DOI: 10.14750/ME.2020.018

MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

HEURISZTIKUS MÓDSZEREK ALKALMAZÁSA LOGISZTIKAI RENDSZEREK TERVEZÉSÉBEN ÉS IRÁNYÍTÁSÁBAN

PhD értekezés tézisei

Készítette:
Veres Péter
okleveles logisztikai mérnök

Hatvany József Informatikai Tudományok Doktori Iskola
Anyagáramlási rendszerek és logisztikai informatika tématerülete

Doktori Iskola vezető:
Prof. Dr. Szigeti Jenő
a matematikai tudományok doktora

Témavezető:
Dr. Bányai Tamás
egyetemi docens

Társ-témavezető:
Prof. Dr. habil. Illés Béla
egyetemi tanár

Miskolc, 2020

DOI: 10.14750/ME.2020.018

Nyilatkozat

Alulírott Veres Péter büntetőjogi felelősség tudatában kijelentem, hogy ezt a doktori értekezést magam készítettem, és abban csak a megadott forrásokat használtam fel. Minden olyan részt, amelyet szó szerint vagy azonos tartalommal, de átfogalmazva más forrásból felhasználtam, egyértelműen a forrás megjelölésével láttam el.

Miskolc, 2020. február18. ……..

Veres Péter

DOI: 10.14750/ME.2020.018

Témavezetői ajánlás

Veres Péter: „Heurisztikus módszerek alkalmazása logisztikai rendszerek tervezésében és irányításában” című PhD értekezéséhez

Veres Péter nappali képzés keretében bánya- és geotechnikai mérnöki alapszakon kezdte meg tanulmányait a Miskolci Egyetemen, majd ezt követően szerzett logisztikai mérnöki mesterdiplomát. Már tanulmányai kezdetén bekapcsolódott a Gépészmérnöki és Informatikai Kar Logisztikai Intézetében folyó kutatómunkába, mely kutatási tevékenységet később PhD hallgatóként folytatott. Aktívan bekapcsolódott az intézet oktatási és kutatási tevékenységébe. Résztvevője volt számos hazai és nemzetközi projektnek. Számos tantárgy gyakorlatvezetője. Aktív részese a nemzetközi kapcsolatok ápolásának. A kutatási téma iránti érdeklődése abból a felismerésből eredt, hogy a negyedik ipari forradalom olyan komplex termelési és logisztikai rendszerek kialakulásához vezet, melyek modellezése, tervezése és irányítása új, hatékony tervezési módszerek kialakítását teszi szükségessé. Véleményünk szerint a kutatási téma eredményei komoly érdeklődésre tarthatnak számot mind a kutatók, mind a vállalati szakemberek részéről.

Veres Pétert nagy munkabírású, érdeklődő, motivált hallgatóként ismertük meg. A Logisztikai Intézet tevékenységébe aktívan bekapcsolódott, ötleteivel, javaslataival hozzájárult az intézeti kollektíva sikeres kutatási tevékenységéhez. Kutatási eredményeit hazai és nemzetközi konferenciákon és folyóiratokban publikálta. A témakörben végzett kutatásairól 13 tudományos folyóiratcikk jelent meg, melyek közül öt külföldi kiadású szakfolyóirat. A konferenciákon tartott előadásai alapján sok dolgozat konferencia-kiadványban került publikálásra. Publikációi közül öt található meg a Scopus adatbázisban.

Az értekezés Veres Péter önálló kutatási eredményeit tartalmazza, és minden szempontból megfelel a Hatvany József Informatikai Tudományok Doktori Iskola szabályzatában előírt követelményeknek.

A fentiek alapján a jelölt számára a PhD cím odaítélését messzemenően támogatjuk.

Miskolc, 2020. február 18.

Dr. Bányai Tamás
témavezető

Prof. Dr. Illés Béla
társ-témavezető

DOI: 10.14750/ME.2020.018

Köszönetnyilvánítás

Elsőként szeretnék köszönetet mondani témavezetőimnek, Prof. Dr. Illés Béla, egyetemi tanárnak és volt Intézetigazgatónak a rengeteg segítségért, lehetőségért és az új logisztikai elvek megismertetéséért (Ipar 4.0), amelyek sok új ötletet adtak; és Dr. Bányai Tamás egyetemi docensnek a több éves aktív közös munkáért, a rengeteg segítségért és folyamatos iránymutatásért.

Emellett szeretném megköszönni a Miskolci Egyetem Logisztika Intézete minden tagjának a segítőkészségét és bátorítását a kutatómunkám elvégzésére és bemutatására, akik közül kiemelném Dr. Bányainé Dr. Tóth Ágota egyetemi docenst, aki rengeteget segített a kutatáshoz kötődő teendők megszervezésében és elintézésében és a motivációm növelésében.

Meg szeretném még köszönni családomnak és barátaimnak a folyamatos támogatást és buzdítást, akik nélkül sose lettem volna képes eddig eljutni.

Ezen kívül szeretném megköszönni mindenkinek, aki segített és támogatott ezen a hosszú úton, hogy befejezhessem és szavakba öntsem kutatásaimat.

Az értekezésben ismertetett kutató munka az EFOP-3.6.1-16-2016-00011 jelű „Fiatalodó és Megújuló Egyetem – Innovatív Tudásváros – a Miskolci Egyetem intelligens szakosodást szolgáló intézményi fejlesztése” projekt részeként – a Széchenyi 2020 keretében – az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg".

DOI: 10.14750/ME.2020.018

Tartalomjegyzék

DOI: 10.14750/ME.2020.018

1. BEVEZETÉS

Napjainkban a vásárlói igények jelentős átalakuláson mennek keresztül. A termékek minőségével szembeni elvárások fokozódása, a termékek életciklusának egyre nagyobb mértékű rövidülése, a termékek és az azokhoz kapcsolódó szolgáltatások folyamatos rendelkezésre állására vonatkozó igények növekedése jelentős kihívások elé állítja a vállalatokat. Ezen kihívások mind technológiai, mind logisztikai vonatkozásokkal bírnak, hiszen egyre egyedibbé váló termékeket kell a tömeggyártást megközelítő hatékonysággal előállítani, a vevőkhöz eljuttatni, illetve biztosítani kell a termék használatához kapcsolódó szolgáltatásokat. A vásárlói igények ilyen értelemben vett átalakulása párhuzamosan jelent meg a szociális viselkedés módosulásával, valamint a kapcsolattartási (kommunikációs) formák fejlődésével. A vállalatok abban az esetben tudják ebben a megváltozott, dinamikusabbá vált piaci környezetben versenypozíciójukat megőrizni és erősíteni, amennyiben időben felismerve „az idő szavát” felkészülnek ezen új vásárlói igények kielégítésére. Ezen felkészülés két fontos területen kell, hogy megvalósuljon. Egyrészről olyan új, dinamikus üzleti modelleket kell bevezetniük, amelyek megteremtik a lehetőségét a horizontális és vertikális együttműködésnek a beszerzés, termelés, értékesítés és inverz folyamatokat magába foglaló komplex ellátási láncok minden területén. Másrészt a dinamikusan változó vásárlói igények kielégítése érdekében az értékteremtő lánc szereplőinek mind termelési, mind szolgáltatási oldalon fejleszteniük kell az Ipar 4.0 képességeiket hatékonyságuk növelése és kapacitásuk bővítése érdekében. A negyedik ipari forradalom az információs és kommunikációs technológiák, valamint az automatizálás egyre szorosabb összefonódását, illetve ezen keresztül a termékek, szolgáltatások, gyártási módszerek és üzleti modellek alapvető megváltozását elhozó időszak összefoglaló neve, amelyben a szenzorok, gépek, munkadarabok és az IT rendszerek a teljes értéklánc mentén összekapcsolódnak [K1]. Ezen digitalizáció-alapú összekapcsolódás eredményei az úgynevezett kiberfizikai rendszerek, melyek mind a termelési, mind az ahhoz kapcsolódó logisztikai folyamatokban megjelennek.

Mivel a diverzifikált vásárlói igények egyre több fajta, egyre komplexebb termék piacon való megjelenését követelik meg, ezért egyre komplexebb, földrajzilag egyre kiterjedtebb, ugyanakkor egyre szorosabb, időben és megbízhatóságban egyre szigorúbb kihívásoknak megfelelő ellátási láncokat, beszállítói és értékesítési hálózatokat kell kialakítani. Ezen komplex ellátási láncok a digitalizáció elterjedéséből adódóan egyre nagyobb mennyiségű információt bocsátanak rendelkezésre, melyekből egyre pontosabb döntések meghozatalára van lehetőség mind a tervezés, mind a végrehajtás szintjén. A sok adat, a sok döntési változó és a dinamikus tervezési és üzemeltetési környezet olyan korszerű, hatékony tervezési és döntéstámogatási módszerek kidolgozását igényli, melyek révén megbízható, költséghatékony rendszerek alakíthatóak ki és működtethetőek, akár valós időben is.

Jelenleg olyan világban élünk, ahol nem elég egyszerre egy problémát megoldanunk, hanem arra is gondolnunk kell, hogy azokkal milyen más hatásokat, változásokat és esetleg új problémákat nyitunk meg. Erre való a komplex tervezés és a komplex rendszerek kutatása, amelynek egy kis szegmensére ebben a dolgozatban megoldásokat lelhetünk.

DOI: 10.14750/ME.2020.018

1.1. A műszaki fejlődés szerepe a logisztikai folyamatokban

A jelenlegi logisztikai trendek és fejlesztések két nagyon fontos tényező miatt alakulhattak ki. Elsősorban a globalizáció, amely fizikailag kötötte össze a világot és alakított ki kapcsolatot szinte minden helyszínnel a Földön, majd az internet elterjedése, amely virtuálisan tette meg mindezt és egy újabb síkkal bővítette a már ismert világunkat.

Mára ez a két fogalom szinte mindennek az alapja lett és életünk részévé vált. A vállalatokhoz naponta érkeznek alapanyagok, félkész és kész termékek a világ bármelyik pontjáról, amelyek kompenzálását internacionális bankhálózatokon keresztül oldják meg. Viszont nem csak a vállalatok, de az átlagember is képes termékeket rendelni bárhonnan, az internet és az online üzletek segítségével. Ma már elképzelhetetlen lenne az életünk az internet és az ott áramló és tárolt adatok nélkül.

Azonban ezek a hatalmas fizikai és virtuális rendszerek olyan méretűvé nőtték ki magukat, amelyeket emberek már nem képesek kezelni. Nem csak az internet, de az ellátási láncok kapcsolati rendszere is egyre jobban kezd hasonlítani egy nem centralizált neurális hálózatra, mint például az agyra. Lassan elérünk abba a korszakba, hogy több eszköz lesz az internetre kapcsolva, mint neuron egy átlagos emberi agyban (100 milliárd), ami kb. 1000 billió különböző kapcsolat kialakítására képes [K2].

2020-ra várhatóan 40 milliárd eszköz fog az internetre csatlakozni és bár nem mindegyik aktív vagy produkál szignifikáns adatforgalmat, mindegyik pontból el kell érni bármelyik másik pontba. Ha az utóbbi állítást igaznak vesszük, akkor az összes kapcsolat számát meg tudjuk határozni úgy, hogy vesszük az „n” sokszög oldalainak számát és átlóinak számának a felét, ahol n=40 milliárd-ot jelent:

$$\binom{n}{2} = \frac{n(n-1)}{2} \approx \frac{n^2}{2} = 8 * 10^{20} \quad (1)$$

Számításom alapján a 40 milliárd eszköz 800.000 trillió kapcsolatot képes létrehozni. Ez a mennyiség már nagyságrendekkel túllépi az emberi agyban lévő kapcsolatok számát. Egy ekkora hálózatnál - ha nem törekedünk az ideális állapotra - óriási veszteségek léphetnek fel, azonban megtalálni a helyes utat szintén nem egyszerű feladat. A matematikusok a hálózat alapú problémákat az NP-hard jellegű problémák közé sorolják, amelyeknek bár meg lehet határozni a probléma egzakt megoldását brute force módszerrel (teljes leszámolás), de az időigény minden egyes hozzáadott paraméterrel vagy elemmel drasztikusan, általában exponencionálisan nő. Ilyenkor használunk olyan algoritmusokat, amelyeket képesek megtalálni egy optimális vagy közel optimális megoldást reális időn belül.

A jelenlegi trendek és fejlődési irányvonalak is ezekben a témákban mozognak. Az iparban jelenleg a negyedik ipari forradalom elve (Ipar 4.0, I4.0) kezd elterjedni, amely gyártási folyamatokban egy olyan jellegű változást jelent, mely során az információs technológiák és az automatizálás között egyre szorosabb kapcsolat jön létre. Azonban ez csak egy keret, amelyen belül többféle technológia fejlődik egyszerre egymást segítve [K3] [K4].

Az egyik legalapvetőbb technológia az Ipar 4.0-án belül a Machine to Machine (M2M) technológia, amely a gépek között való információcserét jelenti emberi beavatkozás nélkül. Ezzel a technológiával egy gyártósoron lévő géphez a kiszolgáló egység automatikusan juttatja

DOI: 10.14750/ME.2020.018

el a szükséges alkatrészek, szerszámokat és készülékeket, vagy egy hiba felmerülése esetén megpróbálják azt saját maguk elhárítani. Közlekedés esetén a Vehicle to Vehicle (V2V) elv felel meg az M2M technológiai megoldásoknak, amely hasonló alapokra épül, csak itt nem gyártósori gépek, hanem járművek és forgalomirányító készülékek kommunikálnak egymással, mint az önvezető autók vagy forgalomszámláló jelzőlámpák [K5].

Szintén nagyon fontos fogalom ebben a témakörben a gépi tanulás (Machine Learning). Ez alatt egy olyan folyamatot értünk, amely segítségével a gépek olyan feladatokat képesek megoldani, amelyre közvetlenül nincsenek beprogramozva. Ilyen lehet a különböző csoportok kialakítása és elemekkel feltöltése, előrejelzések készítése, helyzetfelismerés. Azonban ehhez nagy mennyiségű adatot kell mintaként szolgáltatnunk nekik, amelyből szabályokat tudnak létrehozni. A gépi tanulás típusát az adja meg, hogy a mintakészítés és szabályok létrehozása milyen mértékű emberi felügyelettel történik.

A gépi tanulás és M2M technológia már a mesterséges intelligencia területéhez tartozik, amely a kétes hangulatú köznépi megítéléssel ellentétben (a gonosz MI, átveszi az uralmat) sokkal inkább a gépek logikus gondolkodásra és tanulásra való képességére fókuszál. A végső célja egy olyan gép megalkotása, amely programozásától függetlenül képes felismerni és megoldást keresni problémákra.

A gépek közötti kapcsolat M2M feltétele, hogy legyen egy kialakított információs csatorna, amin tudnak kommunikálni. Ez az alapja a dolgok internetének (Internet of Things=IoT), amely során egyre több és több eszközt és szenzort kapcsolunk a globális hálózatra, amelyeket bármikor elérhetünk. Ez természetesen újabb nagy mennyiségű adattal lát el minket, amelyeket információvá kell alakítanunk.

A Cisco prognózisa szerint 2018-ra a vezetékes és mobil adatkapcsolatokon bonyolított globális IP alapú adatforgalom el fogja érni a 1,6 zettabájtot, azaz a több mint egybillió gigabájtot [K6]. Ezekből az adatokból pedig nekünk valahogy információt kell kiszűrnünk, azaz csak a nekünk fontos és hasznos adatra van szükségünk. Ezt nevezzük Big Data koncepciónak. Rengeteg informatikai cég, amely nagy mennyiségű adattal dolgozik, folyamatosan fejleszti azokat az eszközöket, amelyek gyorsan és effektíven állítanak elő információt számunkra.

Az eddigiekben bemutatott fogalmak azonban nem csupán vállalaton belül, gyártóegységek és gyártósorok között jelennek meg, hanem vállalatok között és makroökonómia szinten is értelmezhetőek.

Európában a 80-as évektől jelentek meg a jelenleg is használatos horizontális és vertikális vállalati együttműködési formák, melyek közül érdemes megemlíteni a virtuális vállalat, a klaszter, konzorcium és keiretsu koncepcióját.

A virtuális vállalat egy olyan együttműködési forma, ahol a szerződő felek egymástól távol is elhelyezkedhetnek és minden tevékenységüket önállóan végzik, azonban egy közös cél érdekében vagy meghatározott időpontig ideiglenesen együttműködnek. Így a kisebb vállalatok könnyebben bekerülhetnek nagyvállalati piacra és könnyebben vehetnek részt a piaci versenyben [K7]. Ilyenek például a franchise-ok.

Egy másik vállalati forma a klaszter. Elsősorban Amerikában jellemzők, de Európában is kezdenek elterjedni. Egy klaszterbe azonos ágazati vállalatok tartozhatnak, amelyek a költségeik csökkentésére és tevékenységeik hatékonyságának növelésére állnak össze. A vállalatok egymás között felosztják a piacot elsődlegesen területi alapon, megelőzve ezzel a vállalatok közötti versengést és egységesítik a technológiát és az árakat.

DOI: 10.14750/ME.2020.018

A konzorciumok olyan vállalati vagy befektetői csoportosulások, amelyek ideiglenesen jönnek létre egy olyan nagyobb pénzügyi művelet lebonyolítása érdekében, mint egy nagy értékű vagy kockázatos befektetés, jelentős összegű hitel nyújtása vagy felvétele, új részvények kibocsátása stb. A konzorcium, mint önálló jogi személy vesz részt a piacon, amelyhez a tagok közül és/vagy külső szervezetekből választanak egy vezető testületet. Az együttműködés időtartamát elsősorban a kitűzött cél időigénye határozza meg, ezért nagyon változatos időtartamokra jönnek létre konzorciumok.

A keiretsu egy kooperációs forma, ahol a gyártóegységek, ellátási láncok résztevői, disztribútorok és finanszírozók külön pénzüggyel, de szoros együttműködésben dolgoznak, hogy biztosítsák egymás sikerét. A megnevezés japánul csoportot jelent és gyakran használják a partner szinonimájaként. A keiretsu lehetővé teszi a gyártók számára, hogy hosszú távú partnerséget alakítsanak ki, ezáltal minden vállalat a fő tevékenységére fókuszálhat a lean alapelvei szerint. A stabilitás azonban hátrányára is fordulat a szervezetnek, ha gyorsan kell a piaci, kulturális vagy technológiai változásokra reagálni. Ezeket a csoportosulásokat a saját kereskedelmi vállalataik vagy bankjaik köré szervezik, amely lehetővé teszi a teljes pénzügyi, technológiai, logisztikai ellenőrzést minden ágazatban [K8].

Ezekből is látszik, hogy mind vállalati szinten, mind technológia és informatikai értelemben szükség van a különböző hálózatok megfelelő irányítására, ugyanis egyre több hálózat alakul ki vagy bővül folyamatosan. Ez a bővülés exponenciálisan írható le mind a hálózati elemek mind a köztük lévő kapcsolatok tekintetében.

1.2. A kitűzött kutatási célok

A hálózatszerűen működő nagy kiterjedésű, komplex ellátási láncok nem csupán az elméleti kutatások tárgyát képezik, hanem a gyakorlatban is egyre inkább előtérbe kerülnek, hiszen a digitalizáció hatására olyan termelési, logisztikai és komplex kiber-fizikai rendszerek iránt jelenik meg a gyakorlati igény, melyek a gyakorlatban is alkalmazhatóak. Ezen komplex rendszerek megjelenése, illetve az iránt való vállalati igény fokozódása indukálta jelen értekezés kutatási témáját, mely a következő célok megvalósítását tűzi ki maga elé:

- Vállalaton belüli és vállalaton kívüli nagyméretű logisztikai hálózatok leírása és kezelését segítő modellek megalkotása.
- Ezen hálózatokban keletkező logisztikai feladatok megoldására olyan új módszerek, modellek és alkalmazások kidolgozása, melyek révén azok működésének hatékonysága javítható.

A dolgozatomban az előbb meghatározott célok eléréséhez először a strukturált irodalomkutatás (SLR) módszerét használtam, melyet az irodalomkutatás fejezetben ki is fejtek. A modellalkotáshoz minden esetben egy általánosan leírt sablont használok, amelyről az Általános modell fejezet szól, amiben ismertetem azokat az alapelveket és általánosított összefüggéseket, melyek alapján a feladatspecifikus modellek megalkothatók. Ezen modellekből hármat mutatok be, melyek a következők: belső milkrun útvonal és raktártervezés, külső raktár pozíciójának meghatározása hozzárendeléssel és installációs költségekkel, automatikus járatmódosítás lehetőségei.

A dolgozatban a korábban említett irodalomkutatás után a logisztikai alapfeladatokról és azok egyszerű megoldási módszereiről esik szó. Leírom ezen egyszerű módszerek előnyeit és

DOI: 10.14750/ME.2020.018

hátrányait, alkalmazhatóságuk határait. Ezek után a heurisztikus algoritmusok általános bemutatása következik, majd azon speciális algoritmusoké, amelyekkel a kutatásaim során dolgoztam és amelyeken olyan változtatásokat eszközöltem, melyek révén azok teljesítménye akár pontosság, akár szükséges futási idő vonatkozásában fokozható. Részletesen ismertetem ezen fejlesztések hatását tesztfeladatokkal és érzékenységvizsgálattal. Mivel a logisztikához kapcsolódó optimalizálási feladatok jelentős részében a heurisztikus módszerekkel történő megoldás során permutációs reprezentációt szükséges alkalmazni (különösen járattervezési feladatok esetében), ezért szükséges volt egy olyan metrika kidolgozása, mely segítségével bizonyítottan lehetséges különböző hosszúságú számsorozatok összehasonlítása, az általuk reprezentált megoldások közötti távolság meghatározása. Erre azért volt szükség, mert azon logisztikai feladatoknál, ahol az eredményt egy sorrend (permutáció) írja le, szükséges definiálni a két sorrend közötti cseretávolságot és erre vonatkozóan nem találtam a szakirodalomban ilyen alkalmas módszert. A dolgozat utolsó részében a nagyméretű logisztikai hálózatok leírására és modellezésére készült egy általános modell-leírás mely tartalmazza azokat az elveket és alapösszefüggéseket, melyeket minden részproblémánál vagy speciális feladatnál használhatunk. Ez a rész tartalmaz három olyan egyedi modellt is, amelyeket speciális logisztikai feladatokra tudunk használni. Ezeken a modelleken is tesztelve vannak a fentebb említett heurisztikus módszerek eredeti és általam továbbfejlesztett példányai is. Záró részként megtalálható még a dolgozatban az egyik általam továbbfejlesztett algoritmusom ipari felhasználásának lehetősége, illetve jövőbeni kutatási terveim ezen a területen.

1.3. Az alkalmazott kutatási módszerek

Az előzőekben már kifejtettem, hogy a világ minden táján rohamoson növekszik a digitalizáció és a vele járó hálózatosodás miatti problémák száma. A legtöbb vállalat elavult módszerekkel próbálja kezelni ezeket a problémákat, de csak felszínesen és rossz költséghatékonysággal tudják ezt kivitelezni. Ma már az egyszerű statikus modellek nem képesek ellátni és feltárni a hibák és veszteségek nagy részét. Dinamikus modellek alkalmazhatók a logisztikai folyamatok, ellátási láncok modellezésére és új könnyen alkalmazható, robusztus módszerekre van szükség a bennük található problémák feltárására és megoldására. Ezek a problémák NP-hard jellegű problémák, azaz reális időn belül egzaktul nem megoldható problémának számítanak számítási igényüket figyelembevéve.

Egy hálózat problémája, bármilyen méretű vagy fajtájú, nem csak egy tudományterületet érint, hanem általában valamilyen szinten több tudományterületre is kiterjed. A legjelentősebb területek ilyen szempontból a gazdasági, mérnöki, informatikai és nyersanyag előállítási területek, azonban legtöbb esetben figyelembe kell venni jogi, szociális, környezetvédelmi és egészségügyi elemeket is. A szakirodalomban ez a tendencia nagyon látványos, ahogy a különböző területek összekapcsolódnak és próbálnak kompromisszumos megoldást találni egy problémára.

