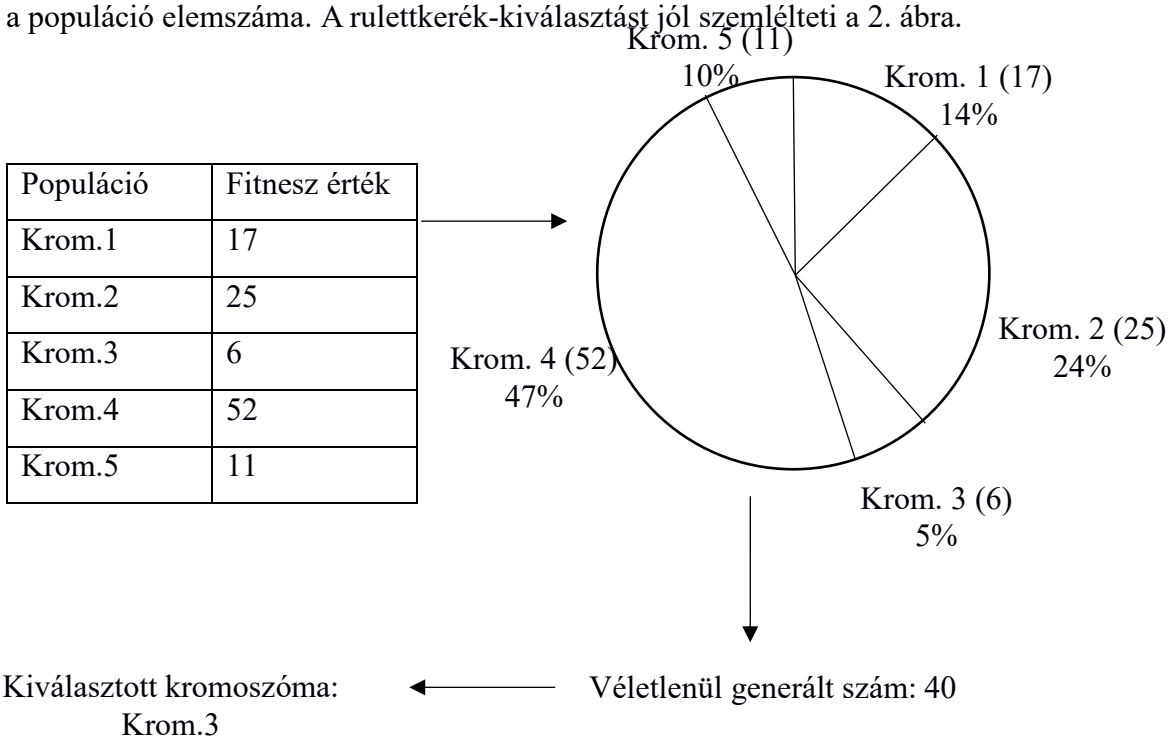
1.2.3. Kiértékelési-, cél- és jósági függvény

A kiértékelési függvény (evaluation function), másnéven célfüggvény (objective function) és az általuk a kromoszómáknak adott értéket felhasználó jósági vagy rátermettségi függvény (fitness function) megválasztása kritikus szempont a genetikus algoritmus kidolgozásánál, mivel ezek sokszor fognak lefutni az iteráció során, ezért fontos, hogy ne legyen nagy az időigénye, könnyen elvégezhető számításokból álljon. A célfüggvény adta értékek megadják, hogy egy megoldás mértéke mennyire jó vagy rossz. Ezeket az értékeket használja a fitneszfüggvény és mondja meg a skálázás során az egyes egyedeknek a többihez viszonyított jóságát, rátermettségét. Ezért a jósági függvényt úgy kell kialakítani, hogy az megfelelően tükrözze az elemek közötti különbséget és a probléma megoldása szerinti jóságát, azaz jobb megoldáshoz nagyobb értéket rendeljen. Alkalmazható módszer például, hogy a jó megoldásoknál a jóságukat adjuk meg, míg a rosszakénál a hozzájuk legközelebb álló jó megoldástól való távolságukat. Másik módszer a büntető függvény, melynek során az egyedek értékét, annak hibáinak elfogadhatatlanságától függően különböző súlyokkal számoljuk, majd ezeken a hibapontokon használjuk a fitneszfüggvényt. Ha ismert a rátermettségi függvény, de túlságosan időigényes vagy túl kevés információ áll

rendelkezésünkre érdemes approximációs függvényt használni, melynek eredménye ugyan nem lesz annyira pontos, de időt lehet vele nyerni és több egyedet megvizsgálni, ami javíthatja a pontatlanságát.

