GENETIKUS ALGORITMUS LOGISZTIKAI ALKALMAZHATÓSÁGA

Szerző:

Csuta Ákos Koppány
Budapesti Műszaki és Gazdaságtudományi Egyetem

Dobák Dávid
Budapesti Műszaki és Gazdaságtudományi Egyetem

Első szerző e-mail címe:
csuta.akoskoppany@edu.bme.hu

Lektorok:

Szilágyi Brigitta (Ph.D.)
Budapesti Műszaki és Gazdaságtudományi Egyetem

Szabó Bendegúz
Knorr-Bremse Fékrendszerek Kft.

…és további két anonim lektor

Absztrakt

A XXI. században nincs olyan terület, ahol nem jelent volna meg a mesterséges intelligencia algortimusok használata. Ezek egyre inkább kutatás tárgyát képezik és az ipar is aktívan érdeklődik irántuk. Cikkünkben egy evolúciós algoritmus, a genetikus algoritmus használatával végezzük el egy logisztikai feladat optimalizációját. Az eredmények kiértékelése során kitérünk a program gyakorlati megvalósíthatóságára is.

Kulcsszavak: evolúciós algoritmus, genetikus algoritmus, optimalizáció
Diszciplinák: mérnöki tudományok, informatika

Abstract

LOGISTICAL APPLICABILITY OF GENETIC ALGORITHM

There is no such field in the 21st century, which is not using the artificial intelligence algorithms. These are even more researched topics and the industry is actively enquiring. In our article we use an evolution algorithm, namely the genetic algorithm for solving a logistic problem. During the evaluation of the results, we emphasize the practical implementation.

Keywords: evolution algorithm, genetic algoritm, optimization
Discipline: engineering, informatics

Csuta Ákos Koppány és Dobák Dávid (2023): Genetikus algoritmus logisztikai alkalmazhatósága. *Mesterséges Intelligencia – interdiszciplináris folyóirat, V. évf. 2023/1. szám.* 21-34. doi: 10.35406/MI.2023.1.21

A XXI. században szinte már nincs olyan mérnöki terület, ahol ne kapnának szerepet a különböző mesterséges intelligencia és optimalizációs algoritmusok. A feladatok és munkafolyamatok ilyen módon történő optimalizálása rendkívül gyümölcsöző lehet. Kiváló példa erre az öbölháborúban (1991) használt DART nevű mesterséges intelligencia program, mely nagyobb megtakarítást eredményezett pusztán a logisztikai műveletek összehangolásával, mint a DARPA 30 éves AI kutatásra fordítandó költségvetése (Hedberg, 2002).

A legelterjedtebb ezen algoritmusok közül a neurális hálók (CNN, Convolutional Neural Network, ill. DNN, Deep Neural Network). Ennek oka, hogy kevés tudással rövid idő alatt igen magas szintű alkalmazást lehet megvalósítani, emellett a vizuális analógia miatt is népszerű (a hálózat felépítése és működése bizonyos fokig hasonlít az emberi agy működésére és felépítésére). A legjobban használható hálózatok alkalmazásához igen mély valószínűségszámítási ismeretek szükségesek (lásd például: Murphy, 2012).

A másik elterjedt csoportba tartoznak a metaheuriszitkus módszerek közé sorolható evolúciós algoritmusok (genetikus algoritmus, bakteriális algoritmus, evolúciós stratégia stb.). Cikkünkben egy ilyen technika, a genetikus algoritmus gyakorlati alkalmazhatóságát járjuk körül. Ez az algoritmus az állati evolúciót modellezi és jól használható optimalizálásra (lásd: Mitchell, 1998). A téma aktualitása miatt a logisztika területét választottuk az algoritmus demonstrálásához. A csomagszállítás egyre nagyobb méreteket ölt a világ minden pontján, így hazánkban is – például: FoxPost (Net1), Magyar Posta. A tengerentúlon az egyik legnagyobb céggé nőtte ki magát az amerikai óriásvállalat, az Amazon. Ennél a cégnél már most is aktívan használnak optimalizációs eljárásokat és mesterséges intelligenciát igénylő alkalmazásokat – például önvezető targonca robotot (Clark, 2022). A trendek szerint tehát ezen a területen is egyre fontosabb az MI. Az általunk vizsgált probléma egy raktárhelyiség optimális kitöltése lesz, ahol a csomagok elrendezését genetikus algoritmus valósítja meg.

Cikkünk felépítése a következő: e bevezetőt követő második szakaszban bemutatjuk a genetikus algoritmust és az alkalmazásához szükséges fontosabb összefüggéseket. A harmadik szakaszban definiáljuk a problémát és az algoritmus bemenetéhez szükséges alakra hozzuk. A negyedik szakaszban bemutatjuk az általunk használt algoritmust, az ötödikben pedig a kapott eredményeket. A hatodik, záró szakaszban pedig értékeljük ezeket, külön kitérve a megvalósíthatóságra.

A genetikus algoritmus (elméleti összefoglaló)

A genetikus algoritmusok tehát az evolúciós algoritmusok (Bäck és Schwefel, 1993) közé sorolandók. Ezekről általánosan elmondható, hogy a darwini evolúciót, mint alapsémát felhasználva oldanak meg optimalizációs feladatokat. Mégis, talán a genetikus algoritmusok azok, amelyeknél a leg-

erősebben tetten érhető ez a fajta viselkedés. Ezt a nagyon erős kötődést tükrözi az elterjedt, egyértelműen a biológia szaknyelvéből származtatott terminus technicus is. Ennek alapfogalmain keresztül ismertetjük röviden a genetikus algoritmusokat.