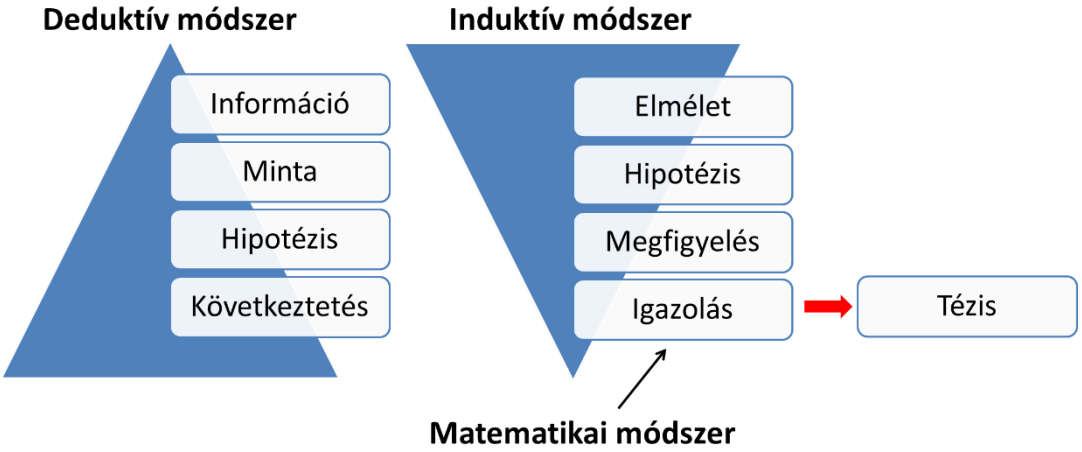
A doktoranduszi tanulmányaim elején (2015-től) tudományos vezetőim bevezettek a fent említett problémakör részletesebb feladataiba és irányt mutattak a logisztikai rendszerek optimalizálása felé. Prof. Dr. Illés Béla elsősorban az ipari jellegű problémák és megoldások majd az Ipar 4.0 elveinek a megismertetésében játszott nagy szerepet. Dr. Bányai Tamás pedig kijelölte a kutatási útirányt a logisztikai problémák matematikai és számítástechnikai

DOI: 10.14750/ME.2020.018

optimalizálással való megoldására. Azt a tudományos rést, amely a heurisztikus eljárások hatékonyságának növelése és a logisztikai problémák modellalkotása és megoldása területén található, a strukturált szakirodalmi feldolgozást tárgyaló fejezetben ismertetem részletesen. Azért választottam ezt a témát, mert a szakirodalmi feldolgozás során kiderült, hogy az Ipar 4.0 terjedésével és új digitális fejlődéssel a megoldások jelenleg nem tudnak lépést tartani és elavult megoldásokat foltoznak és toldanak meg egyes problémák megoldására [K3]. Az új modellek kiegészítésére és problémák megoldására alkalmas heurisztikus megoldásokat kevesen használják, pedig könnyen és gyorsan jó megoldást képesek alkotni.

Disszertációmban deduktív, induktív és matematikai módszereket használok fel kitűzött céljaim elérése érdekében. A szakirodalmi források feldolgozása és a kevésbé kutatott területek azonosítása során deduktív módszert alkalmaztam, ami az irodalomkutatás során egy általánosan használt módszer. Azért tartottam alkalmasnak a deduktív megközelítést, mivel a deduktív gondolkodás során egy igaznak vélt feltételezésből kiindulva alkothatunk meg új tételeket. Ezen módszerek különösen alkalmasak olyan esetekben, amikor a hipotézis csak a probléma eredetének feltárásával oldható meg [K9].

Az új tudás megszerzésének egyik leginkább elterjedt módszere az indukció, mely során megfigyeléseken, feltételezéseken alapuló hipotéziseket próbálunk meg igazolni. Amennyiben a hipotézis nem igazolható, akkor a levont tapasztalatokból új hipotézist kell előállítani, amennyiben a hipotézis igazolásra kerül, akkor abból tézist lehet megfogalmazni. Ezen gondolatmenetet alapul véve döntöttem úgy, hogy a szakirodalmi kutatás és a tudományos fehér foltok deduktív módszerrel történő azonosítását követően matematikai módszerekre alapozott induktív módszerrel valósítom meg kitűzött kutatási céljaimat (1. ábra).



1. ábra: Az értekezésben alkalmazott kutatási módszerek struktúrája

Megfogalmazott téziseimet vagy matematikai bizonyításokon keresztül, vagy ipari problémák felhasználásával validáltam és a tudományos eredményeket 23 publikáción keresztül ismertettem különböző szakmai fórumokon az elmúlt 5 évben. A releváns lista az irodalomjegyzék saját irodalom fejezetében megtekinthető.

DOI: 10.14750/ME.2020.018

2. SZAKIRODALMI ÁTTEKINTÉS

Az utóbbi évtizedben mind fizikailag, mind digitálisan összement az ismert világ. A globális szállítási hálózat kiépítésével, a kapitalizált szállítási vállalatok és az országok kereskedelemre való fokozottabb nyitásával és a szabad internet megjelenésével olyan mérvű áruforgalom indult meg, amelyet a XX. században el se tudtak képzelni. Ezt a hatalmas mennyiségű fizikai árut és információt kezelni csak megfelelő eszközökkel lehet, és mivel az igények és a mennyiségek napról napra nőnek, a szállítási határidők rövidülnek és a költségeket pedig a lehető legalacsonyabban szeretnénk tartani, a rendszer, amely ezeket kezeli, folyamatos fejlesztésre szorul. A hálózat, amely összeköti a csomópontokat, legyen az egy gyáron belül vagy kívül; raktárak, gyártóegységek, kereskedők és felhasználók között, olyan méretűvé nőttek, hogy ha egzakt módon próbálnánk meghatározni egyes tulajdonságait vagy egy problémára megoldást találni, milliós vagy „végtelen” megoldáshalmazból kellene megkeresni az optimális eredményt. Az ilyen jellegű feladatokat NP-hard problémaként kezeljük, amelyek nem determinisztikusan polinomiálisan nehéz problémákat jelentenek. Minden olyan probléma vagy feladat, amelyet véges sok lépésből meg tudunk oldani (vagy egy Turing-gép) azt polinomiális problémának nevezzük [K24]. Azok a problémák, amelyekről a továbbiakban is szó lesz, nem oldhatók meg véges sok lépésből, így az NP, azon belül is az NP-hard kategóriába sorolhatók. Ezeket a problémákat már a 1960-70-es években is próbálták számítógép segítségével megoldani, amikor a számítási kapacitása egy akkori szuperszámítógépnek meg se közelítette a mai átlagos számítógépekét, ezért okos megoldásokra volt szükség. Ezek voltak a heurisztikus algoritmusok, olyan módszerek, amelyeket általában egy természeti jelenség ihletett és elvárható időn belül képesek voltak egy az optimális megoldáshoz közeli eredményt szolgáltatni. Azóta rengeteget fejlődött ez a tudományág, amelyet, ma már minden szektor használ különféle feladatokra. A belső és külső logisztikai rendszerek, ellátási láncok nem tudnának a mai szinten létezni az informatikai háttér és a nagyteljesítményű számítások nélkül, amelyekkel képesek vagyunk valós időben útvonalat tervezni, módosítani és kirajzolni; aktuális raktárkészletet lekérdezni és analizálni pár gombnyomással vagy automatikusan működő eszközöket használni. A kutatómunkámban olyan módszereket fogok bemutatni, melyek képesek megoldani ezeknek a logisztikai feladatoknak a kezelését és irányítását.

Jelen fejezet elsődleges célja bemutatni kutatási témakörömet és megismertetni azokat a szakirodalmi forrásokat és az azokban ismertetett kutatási eredményeket, amelyek megalapozták eddig elért eredményeimet. Ebben már a kutatási fázis elejétől nagy segítségemre voltak olyan nagy repozitóriumok, mint a Scopus és a ScienceDirect.

Mivel én is és a tudományos élet is folyamatosan fejlődik, elengedhetetlen időről-időre megvizsgálni az új tudományos eredményeket, amelyek teljesen más irányba is terelhetik az éppen készülő félben lévő kutatásokat. Velem is ez volt a helyzet. Amikor nekikezdtem, nem láttam a kutatási munkám végét, hogy milyen eszközökkel és milyen eredménnyel fogom zárni az értekezésemet. Rengeteg ötletet merítettem azon emberek munkáiból, akik hasonló területen dolgoznak. Ezt a módszer a szakirodalomban szisztematikus irodalomkutatásnak (Systematic Literature Review=SLR) hívják. A szisztematikus irodalomkutatás a következő lépésekből áll:

1. Kutatási kérdések meghatározása (Ki mit csinált eddig? Ki végezte el, vagy publikálta a kutatást elsőként? Hol vannak a tudományos rések?)

DOI: 10.14750/ME.2020.018

2. Kapcsolódó irodalmak feltérképezése, elsősorban online adatbázisok segítségével.
3. Találatok redukálása, releváns irodalmak kiválasztása és elolvasásukkal a fő kutatási irány meghatározása (extra kulcsszavak megadása, dátum alapján,…).
4. Az irodalmak feldolgozási és analizálási módszerének kidolgozása.
5. Főbb tudományos áttörések és eredmények megfogalmazása.
6. Tudományos rés vagy szűk keresztmetszet meghatározása.

Ezek alapján először kulcsszavakat kell meghatározni, amelyek lefedik a kutatómunkám kérdéskörét [I1].

Doktori képzésem elején kiírt kutatási témám címe: igényvezérelt virtuális logisztikai hálózatok tervezési módszereinek fejlesztése. Szakirodalmat találni erre a kimondott területre szinte lehetetlen, ha egészben szeretnénk kezelni, viszont, ha szétbontjuk altémákra, akkor kiszűrhetjük azokat a kulcsszavakat, amelyek alapján már jóval átfogóbb képet kaphatunk a téma jelenlegi tudományos állásáról. Ebből kifolyólag három altéma-területet határoztam meg:

1. Vevő, gyártás- és szolgáltatóipari logisztikai igények és feladatok.
2. Nagyméretű logisztikai hálózatok.
3. Heurisztikus módszerek.

A következőkben ezen három terület irodalmi hátterét és releváns munkáit fogom bemutatni.

2.1. Logisztikai igények és feladatok a szakirodalomban

A vevők, gyártó- és szolgáltatóipari igények és feladatok jelentik az igényvezérelt logisztikai feladatokat és megoldásokat. Ezek azok a feladatok, melyek egy szállítási/elosztási láncban rendszeresen megjelennek. Ezek lehetnek napi feladatok, telephelytervezéskor/átszervezéskor megjelenő feladatok vagy bizonyos időnként, elsősorban ellenőrzéseknél vagy felülvizsgálatoknál megjelenő feladatok. Ebbe a témába még beletartozhat a virtuális kifejezés is, mivel ezek a kapcsolatok nem mindig fizikai kapcsolatok és/vagy csak átmenetileg állnak fenn. A következő kulcsszavakat használhatjuk az irodalomtárakban kereséskor: *demand-driven, virtual, modern logistic tasks*. Az alfejezetben szereplő ábrák és táblázatok mutatják, hogy milyen eredményeket értem el ezekre és kombinációjukra rákeresve.

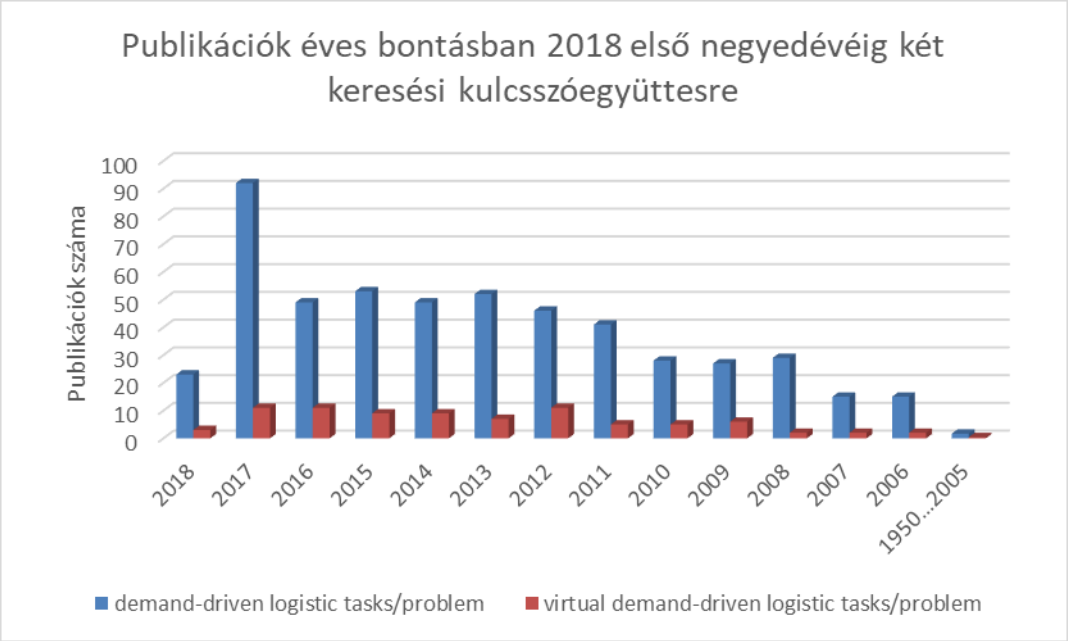
1. táblázat: Kulcsszavak találatszáma Scopus-ban

Kulcsszavak	**Publikációs darabszám**
demand-driven	8999
virtual	1057636
modern logistic tasks	5962
virtual demand-driven modern logistic tasks	7
virtual demand-driven logistic tasks	29
demand-driven logistic tasks	112
demand-driven logistic tasks/problem	612
virtual demand-driven logistic tasks/problem	105

DOI: 10.14750/ME.2020.018

Az 1. táblázat azt mutatja, hogy az eredeti kulcsszavaimat szükséges volt kicsit megváltoztatni, hogy jobb keresési eredményeket kapjak. Az első ilyen változtatás a „modern” szó kivétele volt, amely növelte a találati számot anélkül, hogy bármilyen fontos művet kivett volna. A második fontos változtatással a feladat és probléma szavakat tettem szinonimává, mivel az egyik szó magában rejti a másiknak a valószínűségét is, így nagyban javítva a találati pontosságot. Végeredményként 105 műből válogattam ki a legfontosabbakat.

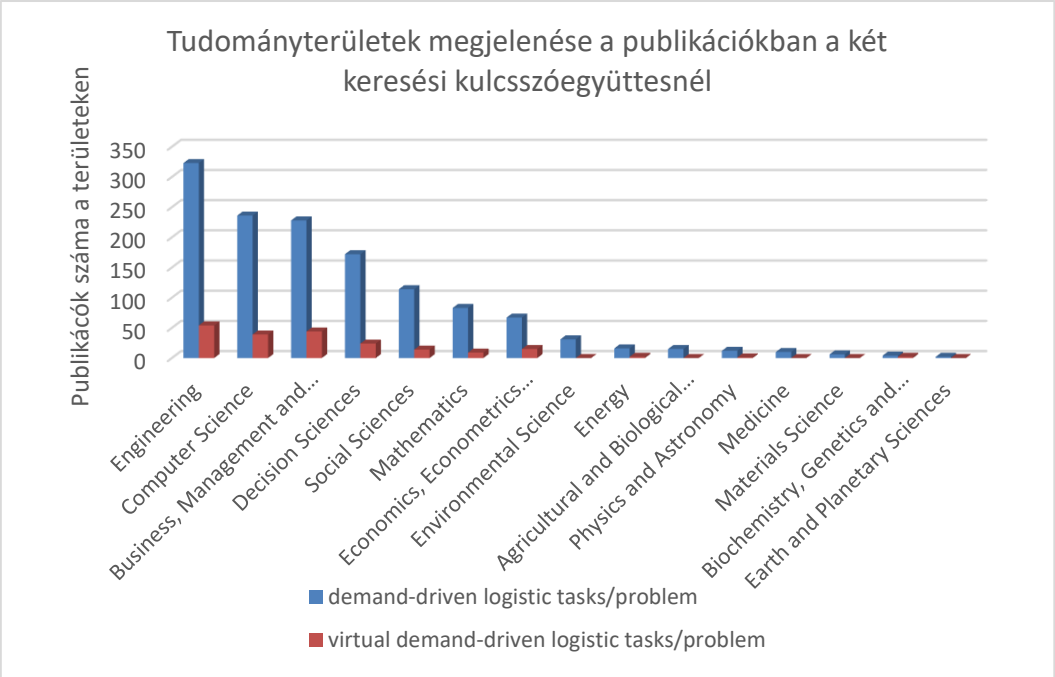
Az utolsó két lépés eredményét mutatja a 2. ábra, ahol az időrendi mennyiségek és tudományágak alapján bontottam fel a keresést. Az ábrán keresési eredmények láthatók 2005 és 2018 között plusz a 2005 előtti átlag, amelyből kitűnik, hogy ezt a terület, bár több évtizede vizsgálják, az utóbbi pár évben megugrott a népszerűsége.



2. ábra: Publikációk száma éves bontásban

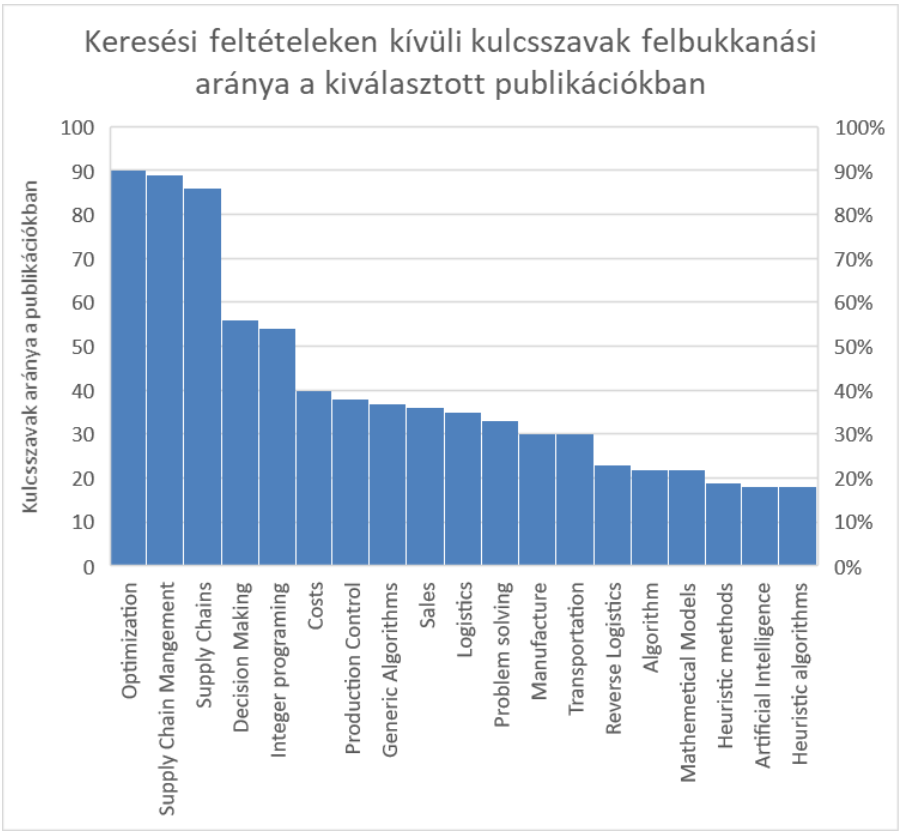
A 3. ábra az egyes tudományterületek súlyát mutatja a publikációs listában. Látható, hogy a kutatási terület multidiszciplináris jelleget mutat, hiszen nem csupán mérnöki tevékenységhez kapcsolódó tudományok találhatók meg a listán, hanem például jelentős mértékű a szociális aspektusokat vizsgáló megközelítés.

DOI: 10.14750/ME.2020.018



3. ábra: Tudományterületek megjelenése a publikációkban

A 4. ábra csak a „*demand-driven logistic tasks/problem*” keresést veszi figyelembe a 612 találatával. Ebből a listából körvonalazódott igazán, merre kell folytatnom kutatásomat, ugyanis olyan kulcsszavak előfordulását tartalmazza, amelyek jelentősen hatottak a kutatási irányom kiválasztásához.



4. ábra: Keresési kulcsszavakon kívüli címkék eloszlása a publikációkban

DOI: 10.14750/ME.2020.018

Ebben a kulcsszó-gyűjteményben megtalálhatunk általánosan használt szavakat, mint az eladások vagy a logisztika, de olyan területeket is mutatnak, melyeket később nagy részben használni fogok, mint az optimalizálás, az ellátási láncok vagy a heurisztikus algoritmusok. Mindezek úgy jelentek meg, hogy nem volt rá utaló szó a keresési szavak között.

Az ellátási lánc menedzsment az egyik leggyakrabban előforduló téma az irodalmak közül. Egy nagyon átfogó elemzést mutat be az „*A multiagent supply chain planning and coordination architecture"* című mű [I2], melyben a fizikai problémák mellett számításba veszi a szoftveres problémákat, melyekre már heurisztikus megoldásokat is alkalmaznak. Emellett rengeteg mű foglalkozik az ellátási láncok egyes tulajdonságaival. Az egyik legnehezebben értelmezhető és számszerűsíthető sajátosság egy rendszer rugalmassága. A következő mű pont ezzel a problémakörrel foglalkozik: *„Supply chain flexibility: A comprehensive review"* [I3].

A jelenlegi elvek és technológiák, mint az Ipar 4.0, megkövetelik a magas fokú automatizáltságot és kommunikációt az eszközök között. Az RFID technológia elterjedése elsősorban az azonosítás-technikában is ennek köszönheti a népszerűséget. Az *„Achieving supply chain integration using RFID technology: The case of emerging intelligent B-to-B e-commerce processes in a living laboratory"* [I4] egy esettanulmányt mutat be, ahol magas fokú kommunikációt alkottak meg RFID technológia segítségével.

A programozás jelentőségét és a vállalat szoftveres irányítását és ellenőrzését ma már senki nem kérdőjelezi meg. Az Ipar 4.0 elvek a vállalati döntések szempontjából azt jelentik, hogy a szoftver nem csak segítséget ad a döntésekben, hanem betanítás után saját maga képes döntést hozni emberi közreműködés nélkül [I5][I6]. Ennek az alapjait és megvalósításának lépéseit mutatja be JAVA programozási nyelven a „*Distributed production planning and control agent-based system*" című mű [I7].

Egy vállalat piaci és financiális helyzetét mindig is a termékei vagy szolgáltatásai iránti kereslet határozza meg. Ezek az igények és a hozzá kötődő döntések képesek olyan elemeket generálni, melyeket egy vállalat magában nem képes kielégíteni. Ilyenkor kooperálnia kell más vállatokkal különböző szektorokból, hogy ne veszítsen piacot. Ez a verseny az e-commerce és e-business világában hatványozottan igaz [I8]. A nagyvállalatok ellátási láncain belüli modern kooperációiról is számos szakirodalmi forrásban lehet olvasni [I9].

A modularizáció egy nagyon hatékony és költségkímélő eszköznek számít szinte minden területen, legyen az termék, szolgáltatás vagy szellemi tulajdon, mint egy szoftver. Maga az elv nem számít újnak, de mindig lehet olyan területet találni, amely még nem vette át. Ezt a szemléletmódot be lehet vezetni anyagáramlás szimulálására, akár újragyártás területén is [I10].

2.2. Nagyméretű logisztikai hálózatok szakirodalma

A következő kutatási altéma a nagyméretű logisztikai hálózatok, amely megfelel a témakiírásban szereplő virtuális logisztikai hálózatoknak. A mai világban rengeteg kiterjedt logisztikai hálózat működik virtuális szinten, egymással összekapcsolódva. Az egyik legjobb példa erre a gépjárműipar, ahol összeszerelő, beszállító, pénzügyi és jogi személyek csoportja dolgozik együtt egy nagyméretű, értékes termék előállításában több szinten. Ugyanis nem csak a végösszeszerelő autógyárak kapnak szinte minden alkatrészt beszállítóktól és azok raktáraitól, hanem a nagyobb alkatrészek beszállítói is a legtöbb alkatrészt más beszállítóktól kapják. Ez a

DOI: 10.14750/ME.2020.018

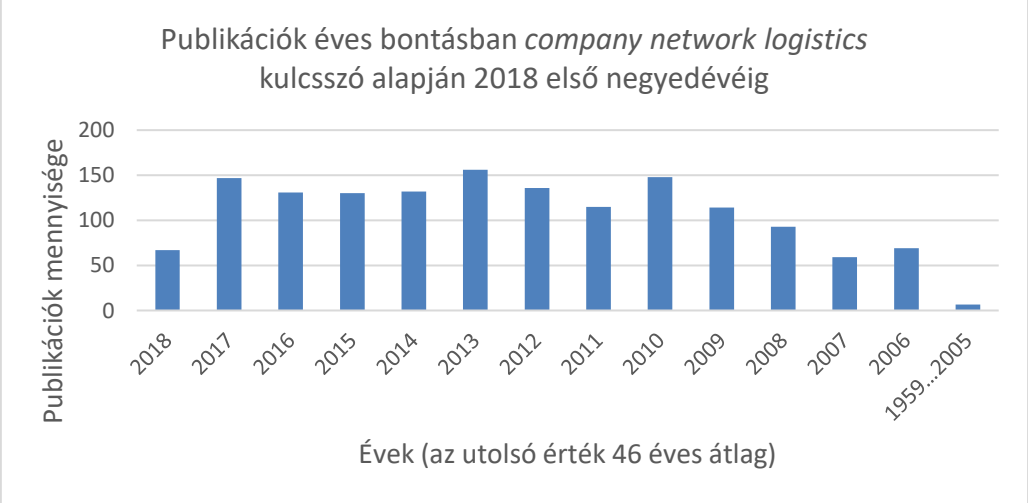
gyártó-beszállító sorrend egészen az alapanyag gyártókig vagy kitermelőkig is elérhet, amely egy olyan nagy és bonyolult hálózatot hoz létre, mely komplex tervezési és irányítási feladatok megoldását teszi szükségessé. Erre a témára a következő kulcsszavakat használtam a repozitóriumokban való keresés során: *multiconnected, connected, big, logistic network, problem, company, complex.* Az alfejezet ábráiban látható, hogy milyen eredményeket értem el ezekre és kombinációjukra rákeresve.