1.2.4. Szelekció

A szelekció során kiválasztásra kerülnek azok az egyedek, amelyek tovább adhatják a génjeiket a következő generáció számára a reprodukció során. Ez a kiválasztás a kromoszómák fitnesz értékén múlik. Általában minél jobb a rátermettség, annál nagyobb eséllyel lesz kiválasztva, így a következő generáció a nagyobb jóságú egyedekből lesz létrehozva, vagyis nagyobb eséllyel jobb lesz a rátermettsége, mint az előző generációnak.
Az egyik legismertebb módszer a rulettkerék-kiválasztás vagy fitneszarányos szelekció (roulette wheel-, fitness based selection). Ennek során a kiválasztás esélye arányos a jóság mértékével, egy egyedet többször ki lehet választani, hiszen a választási műveletek függetlenek egymástól. Ha az egyed jósága *ff*$_i$, akkor kiválasztásának esélye $\frac{ff_i}{\sum_{i=1}^{N} ff_i}$, ahol N a populáció elemszáma. A rulettkerék-kiválasztást jól szemlélteti a 2. ábra.



2. ábra: A rulette kerék kiválasztás

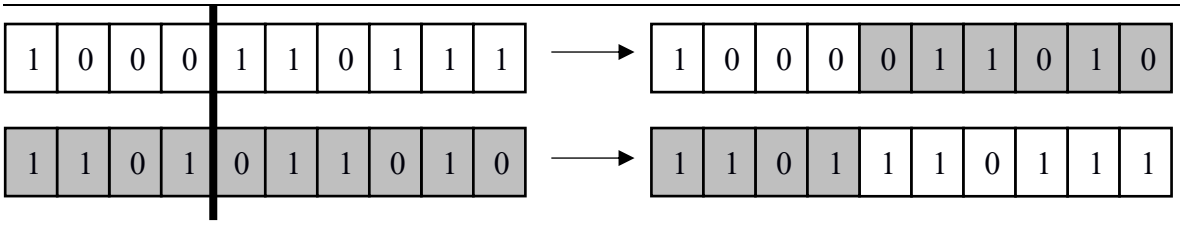
Ezt a módszert könnyű implementálni és sokszor jó eredmény is ad, azonban vannak esetek, amikor más módszer jobb eredményt szolgál, ilyen lehet a versengő kiválasztás vagy a sorrend alapú kiválasztások.

Utóbbihoz tartozik a csonkolásos kiválasztás (truncation selection), ahol csak egy adott jósági szint feletti kromoszómákból választunk, de itt viszont a szelekció valószínűsége egyenlő minden egyednek. Szintén sorrend alapú kiválasztás a lineáris sorrendezésen alapuló kiválasztás (linear ranking selection), ahol fitnesz szerint sorba rendezzük az egyedeket. A kiválasztási valószínűségük csak a sorban elfoglalt helyüktől függ, a valószínűség növekmény állandó.

A versengő kiválasztás (tournament selection) a rulettkerék módszer továbbfejlesztése. Kiválasztunk két egyedet a rulettel majd ezeket „versenyeztetjük” és a jobb jóság értékű kerül ki győztesként, lehet szülő.