Tetszőleges optimalizációs algoritmust tekinthetünk úgy, mint a problémára adott lehetséges megoldások előállításának iteratív sorozatát. Míg egy klasszikus, nem evolúciós algoritmus jellemzően egy megoldást állít elő minden iterációban, egy genetikus algoritmus egy számos példányból álló megoldáshalmazt ad. Ezek összességét populációnak nevezzük. Ekkor persze azt mondhatjuk el, hogy ennek a populációnak a „legjobb" példánya konvergál az optimális megoldáshoz. Ez, ahogyan az alább a gyakorlati példán is bemutatásra kerül, ebben a formában nem helytálló maradéktalanul – ezt a megfelelő helyen tárgyaljuk részletesebben.

Adódik a kérdés, miként fogalmazható meg egy egyed „jósága". Erre a célra definiálandó egy úgynevezett rátermettségi vagy, ahogy az angol eredetijéről magyarítva gyakran használatos: fitnesz függvény. Ennek konkrét felírása persze feladatfüggő, de általánosan elmondható róla, hogy a populáció minden egyedéhez hozzárendel egy „jósági tényezőt", amely aztán lehetővé teszi az egyedek rangsorolását. Utóbbira számos eljárás ismert, ezek közül kizárólag a rátermettség-arányos szelekciót alkalmazzuk. Ez a rendezés határozza meg azt, hogy az aktuális populáció mely példányai vesznek részt a következő generáció előállításánál. Jól látható az evolúcióval való párhuzam, hiszen a természetben is a legrátermettebb egyedek génjei öröklődnek tovább a legnagyobb eséllyel. Fontos felismerésre vezet az előző mondat: egy példány a génjeivel írható le. Ahhoz tehát, hogy implementálható legyen a genetikus algoritmus, szükséges egy megfelelő kódolás meghatározása, amely minden az egyedre jellemző releváns tulajdonságot egy karaktersorra képez le.

Az így felírt genotípus már alkalmas arra, hogy variálásával új példányokat hozzunk létre. A biológiában ismert fogalom a kromoszomális átkerereszteződés, ezzel analóg módon valósul meg az új egyed genotípusának előállítása. A legegyszerűbb esetben ez annyit tesz, hogy két szülő génsorozatát a megfelelő helyen egyenként elfelezzük, majd egymással rekombináljuk. Ettől eltérő, például többpontos vagy egyenletes keresztezés is elterjedt. Ezeket azonban, mivel általunk való alkalmazásukra nem került sor, itt csupán megemlítjük.

Szülőknek persze azokat az egyedeket választjuk, amelyek a korábban leírt rangsorolás szerint jobb tulajdonságokkal bírnak. Ezzel azonban nem garantálható az, hogy előáll az optimális megoldás. Fennáll ugyanis a veszélye annak, hogy egy lokális optimumtól nem képes elszakadni algoritmusunk. Azaz előáll egy olyan példány, amelyhez közeli egyedek nála rendre rosszabbak, hiába létezik nagyobb távolságra nála jobb fitnesz értékű megoldás. Ennek kiküszöbölésére a rekombináció mellé bevezetjük a mutációt, mint operátort. Akárcsak a természetben, itt is egy

véletlen génmódosulásról van szó. Azaz a populáció egyes példányainak valamely génjét vagy génjeit véletlenszerűen megváltoztatjuk.

Egy új generáció létrehozására jellemzően azt jelenti, hogy az előző generáció legjobb megoldásai mellé emelünk annyi új példányt, hogy a populáció egyedszáma ezáltal elérje az előírt értéket.

A probléma definiálása és analóg rendszerek

A mesterséges intelligencia algoritmusok alkalmazásának egyik legnehezebb lépése a problémakör oly módon történő definiálása, hogy az valóban algoritmizálható legyen. Az emberek számára egyszerű feladatok egy számítógép számára rendkívül bonyolulttá is válhat, a számítógépek viszont ezredmásodpercek alatt elvégeznek egyszerű számításokat, ami egy ember számára megoldhatatlan feladat (Hassan és Rizvi, 2019). Jó példa erre az arcfelismerés, hiszen míg ez egy ember számára egyszerű és mindennapos dolog, addig egy gép (alkalmas bemenet kialakítása nélkül) nem tud mit kezdeni a problémával. Megfelelően definiálva viszont egyszerűvé válik a feladat: egy lehetséges megoldást jelent a Haar Alfréd magyar matematikusról elnevezett Haar-kaszkád használata (Viola és Jones, 2001). Az algoritmus ún. feature-öket keres egy képen (általában éleket), és ezek alapján képes klasszifikálni a képen látható tárgyat/embert. Ez egy arcfelismerés esetén a szemöldököket, szájat, orrot stb. jelenti.