A 2. táblázat azt mutatja, hogy hogyan finomodott a keresés a különböző kulcsszavakra keresve. Az utolsó 5 keresési feltételnél már találtam releváns műveket, melyeket érdemesnek tartottam feldolgozni. Meglepetésként ért, hogy a *multiconnected* kulcsra mindössze 356 találatot adott ki a Scopus, amelyből egyik sem tartalmazott szakmailag releváns, kutatási területemhez kapcsolódó információt. Ezután a többi kulcsszó különböző kombinációjával sikerült leszűkíteni a területet. Még egy érdekességet találtam a *connected company logistics problem* kulcsszó-kombinációra, amely csak 37 találatot adott, azonban ezekből 6 db a Miskolci Egyetemen készült, a nagyrészük a Miskolci Egyetem Logisztikai Intézetében dolgozó témavezetőim által [I11].

2. táblázat: Keresési kulcsszavak találatai Scopus-ban

Kulcsszavak	**Publikációs darabszám**
Multiconnected	356
Connected company	8873
Connected company logistics problem	37
Company network	55721
Company network logistics	1847
Company network logistics problem	524
Big company network logistics problem	20
Complex company network logistics problem	88
Complex company network logistics	238

Az 5. ábra a *company network logistics* kulcsszó-kombinációra kapott 1847 találat évek szerinti elosztását mutatja.



5. ábra: Publikációk száma éves bontásban

DOI: 10.14750/ME.2020.018

Ahogy az az 5. ábrából jól kivehető, 2010-ig folyamatosan emelkedett a publikációk száma, majd jelen évtizedünkben viszonylag állandósult évenkénti 150 publikációban a témában írt művek mennyisége. Az utolsó oszlop itt is a 2005 előtti évenkénti átlagszámot mutatja.

A tématerületeket és az egyéb kulcsszavakat nem emelném ki ebben a fejezetben, ugyanis nagyon hasonló arányt mutatnak, mint az előző fejezetben annyi különbséggel, hogy az optimalizálás 158 db (9%) a heurisztika pedig mindösszesen 42 db (2%) publikációban jelenik meg. Ha a 42 darab heurisztikus találatra rákeresünk, akkor kiderül, hogy ezekben is találhatunk elismert publikációkat. Ezek egyike egy részecske raj alapú optimalizáló algoritmus, amelyet járattervezési feladatokra használtak fel inhomogén járműparknál időablakokat figyelembe véve [I12]. A munkában a szerzők a PSO algoritmust keresztezték lokális kereséssel, amellyel nagyon gyorsan pontos járatterveket állítottak elő.

Egy másik nagy probléma, amelyre több publikációt is találtam, a vállalatok közötti kapcsolatok kiépítésre vonatkozott, azon belül is a partnerek/beszállítók kiválasztására. Két közleményt emelnék ki, két különböző megközelítési móddal. Az első közleményben a szerzők egy paraméterek alapján előválogatással és második körben ANP, azaz analitikus hálózati folyamattal oldották meg a partnerek kiválasztását [I13]. A második módszer pedig egy sokkal fiatalabb és több paramétert figyelembe vevő módszert ír le, amelynek egy nagyon fontos komponensese a zöld technológiák és kevesebb szennyezés megvalósítása [I14].

Ebben a keresési fázisban is sokat találkoztam az Ipar 4.0 és a vele kapcsolatos témákkal. Ami nagyon érdekes volt számomra, hogy ezekben a témákban rengeteg magyar alkotás született. A [I15] publikációt több magyar egyetem gazdasági jellegű kutatói írták, amelynek témája az Ipar 4.0 hatása a vállalati stratégia alkotásra. Azonban más országok is megközelítették ezt a témát gazdasági és mérnöki szempontok alapján. A Szlovákiai Zilinai Egyetemen olyan eszközök alapjait alkották meg és fejlesztettek ki, amely funkcionalitásában és ergonómiailag is követi az Ipar 4.0 elveit [I16].

A nagy összekapcsolt logisztikai hálózatokat csak akkor lehet fenntartani, ha létezik fizikai és virtuális kapcsolat is azok entitásai között. A következő három bemutatásra kerülő publikáció alapján kiderül, hogy új elveket bevezetve sem elég szoftvertechnikai szempontból fejlődnünk, hanem a hardvert és a kapcsolódó egyéb fizikai kapcsolatokat is fejlesztenünk kell. A következő három publikáció a szállítmányozás három szintjét jeleníti meg. A HUB-ok hálózatának kialakításával foglalkozik az [I17] publikáció, amelyek légi és közúton szállítanak árukat. Az ilyen HUB-ok általuk leírt legnagyobb problémái a leszállásnál és átrakásnál jelentkeznek. Ezeknek a számát próbálják csökkenteni azáltal, hogy a megfelelő helyet választják ki és a megfelelő szállítóeszközöket. A következő irodalom a szállítójárművek okosításáról szól, amely az egyik alapkövetelménye az Ipar 4.0-nak. A szerzők olyan eszközöket és a hozzájuk kapcsolódó szoftveres megoldásokat mutatnak be, amelyek felhőalapú kommunikációra képesek, ezáltal sokkal könnyebbé tehetnék a járműfigyelést és növelhetnék a rendszer átláthatóságát [I18]. A következő publikáció a szenzor-technikáról szól, melyet ERKE-be és konténerekbe szerelnek be, az áru jobb monitorozása érdekében, hogy egy okos eszközt kapjanak. Az érveik nagyjából megegyeznek az előző publikációban tárgyaltakkal [I19].

DOI: 10.14750/ME.2020.018

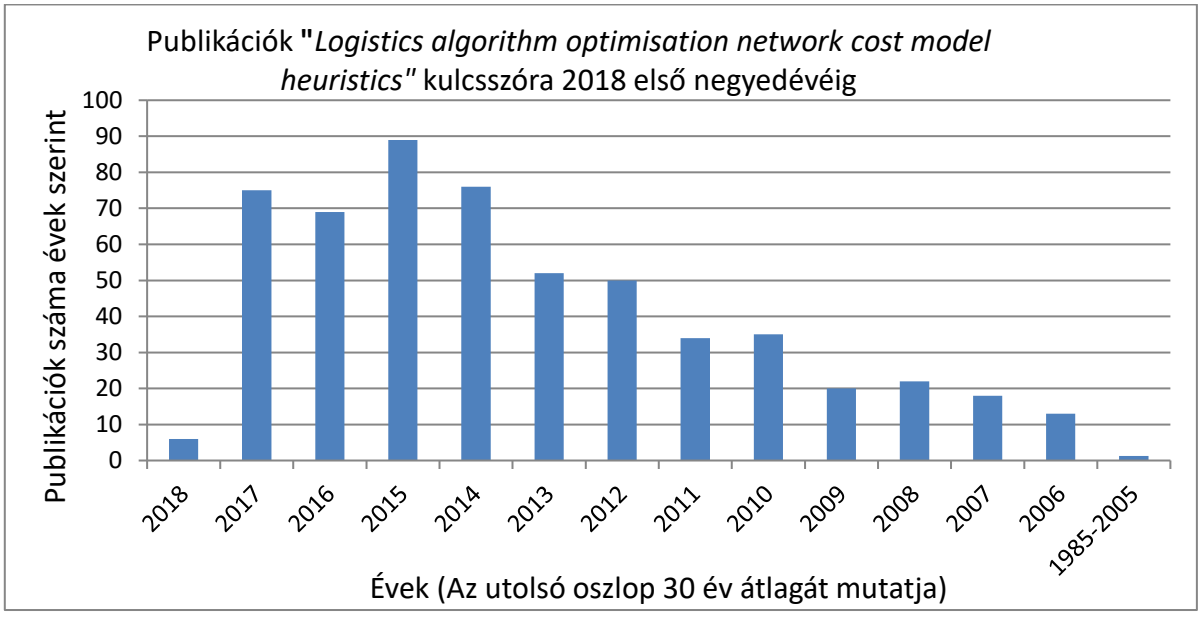
Ezekből a publikációkból is látszik, hogy a világ folyamatosan fejlődik és digitalizálódik, amely egyre nagyobb és komplexebb hálózatokat alkot. Ezek nem csak logisztikai jellegű problémákat, de számításigényes problémákat is felvetnek, amelyre a heurisztika megoldást nyújthat.

2.3. Heurisztikus módszerek a szakirodalomban

A harmadik részt és annak irodalomkutatását csak az előző kettő befejezése és kiértékelése után tudtam elkezdeni, mivel ekkor bizonyosodtam meg arról, hogy a megfelelő kutatási irányt választottam ki. Mint az a logisztikai igények és feladatok és a nagyméretű logisztikai hálózatok témakörben való keresésből kiderült (4. ábra), az optimalizálás szinte mindegyik keresett szakirodalomban szerepelt és a döntéshozatal is nagy számban megjelent. A szakirodalmak több, mint 20%-a pedig heurisztika és mesterséges intelligencia felhasználásáról is tanúskodik. A kereséseket és eredményeit felhasználva eldöntöttem, hogy a heurisztikus optimalizálás és heurisztikus algoritmusok felhasználása a logisztikában lesz a fő kutatási irányvonalam. Már csak egy nagy kérdés marad: A heurisztika mely részével vagy milyen algoritmusokkal kezdjek el dolgozni?

A kérdés megválaszolásához újabb irodalomkutatást végeztem azokon a területeken, ahol heurisztikus módszereket alkalmaztak a fentiekben felsorolt problémák, feladatok és modellek megoldására. Bár ez a téma eléggé szűk területnek tűnhet csak a Scopus adatbázisában keresve 596 tudományos munkát találtam. A kereséshez az alábbi kulcsszavakat használtam: *logistics, algorithm, optimisation, network, cost, model, heuristics.*

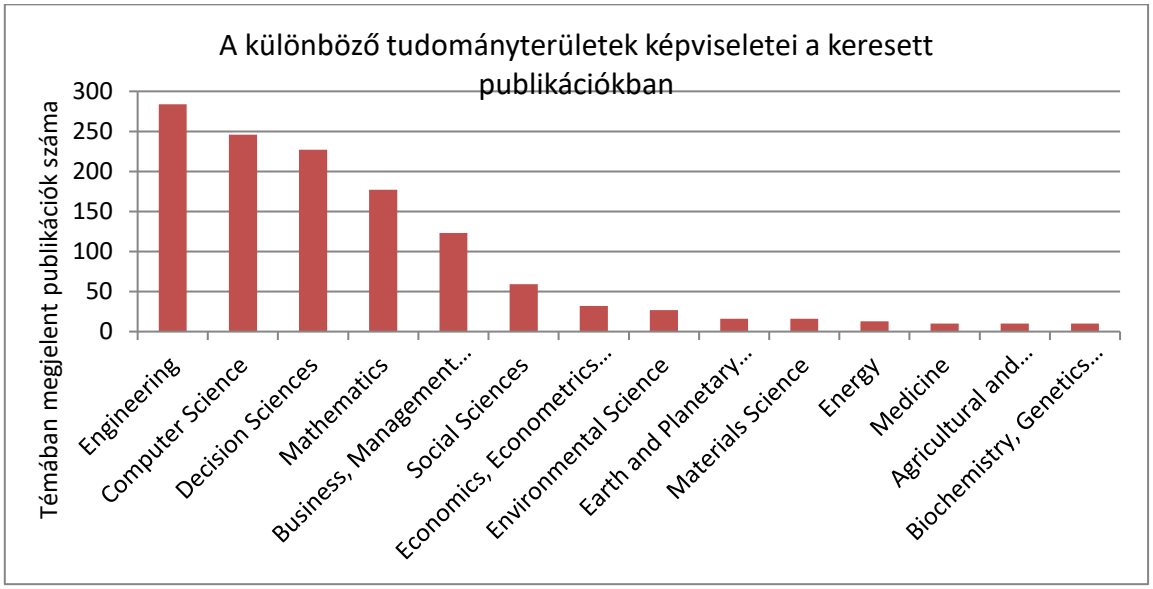
Nem nevezhető túl fiatal témának, hiszen a legrégebbi cikket 1985-ben, több mint 30 éve írták, de azóta folyamatosan fejlődő tendenciát mutat ez a tématerület az azóta megjelent munkák darabszáma alapján, amelyet a 6. ábra tartalmaz.



6. ábra Tudományos munkák mennyisége évente

DOI: 10.14750/ME.2020.018

A kutatási témám a neve alapján szinte egyértelmű, hogy a logisztika és informatika témakörhöz kapcsolódik, azonban lehet olyan felhasználási területeket is találni, amelyek teljesen váratlanul érhetik a kutatót (7.ábra). Azokat a módszereket, amelyeket felhasználok kutatásom során, mások nem csak járattervezésre és készletfigyelésre, de a gyógyászatban, a föld- és a kémiai tudományokban is használják. Ami ezen kívül még szembetűnő, az a teljes találati lista 10%-át kitevő társadalomtudományok területe közé sorolt művek, amelyek például a heurisztikus algoritmusokkal támogatott zöld technológiák hatásait vizsgálják emberekre [I20] vagy a társadalmi tényezők hatásvizsgálatát tárgyalják [I21] a városi közlekedésben és az ellátási láncokban.



7. ábra Tématerületeken megjelent publikációk száma

Az irodalomkutatás kiértékelése során megállapítható, hogy a tématerületemnek nagyon intenzív kutatói ázsiai eredetűek, a 10 legtöbbet publikáló illetőből mindössze 2 nem Kínából származik (8. ábra).

8. ábra: A téma 10 legintenzívebb kutatója a keresési feltételeken belül

DOI: 10.14750/ME.2020.018

A vizsgálatot és kiértékelést követően elkezdtem szűkíteni a keresést azokra a módszerekre és algoritmusokra, amelyeket használtam vagy használni fogok a jövőben. Először a természet alapú heurisztikus algoritmusok keltették fel az érdeklődésemet. Ebbe a csoportba tartozik minden módszer és algoritmus, amelyeket a természet inspirált. Általában könnyedén kezelhető és egyszerű logika alapján működnek, tehát könnyű őket megérteni és manipulálni. Sajnos a keresés nem járt teljes sikerrel, mivel csak 60 publikációt generált, ellenben mikor rákerestem a raj (swarm) alapú módszerekre az 596 publikációt 141-re szűkítette, pedig a raj alapú algoritmusok ugyanúgy a természet alapúak alkategóriájába tartoznak.

Ezek után kibővítettem a keresést annak érdekében, hogy megfelelő algoritmusokat találjak a kutatásomhoz. Három fontos pont alapján választottam ki azokat a metaheurisztikus algoritmusokat, amelyeket a későbbiekben alapos irodalomkutatás után bemutatok és használok a kutatásaimhoz:

- széles körben elterjedt: Genetikus algoritmus
- megbízható és pontos: Genetikus algoritmus/ Harmony search algoritmus
- új ígéretes algoritmusok: Firefly és Black-hole algoritmusok

Több száz lehetséges és húsz potenciális algoritmusból alapos válogatás után, amelynél figyelembe vettem a fent említett indokokon kívül az algoritmus értelmezhetőségét, programozhatóságát és módosíthatóságát, erre a 4 algoritmusra esett a választás. Ezen algoritmusokat alapos irodalomkutatás és módszerismeret elsajátítása után használok fel és a két ígéretes algoritmus esetében át is alakítom kutatásaimhoz.

Időrendi sorrendben a Harmony Search algoritmus volt az első, amelyet teszteltem logisztikai problémákra. Elsősorban Make or Buy döntéshozatalt elősegítő [S1] és hátizsák problémát megoldó [S2] egyszerű szoftverek motorjaként szolgált számomra, és ez szolgáltatta az alapot az egyik legelismertebb publikációmhoz, amelyben a szerelőhálózat járattervét és készletszintjét optimalizáltam [S3]. Azért választottam ezt az algoritmust kezdésnek, mert könnyű volt megérteni, kezelni és rendkívüli a módosíthatósága, egész-, és valósértékű programozásra is alkalmas. Az 596 publikációból mindössze 13 szólt erről a módszerről. Az egyik legjelentősebb publikáció ebben a témában [I22] mű, amely bemutatja a Harmony Search algoritmus problémamegoldó képességét egészértékű programozáson keresztül egy járattervezési és járműpark szabályozási feladat példáján. Bár a következő publikáció nem került bele a találati listába, érdemes megemlíteni, ugyanis a Harmony Search algoritmus e nélkül a cikk nélkül talán nem is létezne, amelyet a kitalálói publikáltak: *Geem, Z.W., Kim, J.H., Loganathan, G.V.: A New Heuristic Optimization Algorithm: Harmony Search* [I23]. A 2001-ben megjelent kiadványt ma már több mint 2300-an hivatkozták és az algoritmus alapjairól és működéséről számol be. A természeti alapokon nyugvó algoritmust zenészek improvizált összjátéka inspirálta. Megfigyelték, hogy egyes zenészek egymásra figyelnek és próbálják egymás játékát követni különböző zenei hanghatásokkal, mint a hangmagasság vagy egy szólam hossza. Ezáltal a zenei szám egyre összetettebb és szebb lesz, megteremtve a zenei harmóniát. Az algoritmus kitalálói azzal az ötlettel álltak elő, ha a zenészeket kicseréljük változókra, amelyek egymásra hatnak és úgy változnak, hogy egy célfüggvényben minél jobb értéket érjenek el, nagyon jól lehet vele optimális megoldásokat keresni egy keresési térben [I23]. Ez az algoritmus alkalmas mind az egész-, mind a részértékű keresésekre és elsősorban optimális helymeghatározásra használják, mint raktárak és gyáregységek telepítése [I24].

Azonban magas fokú alakíthatósága miatt, rendkívül alkalmas más területeken való optimalizálásra vagy optimumkeresési eljárások kiegészítésére.

A raj alapú algoritmusok szintén természet által inspirált módszerek optimumkeresésre. Közös tulajdonságuk, hogy valamilyen állatcsoportnak az együttes mozgását vagy életterét próbálják matematikailag leírni. Az állatokon belül 3 jelentős állattípusra tudjuk bontani őket. A legelterjedtebbek a rovar alapú algoritmusok, mint a hangya az Ant Colony algoritmus, a szentjánosbogár a Firefly algoritmus vagy a méhek az Artifical Bee Colony módszer. Utóbbiról ebben a cikkben bővebben lehet tájékozódni [I25]. Az első kettőről lesz szó a későbbiekben. A második és harmadik állatcsoport a madarak és halak, amelyek rajokban élnek. A madarakból a Kakukk keresési módszert többen is használják [I26] és a halak mozgását követő raj alapú algoritmust írja le a [I27] publikáció. A raj alapú optimumkeresés nagyon elterjedt és a genetikus algoritmus után szintén egy ilyen módszer áll az alkalmazások számát tekintve. A neve Részecske-raj alapú keresés és egy könnyen tanítható általánosított optimalizációs eljárás. Nincs megadva milyen állatfajt utánoz, ugyanis megfelelő paraméterekkel és módosításokkal szinte bármire megtanítható. Erről az algoritmusról többet az alábbi kiadványban lehet megtudni [I28]. A *swarm* szóra rákeresve 141 publikációt találtam, amelyben megemlítették, tehát egy olyan területről beszélünk, amely nagyban befolyásolja a kutatásaimat. Ezt bővebben kifejtem egy összefoglaló jellegű publikációmban, ahol a Harmony Search, Ant Colony és Firefly algoritmussal elért eredményeimet foglalom össze egy másik perspektívából [S4].

Következőnek a Firefly algoritmusra kerestem rá, amelyből ezekkel a keresési feltételekkel csak 5 darabot talált a Scopus. Azonban ezek között két jelentős kiadvány is szerepel. Az első publikációban egy módosított Firefly algoritmusról számolnak be, amely képes meghatározni bizonyos feltételek mellett egymással anyagáramlási kapcsolatban álló és meghatározott kapacitással rendelkező épületek telepítését: *A hybrid Firefly-Genetic Algorithm for the capacitated facility location problem* [I29]. Az algoritmus rajalapú intelligenciára épít azáltal, hogy matematikailag írja fel a szentjánosbogarak párzási szokásait. A nőstény egyedek a hímeket éjszaka a világító potrohuk fényének erőssége alapján választják ki. Az algoritmus leírói ezt a jelenséget akarták kihasználni azáltal, hogy egy rovar fényessége reprezentálja az egyed optimum ponthoz lévő közelségét és minél fényesebb valamelyik egyed, annál jobban közelednek felé a raj halványabb tagjai minden iterációban [I30]. A másik egy review cikk, amely nagyon jó kiindulási alapot ad az algoritmus alapjairól és módosíthatóságáról, majd egy új felhasználási területet mutat be, a matematikából ismert káosz teóriát próbálják meg szimulálni, amely dinamikus hatásokat figyel érzékeny rendszerekben [I31]. Személy szerint a szentjánosbogarak mozgását járattervezési, készletfigyelési, készlet előrejelzési és járműpark méretezési feladatokra használtam [S5].

A Firefly algoritmussal együtt megismertem az Ant Colony algoritmust is, mivel mindkettő rovarok mozgásának matematikai leírásával keletkezett és vannak hasonlóságok. A keresési eredmények sem leptek meg, ugyanis a közel 600-as találati listát ez az algoritmus mindössze negyedére szűkítette, pontosabban 134 publikációban van megemlítve a módszer. Az algoritmus alapjai 1992-ben *Marco Dorigo* PhD munkájaként [I32][I33] születtek meg és azóta folyamatosan fejlesztik, aminek az okán ma az egyik legismertebb raj alapú algoritmus [I34] és rengeteg helyen használják. A járattervezésben felmerülő problémákat szinte egy az egyben át lehet ültetni a hangyák élelemhez történő útvonalkereséséhez. Ha egy hangya megtalálta az élelmet, visszamegy a bolyba egy feromoncsíkot húzva. A többi hangya pedig

eldöntheti, hogy követi-e a csíkot vagy új útvonalat keres. A feromon erősségét és a hangyák mennyiségét és intelligenciáját egyszerű változókkal lehet szabályozni. A legtöbb raj alapú módszerhez híven ez is egészértékű programozással dolgozik, viszont át lehet alakítani más típusú feladatok megoldására is. A robotirányításban, készletfigyelésben és ütemezésben is nagy szerepet kapnak a raj alapú algoritmusok. Ezekről részletesebben az [I35] publikációban olvashatnak. A matematikában főként gráfelméleti területeket érintő problémák megoldására használják [I36].

A Black Hole algoritmus a többihez képest nagyon fiatalnak számít. 2013-ben ismertette *Hatamlou A.* és a természet alapú algoritmusokon belül a populáció alapúakhoz tartozik [I37]. Sajnos a keresési feltételeknek egyetlen dokumentum sem felelt meg és köszönhetően a fiatal kora és az ismertségének hiánya miatt az egész Scopus adatbázisban mindössze 46 dokumentumot töltöttek fel, amelynek köze van a módszerhez, pedig egy könnyen használható gyors és ötletes algoritmusról beszélünk. A módszer a genetikus algoritmushoz hasonlóan fitnesz értéket használ, ez alapján állapítja meg, hogy egy felvett ponthoz milyen értéket rendel. A keresés úgy kezdődik, hogy az általunk kijelölt keresési térben random helyeken elhelyezünk pontokat, ezek reprezentálják az űrben a testeket, mint a nap és bolygók. Mindegyik fitnesz értékét kiszámolva a legnagyobbat kinevezzük fekete lyuknak és az elkezdi vonzani a többi égitestet. Iterációnként változhat a feketelyuk identitása. Ha egy égitest túl közel kerül a feketelyukhoz, tehát átlépi az eseményhorizontot, egyszerűen elnyelődik, azonban ezzel együtt egy új égitest születik egy véletlen pontban a keresési térben. Ez az alap algoritmus leírása, azonban rengeteg apró dologgal ki lehet egészíteni, hogy még pontosabb, gyorsabb és realisztikusabb legyen az eredmény. Ilyen a gravitáció bevezetése és az égitestek saját gravitációs mezeje, röppályák kiszámítása, tömegváltozások figyelembevétele, stb [I37]. Az adatfeldolgozás és csoportosítás területén több publikáció is megjelent az algoritmusról vagy más módszerekkel való összehasonlításáról. A következő cikkben a Kakukk keresési módszerrel von párhuzamot a szerző [I38]. Vannak egyes szerzők, akik a Black hole algoritmust a Particle Swarm algoritmus egy változatának tartják, amelybe bele lett építve egy tehetetlenségi tömeg [I39]. Született néhány olyan publikáció is, amely magas gyakorlati értéket tartogat és az elektromos hálózat kiépítését veszi figyelembe kapacitásszámításokkal [I40]. Egy másik dokumentum szintén az elektromos energiaellátást hivatott optimalizálni, csak itt elsősorban a zöld energia előállító egységek, mint a szél- és naperőművek telepítésével és hálózatba történő bekötésével foglalkozik [I41]. Én logisztikai feladatok megoldására használtam az algoritmust, mint amilyen a kétszintű milkrun ellátási rendszer és közbenső buffer-raktárok telepítése gyáron belül [S4], valamint a hasonló elven működő inverz ellátási rendszer, amely inkább az újrahasznosító üzemek és közbenső állomások telepítésére fektet hangsúlyt [S7].