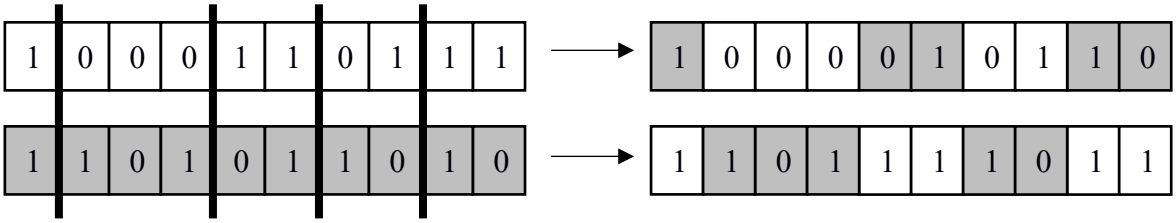
1.2.5. Keresztezés

A keresztezés a genetikus algoritmusok egyik alapvető művelete, mely során általában két szülő egyed különböző részeit összetéve új egyede(ke)t hozunk létre. A legegyszerűbb ilyen módszer az egypontos keresztezés (one point crossover). A művelet során a két szülő kromoszómát ugyanazon pozíciójú gén mellett kettévágunk, és a részeket felcseréljük, ezáltal létrejön két új egyed. Jellemző hibája a módszernek, hogy a pozíciókkal nem egyenlően bánik, például a végpontok minden esetben felcserélődnek, míg a közbülső gének megmaradására jóval nagyobb esély van. A másik elég feltűnő hibája a pozíciós hiba. A vágásoknál ugyanis semmi garancia nincs arra, hogy egy jó génsorozat tovább öröklődik, hiszen arra, hogy a vágás ne a jó sorozaton belül legyen semmi befolyásunk nincs és ez a probléma lehetőség annál nagyobb, minél hosszabb a jó génsorozat. A kisebb géncsoportok viszont könnyen együtt maradhatnak, így létre jöhetnek autóstoppos bitcsoportok, melyeknek semmi funkcionális közük nincs egymáshoz, csak közel helyezkednek el.



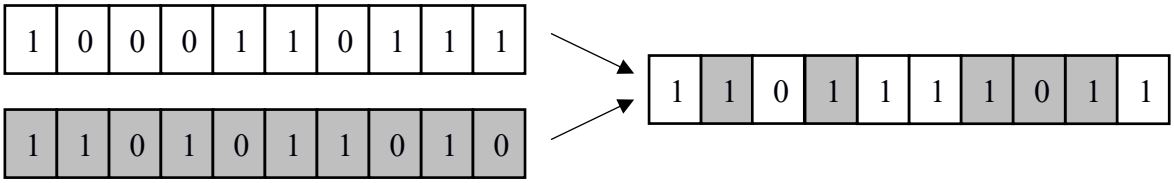
3. ábra: Egy pontos keresztezés

Az egypontos keresztezés könnyen tovább fejleszthető, ilyen például a kétpontos- vagy a több pontos keresztezés, ezeket már nem érintik az előbb említett hibák, de még messze nem tökéletesek. A kétpontos során nem egy, hanem két helyen kerül elvágásra az egyed. A több pontos keresztezés is hasonló elven működik, de itt három vagy több helyen lesz vágás. A vágáspontok itt is, mint az egypontosnál, teljesen véletlenszerűen lesznek elhelyezve. Ezeknél a kiválasztásoknál azonban nem lehet tetszőleges csoportokat ötvözni.