Látható tehát, hogy a feladatot algoritmizálhatóvá és jól definiálttá kell tenni, hogy egyáltalán bármilyen MI algoritmus kiválthassa az emberi operátort. A problémát érdemes összehasonlítani már létező megoldásokkal. A raktárkitöltés feladat egy 3 dimenziós probléma, melynek 2 dimenziós megfelelői az ún. parkettázási problémák. Sok játékot ihlettek meg az ilyen problémák, ahol a darabokkal általában teljes (tökéletes) lefedést szükséges elérni, ilyenek például a népszerű tangram kirakósok. Az előzőnél is jobb példa a Tetris nevű videójáték: ebben a játékosnak felülről érkező, különböző geomatriai alakzatokat kell minél sűrűbben pakolva elhelyezni. A játék fokozatosan gyorsul és egy idő után a játékos nem tudja elég gyorsan kitalálni a következő alakzat legjobb helyét és így az szuboptimális helyre kerül. Az így halmozódó hibák a munkaterület túlcsordulását, ezáltal a játék elvesztését eredményezik. A Tetris az általunk megoldani kívánt probléma tökéletes 2 dimenziós megfelelője. A játékhoz éppen az általunk is vizsgált genetikus algoritmus segítségével készítettek agent-et, amely a játék világrekordját is képes megdönteni (Bui, 2020).

A raktároptimalizációs feladat megoldása során ezért a következő feltételezésekkel élünk:

- a raktározási térfogat téglatest alakú,
- a raktározandó csomagok szintén téglatest alakúak,
- mind a raktározási térfogatot mind a csomagokat reprezentáló téglatestek élhossza valamilyen egység egész számú többszöröse,

- az egyes megoldások (elrendezések) között prioritás definiált.

Ezekkel a probléma már algoritmizálva megoldható.

Genetikus algoritmus megvalósítása

A korábbiakban ismertettük, milyen megfontolások és általános sémák mentén építhető fel egy genetikus algoritmus, mint optimalizációs eljárás. Ebben a fejezetben azt mutatjuk be, hogy mindezt miként implementáltuk az adott probléma esetében.

Elsőként azt kell megfontolni, hogy miként írható fel egy példány, azaz milyen tulajdonságait és hogyan kódoljuk, előállítva így fenotípusát.

Adott egy alaphalmaz, mely tartalmazza a „rakodó térben" elhelyezni kívánt „dobozokat". Egy példány, azaz egy megoldás szükségszerűen azt írja le, hogy mely „dobozok" kerültek kiválasztásra és pontosan hol helyezkednek el a térfogatban. Ez a térfogat – derékszögű koordináta-rendszerben véve – az egyszerűség kedvéért essen a pozitív térnyolcadba és illeszkedjen a tengelyekre, azaz egyik csúcsa legyen az origó. Ekkor könnyedén számszerűsíthető egy-egy doboz helyzete. A pozíció egyszerűen megadható a dobozokat egyetlen pontjukkal reprezentálva, ez a pont implementációnkban rendre a téglatestek origóhoz legközelebb eső csúcsa. Ez azt jelenti, hogy a dobozok helyzete kódolható három változó segítségével, amelyek rendre a kitüntetett pont koordinátái. Egy „doboz" helyzetének leírása nem merül ki ennyiben, hiszen egy pontja mellett meg kell adnunk orientációját is. Megjegyezzük, hogy ezt az orientációt csak a tengelyek körül vett 90 fokos forgatások kompozíciójával állítottuk elő. Nem vezettünk be további géneket az irányultság leírására, ehelyett az alaphalmazból létrehoztunk egy bővített halmazt. Ez tartalmaz minden „dobozt" minden szabályos orientációval. Arra természetesen tekintettel kell lennünk, hogy egyazon „doboz" több különböző orientációban nem kerülhet a „rakodó térbe". A tárgyaltak mellett egyetlen további géntípust vezettünk be annak kódolására, hogy a példány tartalmazza-e a bővített halmaz adott elemét.

A fenti megfontolások alapján levezethető, miként valósítható meg a fenotípus kódolása. Legyen V a „rakodó tér", azaz a megtöltendő térfogat, melynek oldalhosszúságait jelölje rendre v_x, v_y és v_z. Hasonlóan legyen A – (a_x, a_y, a_z) egy „doboz".

Azaz az alaphalmaznak egyetlen eleme van. A korábban tárgyaltak szerint ezt a halmazt bővítjük ki olyan módon, hogy az így kapott halmaz minden megkülönböztetett orientációban tartalmazza az alaphalmaz elemeit – példánkban elemét. Könnyen belátható, hogy ez összesen hat különböző téglatestet állít elő. Ez a megállapítás általánosan is igaz: az alaphalmaz minden eleméhez hat elem tartozik a bővített halmazban. A vizsgált példához visszatérve, előállt tehát egy hatelemű halmaz.

A következő lépés a példány génkészletének definiálása. Arról már szó esett, hogy milyen gének definiálják a példányt. Emlékeztető gyanánt és további részletezés

nélkül: három gén írja le az adott téglatest origóhoz legközelebb eső csúcsának koordinátáit, illetve egy további gén fejezi ki azt, hogy az elem kiválasztásra került-e. Látható tehát, hogy egy példány teljes fenotípusát a bővített halmaz minden eleme után négy gén alkotja. Ez természetesen azt is jelenti, hogy az alaphalmaz minden eleme 24 gént ad a példány teljes génkészletébe. Minthogy az egyes gének valójában egy-egy változót jelentenek, szükséges annak meghatározása, hogy ezek milyen értékeket vehetnek fel.

A kiválasztott csúcs koordinátáit reprezentáló génekre – ezeket jelölje A_x, A_y, A_z – vonatkozó kényszer egyszerűen adódik. Ahhoz, hogy a „doboz" a kijelölt térfogaton belül maradjon, feltétlenül teljesülnie kell a következő egyenlőtlenségnek:

$$0 < A_i < v_i - a_i$$

$$i \in \{x,y,z\}$$

Az egyszerűség kedvéért feltételeztük továbbá, hogy:

$$A_i,\ v_i,\ a_i \in \mathbb{Z}$$

A választ arra a kérdésre, hogy pontosan mit von maga után az egyenlőtlenség jobboldalának negativitása, egyelőre függőben hagyjuk.