A genetikus algoritmust nem kell bemutatnom senkinek, aki optimalizálással és algoritmusokkal dolgozik. A legismertebb és legelterjedtebb módszer az összes közül, nem csoda, hogy az 596 dokumentumnak több mint a felében szerepel, egész pontosan 357 publikációban tesznek róla említést. Egy olyan heurisztikus algoritmusról beszélünk, amelyet a 80-as évek végén kezdtek el intenzíven használni. Az algoritmus alapja a természetes kiválasztódás folyamata. Létrehozunk egy populációt, amelyet elkezdünk szaporítani, minden egyes iterációban, amit itt generációnak hívunk, az egyedeket a szülők tulajdonságai alapján készítjük el. A tulajdonságok a változók, amiket meg akarunk határozni, hogy a legjobb

DOI: 10.14750/ME.2020.018

eredményt adják. Egy célfüggvény segítségével minden egyes egyedre kiszámolunk egy fitnesz értéket, minél magasabb az érték, annál jobban teljesít az egyed. Egy generáció létrehozása során csak a legerősebbek maradnak meg, a legkisebb fitnesz értékűek kiesnek, ez a természetes kiválasztódás. A tulajdonságokat, vagyis változókat örökléssel kaphatják meg az új egyedek, ami az érték teljes másolását jelenti teljes vagy részbeni örökléssel, amely csak egy közelítő értéket jelent a szülő és a gyerek között. Felléphet mutáció is, amikor egy teljesen egyedi értéket kap a gyerek. Az alap genetikus algoritmusról két érdekes és kedvelt publikációt is felhasználtam, az egyik egy 1986-ban írt tanulmány a változók és a mutációk finomhangolásáról [I42]. A másik dokumentum szintén az alap genetikus algoritmust vizsgálja a modern genetikus algoritmusokkal szemben, amelyben a konvergencia, a pontosság, az idő és a számítási kapacitás is összehasonlításra kerül [I43]. Mint már korábban említettem, az evolúciós algoritmus régóta létezik és nem egy fajtája van. Egy 2011-es tanulmány a MEGA5-t, az 5. generációjú Molekuláris Evolúciós Genetikus Analízis optimalizáló szoftvert mutatja be, amelyben egy speciálisan módosított genetikus algoritmus működtet és egyaránt alkalmas általános és speciális feladatokra [I44]. A genetikus algoritmusnak van még egy nagyon fontos szerepe a tudományos életben. Amikor egy új heurisztikus algoritmus megjelenik elsődlegesen ehhez szokták mérni a tulajdonságait és használhatóságát, ugyanis úgy kezelik a legtöbben, mint egy standardot. Ebben a cikkben két széles körben használt módszert, a genetikus algoritmust és a részecske-raj alapú optimalizálást hasonlítja össze a szerző, mivel mind a kettő hasonló alapokon nyugszik [I45].

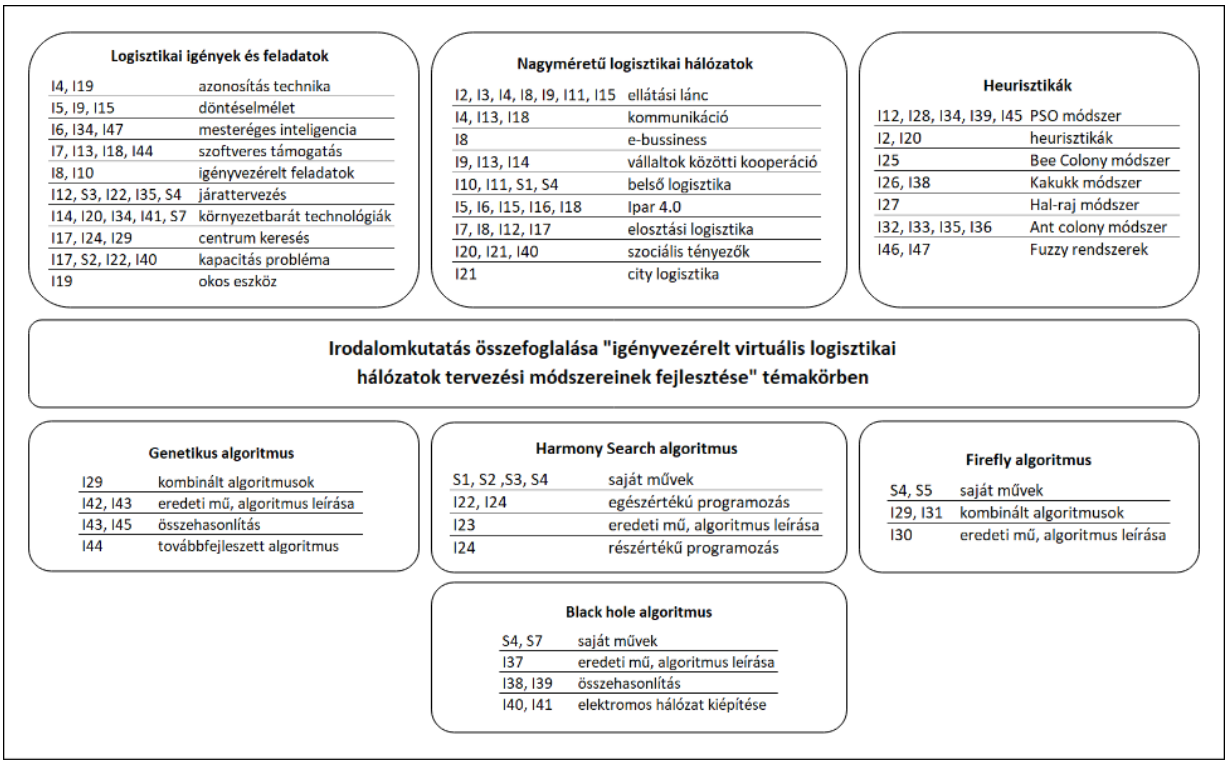
Az eddig bemutatott művekben is látható a tényleges felhasználói irány, miszerint heurisztikát csak akkor alkalmaznak, ha szükséges. A heurisztikák alkalmazásának elsődleges oka a nagy számítási igényű feladatok megoldásainak keresése viszonylag rövid idő alatt egy megfelelő, de nem feltétlen tökéletes megoldással. Egyéb okai lehetnek a hiányzó vagy helytelen adatok, vagy a túl sok paraméter súlyozása, amelyek esetében nem egyértelműen definiálható az eredményre gyakorolt hatás [I12]. Minden más esetben érdemes analitikus matematikai (lineáris programozás) vagy más elven működő (például: mesterséges intelligencia, klaszterezés…) módszert keresni. Azonban a feladatokon belül is lehetnek olyan részproblémák, amelyeket egyik vagy másik módszercsoport jobban tud megoldani. Ilyenkor kombinált algoritmusokat alkalmazunk, amelyekben a különböző feladatokat megvizsgálás és kiválasztás után egy adott módszer végzi el. Ha heurisztikákat is alkalmazunk, akkor kombinált heurisztikus módszernek is nevezzük, amelyben egy vagy több heurisztikus és lineáris programozás módszer is keveredhet [I29]. Az iparban is a legtöbb algoritmust ezzel a módszerrel készítik el, ugyanis ha a részfeladatokra a megfelelő módszert alkalmazzuk és egy részfeladat eredményét egy másik módszer bemenetének tekinthetjük, akkor könnyebben képesek vagyunk párhuzamosítani feladatokat. Ezek együttesen növelik a számítási teljesítményt és a konvergencia sebességet, ezzel együtt pedig csökken a számításhoz szükséges idő [S17].

Utolsóként nem egy meghatározott módszerre kerestem rá, hanem egy metódusra, amelyet bármelyik egzakt algoritmusba bele lehet építeni. A fuzzy logika nagyon jelentős a számítástechnika, matematika, logika és nyelvészet területein. Az elv szerint az elméletben igen, de a gyakorlatban a legtöbb esetben nem tudunk élesen egy pontos határral elválasztani többdöntésű halmazokat. Tehát nem minden igen (1) és nem (0), hanem létezik talán (0,5), inkább nem (0,2) és egyéb értékek is. Ez az informatikában azért jelentős, mert minden felírható

DOI: 10.14750/ME.2020.018

nullát és egyet tartalmazó halmazokra és pontosan így működik az egész értékű programozásoknál is. A változók csak egy bizonyos értékkészletből tudnak választani. Ha ezeket a határokat kicsit elmossuk és engedjük az algoritmusokat szabadabban dolgozni, lehet jobb eredményt kapunk a keresés végén és még a számításigény is csökkenthet. Az [I46] a fuzzy logika alapvető tulajdonságairól beszél. A fuzzy logika, mint már említettem, nagyon jól összekapcsolható különböző keresési módszerekkel és gépi tanulással is. A következő publikációban a K-közép módszerrel és gépi tanulással csoportosítanak adatokat a fuzzy elv figyelembevételével [I47]. A Scopus adatbázisában való keresésem eredményeképpen 171 publikációban találtam nyomokat a fuzzy logika használatára az eddig is kiszűrt 596 darabból, tehát itt is elmondhatjuk, hogy jelentősen képes befolyásolni a kutatási témámat.

Az irodalomkutatási fejezetben bemutatott irodalmakról a 9. ábrán egy egyszerű összefoglalást láthatunk, amely a legfontosabb kulcsszavakat vagy témákat tartalmazza.



9. ábra: Irodalmi kutatás összefoglalása

DOI: 10.14750/ME.2020.018

3. ALGORITMUSOK BEMUTATÁSA

3.1. Általános optimalizálási feladatok megjelenése a logisztikában

Mint ahogy azt már korábban is említettem, egy komplex logisztikai rendszer megtervezése és működésének optimalizálása nagyon komoly és számításigényes feladat. Nem elég, hogy minden bevezetett új objektummal, információval vagy adattal, amivel pontosítani vagy realisztikusabbá akarjuk tenni a kalkulációnkat a legtöbb esetben folyamatosan növeljük a lehetséges megoldások számát (a legtöbb esetben exponenciálisan), de még csak nem is biztos, hogy jobb eredményt kapunk. Ezért nagyon fontos figyelembe venni, hogy mit és hogyan használunk föl, milyen hibalehetőségekkel és mekkora eséllyel vegyünk figyelembe.

A számítógép elterjedése előtti időkben is létezett logisztikai optimalizálás. Azokban az időkben az emberek a józan ész, rengeteg egyszerűsítés, becslés és általános matematikai szabályok alapján képesek voltak a korukhoz képest jól működő rendszereket létrehozni és működtetni. A legtöbb ilyen módszer problémaspecifikus volt, ami azt jelenti, hogy sok feltételeknek kell teljesülnie ahhoz, hogy használható legyen. Legjobb példák erre a gyakorlati képletek vagy gépspecifikus ábrák, melyekről információt olvashatunk le.

Ismerünk és tanítunk is olyan egyszerű technikákat, melyek segítségével bármilyen komolyabb segédeszköz nélkül, mindössze papír, ceruza és józan ész segítségével képesek vagyunk megoldani egyszerű feladatokat. Az egyik ilyen feladat az egyszerű centrumkeresési feladat. Két számítási módszert szeretnék bemutatni, amelyek hasonló elven működnek.

Tesztfeladat:

15 magyarországi telephelyről, amelyek a térképen jelölve vannak, különböző mennyiségű anyagot kell egy központi raktárba szállítani. Az elszállítandó mennyiségek a teherautónyi mennyiség/év értékkel a pontok mellett jelölve vannak. Határozzuk meg a központi raktár helyét, hogy a lehető legkevesebb szállítási munkát jelentse számunkra.

Fizikai eszközöket (méretpontos térképet) használó módszerek:

A területhálós módszer [K10]

Lépései:

1) Kiinduló adatok meghatározása (telepítendő üzemmel szállítási kapcsolatban lévő üzemek léptékhelyes elrendezési rajza, anyagáramlási intenzitások értékei)
2) Léptékhelyes alaprajz elkészítése
3) A hálózat megszerkesztése (a léptékhelyes vázlatra egy hálózatot kell illeszteni úgy, hogy minden üzemet jelölő pont a hálózat külön elemébe kerüljön)
4) A szükséges számítások végrehajtása
    - első oszlopba, illetve sorba soronként, illetve oszloponként összegzett anyagáramlási intenzitás értékek kerülnek,

DOI: 10.14750/ME.2020.018

- a második oszlopba, illetve sorba felülről lefelé, illetve balról jobbra kumulált anyagáramlási intenzitás értékek kerülnek,
- a harmadik oszlopba ill. sorba az alulról felfelé, ill. a jobbról balra kumulált értékek kerülnek,
- az utolsó oszlopba, illetve sorba a soronkénti és oszloponkénti kumulált értékek különbségeit (abszolút értékben) írjuk.

5) Új üzem telepítési körzetének kiválasztása (a minimális differenciájú sor és oszlop kereszteződése)

Terület alapú súlyfelező módszer

Lépései:

1) Kiinduló adatok meghatározása (telepítendő üzemmel szállítási kapcsolatban lévő üzemek léptékhelyes elrendezési rajza, anyagáramlási intenzitások értékei)
2) Léptékhelyes alaprajz elkészítése
3) Összeadjuk az összes anyagáramlási intenzitást egyforma előjellel (abszolút értékben)
4) A következő lépéssorozatot néhányszor megismételjük (négyszer elég)
   - kiválasztunk egy véletlen irányt (lehetőleg egy vonalzó segítségével)
   - erre az irányra merőlegesen a pontok halmazán kívülre ráhelyezünk egy vonalzót (derékszögű)
   - a vonalzót az irányon tartva elkezdjük a ponthalmaz fele húzni
   - amint a vonalzónk áthalad az első ponton az anyagáram intenzitását (értékét) megjegyezzük, a további pontok értékeit, amin szintén áthalad a vonalzó, pedig hozzáadjuk kumuláltan
   - ahol a kumulált érték túllépi az összes anyagáram intenzitás felét, oda húzunk egy vonalat merőlegesen az iránnyal
5) Új üzem telepítési körzetének kiválasztása (Az a terület, amit a húzott vonalak körbe zárnak)

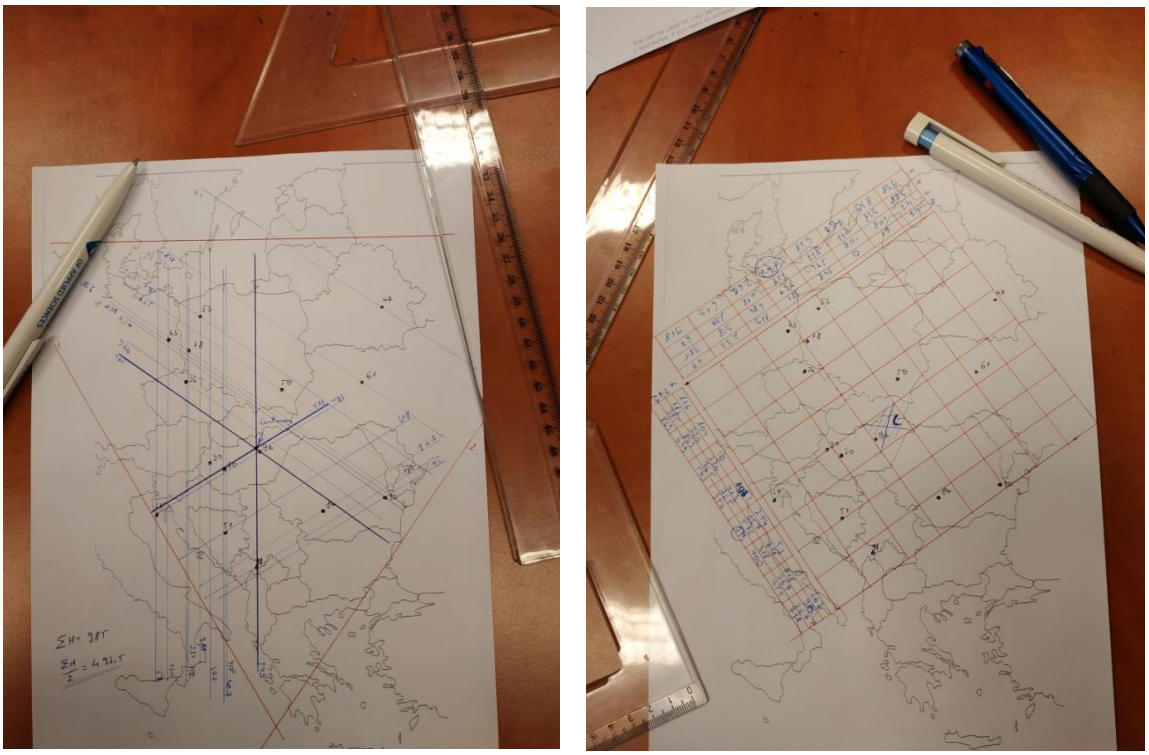
A két feladat megoldását a 10. ábra mutatja. A megoldásokat a szemléltetés érdekében kézzel készítettem és nagyjából 15 percig tartott megoldanom mindkettővel a feladatot. Az is látható, hogy a centrum helye minkét esetben a gépi megoldásokkal is megegyezik (lásd a későbbi fejezetekben).

Megállapítható, hogy a két módszer nagyon gyors és hatékony, azonban az eredmény mindkét esetben pontatlan, és inkább csak egy régiót jelöl ki, mint konkrét helyszínt. Ez viszont csak az egyik probléma. Egyik módszer sem képes teljes újragondolás nélkül másféle adatokkal működni. Néhány példa:

- Nem lehet velük 3D-ben gondolkozni.
- Csak az anyagáram intenzitást veszik figyelembe,
- A mai világban elengedhetetlen logisztikai elveket figyelmen kívül hagyja, úgy mint
  - üresjáratok száma,
  - gyűjtő/elosztó járatok létrehozásának esélyei,
  - domborzat, infrastruktúra, lokális árak és kedvezmények,
  - munkaerő,

DOI: 10.14750/ME.2020.018

    - egyéb.
- Nem lehet több raktár helyét megállapítani velük.



10. ábra: Centrumkeresési megoldások kézzel készítve

Ezek ellenére bizonyos megkötésekkel és kis hibarátával lehetett ezeket a módszereket használni. Azokban az időkben még csak nagyon minimálisan létezett szervezett nemzetközi termék és tőkeáramlás, amely legtöbbször az országok vezetőinek a kezében összpontosult. Azonban az 1950-es évektől megkezdődött a globalizáció és elkezdtek kialakulni internacionális vállalatok, amelyek a világ különböző pontjairól kezdtek el nyersanyagot vagy alkatrészt beszerezni és küldeni termékeket szintén országhatáron kívülre. Ezekből lettek később az ellátási láncok. Ezeket már nem lehet egyszerű feladat-specifikus módszerekkel optimálisan kialakítani vagy működtetni, mert túl sok tényező van, amelyek más tényezőket vagy saját magukat is befolyásolják. Ezeknek a megoldására használunk a mai világban szimulációs programokat és optimalizáló algoritmusokat.

Az előző feladatot nagyon egyszerűen és gyorsan meg tudjuk oldani, közel tökéletes megoldással a szinte mindenki számára hozzáférhető Excel segítségével, mindössze egy Solver bővítmény kell hozzá és minimális logika.

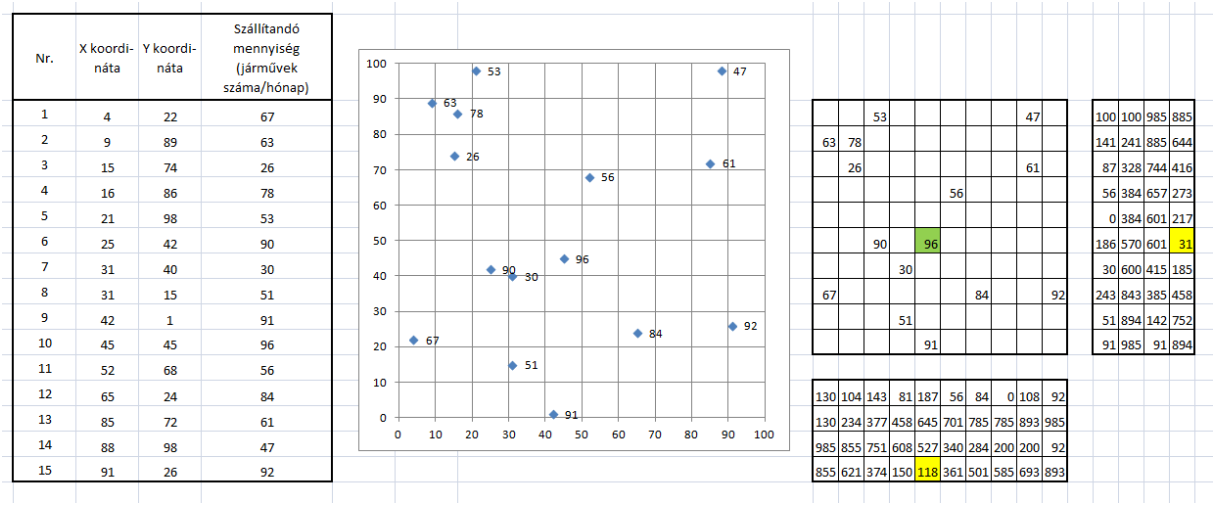
Excel és Solver kép és megoldás

A 11. és a 12. ábra az Excelben készült megoldásokat szemlélteti. A 11. ábra a szimpla területhálós módszert mutatja, amelyet már papíron is láthattunk és ugyanazt a megoldást szolgáltatja. A 12. ábra viszont már egy tényleges megoldást mutat be az Excel Solver

DOI: 10.14750/ME.2020.018

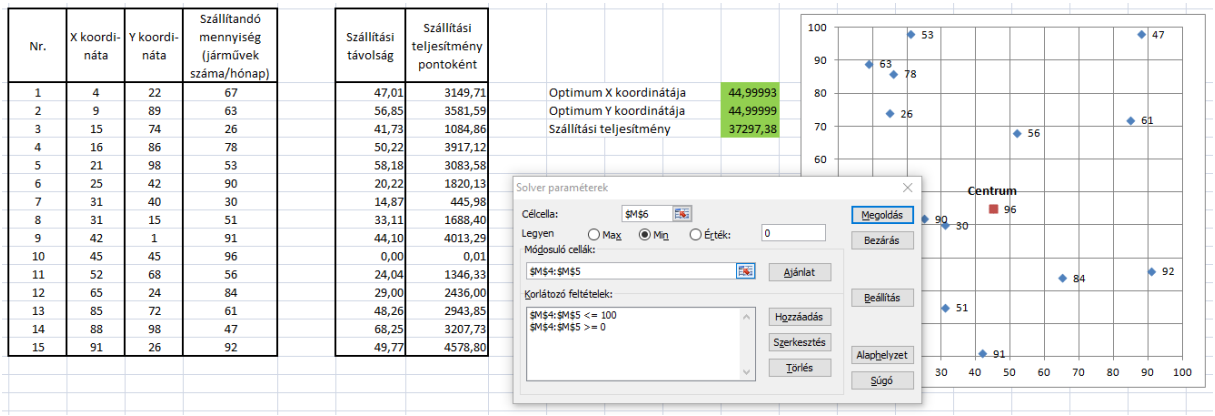
bővítményének a segítségével. Ez a módszer már olyan beépített funkciókat használ, melyek képesek tényleges megoldást nyújtani, mint azt itt is láthatjuk, mely szerint a fenti problémára a megoldás (44,999933; 44,999999) ponton található. Látható, hogy ez is csak egy közelítő megoldás, amit a másodperc töredéke alatt talált a programunk, mivel (45; 45) –ben van a tényleges optimális megoldás, de a hibahatárunk jelenleg:

$$\frac{1}{100} * ABS\left[1 - \frac{\sqrt{44{,}999933^2 + 44{,}999999^2}}{\sqrt{45^2 + 45^2}}\right] = 1{,}8556 * 10^{-8} \qquad (2)$$



Nr.	X koordináta	Y koordináta	Szállítandó mennyiség (járművek száma/hónap)
1	4	22	67
2	9	89	63
3	15	74	26
4	16	86	78
5	21	98	53
6	25	42	90
7	31	40	30
8	31	15	51
9	42	1	91
10	45	45	96
11	52	68	56
12	65	24	84
13	85	72	61
14	88	98	47
15	91	26	92

11. ábra Területhálós módszer Excellel



Nr.	X koordináta	Y koordináta	Szállítandó mennyiség (járművek száma/hónap)	Szállítási távolság	Szállítási teljesítmény pontoként
1	4	22	67	47,01	3149,71
2	9	89	63	56,85	3581,59
3	15	74	26	41,73	1084,86
4	16	86	78	50,22	3917,12
5	21	98	53	58,18	3083,58
6	25	42	90	20,22	1820,13
7	31	40	30	14,87	445,98
8	31	15	51	33,11	1688,40
9	42	1	91	44,10	4013,29
10	45	45	96	0,00	0,01
11	52	68	56	24,04	1346,33
12	65	24	84	29,00	2436,00
13	85	72	61	48,26	2943,85
14	88	98	47	68,25	3207,73
15	91	26	92	49,77	4578,80

Optimum X koordinátája	44,99993
Optimum Y koordinátája	44,99999
Szállítási teljesítmény	37297,38

12. ábra Centrumkeresési feladat Solverrel megoldva

Tehát nagyságrendileg a százmilliomod tévedési határon belül vagyunk, ha a keresési tér 0 és 100 közötti mindkét koordinátára, amelyet az 1/100 érték reprezentál a számítás elején.

A beépített funkciók a Solverben a heurisztikus és metaheurisztikus módszerek közé tartoznak, ugyanis közelítéssel és iterációs elven próbálják megalkotni a megoldást a megadott feltételekkel. Az egyik ilyen módszer az evolutív keresés, melyről a későbbiekben lesz szó.