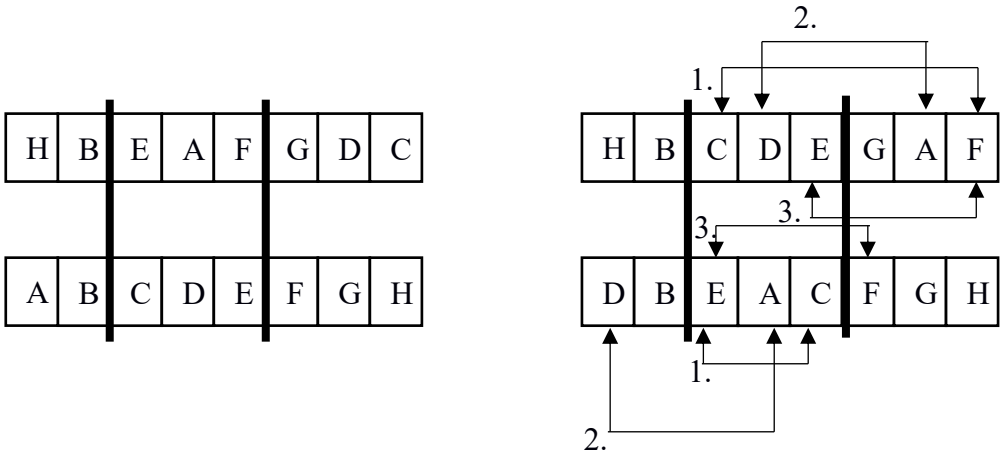
4. ábra: Több (4) pontos keresztezés

Említésre méltó még az egyenletes keresztezés (uniform crossover), ahol az új kromoszóma minden génje egyenlő, 50-50 %-os eséllyel kerül át a szülőktől. Ezt a műveletet már nem érinti a vágásokat jellemző hibák. Ugyanakkor a pozíciós hiba hiánya miatt az adaptálódó génértékek nehezebben tudnak kialakulni, mivel ez a fajta keresztezés könnyen szét is darabolhat bármilyen géncsoportot



5. ábra: Uniform keresztezés egy utóddal

A pozíciós hiba kiküszöbölésére alkalmas módszer még a csúsztatott keresztezés, itt a csere előtt mindkét szülőben eltoljuk a biteket egy random mértékkel, majd utána fordított irányba is elvégezzük a tolást.

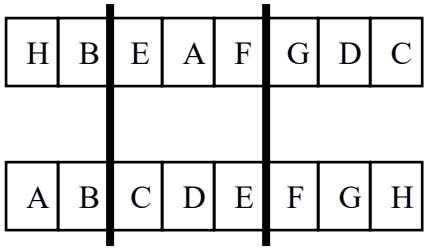


6. ábra: PMX működése

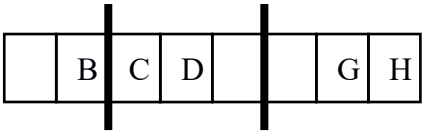
Az említett módszerek olyan esetekben használhatók, ha a gének értékei megegyezhetnek, viszont ha az allélok egyediek más műveleteket kell használni, ezeket átrendező operátoroknak (reordering operators) hívjuk.

A részlegesen megfeleltetett keresztezés (partially matched crossover, PMX) során a szülőkben két véletlenszerű vágással egy megfeleltetési szakaszt hozunk létre, melynek elemeit a szülőkön belül megcseréljük a szakaszon belüli párjukkal, a 6. ábra egy ilyen keresztezést mutat be.

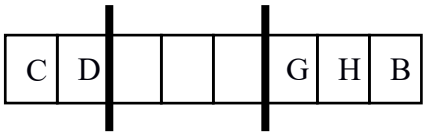
Az átrendezési keresztezés (order crossover OX) is hasonló képen indul, mint a PMX, létrehozunk egy megfeleltetési szakaszt. Először az egyik egyedből kitöröljük a másik megfeleltetési szakaszában lévő elemeit, majd ezeket az üres géneket úgy tesszük a két vágáspont közé, hogy a kromoszómát két végénél fogva összefűzöttnek tekintjük, és a 2. vágásponttól balra lévő elemek addig toljuk balra, amíg az üresek középre nem tolódnak. Végül az első elem megfeleltetési szakaszát a második helyére másoljuk. Az OX műveletét a 7. ábrán szemléltetem egy utód létrehozásával, a másik gyerek kromoszóma értéke „AFCDEGHB” lenne.