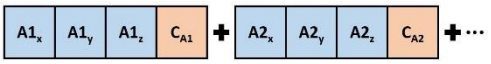
Egyértelműnek látszik a negyedik gén felírása, amely vagy csak 0, vagy csak 1 lehet függően attól, hogy az adott „doboz" a „rakodó felületre" került vagy sem. Ez a leírás önmagában nem elégséges, alkalmaznunk kell egy megszorítást. Elképzelhető ugyanis, hogy a téglatest valamely éle hosszabb a „rakodó tér" megfelelő dimenziójánál. Egyszerűen megfogalmazva ez annyit tesz, hogy az adott „doboz" az adott orientációban nem fér el, nem helyezhető el a céltérfogatban. Az ilyen elemek kiválasztottságot reprezentáló génjéhez, hogy hatékonyabb legyen algoritmusunk, feltétlenül 0 értéket rendelünk. Ezzel választ adtunk arra a korábban nyitva hagyott kérdésre is, hogy miként szükséges eljárni abban az esetben, ha:

$$v_i - a_i < 0$$

Megjegyezzük, hogy az ilyen elemeket el is távolíthatnánk halmazunkból, ám megfelelően implementált algoritmus esetén ezzel azonosan eredményes az alkalmazott eljárás.

Az 1. ábra egy példány fenotípusának egy részletét mutatja. Megfigyelhető a bővített halmaznak az alaphalmaz A eleméhez tartozó A1 és A2 elemére vonatkozó négy-négy gén szekvenciája. A 2. ábra az egyes génekhez rendelhető értékeket szemlélteti.

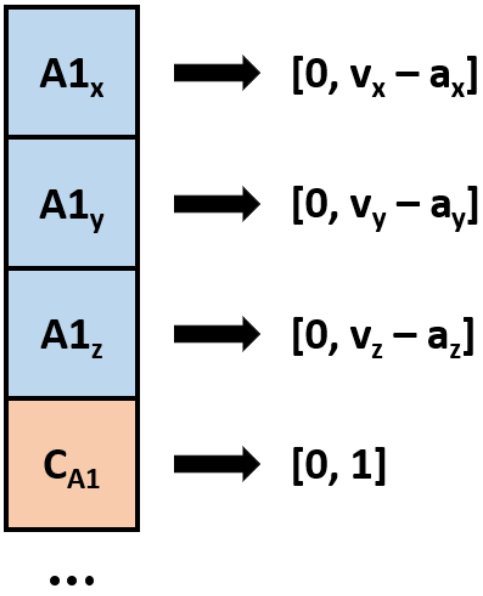
1. ábra: Gének szerkezete (forrás: a Szerzők)



Adott tehát, hogy miként írható le egy példány. Az így felírt példányok sokasága megad egy generációt. Genetikus algoritmusról lévén szó, szükséges a példányok rangsorolására alkalmas viszonyszám meghatározása. Ebben a bekezdésben az erre szolgáló fitnesz-függvényt ismertetjük.

A példányok rendezéséhez olyan függvény definiálása szükséges, amely egy lehetséges megoldáshoz annak „jóságát" rendeli. Ez valóban igaz, ám kevéssé praktikus. Az előző állítás helyett alkalmazzuk azt a meg-

2. ábra: Az egyes gének értékkészlete (forrás: a Szerzők)



közelítést, hogy egy példányhoz hozzárendeljük, mennyire „rosszul" felel meg. Ekkor a „legjobb" megoldás természetesen a legkevésbé „rossz" egyed. Jó és rossz persze szubjektív, de jelen feladat kapcsán jól körülhatárolható fogalmak. Belátható, hogy nagyon rossz megoldás az, amely a bővített halmaz több olyan elemét is tartalmazza, melyek egyazon alaphalmazbeli elemhez tartoznak. Ugyancsak nagyon rossz az a megoldás, amely olyan koordinátákat rendel a tartalmazott „dobozok" kitüntetett csúcsához, hogy az egymást metsző testeket eredményez. Egyértelműen adódik, hogy ilyen megoldásról beszélhetünk, ha a kiválasztott objektumok egyenként vett térfogatainak összege nagyobb, mint a berendezett elemek összessége által alkotott test térfogata. A multiplikációktól terhelt és metsző téglatesteket tartalmazó megoldások nyilvánvalóan elfogadhatatlanok is egyúttal.

Egy további „jósági faktor" annak mértéke, hogy a kiválasztott és a „rakodó térben" elhelyezett testek összes térfogata mennyivel marad el a teljes rendelkezésre álló térfogathoz képest. Ellentétben az előző két esettel, az itt megfogalmazott „jóság" már nem az adott megoldás elfogadhatóságára, hanem az elvben megfelelő megoldások összehasonlíthatóságára szolgál. Ezzel összefoglaltuk a definiálandó fitnesz-függvény minden tényezőjét. Látható, hogy minden egyes faktor könnyen számszerűsíthető. A multiplikációk számát jelölje α. Az egyenkénti és az egybevont térfogatok különbségét reprezentálja β. A „rakodó tér" betöltöttségének mértékét leíró változó legyen γ. Ekkor egy megoldásra felírható a fitnesz-függvény:

$$f = -\lambda_1 \cdot a - \lambda_2 \cdot \beta - \lambda_3 \cdot \gamma$$

Az itt megjelenő, λ-val jelölt „büntetések" értékét úgy választottuk meg, hogy a fent leírtakat tükrözzék. Ennek megfelelően:

$$\lambda_1 \gg \lambda_2 > \lambda_3$$

Erre azért van szükség, hogy egy elvben helyes megoldás mindenkor előnyt élvezzen olyan példányokkal szemben, amelyek kivitelezhetetlenek a gyakorlatban. A megvalósítható példányok közötti sorbarendezhetőség persze ekkor sem szenved csorbát.