DOI: 10.14750/ME.2020.018

3.2. Heurisztikus módszerek bemutatása

A heurisztika rátalálást jelent, amikor a megoldást véletlenül, nem logikai következtetések útján vagy egy nem bizonyított módszerrel kapjuk meg. A heurisztikus módszerek már évezredek óta használatosak emberek számára, csak nem mindig vettük észre őket. Az emberi problémamegoldó folyamatok kutatásában forradalmian új szemléletet képviselt Allen Newell és a Nobel-díjas Herbert A. Simon „Általános Problémamegoldó Modell”-je, melyet az 1970-es években alkottak meg [K11]. A könyvben szó esik olyan, az emberekbe születésükkor „beépített” és életük során folyamatosan fejlesztett képességekről és funkciókról, melyek heurisztikus úton működnek és legtöbbször a tapasztalaton alapulnak. Ilyen alapvető módszereink: a becslés, a józan ész, az intuíció vagy a körvonalazás.

Mérnöki tudományokban heurisztikus módszereknek nevezzük azokat a módszereket, melyek úgy képesek megoldást nyújtani egy problémára, hogy nem tudjuk teljesen logikusan megmagyarázni a működésüket, tehát nem vezethetők le tényleges matematikai és fizikai axiómákból. Ezek a megoldások csak bizonyos feltételek mellett használhatók fel; még úgy is, ha a kitalálójuk vagy továbbadójuk nem figyelmeztet rájuk [K12].

Heurisztikus módszerek és megoldások közé tartoznak a gyakorlati és tapasztalati képletek, melyek arra hivatottak, hogy egyszerűbbé tegyék valamilyen fizikai tényező kiszámítását. Olyan esetekben alkalmazzák őket, amikor a logikusan is levezethető képletben olyan paraméterek szerepelnek, melyeket nem, vagy csak nagyon nehezen lehetne meghatározni. A legtöbb esetben olyan paraméterekre kell gondolni, amelyek csak akkor derülnek ki, ha már a folyamat, melyhez a képlet kellett, már véget ért. Ilyenkor ezeket a paramétereket becsléssel, állandók meghatározásával, vagy tapasztalat alapján határozzák meg. A másik nagy területe a heurisztikáknak a kísérletekkel és mérésekkel meghatározott mérnöki sablonok, ábrák és paraméterek. Ilyenkor különböző beállításokkal nagyszámú kísérletet végzünk egymás után, hogy egyes paramétereket vagy arányokat meg tudjunk határozni bizonyos hibatűrés mellett. Ilyenkor maga a folyamat a heurisztika és a megoldásként kapott információ fog szolgálni a további mérésekhez és felhasználáshoz. Példa ezekre: a vas-karbon diagram vagy egy motor teljesítmény-fordulatszám görbéje.

3.2.1. A metaheurisztikus algoritmusok

A metaheurisztika egy magasabb szintű heurisztikus eljárás, amely generál, keres vagy kiválaszt egy heurisztikus módszert az optimalizálási problémára azért, hogy effektíven jó megoldás készülhessen. A metaheurisztikákat csak az informatikai tudományokban használják effektíven. Ezek a módszerek általában összefognak és irányítanak más módszereket, hogy minél gyorsabban és minél kevesebb energia és számítási igény felhasználásával jó megoldást produkáljanak. Magukba olvaszthatnak szinte minden keresési metódust egy egyszerű lokális kereséstől kezdve tanuláson alapuló mesterséges intelligenciáig. A metaheurisztikák sokkal inkább irányítási stratégiák, mintsem különálló módszerek. [K13]

Ezeken felül fontos tulajdonságuk, hogy kevésbé érzékenyek hiányos vagy téves információra más algoritmusokkal szemben.

A metaheurisztikus módszerek általában egy vagy több célfüggvény alapján keresik a megfelelő optimumot. Ha a célfüggvényt és a paramétereit meg tudjuk határozni és be tudjuk

DOI: 10.14750/ME.2020.018

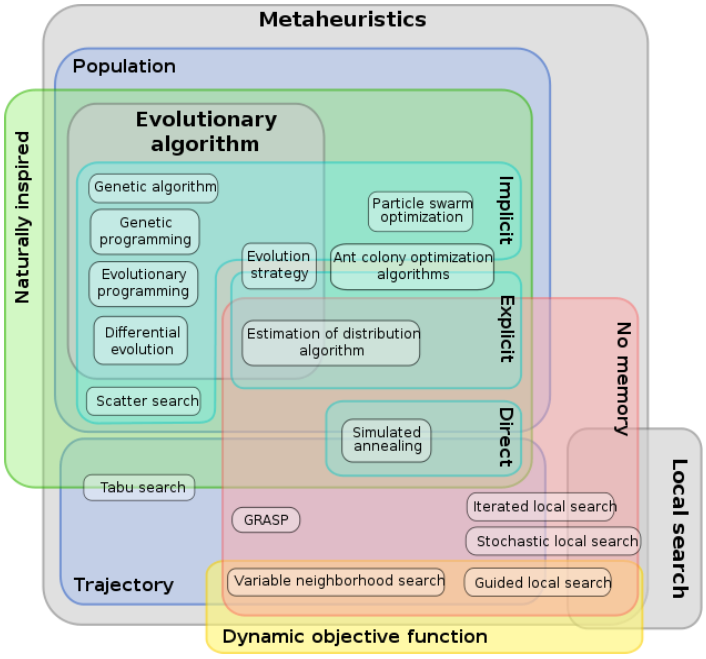
vinni programozható környezetbe, akkor bármilyen problémára képesek megoldást találni, racionális időn belül. A paramétereit és azok értékeit, értéktartományát és feltételeit beállításoktól függően képesek ezek a módszerek nagyon rugalmasan, szigorúan vagy robosztusan kezelni.

A heurisztikus algoritmusokat olyan esetekben alkalmazzuk, amikor nagy a keresési tér és sok a megoldásváltozat, sok a paraméter (akár több 100 vagy 1000 is lehet), a paraméterek és változók között komplex kapcsolat áll fenn, mint például a paraméterek egymást határozzák meg, vagy olyan feltételek is szerepet játszanak, amelyek megtiltják vagy lehetetlenné teszik egyszerűbb matematikai módszerek alkalmazását.

3.2.2. Metaheurisztikus algoritmusok fajtái

A metaheurisztikus algoritmusokat szoktuk típusonként/fajtánként csoportosítani, azonban egy algoritmus többe is beletartozhat, sőt a nagytöbbségük bele is tartozik. Ez a csoportosítás inkább csak az algoritmusok jellegére és egyes jegyek, működési stratégiák könnyebb megértésére létezik, mintsem határvonalként. A legtöbb algoritmus besorolható a következő kategóriákba [K13], amelyek kapcsolódását és fontosabb módszereit a 13. ábra is mutatja (Az ábra nem került lefordításra, mivel több elemnek nincs magyar neve):

- Természet inspirálta,
- Populáció alapú,
- Közvetlen,
- Közvetett,
- Memóriával ellátott,
- Memória nélküli,
- Dinamikusan változó,
- Lokális keresés,
- Globális keresés,
- Útvonal alapú,
- Evolúciós.



13. ábra: Metaheurisztikák fajtái [K13]

A 13. ábra jól mutatja, hogy a legtöbb algoritmus működésében nagy az átfedés. Egy példát kiemelve, a genetikus algoritmus természet inspirálta, populációt használó memóriával ellátott evolúció alapú keresési módszer.

DOI: 10.14750/ME.2020.018

3.2.3. Heurisztikus algoritmusok alkalmazása

A heurisztikus algoritmusok alkalmazásának előnyeit és hátrányait már leírtam, de egy konkrét példával is szeretném bizonyítani. Az első- és másodfokú függvényekről a legtöbben már általános iskolában tanultunk és a megoldóképletüket is ismerjük, amely egy-egy képlet. Említésképpen elmondták, hogy elméletben létezik a harmad- és negyedfokú függvényeknek is megoldása, azonban azok már annyira komplexek, hogy nem tanítják meg őket. Emellett azt is megtanultuk, hogy ahanyadik fokú egy függvény, annyi gyöke van és ezek konkréten meghatározhatók a megoldóképlettel.

Elsőfokú függvény és megoldóképlete:

$$ax + b = 0$$
$$x = -\frac{b}{a} \tag{3}$$

Másodfokú függvény és megoldóképlete:

$$ax^2 + bx + c = 0$$
$$x_{1,2} = -\frac{b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4}$$

Harmadfokú függvény és megoldóképlete [K14]:

$$ax^3 + bx^2 + cx + d = 0$$

$$x_1 = -\frac{b}{3a} - \frac{\sqrt[3]{-b^2+3ac}}{3a\cdot\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}} + \frac{\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}}{3a\cdot\sqrt[3]{2}}$$

$$x_2 = -\frac{b}{3a} + \frac{(1+i\sqrt{3})(-b^2+3ac)}{3a\cdot 2^{2/3}\cdot\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}} - \frac{(1-i\sqrt{3})\cdot\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}}{6a\cdot\sqrt[3]{2}} \tag{5}$$

$$x_3 = -\frac{b}{3a} + \frac{(1-i\sqrt{3})(-b^2+3ac)}{3a\cdot 2^{2/3}\cdot\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}} - \frac{(1+i\sqrt{3})\cdot\sqrt[3]{-2b^3+9abc-27a^2d+\sqrt{4(-b^2+3ac)^3+(-2b^3+9abc-27a^2d)^2}}}{6a\cdot\sqrt[3]{2}}$$

Ha van egy egyenletünk, ami beleillik valamelyikbe a fent említett három kategóriába, akkor nem a számítási idő fogja az idő nagy részét felemészteni, hanem az, hogy három különböző feladatként kell megírni a megoldóprogramot, amelyben külön figyelni és kezelni kell a függvény fokát és bár az első kettő megoldó képlet nagyon kicsit, a harmadfokú egyenlet megoldóképletének nagysága és komplexitása már nem olyan könnyen kezelhető és ez okot adhat a hibázásra.

Ezzel szemben egy heurisztikus optimalizáló eljárásnak nincs szüksége megoldóképletre, csak a normál függvényre és nem csak 4. fokig tudhat megoldani egyenleteteket, hanem bármilyen $n \in \mathrm{Z}$ fokig. Emellett nem kell különböző programot írni akkor se, ha esetleg nem egyváltozós, hanem több változós egyenlettel van dolgunk, mindössze ki kell bővítenünk a változók számát. Azonban nem csak előnyei vannak a heurisztikus eljárásoknak, két nagy hátránnyal is rendelkeznek. A számítás nem véges és becslő módon történik, tehát nem biztos, hogy tényleg az egzakt megoldást kapjuk és nem mondható meg előre, hogy a számítás meddig fog tartani.

DOI: 10.14750/ME.2020.018

3. táblázat: Megoldóképlet és Solveres heurisztika összehasonlítása

Megoldandó képletek	Fokok paraméterei						változók neve	Megoldás megoldóképlettel	Megoldás Solver heurisztikával	Solver heurisztika ellenörző eredmény
	n*X^8	n*X^6	n*X^3	n*X^2	n*X^1	c				
3X-6=0					3	-6	X	2,0000	2,0000	0
15X^2-2X-15=0				15	-2	-15	X1	-0,9356	1,0689	0,00040815
							X2	1,0689	-	-
8X^3+25X^2-150X-10=0			8	25	-150	-10	X1	3,0847	-6,1439	-0,06259613
							X2	-6,1438	-	-
							X3	-0,0659	-	-
4X^8-6X^6-1800=0	4	-6				-1800	X1	-	2,2432	0,039575545

A 3.táblázatban láthatjuk a leírtakat. Az Excel Solver bővítményébe beépített evolutív heurisztikus algoritmus tökéletes megoldást csak a legelső esetben talált, azonban a többi eredménynél az eltérés az ezrelék alatt található és képes volt az utolsó nyolcad fokú egyenletnek is megoldást találni.

3.2.4. Célfüggvények és paraméterek

A metaheurisztikus algoritmusok szinte mindegyike célfüggvények alapján számolja ki, hogy egy megoldás, amit egy lépésben megalkotott, mennyire jó az eddig már elkészített megoldásokhoz képest. A célfüggvények olyan függvények vagy képletek, melyeket egy adott feladatra készítünk a meglévő információk és paraméterek alapján. A célfüggvényeket úgy kell megalkotni, hogy a lehető legjobban képezzék le a valóságot vagy legjobban idomuljanak saját elképzelésünkhöz. A legtöbb célfüggvény, amelyet megalkotunk, gyakorlati és tapasztalati képletek megalkotásának módszerével készül, azaz ha fel is használnak elfogadott matematikai vagy fizikai összefüggéseket, nem lehet őket teljesen visszavezetni rájuk. A megoldásukból kapott érték is csak egy arányszámnak tekinthető, amelyet a többi megoldás értékével tudunk összehasonlítani, amelyből kiderül a megoldások jósága és rangsora.
A célfüggvényeket általában három keresési technikára alkalmazzuk:

- minimumkeresés,
- maximumkeresés,
- előre meghatározott érték vagy értéktartomány keresése.

Jelölni a következőképpen szoktuk:

$$f(A) \rightarrow \begin{matrix} min. \\ max. \\ c, \end{matrix} \qquad ahol\ c = 10\ vagy\ c \in [0,1] \tag{6}$$

Tapasztalataim szerint a legtöbb módszert a minimumkeresésre készítették el. Ilyenkor a keresési térben azt a pontot (paraméterek értékeit) szeretnénk megtalálni, ahol a célfüggvény értéke a legkisebb. Egy minimumkeresésre megalkotott módszert a legtöbb esetben nagyon könnyű átírni maximumkeresésre és ez fordítva is igaz. Elég csak annyit tenni, hogy a megoldások célfüggvény értékeiből kivonunk egy mindegyiknél nagyobb számot, de sokkal egyszerűbb az előjeleiket megváltoztatni, úgy hogy megszorozzuk mindegyiket „-1” értékkel. A negatív számokból így pozitív, a pozitívből pedig negatív lesz. Így például növekvő

DOI: 10.14750/ME.2020.018

sorrendbe rendezett sorozatokat teljesen meg tudunk fordítani és a legkedvezőbb értékből a legkedvezőtlenebb értéket tudunk alkotni:

-6	-4	-2	1	3	5	8	10

-10	-8	-5	-3	-1	2	4	6

Azoknál a kereséseknél, amelyeknél bizonyos értéket vagy értéktartományt keresünk, egy megoldás „jóságát” a célfüggvény érték és a meghatározott érték vagy közelebbi határérték közötti különbség adja meg. Minél kisebb ez az érték, annál jobb egy megoldás. Innentől kezdve ezt a feladattípust is át lehet írni minimumkeresési feladattá, ahol a két érték különbségének minimumát keressük abszolút értékben.

A célfüggvényeket felépítő paramétereknek szintén három fajtáját különböztetjük meg:

- valós számokat használó (float, dec),
- egész számokat használó (int),
- számsorozatokat használó (seq).

A legtöbb célfüggvény általában egy fajtát használ, azonban bonyolultabb célfüggvények, amelyek teljes rendszerek működését képesek kiértékelni, felhasználhatnak többet is. Ezek a paraméterfajták sokkal inkább a programozás tekintetében jelentenek különbséget, mint elméleti síkon.

A legtöbb metaheurisztikus algoritmusról elmondható, hogy eredetileg valós számokkal történő munkára készítették őket, amelyben a legkevesebb a megkötés és végtelen sok megoldás létezik. Az általam bemutatásra kerülő 4 algoritmus is eredetileg a paramétereit a valós számok halmazán generálta le, azonban a genetikus és a harmony search algoritmusnak létezik olyan változata, amely bár később került bemutatásra, de képes sorozatokkal és egész számokkal is dolgozni.

Azokat a paramétereket melyek valós számokat tartalmaznak, decimális paramétereknek is szoktuk nevezni. Ebben az esetben a paraméter bármilyen valós értéket felvehet, amelyet az adott probléma vagy a keresési tér megenged. Mivel bármely két szám között végtelen mennyiségű szám található, ezért elméletileg az algoritmus által előállítható megoldások száma is végtelen. Azonban a számítógép feldolgozó képessége véges, ezért a legtöbb esetben speciális programozás nélkül egy bizonyos pontosságnál nem vagyunk képesek jobb értékeket előállítani. Emellett fontos megjegyezni, hogy a valós szám típusú paramétereket a legkönnyebb és leggyorsabb programozni. A logisztikában olyan feladatoknál jelentkezik ez a fajta paraméter, amelyeknél egy vagy több objektum helyét kell meghatározni (centrumkeresési feladat), amelyet már bemutattam az előzőekben. Emellett függvényekben a súlyok pontos meghatározása is ide tartozik (súlyozási feladat).

Egész számokat felhasználó paramétereket az angolban integer, azaz egészértékű paraméternek is nevezik. A metaheurisztikus algoritmusok, amelyeket valós számú paraméterekre találtak ki, bizonyos megkötések között képesek dolgozni egész értékekkel is, elsősorban kerekítések felhasználásával. Azonban nagyon fontos, hogy a kerekítéseket jó helyen és szabályosan végezzük el, és ha az általános kerekítési szabályok nem működnek jól az algoritmussal, akkor saját magunknak kell újat alkotni.

DOI: 10.14750/ME.2020.018

Példa a kerekítési szabályok rossz használatára

Feladat: Folytonos eloszlással véletlenszerűen hozzunk létre 1 és 10 között egész számokat, úgy hogy a random generátor 0 és 1 között generál számokat.

Rossz megoldás:

$$C_{int} = round\ int(rand(0 \dots 1) * 9) + 1 \quad (7)$$

Ebben az esetben az egyenletes eloszlás szabálya nem érvényesül az első és utolsó elemre, ugyanis azoknak csak feleakkora esélyük van a megjelenésre, amit a következő táblázat is mutat. Ez azért van, mert nem 10 számértéknek generáltuk le az esélyét a szorzóval, hanem csak 9-nek. Ha „0.001" érték jönne ki a véletlen szám generálása során, akkor a képlet szerint „0.001*9=0.009" értéket 0-ra kerekítünk és hozzá adunk egyet. Azonban a kerekítési szabály kimondja, hogy 0-ra kerekítés az -0,4999…0,4999 között történik. Mivel nem jöhet ki -0,4999 és 0 között érték ezért annak az esélyét elvesztjük. Ugyanez vonatkozik a felső értékre is.

4. táblázat: Rossz random generálás eredménye

Számértékek	1	2	3	4	5	6	7	8	9	10	Összesen
Darabszám	54	92	118	105	119	125	98	120	111	58	1000

Jó megoldás:

$$C_{int} = round\ int(rand(0 \dots 1) * 10 - 0{,}5) + 1 \quad (8)$$

Egyszerűsítve:

$$C_{int} = round\ int(rand(0 \dots 1) * 10 + 0{,}5) \quad (9)$$

Ebben az esetben már 10 értéket osztunk szét és úgy garantáljuk a fél értéket, hogy kivonunk a kerekítés előtt a generált számokból felet. Ha a random generátor 0.001-es értéket generál, akkor „0.001*10-0,5=-0,49" amelyet a gép szintén 0-ra kerekít. Ez a módszer a felső határnál is működik. Ezt az alábbi táblázat is alátámasztja. Viszont még ez sem tökéletes megoldás, ugyanis, ha a két határszámot adja ki a randomgenerátor, akkor a kerekítési szabályok szerint -1 és 11 fog kijönni, amelyek nem várt eredmények. Ezeket a képlet finomításával vagy a programban lévő ellenőrzéssel ki lehet küszöbölni.

5. táblázat: Random generálás helyes módon

Számértékek	1	2	3	4	5	6	7	8	9	10	Összesen
Darabszám	103	101	92	91	114	101	92	94	98	114	1000

Az egész számokkal dolgozó algoritmusokat egy kicsit nehezebb programozni és nagyobb a számítási igényük, mint a valós számokat használóknak, amely a plusz feltételek, kikötések és ellenőrzések miatt alakul ki. Elméletileg véges számú megoldás is létezhet egy

DOI: 10.14750/ME.2020.018

problémára, ha az összes paraméter a keresési térben legalább egy alsó és felső korláttal rendelkezik. Azonban a probléma bonyolultságától, paraméterek számától és a korlátok nagyságától függően a legtöbb problémára olyan nagyszámú megoldás létezik, hogy a teljes leszámolás módszerével reális időn belül nem tudná semmilyen eszköz megvizsgálni.

A logisztika témakörében egész értékű paraméterezéssel elsősorban a kapacitási problémáknál (raktár/tárhely méretezés), hozzárendelési és csoportosítási problémáknál, ezen felül pedig a Make of Buy jellegű döntéseknél találkozhatunk [S1].

Számsorozatokat használó paramétereket szokás permutációs vagy szekvenciális paramétereknek is nevezni. Ebben az esetben nem egyszerű egész értékekkel, hanem előre meghatározott elemből építkezik az adott paraméter. Ez azt jelenti, hogy egy paraméter csak egy adott halmazból vehet fel értéket és csak annyit, amennyit a halmaz tartalmaz. Mivel ebben a paramétertípusban rengeteg a megkötés, nagyon ritkán látni olyan metaheurisztikus algoritmusokat, amelyet a decimális vagy egészértékű leírás mellett rendelkeznek ilyen típussal is. Vannak olyan algoritmusok, melyeket kimondottan erre a feladatra készítenek, mint a tabu keresés vagy a memetikus algoritmus, amely az evolúciós algoritmusok családjába tartozik, de a genetikus algoritmusnak is készült olyan változata, mely megoldja a permutációs problémákat [S5].

A logisztikában az egyik legismertebb probléma az utazóügynök probléma (járattervezés), amelynek a parametrizálása permutációkban történik. Ilyenkor egy olyan útvonalat kell keresnünk előre kijelölt helyek között, amelynek a kiinduló és kezdőpontja azonos, minden pontot érint és a lehető legrövidebb az összesített távolsága. Ez a leírás csak az alap feladatnak minősül. Ez a feladat még kiegészül időkapukkal, amire oda kell érnie az adott járműnek, amihez ütemtervet kell készíteni. Figyelni kell saját és mások kapacitását és előre kell tervezni, amennyire csak tudunk és több jármű együttes menedzselését is meg kell oldanunk [S6].

A permutációkkal vagy sorozatokkal van egy hatalmas baj, amely nagyon sok metaheurisztikus algoritmust tesz használhatatlanná magával szemben vagy teljesen más megközelítést követel. Ez pedig a permutációk közötti távolságmérés. Amikor két megoldást szeretnénk összehasonlítani, amelyek permutációs paramétereket is tartalmaznak, a célfüggvény értéküket könnyedén össze tudjuk vetni, azonban a paramétereikkel már más a helyzet. Két ugyanabból az elemekből felépülő számsorozatban csak elvontan lehet távolságot keresni. Nagyon sok metaheurisztikus algoritmus elsősorban populáció alapú, ezek úgy generálnak új egyedeket, azaz megoldásokat, hogy figyelembe veszik a távolságokat más megoldások paraméterei között a célfüggvény-értékékek függvényében.

Az egyik legjobban elterjed permutációk közötti távolságmérési módszer a cseretávolságok meghatározása. Ez azt jelenti, hogy hány cserével tudjuk előállítani az egyik sorrendből a másikat. Azonban még így se egyértelmű, hogy mi számít egy cserének. Egyes módszerek szerint bármely két érték felcserélhető megkötés nélkül egy számsorozatban, míg mások szerint csak az egymás mellett lévők cserélhetők fel. Erre a kérdésre egy teljes tudományterület épült, amelyben személy szerint elmerültem kicsit és a következő rész ezt fogja bemutatni.

DOI: 10.14750/ME.2020.018

4. TÁVOLSÁGMÉRÉS PERMUTÁCIÓK KÖZÖTT KERESÉSI TÉRBEN

A logisztikában, mint minden műszaki területen nagyon fontos szerepe van a matematikának. Ez adja meg az alapot arra, hogy megvizsgálhassunk folyamatokat, adatokat gyűjtünk, elemezzük őket, különböző indikátorokat és határokat állítsunk fel, majd megoldásváltozatokat készítsünk és döntsünk. Akár tudjuk bizonyítani egy módszer vagy megoldás működését, akár nem, matematikai módszerekkel dolgozunk. Erre láthatunk több példát is a 3. fejezetben és a soron következőkben egyaránt. Egy másik nagyon fontos terület, amely elképzelhetetlen matematika nélkül, az a számítástechnika, ugyanis bármit készítünk vagy használunk számítógépen, az csak matematikai műveletekkel képes dolgozni. Mindezekre szükségünk van, ha megalapozott döntéseket szeretnénk hozni. Azonban nem csak jó döntéseket kell hoznunk, hanem gyors döntéseket is. Ezért szükségünk van a döntéstámogató módszerekre és eszközökre. Az iparban a döntések nagy részénél analitikus és vizualizáló szoftverek használnak, amelyeket nagyon jól megírt kombinált algoritmusok vezérelnek. Ezekben megtalálhatók egyszerű matematikai, LP és heurisztikus algoritmusok is. Minden ilyen szoftver kisebb módszerek hadával és megfelelő irányítással képes effektíven dolgozni. Ebben a fejezetben egy általam alkotott módszer kerül bemutatásra, amely akár egy ilyen szoftver része is lehet.

A továbbiakban a módszer leírását és bizonyítását fogjuk látni, amelyet egy logisztikai feladat inspirált. A módszer egy olyan járattervezési feladat heurisztikus megoldásának az elkészítésében segít, amelynél a járat hossza és a csomópontok sem előre meghatározottak. Erről a feladatról szól a 7.2-es fejezet [S5].