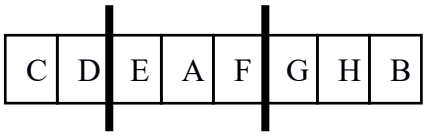
1. Eltávolítjuk a megfeleltetési szakasz elemeit



2.Addig toljuk balra az elemeket amíg az üres gének középre nem kerülnek



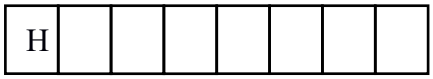
3. Végül beillesszük a megfeleltetési szakaszt a másik kromoszómából



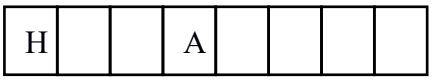
7. ábra: Az OX működése

Az utolsó általam szemléltetett operátor a ciklikus keresztezés (cycle crossover CX) lesz. Itt nem használnak vágópontokat. Az első lépés, hogy az egyik szülő első génjét átmásoljuk a

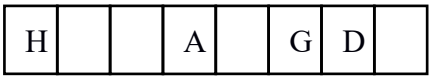
1. Átmásoljuk az első allélt az első szülőből



2. Megkeressük az előbb átmásolt gén helyén a második egyedben lévő alléljét az első egyedben és áthelyezzük az új kromoszómában



3. …és ezt addig folytatjuk amíg el nem érünk egy olyan allélig ami már szerepel az új egyedben, ami most a negyedik lépés után következett be.



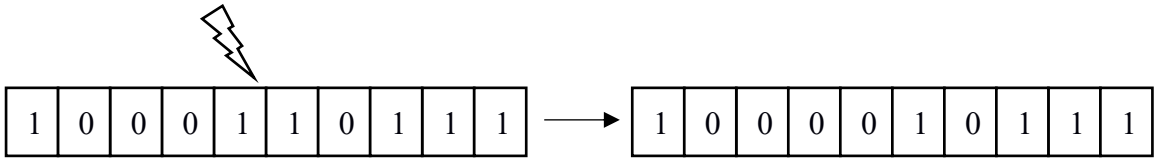
4. Végül a második kromoszóma génjeit átmásoljuk az új egyed üres helyeire.

8. ábra: Az CX működése

leszármazottba. Majd a másik szülő azonos pozíciójában lévő allélját megkeressük az első szülőben és azt is átmásoljuk az új kromoszómába, de abba a pozícióba, ahol az egyes szülőben helyet foglalt. Ez után megnézzük, hogy az előbb átmásolt allél álláspontjában a második szülőben milyen értéke szerepel, amit megkeresünk az első szülőben és átmásoljuk azonos helyre az utódba. Ez így folytatódik, amíg nem jutunk el olyan allélhoz a második szülőben, ami már szerepel a leszármazottban, ekkor a maradék helyekre a második szülő értékeit másoljuk át a pozíciókat megtartva. Működését a 8. ábrán lehet megtekinteni.
Másik utód értéke a művelet elvégzése után ABEDFHGC lenne.

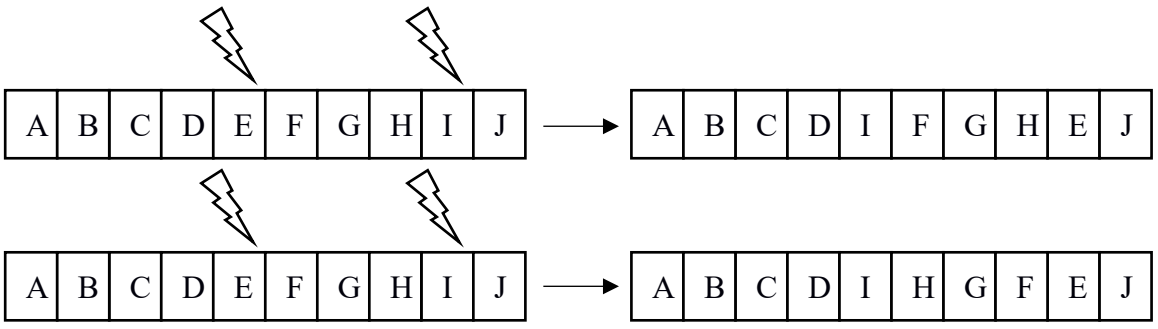
1.2.6. Mutáció

A mutáció lényege, hogy a meglévő populációból kis valószínűséggel ($<=10^{-3}$) módosítsa a kromoszómák értékét, ezáltal segítve a keresési tér eddig fel nem fedezett területeit