Adott tehát a gének kódolása és a fitnesz-függvény, ezután felírható maga a megoldáskereső algoritmus is. Hozzunk létre egy kiindulási generációt, azaz példányok egy

meghatározott számú sokaságát. Hattassuk a fitnesz-függvényt a teljes populációra, azaz rendeljük hozzá minden egyedhez annak „jóságát". Rendezzük sorba az egyedeket jóságuk szerint. Állítsuk elő az új generációt olyan módon, hogy a rangsorolt példányok felét megtartjuk, majd a populációt feltöltjük új egyedekkel. Ez utóbbiakat az előző generációból megtartott példányok egypontos keresztezésével kapjuk. Az eredeti genotípust mindig ugyanott választjuk ketté, méghozzá a génszekvencia felénél. Ezután a mutációk következnek. Az új egyedek egyes véletlenszerűen kiválasztott génjeit megváltoztatjuk. Természetesen nem arról van szó, hogy ezen génekhez teljesen tetszőleges értékeket rendelünk. A mutáció olyan formán valósítandó meg, hogy annak nyomán az adott gén egy véletlen, de a korábban ismertetett értékkészletbe eső számot hordozzon. Ezzel előállt az új generáció. A populációra újra alkalmazzuk a fitnesz-függvényt, ami alapján aztán rangsoroljuk az egyedeket, majd végrehajtunk minden további, fentebb ismertetett lépést.

Kérdés, meddig ismételjük ezt újra és újra. Az egyik lehetőség a generációk számának előírása, ekkor tehát előre definiáljuk, hány iterációt hajtson végre az algoritmus. Ez minden esetben lehetséges, de nem túl hatékony módszer. Egyes esetekben a szükségesnél több ideig futhat az algoritmus, míg máskor fennállhat, hogy a megadott lépésszám mellett nem állítja elő az elérhető legjobb megoldást.

A másik megközelítés szerint valamilyen leállási feltételt fogalmazunk meg. Például: álljon meg a keresés, ha az aktuális populáció legjobb egyedének rátermettsége elér egy meghatározott értéket. Ez egy lehetséges megoldás, de nem alkalmazhatjuk, ha az elérhető legjobb megoldás fitnesz-értéke ismeretlen. Jelen példában akkor lenne jól használható ez a leállási feltétel, ha tudnánk, hogy előállítható olyan példány, amihez a fitnesz-függvény 0 értéket rendel. Másként megfogalmazva, ha biztosak lennénk abban, hogy a „rakodó tér" teljes mértékben és résmentesen feltölthető „dobozokkal". Mivel tetszőleges alaphalmazra alkalmazható algoritmust kívánunk létrehozni, más megoldást választottunk. Implementált algoritmusunkban akkor kerül sor a ciklusból való kilépésre, ha az újonnan előálló populáció legjobb példányának rátermettsége már nem mutat emelkedést, vagyis megadott számú iteráción keresztül rendre azonos. Ez a példány lesz az algoritmus feladatra adott megoldása.

Az algoritmus leírásában egyes változók, például a populáció nagyságának esetében úgy fogalmaztunk, hogy értékük „megadott". Ezeknek a paramétereknek tetszőlegesen adhatunk értéket, helyes megválasztásukkal az algoritmus hatékonyságát befolyásolhatjuk. Ennek részletesebb kifejtése az eredményekkel foglalkozó szakaszban olvasható.

Eredmények bemutatása

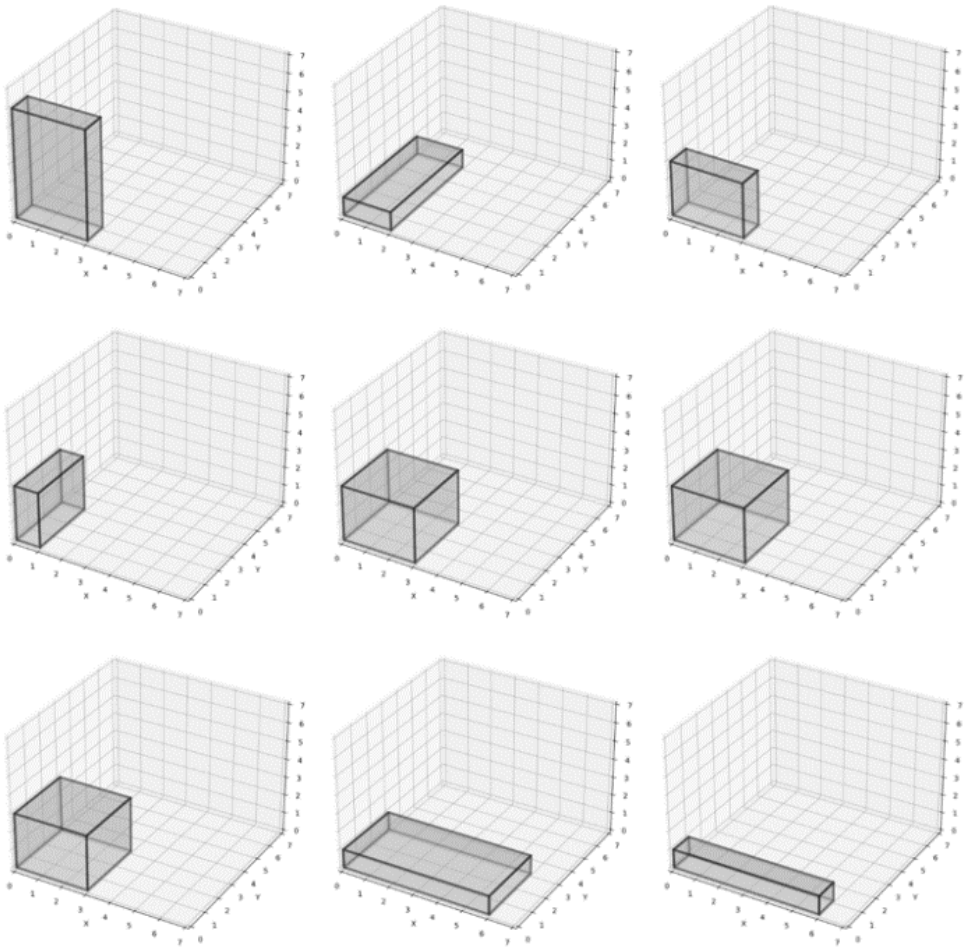
Az implementált algoritmust számos paraméter-kombinációval és alaphalmazzal teszteltük. Ezek kimerítő összefoglalása