A kutatásaimhoz használt algoritmusok leírásának többségében megtalálható a távolság szó, amelyet ahhoz használnak ezek az algoritmusok, hogy meghatározzák az egyes részmegoldások (egyedek) egymáshoz vagy az optimumponthoz vett távolságát. Ezzel mindaddig nincs semmilyen baj, amíg egy megoldásban csak valós számok szerepelnek, mindegy hogy hány paraméterre, ugyanis súlyozással és többdimenziós távolságméréssel képesek vagyunk használható távolságot kapni. A probléma a permutációk és szekvenciák egymástól való távolságában van, mivel itt egy érték nem egy bizonyos tulajdonságot meghatározó szabad változó, hanem egy előre meghatározott halmaz elemeként tekinthető és bár számokkal dolgozunk a legtöbbször, ezt azért tesszük, hogy gépeink könnyebben feldolgozzák, ugyanis bármilyen más szimbólummal is tudnánk őket jelölni. Ez a probléma a logisztikában elsősorban a járattervezésnél jön elő, ahol útvonalakat kell alkotnunk. Két ilyen problémát is ismertetek a későbbiekben az egyedi modellek tesztelésekor.

A következőkben megvizsgáljuk az egyes megoldási változatokat reprezentáló permutációk közötti távolságokat definiáló metrikákat, hiszen azok befolyásolhatják az algoritmus konvergenciáját. A heurisztikus és metaheurisztikus optimumkereső algoritmusok esetében különböző állapotterekben kell a lehetséges megoldásváltozatok közül a legjobbat megkeresni. A raj intelligencia típusú algoritmusok esetében (hangya kolónia algoritmus, méh kolónia algoritmus, szentjános bogár algoritmus) az állapottér függvényében különböző módszerek használhatók az egyes megoldások közötti távolságok mérésére. Amennyiben a keresési térben az egyes megoldási változatok binárisan kódoltak, akkor a Hamming távolság

DOI: 10.14750/ME.2020.018

egy alkalmas metrika. Valós vektorokkal leírt megoldási változatok esetében Euklidészi vagy Chebisev távolság használható távolságmérésre. Jelen esetben diszkrét számokat tartalmazó vektorok írják le az egyes megoldási változatokat.

Ha a Firefly algoritmus vagy a Black hole algoritmus teljes leírását követnénk, akkor tényleges távolságokkal kellene dolgozunk, amelyek az objektumaink között vannak és a optimalizálási folyamat legvégén az összes objektumunk egy rajba rendeződne. Azonban mi nem ezt akarjuk, ugyanis a mi objektumaink fix helyen állnak és a köztük lévő útvonalakat kell változtatnunk. Ilyenkor permutációkat használunk arra, hogy egy lehetséges járatverziót felírjunk, amelyek az objektumaink között közlekednek.

Ahhoz, hogy új útvonalakat, azaz megoldásokat kapjunk egy már adott útvonal felhasználásával, képesek vagyunk a permutációkban cserélni, hozzáadni vagy elvenni objektumot. Ha ezeknek a műveleteknek a minimális számát meghatározzuk, akkor megkapjuk a távolságát a két permutációnak. Ezeknek az útvonalaknak meg tudjuk határozni a célfüggvény értékeit, azonban a köztük lévő távolságot már nehéz értelmezni.

4.1. Permutációk közti cseretávolság számításának lehetőségei

A permutációk közötti cseretávolság azt jelenti, hogy hány cserével tudjuk az egyik permutációból előállítani a másikat. Ez két dolgot feltételez:

- A két permutáció ugyanannyi elemből áll.
- A két permutáció ugyanazokból az elemekből áll.

Az ilyen esetekre rengeteg megoldás van a szakirodalomban, a következőkben a legelterjedtebb módszerek kerülnek bemutatásra.

Az egyik legegyszerűbb módszer a Hamming távolság [K23], mely a nem azonos elemeket tartalmazó pozíciók száma két permutációban:

$$d_{Ham}(s_1, s_2) = \sum_{i=1}^{n} x_i \quad ahol \;\; x_i = \begin{cases} 0 \; ha \; s_1(i) = s_2(i) \\ 1 \; más \; esetekben \end{cases} \tag{10}$$

A Hamming távolság normalizálható, ebben az esetben a két permutáció valós Hamming távolságát osztani kell a maximális távolsággal:

$$d^*_{Ham}(s_1, s_2) = \frac{d_{Ham}(s_1, s_2)}{n} \tag{11}$$

A különbség távolság a két permutáció egyes elempárjai közötti távolságok különbségeinek összege [K23]:

$$d_{különbség}(s_1, s_2) = \sum_{z=1}^{n} |s_1(z) - s_2(z)| \tag{12}$$

DOI: 10.14750/ME.2020.018

A különbség távolság normalizált értéke az egyes permutáció vektorok elemszámának függvényében az

$$d^*_{különbség}(s_1, s_2) = \frac{2 \cdot d_{különbség}(s_1, s_2)}{n^2} \; és \; d^*_{különbség}(s_1, s_2) = \\ = \frac{2 \cdot d_{különbség}(s_1, s_2)}{n^2 - 1} \tag{13}$$

összefüggésekkel számítható.

Amennyiben az egyes permutációk közötti távolságokat nagy távolságok esetében fokozottan kell figyelembe venni, akkor a négyzetes távolságok alkalmazása lehet célszerű:

$$d_{négyzeteskülönbség}(s_1, s_2) = \sum_{z=1}^{n} \big(s_1(z) - s_2(z)\big)^2 \tag{14}$$

melynek normalizált értéke a következő összefüggéssel számítható:

$$d^*_{négyzeteskülönbség}(s_1, s_2) = \frac{3 \cdot d_{különbség}(s_1, s_2)}{n^3 - n} \tag{15}$$

Az utazó ügynök típusú feladatok esetében igen gyakran jelentkezik igényként az, hogy bizonyos pozíciókat vagy felkeresendő objektumokat prioritással vegyünk figyelembe. Amennyiben a felkeresési sorrendek esetében szükséges prioritásokat alkalmazni, akkor a súlyozott különbség metrika jó alkalmazható:

$$d_{súlyozott\ különbség}(s_1, s_2) = \sum_{z=1}^{n} |s_1(z) - s_2(z)| \cdot w_z \quad ahol \quad \sum_{z=1}^{n} w_z = 1 \tag{16}$$

Ezen módszert lehet négyzetes különbség esetében is alkalmazni:

$$d_{súlyozottnégyzetes}(s_1, s_2) = \sum_{z=1}^{n} \big(s_1(z) - s_2(z)\big)^2 \cdot w_z \quad ahol \quad \sum_{z=1}^{n} w_z = 1 \tag{17}$$

A másik metrika esetében definiálni kell egy határeltérést, és ezt követően összegezni kell a határeltérésnél nagyobb eltérések számát:

$$d_{határeltérés}(s_1, s_2) = \sum_{i=1}^{n} x_i \quad ahol \quad x_i \\ = \begin{cases} 1 \; if \; |s_1(i) - s_2(i)| > dev_{határ} \\ 0 \; egyébként \end{cases} \tag{18}$$

DOI: 10.14750/ME.2020.018

Az utazóügynök probléma (Traveling Salesman Problem, röviden: TSP) esetében a leghosszabb közös string, mint távolság igen jól használható, hiszen segítségével azt mérhetjük, hogy milyen hosszú azon részkörútnak a hossza, mely megegyezik a két permutáció esetében. Mivel permutációs problémák közül a legfontosabb a TSP megoldása, ezért kidolgozásra került általam egy olyan új metrika, mely kifejezetten ilyen típusú feladatok esetében alkalmas a permutációk közötti távolságok leírására [S6].

A legegyszerűbben felhasználható és a legkisebb számítási igénnyel rendelkező metrika a Hamming távolság. A módszer lényege, hogy megvizsgálja két ugyanolyan hosszú permutációban, hogy ugyanazok az elemek állnak-e ugyanabban a pozícióban. Ha nem ugyanaz a két elem, akkor a távolság értékét a módszer növeli eggyel. Ezzel megkapunk egy olyan értéket, amely csereszámból biztosan át lehet alakítani az egyik permutációt a másikká. A módszer nagyon gyors, viszont nagyon pontatlan is, ugyanis akár a szükséges csereérték helyett akár kétszer nagyobb értéket is tud mutatni. Akkor alkalmazható, ha egy maximum értéket szeretnénk kapni.

Az alap utazóügynök problémánál azonban mostanában jóval változatosabb és összetettebb problémákkal találkozunk. Járattervező szoftverek egyszerre több járművet és nagyszámú változó ponthalmazt is figyelembe vesznek, megesik, hogy két útvonalmegoldás nem ugyanolyan hosszú és többször is tartalmazza ugyanazokat az elemeket (rövidtávú szállítások során a jármű általában a kiinduló helyre tér vissza és esetleg egy új szállításba kezd). Ezeket az eseteket nem hívhatjuk többé permutációnak, ezek már szekvenciák vagy számsorozatok. Sajnos a fent említett módszerekből egyik se használható ezekre a feladatokra.

Az irodalomban a legközelebb ehhez az esetekhez a nyelvhelyesség figyelő algoritmusok állnak, ahol azt vizsgálják, hogy egy szóban hol lehet elírás. Ennek az egyik legismertebb módszere a Levenshtein távolság, ami vizsgálja, hogy „a” (beírt) és „b” (javasolt) szó között mekkora eltérés mutatkozik, ha megváltoztatunk, hozzá adunk vagy kivonunk belőle egy betűt. A Levenshtein távolság matematikailag a következőképpen fogalmazható meg [K15]:

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & egyébként \end{cases} \quad (19)$$

ahol $i = |a|$ és $j = |b|$, azaz a két szó hossza, és ahol $1_{(a_i \neq b_j)}$ 0 értéket vesz fel, ha $a_i = b_j$, máskülönben az értéke 1. Emellett $lev_{a,b}(i,j)$ a távolságértéket adja meg „a” szó *i.* karaktere és „b” szó *j.* karaktere között.

Ez a módszer nagyszerűen használható nyelvhelyesség ellenőrzésre, de sajnos nem képes lekezelni szekvenciális távolságmérést logisztikai feladatokban. A módszernek ugyanis szüksége van egy speciális táblázatra, mely szerint egyes gombok és karakterek hogyan viszonyulnak egymáshoz, és csak egy karakternyi hosszúságkülönbséget enged meg.

DOI: 10.14750/ME.2020.018

4.2. Elempárkereső (SPS, sequencial pair search) távolságmérés

A sikertelen módszerkeresés után létrehoztam egy egyszerű és gyors távolságmérő módszert, mellyel különböző hosszúságú és különböző elemekből felépülő szekvenciák távolságát is meg lehet becsülni. Erre azért volt szükségem, mivel a legtöbb mostani TSP problémánál előfordul, hogy egy objektum hozzáadásra vagy elvonásra kerül, akár a jármű vagy optimalizálási folyamat közben. A módszerrel meg lehet becsülni két permutáció vagy szekvencia csereértékét, azaz, hány cserét kell végrehajtani ahhoz, hogy az egyik számsorozatból megkapjuk a másik számsorozatot vagy egy olyat, amely legközelebb áll hozzá. A módszert elempárkereső módszernek neveztem el és működése során a két számsorozatban végignézi az összes elempárt (éleket) és összeszámolja mennyit talált. Ezek az egyezések számai, amelyet kivon a legnagyobb elemszámú szekvencia elemszámából, majd az egészből kivonunk egyet. Az utóbbi azért szükséges mivel eggyel kevesebb párt tudunk készíteni, mint amennyi elemszámunk van:

$$d_{SPS}^{1}(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - 1 \tag{20}$$

ahol az „E” az éleket, azaz az elempárokat jelenti, és a következőképpen csoportosítjuk:

$$E_a = \{(a_1,a_2),(a_2,a_3),\dots,(a_{n-1},a_n)\} \tag{21}$$

$$E_b = \{(b_1,b_2),(b_2,b_3),\dots,(b_{m-1},b_m)\} \tag{22}$$

$$E_b' = \{(b_2,b_1),(b_3,b_2),\dots,(b_m,b_{m-1})\} \tag{23}$$

Van még egy kiegészítő szabály, amelyet nehéz megfogalmazni matematikailag, és amely kijelenti, hogy ha egy számpárból több ugyanolyan is létezik akármelyik sorozatban, akkor csak egyszer számoljuk és a többi számpár vizsgálatnál már nem számoljuk bele.

Még mielőtt a matematikai bizonyításra kitérek, meg kell említenem, hogy a módszert egyirányú és kétirányú változatra is elkészítettem. Az egyirányú változat képlete a bekezdés felett található és akkor alkalmazzuk, ha azt akarjuk, hogy a párok orientációja megmaradjon. Azaz a[a_1,a_2,…,a_n] számsorozatból a másik számsorozatban a_1-a_2 párokat keressünk (irányított éleket) csak és nem számítjuk az a_2-a_1 számpárokat. Ez a logisztikában szintén fontos és megkülönböztetendő, mivel a rövidtávú szállítások, amilyenek a városon, üzem területén és üzememen belüli szállítások, rengeteg egyirányú útszakasszal, kereszteződéssel és speciális állomásokkal vannak ellátva és ezek orientációját figyelembe kell venni tervezéskor. Ilyenkor az objektumok közötti távolságot leíró mátrix sem szimmetrikus a főátlóra. A másik eset a nagy távolságú szállításoknál jön elő, ahol elhanyagolható mértékűek az orientációval való paraméterek és minden kétirányúnak tekinthető és szimmetrikus a távolságmátrix is. Ilyenkor a párkeresési algoritmus nem csak az a_1-a_2, hanem az a_2-a_1 számpárokat is keresi irányítatlan élként. Ennek a képlete a következő:

$$d_{SPS}^{2}(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - |E_a \cap E_b'| - 1 \tag{24}$$

DOI: 10.14750/ME.2020.018

Állítás: az előzőkben megfogalmazott d^1 és d^2 teljesíti a távolságaxiómákat.

A módszer bizonyítása

Ahhoz, hogy bebizonyítsuk, hogy egy permutációs vagy szekvenciális módszer működőképes és használható négy feltételnek kell megfelelnie:

1. Egyezőség
2. Szimmetrikusság
3. Távolság nem-negativitás
4. Háromszög-egyenlőtlenség

Az egyezőség azt jelenti, hogy két megegyező számsorozat távolságának nullának kell lennie:

$$0 = d(a,b) \leftrightarrow a = b \tag{25}$$

A bizonyítása ennek meglehetősen egyszerű mindkét esetben. Ha van két teljesem megegyező a = b számsorozatunk, akkor a hosszuk $|a| = |b| = n = m$ darabból állnak. Két egyenlő szám közül a legnagyobb érték az $n = m$, az egyszerűség kedvéért válasszuk ki közülük n-et. n darabból álló szekvenciából $n - 1$ darab elempárt lehet készíteni, ennyi éllel lehet őket sorban összekötni. A kétirányú változatnál az E_b' azokat az elemeket tartalmazza, amelyek a párok fordítottjaira igazak. Jelen esetünkben, mivel mindkét számsorozat ugyanazokkal az elemekkel van ellátva, ezért nem lehet egyetlen fordított elempár benne, tehát $E_b' = 0$.

Képletesen leírva:

$$\begin{gathered} 0 = d_{SPS}^1(a,b) = \text{n-(n-1)-1=0} \\ 0 = d_{SPS}^2(a,b) = \text{n-(n-1)-0-1=0} \end{gathered} \tag{26}$$

Akkor beszélhetünk szimmetrikusságról, ha ugyanazt a távolságértéket kapjuk, ha az egyiket nézzük a másikhoz és fordítva:

$$d(a,b) = d(b,a) \tag{27}$$

A második feltétel is könnyen belátható, ugyanis ha felcseréljük a tagokat, akkor sem lesz olyan paraméter, amely az értékeket megváltoztatná, mivel mindegyik művelet asszociatív. A legnagyobb elemszám kiválasztása nem sorrendfüggő, ezért mindig a legnagyobbat fogja kiválasztani, az elempárok metszete pedig szintén asszociatív művelet, azaz:

$$\begin{gathered} d_{SPS}^1(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - 1 = \\ = d_{SPS}^1(b,a) = \max(|b|,|a|) - |E_b \cap E_a| - 1 \end{gathered} \tag{28}$$

DOI: 10.14750/ME.2020.018

Ugyanezek igazak a kétirányúsított változatra is, azzal a felismeréssel, hogy ha vesszük a két sorozat és összehasonlítjuk, akkor, ha az élek halmazának minden élében a csúcspontokat felcseréljük, akkor nem változik meg az eltérések száma:

$$|E_a \cap E'_b| = |E_b \cap E'_a| \tag{29}$$

Ez alapján igaz a következő összefüggés:

$$\begin{gathered} d^2_{SPS}(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - |E_a \cap E'_b| - 1 = \\ = \max(|b|,|a|) - |E_b \cap E_a| - |E_b \cap E'_a| - 1 = \\ = d^2_{SPS}(b,a) = \max(|b|,|a|) - |E_b \cap E_a| - |E'_b \cap E_a| - 1 \end{gathered} \tag{30}$$

A harmadik feltétel megszabja, hogy a távolság értéke nem lehet negatív szám. Ezt könnyű belátni, ugyanis a legnagyobb elemszám, mindig nagyobb vagy egyenlő, mint a belőle készített párok plusz egy:

$$\max(|a|,|b|) \geq |E_a \cap E_b| + 1 \tag{31}$$

Ebből következik mindkét távolságnál, hogy teljesül a feltétel:

$$\begin{gathered} d^1_{SPS}(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - 1 \geq 0 \\ d^2_{SPS}(a,b) = \max(|a|,|b|) - |E_a \cap E_b| - |E_a \cap E'_b| - 1 \geq 0 \end{gathered} \tag{32}$$

A negyedik a feltétel a háromszög egyenlőtlenség, amely kimondja, hogy három távolság közül, ha két távolságot összeadunk, akkor annak nagyobb vagy egyenlőnek kell lennie, mint a harmadik távolság:

$$d(a,b) \leq d(a,c) + d(b,c) \tag{33}$$

A már bizonyított nem-negativitás (32) feltételből adódik a következő összefüggés:

$$\begin{gathered} \left|E_{a/b} \cap E_{c/b}\right| + \left|E_{c/a} \cap E_{b/a}\right| + |E_a \cap E_b \cap E_c| \leq |c| - 1 \\ \leq min(|c|,|b|) \end{gathered} \tag{34}$$

Legyen:

$$f = max(|a|,|c|) \tag{35}$$

A (34)-es összefüggés egyszerűsítve $|E_a \cap E_b \cap E_c|$-el

$$\begin{gathered} max(|a|,|b|) - \left|E_{a/c} \cap E_{b/c}\right| \overset{?}{\leq} max(|a|,|c|) - \left|E_{a/b} \cap E_{c/b}\right| \\ + max(|b|,|c|) - \\ - \left|E_{b/a} \cap E_{c/a}\right| - |E_a \cap E_b \cap E_c| - 1 \end{gathered} \tag{36}$$

DOI: 10.14750/ME.2020.018

$$max(|a|,|b|) - \left|E_{a/c} \cap E_{b/c}\right| \leq f - \left|E_{a/c} \cap E_{b/c}\right| \overset{?}{\leq} f - \left|E_{a/b} \cap E_{c/b}\right| + max(|b|,|c|) - \left|E_{b/a} \cap E_{c/a}\right| - |E_a \cap E_b \cap E_c| - 1 \quad (37)$$

$$f - \left|E_{a/c} \cap E_{b/c}\right| \overset{?}{\leq} f - \left|E_{a/b} \cap E_{c/b}\right| + max(|b|,|c|) - \left|E_{b/a} \cap E_{c/a}\right| - |E_a \cap E_b \cap E_c| - 1 \quad (38)$$

Innen

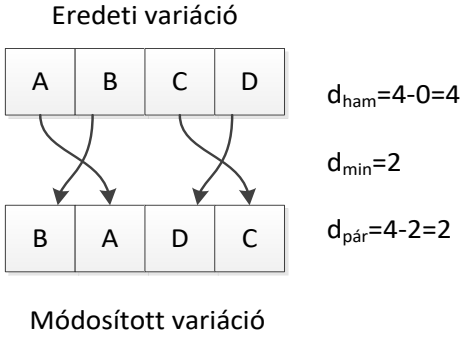
$$-\left|E_{a/c} \cap E_{b/c}\right| \overset{?}{\leq} max(|b|,|c|) - \left|E_{a/b} \cap E_{c/b}\right| - -\left|E_{b/a} \cap E_{c/a}\right| - |E_a \cap E_b \cap E_c| - 1 \quad (39)$$

A (34)-es összefüggés miatt a kifejezés jobb oldala nem negatív, a bal oldala viszont nem pozitív. Ebből következik, hogy az állítás igaz (az egyenlőtlenségről a ? elhagyható). Ez a levezetés a $d^1_{SPS}(a,b)$ távolságra lett alkalmazva. Ugyanez a gondolatmenet alkalmazható a $d^2_{SPS}(a,b)$-es távolságra is. Ezen fenti bizonyításért köszönet illeti Dr. Gubán Miklóst.

A fenti bizonyításokból látszik, hogy a módszer képes kielégíteni minden feltételt és alkalmazható távolságmérésre.

A módszer működésének és hatékonyságának tesztelésére nézzünk néhány példát:

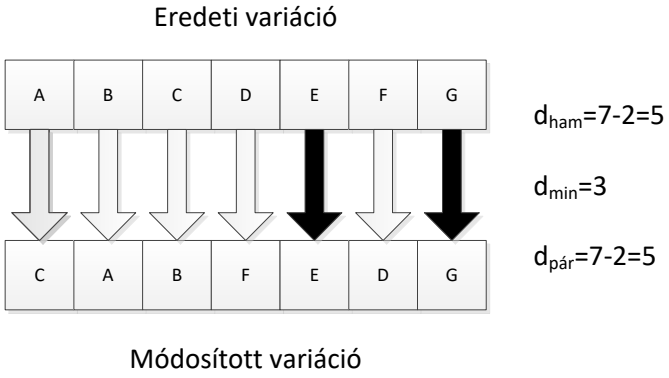
Az első permutáció párban 4-4 elem található és ugyanabból az elemekből állnak. Ezek között egyszerűen lehet cseretávolságot mérni. A cseretávolság mérésénél az a jobb, ha az eredmény minél kisebb értéket ad. Az alábbi ábrában látható, hogy a Hamming távolság szerint 4 cserével biztosan megoldható a feladat, így ez egy felső korlátnak tudható be, viszont a minimális csereszám meghatározása során, amelyre számos biztos megoldás létezik, 2 cserével már megkaphatjuk az eredeti megoldást. A saját párkereső megoldásunk is megoldásnak kettőt ad, tehát ebben az esetben az ideális megoldást kapjuk meg tőle. A nyilak az ábrában a cseréket jelentik, amiket a minimális csereszám eléréséhez kell megtennünk.



14. ábra: 4 elemű példafeladat ugyanazon elemekkel

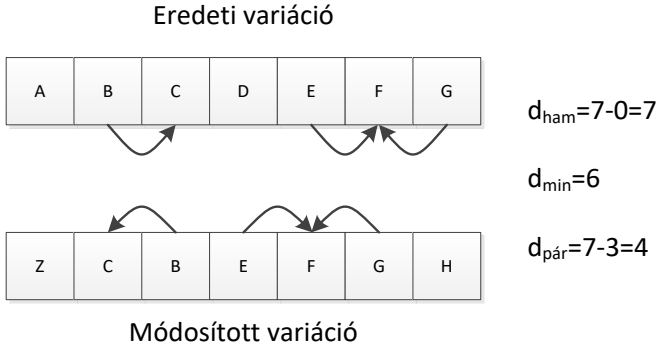
DOI: 10.14750/ME.2020.018

A következő példában két darab 7 elemű permutációt adtam meg, amin bemutatom a Hamming távolságmérés alapját. Az üres nyilak azokat az elempárokat jelölik, amelyek nem egyeznek meg. Az üres nyilak mennyisége fogja megadni a metódus által meghatározott cserék számának szükségletét. A saját módszerem is ugyanarra az eredményre jutott, amely 5 cserét határoz meg. Azonban más metódusok szerint már 3 cserével visszakaphatjuk az eredeti variációt.



15. ábra: 7 elemtagú példafeladat ugyanazon elemekkel

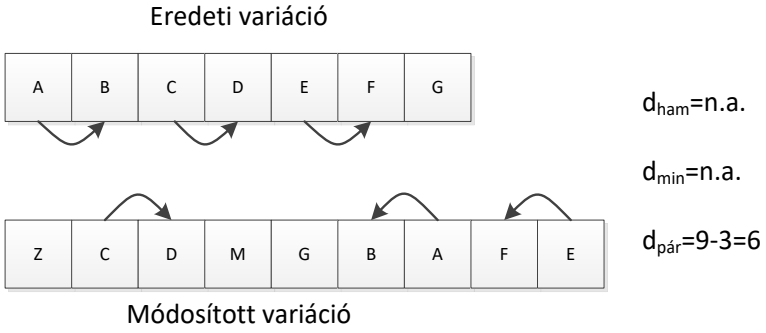
A 3. példában szintén két 7 elemű permutációt adok meg, viszont itt már olyan elemek is találhatók a variációkban, amelyek a másikban nincsenek meg. Ezt a Hamming távolság még mindig tudja kezelni, azonban a minimumkeresési módszerek nagy része már nem használható. A minimumkeresési metódusok úgy tudnak tovább keresni, hogyha 1 cserével kicserélik az elemet, amely a másikban nem található egy olyan elemre, amely viszont található. Ezért jön ki több cserelehetőség ennél a lehetőségnél, mint a párkeresésnél. Az ábrán a nyilak a párokat jelentik a két variációban.