9. ábra: Gén értékének vátozása

felkutatni. Mutáció lehet például bitsztringek esetén egy kiválasztott gén értékének véletlenszerű megváltoztatása, vagy minden gén értékének kis valószínűséggel való módosítása.
Egyedi allélok esetén egyik megoldás lehet ha egy kromoszómán belül két véletlenszerűen kiválasztott gén értékét megcseréljük vagy használjuk az inverziót és egy szakasz sorrendjét megfordítjuk.



10. ábra: Gén értékek cseréje és az inverzió

1.2.7. Reprodukció

A szelekció, keresztezés, mutáció során létrejött köztes populációból és az előző generációból a reprodukció során jön létre az új generáció. Legegyszerűbb módja ennek az ha az előző populációt teljes egészében leváltja a köztes állomány, ezt egyszerű vagy generációs reprodukciónak hívják (simple, generational repoduction). Mivel semmi garancia nincs arra, hogy a köztes generáció tagjai jobbak, mint az előzők érdemes lehet a legjobba(ka)t megőrizni és biztosítani a konvergenciát. Ezt hívják elitizmusnak (elitism). A megtartott tagokat kétféle módon vihetjük tovább: ha kisebb a köztes generáció, mint az előző, akkor csak hozzáadjuk az elit tagokat. Amikor az ideiglenes populáció mérete megegyezik az előzővel, akkor vagy a legrosszabb fitnesz értékű vagy véletlenszerűen kiválasztott kromoszómák kerülnek leváltásra.

1.2.8. Kilépési feltétel

A kilépési függvény különféle feltételek beteljesülését figyelheti, majd állíthatja le az iterációt. A legegyszerűbb megoldás, ha adott számú iteráció után leáll az algoritmus. Ehhez előismeretek szükségesek a problémával kapcsolatban. A leállás függhet az aktuális populáció állapotától is. Ha egy egyed jósága nagyon eltér az átlagos fitneszértéktől, akkor feltételezhetjük, hogy van egy kiemelkedő megoldás. Hasonló megoldás ha a szórást vizsgáljuk és az egy adott értéken belül esik, ekkor a kromoszómák a keresési tér egy pontja köré csoportosulnak. Ezeken kívül még kilépési ok lehet, ha a generáció átlagos fitnesz értéke nagyon hasonló vagy megegyezik az előző generációk rátermettségével. A legjobb egyedek konvergenciája is kilépést okozhat. A bináris kromoszómáknál egy gén konvergál, ha például 95%-ban azonos az értéke a populációban, egy populáció konvergál, ha minden gén konvergál.

2. FELHASZNÁLT TECHNOLÓGIÁK ISMERTETÉSE

Az alkalmazás során több különböző technológiát használtam be, melyeket a következő fejezetekben ismertetek röviden.

2.1. Verziókezelő rendszer

„Verziókezelés alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a mérnöki tudományokban és a szoftverfejlesztésben használnak verziókezelő rendszereket fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg.” [https://hu.wikipedia.org/wiki/Verzi%C3%B3kezel%C3%A9s]
Főbb előnyei az ilyen rendszereknek:

- Backup és visszaállítás
- Szinkronizáció
- Változások követése
- Felelősség követése
- Elágazás (branching) és összefésülés (merging)

A verziókezelőknek két fő fajtájavan:

- Centralizált. A hagyományos verziókezelők központosított modellel dolgoznak, ahol minden verziókezelési művelet egy közösen használt szerveren történik. Ha többen egyszerre akarnak módosítani egy fájlt, akkor valahogy le kell kezelni a konkurens viselkedést. Kétféleképpen oldják meg ezt a problémát: zárolással és összefésüléssel.
- Decentralizált. Itt egy központi tároló (angolul repository) helyett minden felhasználó gépe egy-egy külön tárolóként jelenik meg. A szinkronizáció az egyes gépek között küldött patch-ek (módosításcsomagok) által valósul meg. Ez a megközelítés jelentős változásokat okoz (nincs nagy központi adatbázis, gyakori műveletek gyorsabbak, minden munkamásolat egy-egy backup). Két fajta elosztott verziókezelő létezik, a nyitott és a zárt. [https://hu.wikipedia.org/wiki/Verzi%C3%B3kezel%C3%A9s]

Munkám során egy decentralizált verziókezelőt használtam, melyet már előző projektjeim során megismertem, a Git-et.