helyett egyetlen szemléletes példán mutatjuk be a kapott eredményeket.

Legyen adott egy olyan „rakodó tér", melyet rendre (6, 4, 4) oldalhosszakkal jellemezhetünk. Emellett „dobozaink" alkossák a 3. ábrán látható alaphalmazt.

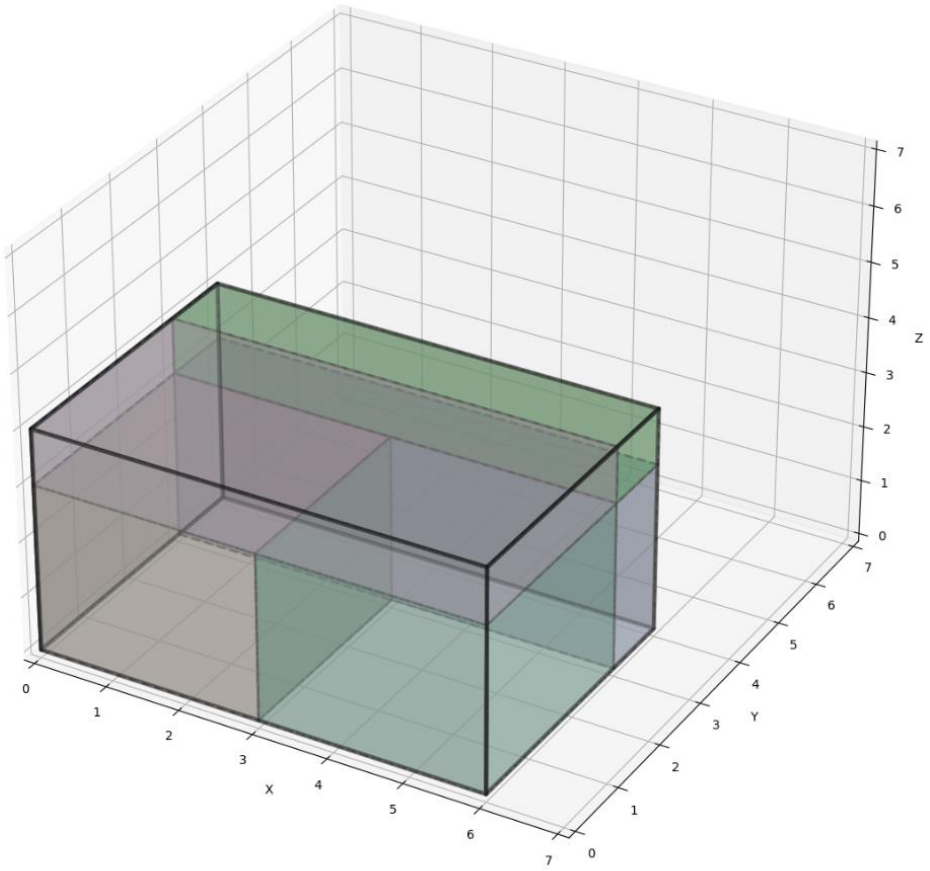
Kis gondolkodás után ez a probléma megoldható gépi segítség nélkül is, és belátható, hogy a „rakodó tér" teljesen feltölthető a „dobozok" egy megfelelő kombinációjával. Azaz azt az elvárást fogalmazhatjuk meg az algoritmussal szemben, hogy állítson elő olyan egyedet, melyre hattatva a fitnesz-függvényt 0 adódik. Ennek tesztelése megkívánja bizonyos paraméternek a megadását.

3. ábra: Dobozok alaphalmaza (forrás: a Szerzők)



Rögzítendő a populáció mérete, az aktuális generációból megtartott és szülővé emelt egyedek száma, valamint a mutációk száma. Ezek megválasztására visszatérünk, kezdetben fogalmazzunk úgy, hogy egy kellően jó kombinációt választottunk. Ekkor az algoritmus a 4. ábrán látható elrendezést javasolta.