16. ábra: Példafeladat fix elemszámmal, de különböző elemekkel

Az utolsó példában már két olyan permutációnk van, amelyek már elemszámban is különböznek. Ilyenkor a szokásos permutációs távolságmérési módszerek nem alkalmazhatóak, semmilyen eredményt nem képesek produkálni. A párkereső módszer itt is tud keresni, azonban ilyenkor nincs mihez viszonyítanunk a hatékonyságát. Az ábrán a nyilak a párokat jelentik a két variációban.

DOI: 10.14750/ME.2020.018



17. ábra: Különböző elemszámú példafeladat

A módszer megalkotásának szükségességét és az alkalmazhatóságát a 7.2-es „Integrált járattervezési probléma megoldása” fejezetben találhatjuk, amelyben egy konkrét példán keresztül kerül felhasználásra.

I. TÉZIS: Kidolgoztam egy új, szekvenciák közötti távolságok mérésére alkalmas módszert, bizonyítottam annak alkalmasságát különböző hosszúságú és felépítésű szekvenciák közötti távolságok mérésére vonatkozóan az egyezőség, a szimmetria és a távolság nem-negativitás szempontjából. A kidolgozott módszer alkalmas különböző járattervezési feladatokban az egyes megoldásváltozatokat reprezentáló permutációs egyedek közötti távolságok meghatározására.

A tézis állítását alátámasztó saját publikációk és előadások: [S5] [S8] [S9] [S10].

DOI: 10.14750/ME.2020.018

5. HEURISZTIKUS ALGORITMUSOK FEJLESZTÉSE

A 4. fejezetben bemutatásra került a heurisztikus módszerek működése, tulajdonságai és fajtái. Ezekből látható, hogy a heurisztika egy nagyon tág fogalomkör és elmélyülni csak akkor lehet benne, ha ezen belül is kisebb csoportokkal dolgozunk. Az általam választott csoportkör a heurisztikus algoritmusok, azon belül is a természet-inspirálta, populáció alapú algoritmusok. Ezekből is négy módszerrel foglalkoztam részletesebben doktoranduszi éveim alatt. A továbbiakban a 4 algoritmus működését fogom bemutatni, amelyre azért van szükség, hogy megértsük a rajtuk végzett fejlesztések és módosítások szükségességét és eredményességet.

5.1. Black hole algoritmus

5.1.1. Eredeti Black hole algoritmus

A Black hole (továbbiakban BH) algoritmus egy populáció alapú természet inspirálta optimumkeresési stratégia [I37]. A többi populáció alapú kereséshez hasonlóan az egyedek, amelyek tárolják a különböző megoldások paramétereit, egy előre meghatározott keresési térben vesznek fel először véletlenszerű értékeket. A populáció alapú algoritmusok nagy része, mint ez is, folyamatosan fejlesztik az egyedeiket speciális módszerekkel evolutív módon. A genetikus algoritmusnál például ezt mutációval és keresztezéssel tesszük meg.

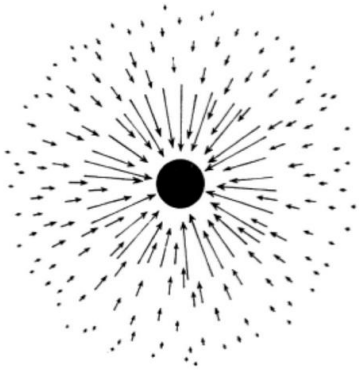
A BH algoritmusban a populáció fejlesztése úgy történik, hogy minden iterációban az egyedeket (égitestek) a legjobb egyed felé mozdítjuk el, nevezetesen ez az egyed lesz a fekete lyuk. Ha viszont egy egyed túl közel megy a fekete lyukhoz, akkor azt az egyedet a fekete lyuk elnyeli, viszont helyette egy új égitest jön létre egy véletlen helyen a keresési térben.

A fekete lyuk kifejezetést és a módszer alapjait először egy benchmark funkciók megoldását vizsgáló tanulmányban használták, mely 2008-ban jelent meg. A tanulmányban a részecske raj optimalizálás (particle swarm optimization=PSO) módszerét szerették volna gyorsabbá és pontosabbá tenni [K16].

A módszer egy új mechanizmust vezetett be a PSO keresési stratégiái közé. A módszerben minden iterációnál egy új részecskét állítunk elő véletlenszerűen, de a legjobb részecske közelében. Ezután két véletlenszerűen generált szám alapján az algoritmus frissíti a részecskék helyét egy PSO módszer vagy az új mechanizmus segítségével. Más szóval, ez a módszer a PSO kiterjesztésének minősül és bevezet egy új részecskét, melyet Black hole-nak, azaz fekete lyuknak hív, amely más részecskéket vonz bizonyos feltételekkel, amelyek növelik a konvergencia sebességet és megakadályozzák a hosszú távú lokális optimumba ragadás problémáját. Ebben a metódusban még nem szerepel az eseményhorizonton belüli egyedek (égitestek) megsemmisítése, amely mint későbbiekben láthatjuk, még jobban növeli a konvergencia sebességet azáltal, hogy kordában tartjuk a populációt [I37].

Az általam bemutatásra szánt és mások által is használt Black hole algoritmus inkább hasonlít a természetben megjelenő fekete lyukakhoz, mint az előbb felvázolt PSO módszerhez.

DOI: 10.14750/ME.2020.018



18. ábra A fekete lyuk húzása és az eseményhorizont [K17]

Az alap BH algoritmus úgy indul, hogy a keresési térben meghatározott számú egyedet létrehozunk, amelyek a populációt alkotják. Minden egyes egyed paraméterét feltöltjük véletlenszerű értékekkel a keresési térnek megfelelő korlátok között. Ezután minden egyes iterációban kiválasztjuk a legjobb célfüggvény értékkel bíró egyedet, amelyet kinevezünk a fekete lyuknak. Ebben a módszerben a fekete lyukat nem véletlenszerűen állítjuk elő, hanem tagja a populációnak. Még ugyanabban az iterációban az összes többi egyed (égitest vagy csillag) elmozdul a fekete lyuk irányába véletlen mértékben, amelyhez tudni kell minden egyed aktuális pozícióját (P_n^{i+1}), a fekete lyuk pozícióját ($P_{n,BH}^i$) és generálni kell egy véletlen számot 0…1 között ($rand(0 \dots 1)$). Az elmozdulás mértékét külön nem számítjuk, sokkal egyszerűbb megadni és számolni a következő iterációban a csillagok helyét, amelyet az alábbi képlet ad meg, minden egyes *n* paraméterre (P_n^{i+1}):

$$P_n^{i+1} = P_n^i + (P_{n,BH}^i - P_n^i) * rand(0 \dots 1) \tag{40}$$

Ha csak ennyiből állna az algoritmus, akkor semmi nem segítené abban, hogy egy lokális optimum pontot el tudjon hagyni. Erre is ad a természet egy nagyon hasznos mintát, a feketelyukat övező eseményhorizontot. Az algoritmusban is szerepel ez a jelenség: ha egy csillag vagy égitest túl közel kerül a fekete lyukhoz és a célfüggvény értéke nem jobb nála (mindig az adott iteráció elején vizsgáljuk meg és választjuk ki, hogy abban az iterációban melyik a feketelyuk), akkor a feketelyuk elnyeli. Amikor a fekete lyuk elnyel egy csillagot, egy új keletkezik véletlenszerűen a keresési térben. Ez a rész védi ki a lokális optimum problémáját. Az alap BH-algoritmusnál az alábbi képlet határozza meg az eseményhorizont rádiuszának nagyságát:

$$R^i = \frac{f_{BH}^i}{\sum_{N=1}^{pop} f_N^i} \tag{41}$$

ahol, R^i az eseményhorizont rádiusza az *i.* iterációban, f_{BH}^i a célfüggvény értéke a legjobb egyednek, azaz a feketelyuknak, és $\sum_{N=1}^{pop} f_N^i$ pedig az összes csillag célfüggvény értéke összeadva.

DOI: 10.14750/ME.2020.018

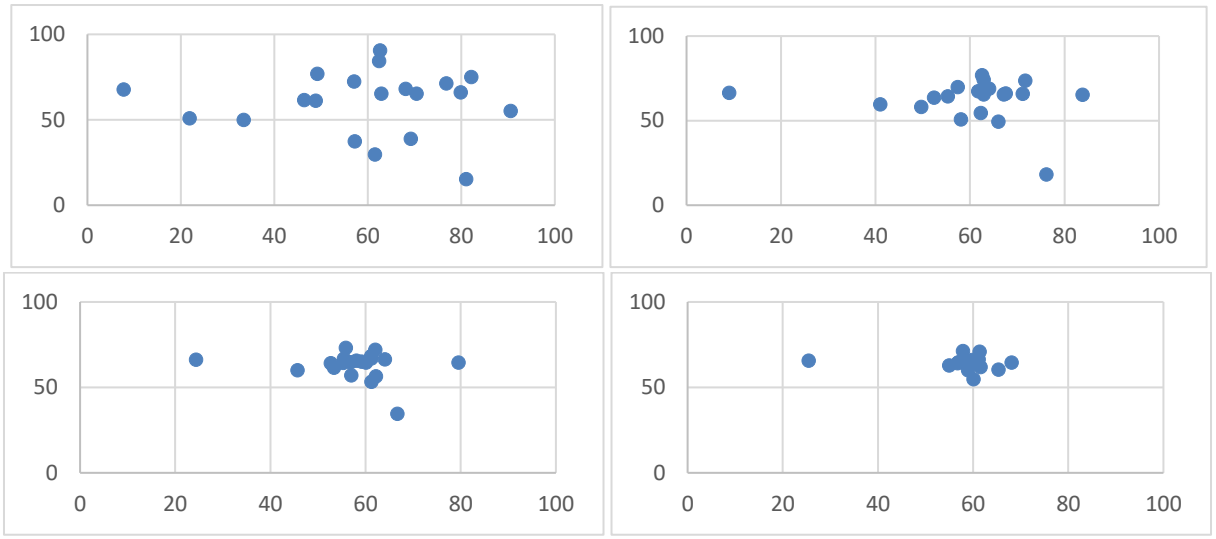
A Black hole algoritmus pszeudokódja [I37]:

```
Begin
Input: Induló adatok, melyek az optimalizálás alapját adják
Input: Az algoritmus vezérléséhez szükséges paraméterek:
       Kilépési kritérium:  iteráció (I) vagy
                            pontosság (ACU)
       Populáció (POP)
       Paraméterek típusa (TYP), mennyisége (PQ), min (MIN) max (MAX)értéke
Célfüggvény meghatározása: f(x)
0. generáció random generálása, POP mennyiségű egyed, PQ mennyiségű paraméterrel

for i=1:I vagy while SOLreal-SOLbest>ACU

A legjobb értékű egyed kiválasztása, és dedikálása mint feketelyuk

Minden egyed, minden paraméterének újraszámolása a (40) elmozdulás képlet alapján,
minden paraméterre (P):
       Púj=Prégi+(Pblackhole-Prégi)*rand(0…1)

Eseményhorizont meghatározása a fenti képlet alapján
       for j=1:Eseményhorizonton belüli egyedek
       Az az egyed, amely paraméterei alapján beleesik az eseményhorizontba
       kitörlődik
       Helyére random paraméterekkel új egyed készül
       end for
end for/while
A legjobb megoldás megjelenítése
END
```

5.1.2. Black hole algoritmus futási példa

A blackhole algoritmus alapú optimalizálás a raj alapú heurisztikához tartozik. A szakirodalomban a módszer konvergenciáját a CEC 2015 referenciafüggvényein alapuló különböző tesztfüggvényekkel vizsgálták [K18], melyek eredményei nagyon jó felhasználhatóságot mutatnak. A következő négy egymás utáni ábrasorozat egy kétváltozós függvény futás közbeni állapotait mutatja. A függvény optimumpontja a (65,60) paramétereknél található. Ezt a 20 egyedből álló black-hole alapú megoldást csak a szemléltetés és az algoritmus könnyebb megértése miatt mutatom be. Az ábrákon a pontokkal jelölt megoldások, jelen esetünkben a csillagok mozgási folyamata figyelhető meg, és a fekete lyuk vonzása, ahogy szépen lassan megközelítik a legjobb megoldást erre a kétváltozós tesztfüggvény problémára. Figyelemre méltó, hogy a részecskék (csillagok) sebessége különbözik, és ezen négy iterációs lépésben nem keletkezett új csillag, ami azt jelenti, hogy az új részecskék paraméterei nem voltak jobbak, mint a már meglévő legjobb megoldás, azaz a mindenkori feketelyuk paraméterei.

DOI: 10.14750/ME.2020.018



19. ábra: Black hole algoritmus működésének kirajzolása négy egymás követi iterációs lépésben (balról jobbra, fentről lefelé haladva)

5.1.3. Módosított Black hole algoritmus

Az általam bemutatott és használt algoritmusok közül csak a genetikus algoritmusnak és a Harmony Search algoritmusnak van olyan leírása, mely képes kezelni permutációs paramétert, mely alapvető reprezentációs forma olyan járattervezési feladatok esetében, mint az utazó ügynök probléma, ahol egy optimális sorrend meghatározása cél. Mivel a módszer távolságokat figyel és két sorrend között csak nagyon elvonatkoztatva lehet távolságot találni ezért eredetileg nem képes megoldani ezeket a feladatokat.

A 4.2-es fejezetben bemutatott kutatásom alapján, melyet elsődlegesen a Firefly algoritmus hasonló problémája miatt készítettem, képesek vagyunk elég jó közelítéssel meghatározni távolságokat sorrendek között. Ez lenne a cseretávolság mely két pont kapcsolatát (tehát az élek meglétét) figyeli. Ezt felhasználva olyan módosításokat végeztem el az algoritmuson, amely képes a sorrenddel ellátott feladatot is megoldani. A 6. táblázatban ennek MSEQ-BH (Modified SEQuence Black Hole) a neve [S6].

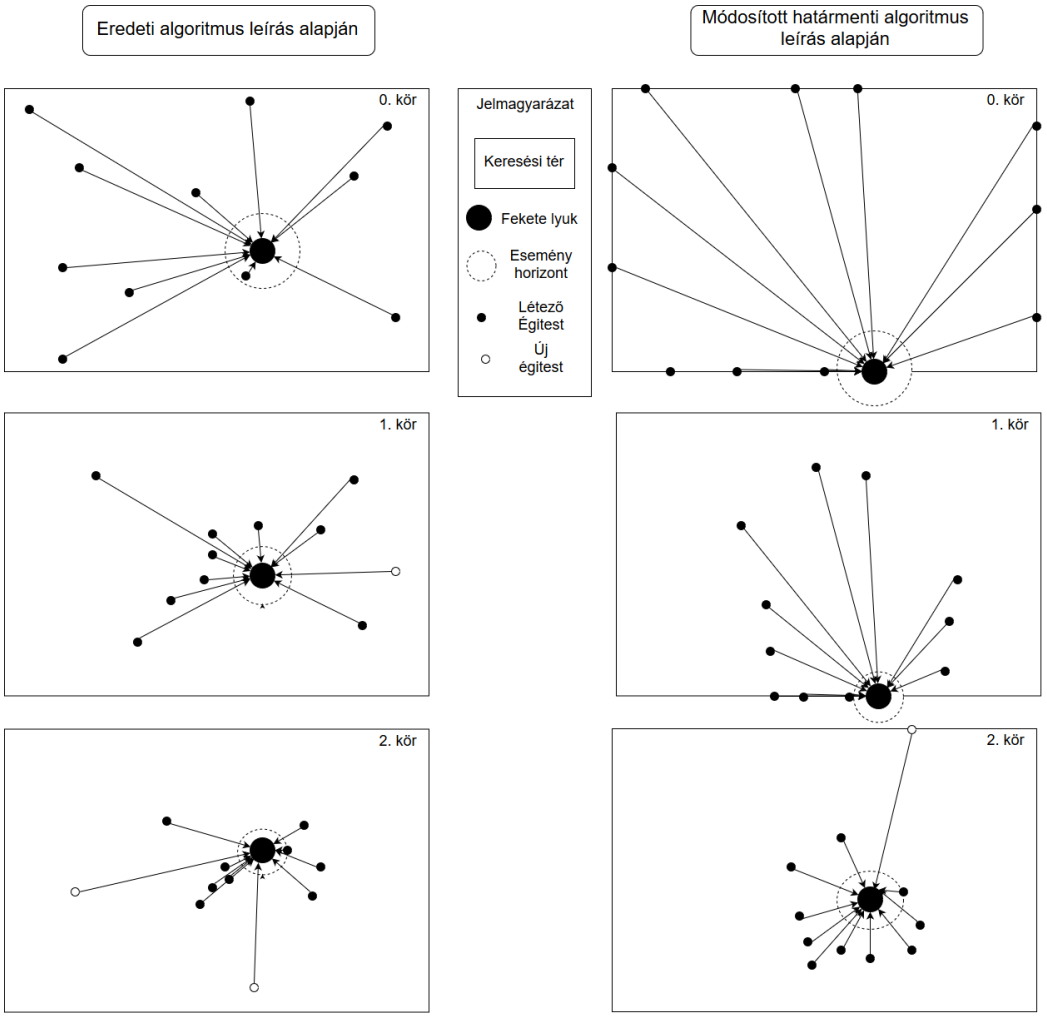
A MSEQ-BH pszeudokódja a következő:

```
Begin
Input: Induló adatok, melyek az optimalizálás alapját adják
Input: Az algoritmus vezérléséhez szükséges paraméterek:
       kilépési feltétel:   iteráció (I) vagy
                            pontosság (ACU)
       Populáció (POP)
       Paraméterek típusa (TYP), mennyisége (PQ), min (MIN) max (MAX)értéke
Célfüggvény meghatározása: f(x)
0. generáció random generálása, POP mennyiségű egyed, PQ mennyiségű paraméterrel
Generálásnál minden egyednél rand(1…PQ/2) paraméter kiválasztása mely megkapja a
paraméterre vonatkozó P(MIN) vagy P(MAX) értéket.

for i=1:I vagy while SOLreal-SOLbest>ACU

A legjobb értékű egyed kiválasztása, feketelyukká tétele
```

DOI: 10.14750/ME.2020.018

```
Minden egyed, minden paraméterének újraszámolása a fentebb említett elmozdulás
képlet alapján, minden paraméterre (P):
       Púj=Prégi+(Pblackhole-Prégi)*rand(0…1)

Eseményhorizont meghatározása a fenti képlet alapján
       for j=1:Eseményhorizonton belüli egyedek
       Az az egyed, amely paraméterei alapján beleesik az eseményhorizontba
       kitörlődik
       Helyére random paraméterekkel új egyed készül:
       Generálásnál minden egyednél rand(1…PQ/2) paraméter kiválasztása, mely
       megkapja a paraméterre vonatkozó P(MIN) vagy P(MAX) értéket.
       end for
end for/while
A legjobb megoldás megjelenítése
END
```

A második módosításom sokkal látványosabb és a gyakorlati képletekkel való munkák alapján készült el. Észrevettem, hogy a gyakorlati képleteknél szinte minden esetben vannak olyan változók, melyek valamelyik határértéket veszik fel a paraméterüknek. Ezen ötlet alapján elkészítettem egy olyan módosítását a Black-hole algoritmusnak, mely egy egyedben legalább egy legfeljebb pedig a paraméterek felének értékeit úgy határozza meg, hogy az egybeessen vagy az alsó vagy a felső határértékével.



20. ábra: Normál BHA algoritmus (balra) és Módosított Határmenti BHA működése (jobbra)

DOI: 10.14750/ME.2020.018

Ezt szemlélteti a 20. ábra jobb oldala. Mind a kezdeti 0. kör, mind az új egyedek így kezdenek. Az ábrán egy kétváltozós (x,y) koordináta alapú feladat látható, amelyet szemléltetésnek választottam. Ez a hatás sokkal nagyobb jelentőséggel bír nagy mennyiségű paraméter esetén, viszont kevésbé látványos és nehezen vizualizálható. A Módosított Határmenti Black-hole algoritmus a futtatási táblázatokban az MBS-BH nevet viseli. Az O-BH az eredeti algoritmust mutatja be, a GA a Genetikus és a HS a Harmony Search algoritmus számítási eredményeit tartalmazza. A következő futtatási táblázatok eredményei sokkal jobban megérthetők, ha előtte elolvasásra kerül a 6. fejezet, amely a heurisztikus algoritmusok teszteléséről és tesztkörnyezetéről szól. Ott kerülnek bemutatásra a tesztfüggvények és az egyszerű tesztfeladatok, amely a táblázat felső felében mutatja az eredményeket. A táblázat alsó felében az összetettebb és a 7. fejezetben bemutatott modellek alapján készült feladatok találhatók. A dolgozat logikai felépítése miatt volt szükség később bemutatni a felhasználásukat a kidolgozott algoritmusoknak.

6. táblázat: Algoritmusok összehasonlítása BH és módosított BH-val, populáció mérete 300 egyed

			Populáció 300									
	változó típus		**Pontosság (log10)**; Iteráció 1000					Idő (s)				
		Algoritmus	GA	HS	O-BH	MBS-BH	MSEQ-BH	GA	HS	O-BH	MBS-BH	MSEQ-BH
Tesztfüggvények	dec	2 változós parabolikus	-143	-121	-312	-312	-	4,5683	1,6379	4,7932	4,6206	-
	dec	10 változós parabolikus	-104	-13	-312	-312	-	5,3259	1,8762	5,8736	5,6915	-
	dec	Goldstein price	-96	-67	-223	-242	-	3,7634	1,5541	4,7352	4,7494	-
	dec	Easom	-52	-8	-74	-72		4,3272	1,7512	4,5914	4,6153	
	dec	Himmelblau	-113	-59	-207	-204	-	4,7895	1,6783	4,5782	4,4088	-
Tesztfeladatok	dec	10 város centrum keresés	-65	-14	-82	-95	-	7,2458	2,2475	8,2511	8,1933	-
	dec	100 város centrum keresés	-43	-11	-56	-54	-	9,6514	3,7983	10,3568	10,9679	-
	dec	30 város 3 centrum(multi) keresés	-25	-6	-37	-41	-	18,4768	10,7327	20,7895	21,5171	
	dec	Esettanulmány: Integrált elosztási rendszer (7.3. fejezet)	-14	-	-18	-21	-	54,2386	-	61,7892	63,8941	-
			Pontosság 0; **Iterációs szám (db)**					Idő (s)				
	int	8x5 szállítási feladat	247	1764	226	197	-	1,3684	3,8753	1,2753	1,1597	-
	bin/int	100 db-os hátizsák feladat	72	151	104	119	-	0,4782	0,3492	0,7589	0,9265	-
	seq	20 város körjárat tervezés	580	2866	-	-	943	4,5159	8,8617	-	-	6,9242
	seq	50 város körjárat tervezés 2 centrum	3875	20000+	-	-	8736	27,6395	120+	-	-	51,9753
	seq	Esettanulmány: Automatikus járatterv (7.2. fejezet)	3264	-	-	-	5971	52,9758	-	-	-	71,5672
Átlag (dec feladatoknál)			-73	-37	-147	-150		12,4874	3,1595	13,9731	14,2953	
Százalék (GA=100%)			100	51	202	207		100	25	112	114	

7. táblázat: Algoritmusok összehasonlítása BH és módosított BH-val, populáció mérete 30 egyed

			Populáció 30									
	változó típus		**Pontosság (log10)**; Iteráció 1000					Idő (s)				
		Algoritmus	GA	HS	O-BH	MBS-BH	MSEQ-BH	GA	HS	O-BH	MBS-BH	MSEQ-BH
Tesztfüggvények	dec	2 változós parabolikus	-123	-151	-312	-312	-	1,7804	1,1705	1,5249	1,2090	-
	dec	10 változós parabolikus	-83	-15	-264	-242	-	1,6406	1,3923	2,0318	1,7454	-
	dec	Goldstein price	-77	-84	-164	-160	-	1,3344	1,2650	1,4779	1,2794	-
	dec	Easom	-43	-9	-52	-47		1,7292	1,3172	1,3993	1,2888	
	dec	Himmelblau	-94	-66	-165	-161	-	1,4455	1,5081	1,3617	1,3908	-
Tesztfeladatok	dec	10 város centrum keresés	-57	-16	-55	-75	-	2,2414	1,6331	2,1361	2,2540	-
	dec	100 város centrum keresés	-35	-14	-39	-42	-	3,3678	3,1618	3,1208	3,7754	-
	dec	30 város 3 centrum(multi) keresés	-21	-7	-23	-27	-	6,1986	7,7001	5,2692	7,0557	
	dec	Esettanulmány: Integrált elosztási rendszer (7.3. fejezet)	-8	-	-11	-14	-	26,5792	-	29,6823	25,9632	-
			Pontosság 0; **Iterációs szám (db)**					Idő (s)				
	int	8x5 szállítási feladat	674	1975	896	927	-	1,5893	2,9765	1,8763	2,0193	-
	bin/int	100 db-os hátizsák feladat	352	206	294	283	-	0,6928	0,4367	0,6193	0,6573	-
	seq	20 város körjárat tervezés	1891	8796	-	-	4761	7,3649	27,9527	-	-	21,9856
	seq	50 város körjárat tervezés 2 centrum	5967	20000+	-	-	14698	38,6915	120+	-	-	86,3291
	seq	Esettanulmány: Automatikus járatterv (7.2. fejezet)	5873	-	-	-	8364	61,4379	-	-	-	95,6914
Átlag (dec feladatoknál)			-60	-45	-121	-120		5,1463	2,3935	5,3338	5,1069	
Százalék (GA=100%)			100	75	201	200		100	47	104	99	

DOI: 10.14750/ME.2020.018

Ahogy az a 6. és 7. táblázatokban láthatók, az algoritmusokat a pontosságuk/iterációs számuk és az idő szerint hasonlítottam össze 5 tesztfüggvény és 9 logisztikai alapú tesztfeladat segítségével. A dec változó típusú feladatoknál a pontosságot 10^x formában definiáltam, tehát nagyobb negatív szám pontosabb megoldást jelent. Az int/seq típusú feladatoknál nem tudunk ilyen módon pontosságot értelmezni, mert ott meghatározott vagy korlátozott számokból lehet csak választani. Ezekben az esetekben akkor állt le a futás, amikor elérték a tényleges optimumot. Ennek az iterációs száma és a futási ideje van feltüntetve a táblázaton.