2.1.1. Git

A Git-et eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez. Minden Git munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől. [https://hu.wikipedia.org/wiki/Git]

2.2. Frontend technológiák

A frontend technológiák felelősek a felhasználóval közvetlenül interakcióba kerülő részekkel. A weboldal kinézetéért és a közvetlen felhasználói tevékenyégek kiszolgálására 4 technológiát használtam, ezek:

- HTML
- CSS
- JavaScript

2.2.1. HTML

„HyperText Markup Language (HTML) a weboldalak és más böngészőben megjelenő dokumentumok elkészítéséhez használt nyelv. Pontosabban, a HTML az a nyelv, amely a dokumentum struktúráját és a szemantikus tartalmát jellemzi.” [https://developer.mozilla.org/hu/docs/Web/HTML]

A tartalom olyan HTML elemekkel van megjelölve, mint például: Tartalom . Az előbbi példában a kezdő- és zárócímke jelzik, hogy a tartalma egy szövegbekezdés. (A „p” az angol paragraph rövidítése.) A *class*-t attribútumnak nevezzük melynek az értéke az „=” jel után idézőjelekben van.

„HTML egy nemzetközi szabvány, amelynek a leírása a World Wide Web Konzorcium és a WHATWG által karbantartott. A HTML egy élő szabvány, amely technikailag folyamatos fejlesztés alatt áll. A jelenlegi HTML szabvány verziójára HTML5 ként hivatkozunk.” [https://developer.mozilla.org/hu/docs/Web/HTML]

Három fő részre osztható egy HTML:

- Dokumentum típus deklaráció. Ennek a szöveg legelején kell elhelyezkedni. HTML5 esetében ez: *<!DOCTYPE html>*
- Fejléc. Ide olyan dolgok kerülnek, amiket egy átlagos felhasználó nem lát, de fontos információkat tárol a weboldallal kapcsolatban, mint pl. karakterkódolás, nyelv, cím, stb. Ezek a *<head></head>* HTML elemek közé kerülnek.
- Törzs. Ide kerül a weboldal lényegi tartalma, ezt látja a felhasználó. Ez a rész a *<body></body>* HTML elemek közé kerül.

2.2.2. CSS

Cascading Style Sheet (CSS) egy stílusleíró nyelv, ami használható egy HTML oldal megjelenítésének formázására.

Egy CSS stíluslap szabályokból épül fel. Egy szabály tartalmaz egy szelektort és egy deklarációs részt. A szelektor adja meg, hogy mely elemkere kell használni a szabályt. Lehet egy HTML elem vagy egy elem attribútuma (class, id). A deklarációk kinézete a következő: „tulajdonság: érték;”. A tulajdonságok angol kulcsszavak, míg az értékek szintén angol szavak, számok és mértékegységek lehetnek (pt, px, %, stb.).

Használatának számos előnye van. Csökkenti a redundanciát a HTML kódban, nem kell minden elemhez ugyanazt a stílust többször leírni, csak létre kell hozni egy szabályt egy adott szelektorhoz és a böngésző érvényesíteni fogja azt. Így a karbantartás is leegyszerűsödik, hiszen ha több elem kinézetét akarjuk módosítani elég csak egy helyen elvégezni a műveletet. Csökkentheti az adatforgalmat, mivel a böngészők gyakran tárolják a CSS fájlokat a gyorsítótárban, így azt nem kell többször lekérni.