4. ábra: A példa megoldása (forrás: a Szerzők)



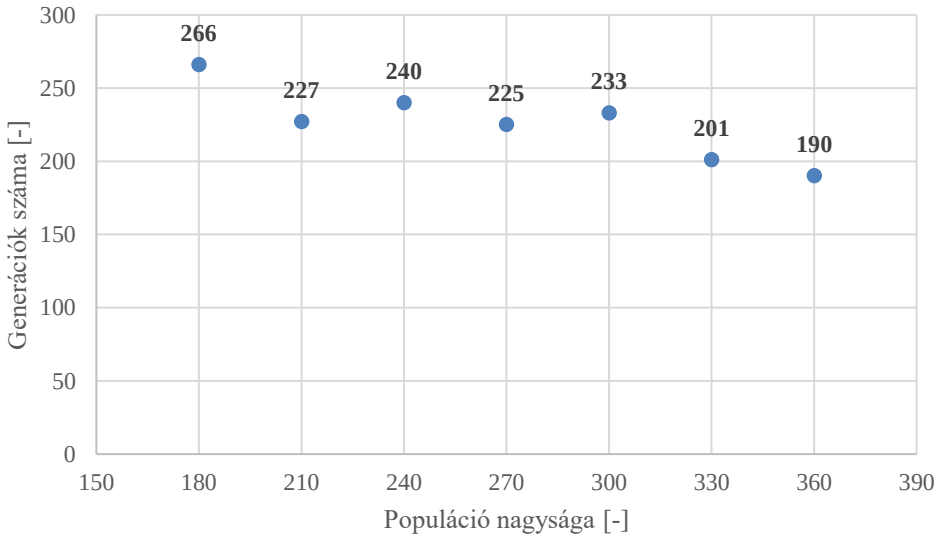
Látható, hogy ezt a megoldást kerestük. Pontosabban fogalmazva, ez egy olyan megoldás, amilyet kerestünk. Belátható, hogy ugyanezen elemeket felhasználva más elrendezés is tökéletes megoldást ad. A keresés tehát sikerrel járt. Az algoritmus hasonlóan eredményes más alaphalmazok esetén is.

A következő kérdés, amely válaszra szorul, a paraméterek megválasztására vonatkozó. Ez azért fontos, mert a megvalósított algoritmustól nem csupán eredményességet, hanem jó hatékonyságot is elvárunk. Azaz azt, hogy az optimális megoldás minél hamarabb előálljon.

Elsőként tekintsük azt a számot, ami megadja, egy új generáció előállításakor az előző egyedek közül hányat tartunk meg és alkalmazunk szülőként. A vizsgálat során a populáció nagysága és a mutációk száma állandó volt. Mivel olyan szabályszerűséget nem találtunk erre a paraméterre vonatkozóan, ami azonosan jól alkalmazható lenne minden esetre, egy egyszerűsítéssel éltünk. Minden esetben úgy jártunk el, hogy az aktuális populáció egyedeinek – természetesen jobb – felét tartottuk meg. Ez viszonylag jó kompromisszumnak mutatkozott a kisebb és a nagyobb bonyolultságú feladatokra is.

A második vizsgálandó paraméter a populáció nagysága, tehát az a szám, ami megadja, egy generáció hány egyed összessége. Az 5. ábra mutatja, hány generáció létrehozása volt szükséges a korábbi példa esetében ahhoz, hogy előálljon a keresett megoldás. A megjelenített értékeket egyenként 10 futtatásra kapott eredmények átlagolásával, illetve egészre kerekítésével képeztük.

5. ábra: A létrehozott generációk száma a populáció nagyságának függvényében (forrás: a Szerzők)



Látható, hogy bár nem szigorúan monoton módon, de mégis csökkenő trendet mutatnak az értékek. Ebből – tévesen – azt a következtetést vonhatnánk le, hogy minél nagyobb populációk alkalmazása a legkedvezőbb megoldás. Belátható azonban, hogy elsődleges fontossága nem annak van, hány generáció létrehozása szükséges a legjobb megoldás előállításához. Sokkal relevánsabb információ az ehhez szükséges futási idő. Természetszerű, hogy egy nagyobb méretű populáció egy-egy generációjának létrehozása, kiértékelése és rangsorolása több időt vesz igénybe, mint ugyanezen lépések elvégzése kisebb egyedszám mellett. Az 1. táblázat éppen ezt példázza.

1. táblázat: Generációk száma és futtatási idő (forrás: a Szerzők)

Populáció mérete [-]	Generációk száma [-]	Futási idő [s]
210	227	5,19
360	190	8,12

Noha a nagyobb populáció kevesebb iterációban jutott el az optimális megoldásig, ahhoz jóval több időre volt szüksége. Éppen ezért az első paraméterezést tekintjük kedvezőbbnek. Megjegyezzük, hogy – a létrehozandó generációk számával ellentétben – a futási időt erősen befolyásolja a használt futtatókörnyezet sebessége. Az adatok és az abból levont következtetések így csupán a rendelkezésünkre álló erőforrások mellett érvényesek.

A harmadik paraméter a mutációk száma. Valójában itt két paraméterről is beszélhetünk. Az egyik azt írja le, az előállított egyedek mekkora hányada szenved el mutációt, illetve egy másik azt, hogy a mutáció hány gént érint. Az implementált algoritmusban minden újonnan létrehozott egyeden alkalmaztunk mutációt, így a folyamat egy paraméterrel – a gének számára vonatkozóval – leírható. A bemutatott példában az alaphalmaz 9 elemből áll, így a bővített halmaz 54-ből, ami a korábban ismertetett módon 216 génből álló genotípust eredményez. A mutációk száma azt mutatja meg, hogy ebből a 216-ból hány génhez rendelünk véletlenszerűen egy új szabályos értéket minden egyes létrehozott egyedben. Ez nem egészen egyenértékű azzal a megfogalmazással, hogy hány gént változtatunk meg az egyed fenotípusában, hiszen az új érték bizonyos valószínűséggel egyezik a korábban hordozottal.

A megfelelő szám megtalálása nem bizonyult egyszerű feladatnak. Túl nagynak vagy túl kicsinek megválasztva a mutációk számát egyaránt ronthatunk az eredményességen és a hatékonyságon. Könnyedén lokális optimumokban ragadhat algoritmusunk, vagy éppen az elfogadhatónál sokkal lassabban éri el az abszolút legjobb megoldást.

A sokat taglalt példa esetében a mutációk számát 20-nak választottuk. Általános esetben a 216-ra rögzített értéket tekintettük referenciának és az 2. táblázatnak megfelelő módon igazítottuk az adott feladathoz.

Megjegyezzük, hogy a 216 alatti tartomány nagyon kis alaphalmazokból áll elő, mely esetek vizsgálatától eltekintettünk.

2. táblázat: Mutációk száma a gének számának függvényében (forrás: a Szerzők)

Genotípus hossza [-]	Mutációk száma [-]
[216; 432]	20
[432; 648]	25
[648; 864]	30
…	…

Egy további számhármasról is beszélhetünk, mint az implementált algoritmus paramétereiről. Ezek pedig a fitnesz-függvény egyes tényezői. Ezeket a „büntetéseket", alkalmazva korábbi eljelölésüket, a következőképpen választottuk meg:

$$\lambda_1 = 100000$$
$$\lambda_2 = 10$$
$$\lambda_3 = 1$$

Összefoglalva a leírtakat, a példa a 3. táblázat szerinti paraméterekkel került megoldásra.

3. táblázat: Paraméterkészlet (forrás: a Szerzők)

Paraméter	Érték [-]
Populáció nagysága	20
Szülő egyedek száma	40
Mutációk száma	60
Fitnesz-függvény tényezői	10000
	10
	1

Konklúziók

Az algoritmus megvalósítása sikeres volt, tesztelése megmutatta, hogy képes a megfogalmazott feladat elvárt módon történő megoldására. Az implementálás során megmutatkozott, milyen érzékeny az eljárás egyes állandóinak megválasztására. Ezen paraméterek hangolására hatásos és nem iteratív eljárást nem találtunk, ez pedig sarkalatos pontja a módszer alkalmazhatóságának.

Emellett elmondható, hogy számos kihívást rejt még magában a vizsgált probléma. Szándékunkban áll a paraméterek helyes megválasztásának további vizsgálata, hogy az alaphalmaz elemszámának nagyságától mindinkább független legyen az algoritmus hatékonysága. Jelenleg is dolgozunk egy olyan algoritmus kidolgozásán, ahol több fázisból áll az „evolúció", és az egyes szakaszok eltérő paraméter-készletekkel operálnak. Másfajta megközelítéssel fogunk élni a mutációk kapcsán is: eltérő valószínűséget rendelünk a dobozok kiválasztására és pozíciójára vonatkozó gének megváltoztatására. Azt reméljük, hogy ezáltal egy a már megvalósítottnál is hatékonyabb algoritmus áll elő.

Továbbá szeretnénk átlépni a már megvizsgált feladat határain is. A „dobozokhoz" további jellemzőket fogunk rendelni, mint például azok tömegét és szállítmányozásuk prioritását, ezzel téve komplexebbé és valóságközelibbé a problémát. Bár algoritmusunk már jelen formájában is alkalmas a megfogalmazott optimalizációs feladat elvégzésére, a gyakorlatban felhasználhatóvá ezen kiegészítések után válhat.

Távlati cél egy megfelelő hardveres környezet kialakítása, mely alkalmas a csomagok katalogizálására, és az algoritmus által előírt berakodás megvalósítására. Ezzel pedig eljutunk egy jól hasznosítható, korszerű, intelligens rendszer megalkotásához.

Irodalom

Bui, D. A. (2020). Beating the world record in Tetris (GB) with genetics algorithm. Letöltés: 2023.03.23. Web: https://towardsdatascience.com/beating-the-world-record-in-tetris-gb-with-genetics-algorithm-6c0b2f5ace9b

Bäck, T., és Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1), 1-23.

Clark, M. (2022). Amazon announces its first fully autonomous mobile warehouse robot. Letöltés: 2023.03.23. Web: https://www.theverge.com/2022/6/21/23177756/amazon-warehouse-robots-proteus-autonomous-cart-delivery

Hassan, M. A., és Rizvi, Q. M. (2019). Computer vs human brain: An analytical approach and overview. *Computer, 6,* 580-583.

Hedberg, S. R. (2002). DART: Revolutionizing logistics planning. *IEEE Intelligent Systems, 17*(3), 81-83. DOI: 10.1109/MIS.2002.1005635

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press. DOI: 10.7551/mitpress/3927.001.0001

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Net1: Ezer fölé növelné csomagautomatái számát a FoxPost. Letöltés: 2023.04.05. Web: https://www.vg.hu/logisztika/2022/10/ezer-fole-novelne-csomagautomatai-szamat-a-foxpost

Viola, P., és Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (Vol. 1, pp. I-I). Ieee. DOI: 10.1109/CVPR.2001.990517