2.2.3. JavaScript

„A JavaScript (JS) egy kis erőforrás-igényű, értelmezett vagy JIT-fordított programozási nyelv első osztályú függvényekkel. Bár legtöbben weboldalak parancsnyelveként ismerik, sok webböngészőn kívüli környezetben is használják. Ilyen a node.js, az Apache CouchDB és az Adobe Acrobat. A JavaScript egy prototípus-alapú, többparadigmás, dinamikus nyelv, ami támogatja az objektumorientált, imperatív és deklaratív (pl. funkcionális) programozási stílusokat.” [https://developer.mozilla.org/hu/docs/Web/JavaScript]

JavaScript legtöbbször kliens oldalon fut és segítségével megadható az oldal működése különböző események bekövetkeztében (oldal betöltésénél, kattintásnál, stb.).
JavasScript lehet HTML dokumentáción <*script*> elemen belül, vagy külön .js kiterjesztésű fájlban.

2.3. Backend technológiák

A backend egy adott rendszer alsóbb, a tényleges feldolgozást végző rétege. A back-end réteg feladata a front-end réteg felől érkező adatok feldolgozása, ill. a keletkezett eredmény a front-end számára történő visszajuttatása.

IRODALOMJEGYZÉK

[1] Darwin, C. R.: *The Origin of Species,* John Murray, London (1859).
[2] Fogel, L. J., Owens, A. J., Whals, M. J.: *Artificial Intelligence through Simulated Evolution,* Wiley, (1966).
[3] Hatwágner, F. M. : *Nagy számításidejű, folytonos változójú célfüggvények optimalizációja evolúciós számítások segítségével,* PhD disszertációm, Széchenyi István Egyetem, Műszaki Tudományi Kar, (2012).
[4] Holland, J. H.: *Adaptation in Natural and Artificial Systems,* The University of Michigan Press, Ann Arbor, (1975).
[5] Mitchell, M.: *An introduction to Genetic Alogrithms,* A Bradford Book The MIT Press (1998)

[6] Rechenberg I.: *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment (1965)

[7] Rechenberg, I.: *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologische Evolution*, Fromman-Holzboog, Stuttgart, (1973).

[8] Sivanandam, S.N., Deepa, S. N.: *Introduction to Genetic Algorithms,* Springer-Verlag Berlin Heidelberg, (2008)

[9] Várkonyiné, K. A. : *Genetikus algoritmusok*, Typotex kiadó, Bp., (2002).

[A felhasznált szakirodalom megadása]

[A szerzők nevét mindenütt “Családnév, X.” formában kell megadni, ahol X. a szerző keresztnevének (keresztneveinek) kezdőbetűje. Magyar cikk esetén a vessző a családnév és a keresztnév kezdőbetűje közt elhagyható. Ha az egyértelműség megkívánja, a keresztnév kiírható teljesen is.

A dolgozat szerzője szabadon választhat, az A vagy a B típust használja.]

[A-típus:

A cikkekre való hivatkozás egy []-be írt sorszámmal történik. A sorszámozást folytonosan kell megtenni, a sorba rendezés alapja az első szerző családneve.]

[1] Szerző1 (, Szerző2 ...): *Cikk címe*

Folyóirat neve, sorszám, kezdőoldal-végoldal, év.

[2] Szerző1 (, Szerző2 ...): *Konferencia-kiadvány-beli cikk címe*

„Konferenciakiadvány:” Konferencia neve, hely, kezdőoldal-végoldal, év.

[3] Szerző1 (, Szerző2 ...): *Könyvcím*

„Könyv:” Kiadó, hely, oldalszám, év.

[4] Szerző1 (, Szerző2 ...): *Kutatási jelentés címe (csak publikus elérhető jelentés!)*

„Kutatási jelentés”: Kutatási projekt neve, intézet, oldal, év.

[5] Szerző: *Disszertáció címe*

„PhD/kandidátusi/stb. disszertáció”: Egyetem, kar neve, év.

[6] *Internetes oldal elnevezése:*

URL, letöltés ideje (csak konkrét dokumentumra mutató URL adható meg!)

MELLÉKLETEK

[A dolgozat mellékletei, ha vannak]