Még egy nagyon fontos értéket meg kellett adni minden futtatás elején, ez pedig a populáció mérete volt. Ez egy nagyon fontos paraméter, amelyet nagyon jól meg kell választani és mindig előre kell definiálni minden egyes futtatásnál. A két táblázaton a populáció megválasztásának hatásait is szeretném érzékeltetni, mint érzékenységvizsgálati eredményt. Az első táblázatban a populáció mérete 300 egyed, míg a második táblázatban 30.

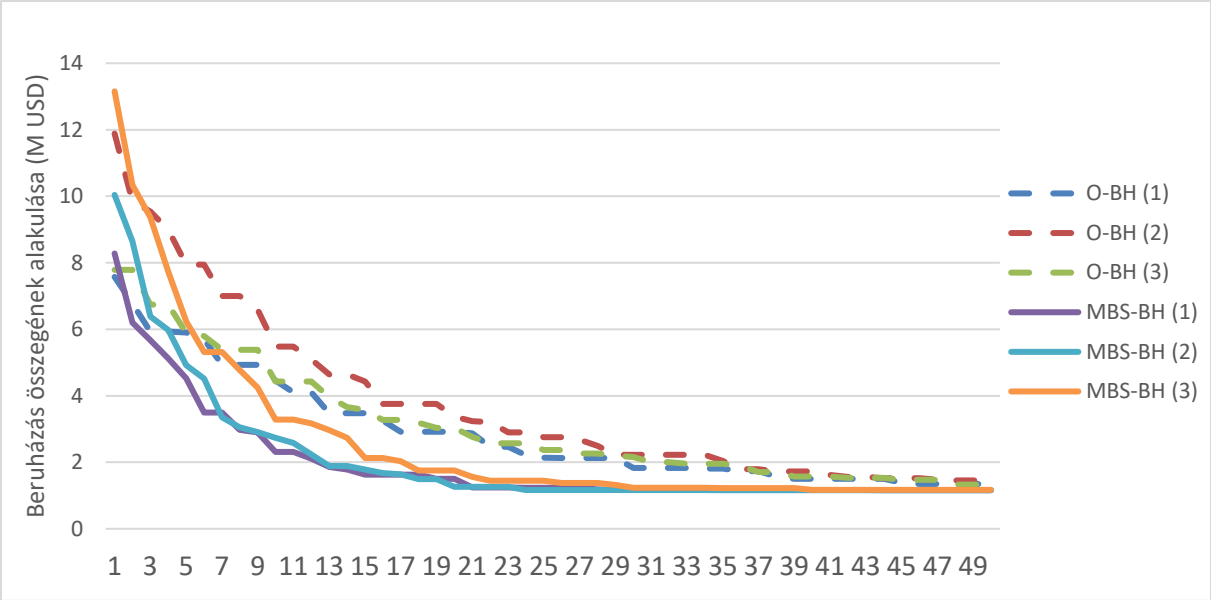
A genetikus, a Firefly és a Black hole algoritmus nagyon dinamikusnak számítható, ha populációról van szó, ugyanis ezek minden egyes lépésben megváltoztatják az összes egyedük paramétereit, így egyre gyorsabban haladnak az optimum felé, minél többen vannak, azonban ehhez több időre van szükségük. Ezzel ellentétben a Harmony Search algoritmusban minden körben csak a legrosszabb (vagy a legrosszabb néhány) esetek alakulnak át, így a körönkénti számítási teljesítmény nagyon kicsi (ami jó), de az optimum felé haladás nagyon lassú és ez a táblázatokból nagyon jól észre is vehető. A genetikus algoritmus folyamatosan jó számítási eredményeket ad, közepes futási idővel. Ebből is látszik, hogy még mindig egy jól használható algoritmus.

A táblázatokból nagyon szépen kivehető, hogy a populáció tizedére való csökkentése a pontosságot mind a GA, O-BH és MBS-BH esetében 20-40%-al csökkenti, azonban hozzá képest mindhárom módszernél a futási idő nagyjából harmadára csökken. Ezek alól az esettanulmányok és komplexebb feladatok a kivételek. A legnagyobb változás a Harmony Search algoritmusban mutatkozik, amely ezen táblázatok alapján nagyon érzékeny a populáció helyes megválasztására, ugyanis a kisebb populáció felgyorsította az eljárást és volt olyan feladat, ahol pontosságában és sebességében megközelítette a többi algoritmust.

A BH algoritmusok kiemelkedően teljesítettek a dec-típusú feladatoknál legyen szó tesztfüggvényről vagy tesztfeladatról. Ez annak is köszönhető, hogy pontosan ezekre a feladatokra lettek kitalálva. A legtöbb esetben nagyságrendek többszörösével készítenek jobb megoldást, mint a második helyen lévő genetikus algoritmust úgy, hogy csak nagyon kicsivel, 5%...10%-al hosszabb a futásidejük, mint a GA-nak.

Azonban a helyzet megfordul, mikor a többi feladatra térünk rá. Láthatjuk, hogy a szekvenciális BHA nem teljesít túl jól, mind időben mind pontosságban alulmarad a genetikus algoritmushoz képest. A határmenti módosítás pedig a normál tesztfeladatoknál nem ad szignifikáns különbséget az eredetitől. Azonban a feladatok között megtalálható két komplexebb feladat (30 város 3 centrum (multi) keresés és esettanulmány: integrált elosztási rendszer) amelyeknél nem csak a genetikus algoritmust, hanem az eredeti BH algoritmust is túlszárnyalja. Ennek az az oka, hogy ezeknél a feladatoknál több paramétert kell figyelembe venni és több egyszerű döntési változó van benne úgy, mint egy gyakorlati feladatban.

DOI: 10.14750/ME.2020.018



21. ábra: Raktártelepítés beruházási költségeinek optimalizálási folyamata a két BH algoritmus iterációs lépésein keresztül

A 21. ábra nagyon jól szemlélteti az előző állítást, amely a 7.3. fejezetben részletesebben bemutatásra kerülő integrált elosztási rendszer tervezése esettanulmányán kerül bemutatásra. Az ábrán egy raktártelepítés beruházási költségeinek alakulása látható az eredeti Black hole (O-BH (1…3)) és a módosított határmenti keresés BH (MBS-BH(1…3)) keretei között. Mindkét algoritmus 3-3 alkalommal futott le. Az ábrából nagyon jól látható, hogy a futtatások kezdetén a szaggatott vonallal rajzolt eredeti algoritmus jócskán alulmarad a folyamatos vonalú módosított algoritmussal szemben. A méréseket csak 50 iterációig végeztem, ugyanis utána, a két algoritmus sebessége megegyezik, ami azt jelenti, hogy a módosítás viszonylag nagy előnyt jelent rövidtávon a gyors kereséseknél.

II. TÉZIS: Kidolgoztam egy olyan új paramétergenerálási módszert a hagyományos black hole heurisztikára, melynek révén annak konvergenciasebessége – különösen a keresési fázis elején – szignifikáns mértékben javítható. Kifejlesztettem a black hole heurisztikák egy olyan változatát, mely alkalmas szekvencia-problémák, például járattervezési feladatok megoldására.

A tézis állítását alátámasztó saját publikációk és előadások: [S3] [S6] [S7]

DOI: 10.14750/ME.2020.018

5.2. Firefly algoritmus

5.2.1. Eredeti Firefly algoritmus

A Firefly algoritmus egy természet által inspirált metaheurisztikus algoritmus, amelyet a szentjánosbogarak viselkedésének megfigyelése alapján alkottak meg. A szentjánosbogarak fényfelvillanásokon alapuló jelzésrendszert használnak éjszaka figyelemfelkeltetés céljából, amellyel magukhoz vonzzák az ellentétes nemű szentjánosbogarakat. Xin-She Yang alkotta meg az algoritmus első változatát 2008-ban, a következőképpen fogalmazta meg az algoritmus alaptételeit [I28]:

- Minden szentjánosbogár uniszex, tehát mindegyik vonzó hatással van az összes többi egyedre.
- A vonzás mértéke egyenesen arányos a fényességükkel. Bármelyik két szentjánosbogárra igaz, hogy a fényesebb vonzza magához a kevésbé fényeset. Viszont a bogarak egymáshoz viszonyított fényessége a távolságukkal is arányos.
- A legfényesebb bogár véletlenszerűen mozdul el, mivel annál nincs fényesebb, aki felé mozogjon.

Az algoritmus maximumkeresésre lett kifejlesztve, ahol a szentjánosbogarak fényessége arányos a célfüggvénnyel, azonban minimumkeresés is könnyedén megvalósítható.

A firefly algoritmust alapvetően folytonos, valamint decimális változókkal leírható problémák megoldására készítették, de speciális kikötésekkel és átértelmezésekkel diszkretizálható, így permutációs problémák is megoldhatók vele. Az egyedek mozgása (bármely a x_i és x_j szentjánosbogár esetén a t+1 iterációban) az alábbi két összefüggéssel határozható meg [I28]:

$$x_{i+1} = x_i + \beta_0 \cdot e^{-\gamma \cdot r_{ij}^2}(x_j - x_i) + \alpha(rand() - \frac{1}{2}), \tag{42}$$

vagy egyszerűbben:

$$\begin{gathered} \beta = \beta_0 \cdot e^{-\gamma r} \\ x_{i+1} = x_i \cdot (1 - \beta) + x_i \cdot \beta + \alpha(rand() - \frac{1}{2}), \end{gathered} \tag{43}$$

ahol:

- β_0: maximális attraktivitási érték;
- γ abszorpciós koefficiens: Általában az értéke 0.1 és 10 közé esik. Egy másik meghatározása lehet a $\gamma = \frac{1}{\sqrt{L}}$, ahol L a probléma mérete;
- r_{ij}: Az *i.* és a *j.* szentjánosbogár távolsága (metrikákra lásd példát a távolságmérés fejezetben).
- α_t randomizációs paraméter: A lépések nagyságát, nálunk a cserék mennyiségét adja meg.
- ϵ_t randomizációs paraméter: Egy véletlen szám vagy vektor, amelyet legtöbbször Gauss vagy egyenletes eloszlás generátorral gyártunk.

DOI: 10.14750/ME.2020.018

A Firefly algoritmus pszeudokódja kapacitásfüggő járattervezési feladatra az alábbiakban látható:

```
Begin
Induló töltöttség (IND), átlag fogyás (ATL), koordináták (X,Y,[Z…]) vagy
útmátrix (L) megadása vagy kiszámítása
Lépésköz (S), iteráció (I), napok számának (D) és szentjánosbogár populáció
számának (X) megadása
Célfüggvény meghatározása: f(x)
for i=1:100/S
Aktuális minimum meghatározása: AKTS= 1+i*S
     for j=1:D
     Útvonal elemeinek meghatározása:
         if (IND_j<AKTS),
          Az elem az útvonalban van
         end if
      Random útvonalváltozatok generálása: SOL[]
       for k=1:elemszám+1
       Útvonalhosszak megadása: M[]=L(SOL[k];SOL[k+1])
       end for k
       Fényesség függvényének meghatározása: M_ij, M_ij∝f(X), vagy egyszerűen
       M_ij=f(X)
       for l=1:I (iterációs szám)
              Legfényesebb (minimum) változat kiválasztása
              A többi változat megvizsgálása; hány elemben tér el a
              sorrendje a legjobbtól: A(X)
              Random szám generálása: B=Rand(1:A(X)-1)
              Minden a legjobbtól különböző változatban B-szer cserélni
              egymás mellett lévő random elemeket
              A legjobb változatban egyszer cserélni egy random elemet
              Fényesség újbóli meghatározása: M_ijl
       end for l
      napi legjobb útvonal kiválasztása: M_ij
  end for j
  Adott töltöttségre vonatkozó összes nap útvonalainak összege: M_i=SUM
[M_ij]
end for i
Legkevesebb útvonallal járó töltöttség kiválasztása
eredmények feldolgozása, vizualizáció;
End
```

5.2.2. Módosított Firefly algoritmus

A firefly algoritmus hasonlít a Black-hole algoritmusra, abban a tekintetben, hogy a legjobb kiválasztott egyedet követi a többi egyed. Azonban ennél az algoritmusnál nincs olyan jól elkülönített lokális optimumpont elkerülő mechanizmus, mint a Black-hole-nál (ha túl közel megy elnyelődik) így sokkal nagyobb eséllyel talál ez a keresési módszer lokális optimumpontot, melyből nem fog kijönni.

Ez a probléma könnyen kiküszöbölhető két általam meghatározott módszerrel is, amely követi a naturális természetét az algoritmusnak:

Az első módszernél bevezetünk egy **randomizált életciklust** (táblázatban RE-FF), ahol meghatározzuk, hogy mennyi lehet egy egyed élettartama (ciklusszám), ezután elhalálozik és helyet ad egy teljesen random új egyednek. Azért nem lehet mindegyiknek egyforma ez az

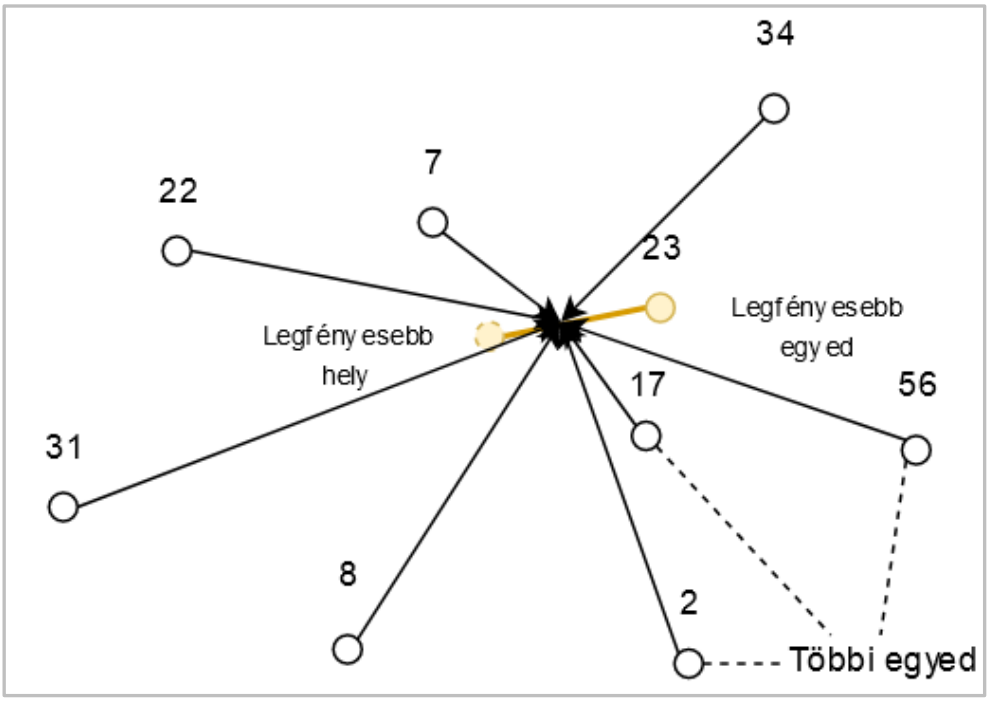
DOI: 10.14750/ME.2020.018

élettartam, ugyanis az összeset egyszerre generáltuk, így az összes egyszerre halna meg. Úgy kell meghatározni a minimális élettartamot, hogy egy egyed képes legyen nagyon közel kerülni az optimális ponthoz. Számításaim során a ciklusidő a 10-100 intervallumban volt vizsgálva. Így bőven van ideje elérni a céljait. Minden egyed a generáláskor kapja meg ezt a random számot, amelyet minden iteráció eggyel csökkent.

A második módszernél bevezetünk egy **elhalálozási rátát** (táblázatban EHR-FF), amely annyit tesz, hogy a legrosszabb N db egyedünk, amelyik nem talált párt az törlődik, majd a helyére új külső egyedek jönnek (randomizálással) [S5].

Mindkét módszernél fontos az új egyed generálásával a robusztusság növelése, és a lokális minimumpontokban ragadás esélyének a csökkentése. Bár a két módszernek hasonló a feladata és az eredménye, mint az a 8. táblázatból is látszik, azonban a megvalósítása matematikai és programozási szempontból más. Az elhalálozási ráta figyelembe veszi a saját megoldás jóságát a többi megoldáshoz képest, így egy irányított módszerről beszélünk, ahol a jobban teljesítő egyedek, azaz megoldások megmaradnak. A hátránya ennek módszernek, hogy nagyobb a számítási igénye, mint a randomizált életciklusnak. Ez a táblázatból kiolvasható kb 10%-os futásidő-növekedést jelent minden esetben. A másik módszer ezzel szemben minden egyed eltörlésére ugyanakkora esélyt ad, bármilyen paraméter figyelembevétele nélkül. A két módszert nem kell feltétlenül külön alkalmaznunk, képesek lehetünk együtt is használni őket.

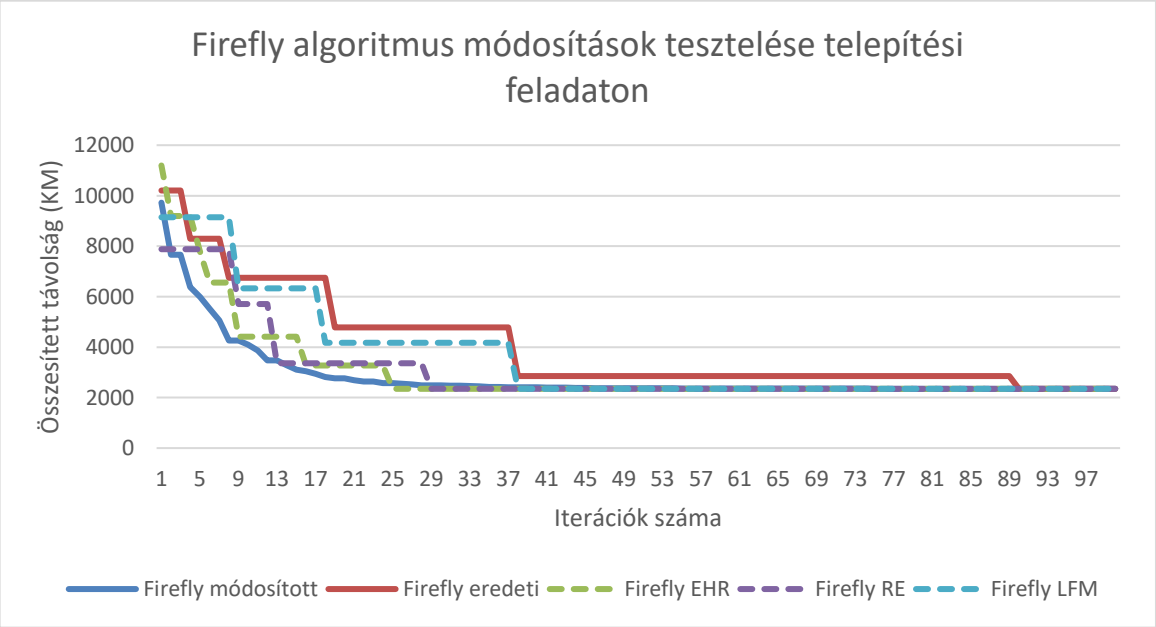
Ezeken felül egy kiegészítéssel látnék el minden verziót, ez pedig a **legfényesebb memória** (táblázatban LFM-FF). Az algoritmus leírásában szerepel, hogy a legfényesebb egyedet követi az összes többi egyed, azonban a legfényesebb egyed is mozog az iterációk között és ez a mozgás random irányba történik. Bevezettem egy olyan apró módosítást, miszerint a bogarak megjegyzik a mindenkori legfényesebb helyzetet és azt a pontot követik, mely a legfényesebb bogár és a legfényesebb helyzet közötti szakaszon a fényességgel (fitnesz értékkel) egyenes arányban oszlik meg.



22. ábra: Legfényesebb memória és randomizált életciklus használata

DOI: 10.14750/ME.2020.018

A 22. ábrán látható a két módosítás együttes használata, ahol a sárga kör a legfényesebb egyed helyét (jelenlegi legjobb megoldás) jelöli, a szaggatott sárga kör pedig a valaha létezett legjobb megoldást. A legfényesebb memória módszere nem hagyja, hogy ez a megoldás elfelejtődjön. Ahhoz, hogy megmaradjon a rendszer robosztussága, ki kell egészíteni az életciklussal, amely az egyedek fölötti szám az ábrán. Amikor ez letelik, az egyed helyett egy új jelenik meg valahol a keresési térben.



23. ábra: Módosított és eredeti Firefly algoritmus futtatási képe

A 23. ábrán látszik a telepítési feladat megoldása, amelynél 10 üzem közötti gyűjtőraktár helyét kellett megállapítani. Az ábrán látszik, ahogy az eredeti Firefly algoritmus nagy lépcsőkben érkezik meg egy-egy lokális minimumponthoz, amelyeket a módosított algoritmus, amely tartalmazza mindhárom fejlesztést, hatékonyabban kikerül. Nagyon jól látszik, hogy az elhalálozási ráta és randomizált életciklus felhasználásával sokkal rövidebbek az egyenes szakaszok, amikor az algoritmus próbál kitörni egy lokális optimumpontból. A legfényesebb memória bevezetésével nagyjából ugyanazt a görbét kapjuk, mint az eredetinél, csak egy kicsit jobb a konvergenciasebesség

Mint előző fejezteben is leírtam a Black hole algoritmus eredményeit bemutató táblázatnál, úgy itt is ki kell emelnem, hogy a Firefly algoritmussal végzett futtatások eredményeit tartalmazó táblázat sokkal könnyebben értelmezhetővé válik, ha előtte elolvasásra kerül a 6. és 7. fejezet. A 8A és 8B táblázat felső felében az algoritmusokat tesztfüggvényeken és az egyszerű tesztfeladatokon vizsgálom, amelyek a 6. fejezeteben kerülnek bemutatásra. A táblázat alsó felében az összetettebb és a 7. fejezetben bemutatott modellek alapján készült feladatok találhatók. A dolgozat logikai felépítése miatt volt szükség később bemutatni a felhasználásukat a kidolgozott algoritmusoknak.

DOI: 10.14750/ME.2020.018

8. táblázat A és B:Firefly algoritmusok teljesítményei a genetikus algoritmushoz képest

A			Populáció 300						
			Pontosság (log10); Iteráció 1000						
	változó típus	**Algoritmus**	GA	HS	O-FF	RE-FF	EHR-FF	LFM-FF	M-FF
Tesztfüggvények	dec	2 változós parabolikus	-143	-121	-97	-99	-102	-94	-104
	dec	10 változós parabolikus	-104	-13	-69	-68	-71	-73	-82
	dec	Bukin No 6.	-121	-70	-85	-82	-88	-86	-98
	dec	Goldstein price	-96	-67	-92	-98	-95	-93	-101
	dec	hatpúpú teve	-109	-62	-72	-73	-72	-72	-83
	dec	Easom	-52	-32	-26	-27	-28	-25	-36
	dec	Himmelblau	-113	-59	-93	-97	-92	-92	-86
Tesztfeladatok	dec	10 város centrum keresés	-65	-14	-23	-24	-28	-26	-42
	dec	100 város centrum keresés	-43	-11	-11	-14	-19	-13	-31
	dec	30 város 3 centrum(multi) keresés	-25	-6	-5	-	-	-	-14
	dec	Esettanulmány: Integrált elosztási rendszer (7.3. fejezet)	-14	-	-6	-	-	-	-11
Átlag (dec feladatoknál)			-94	-50	-63	-65	-66	-64	-74

B			Populáció 300						
			Idő (s)						
	változó típus	**Algoritmus**	GA	HS	O-FF	RE-FF	EHR-FF	LFM-FF	M-FF
Tesztfüggvények	dec	2 változós parabolikus	4,5683	1,6379	2,7898	2,9081	2,9931	2,8830	3,0706
	dec	10 változós parabolikus	5,3259	1,8762	3,4935	3,5305	3,8743	3,5525	3,7484
	dec	Bukin No 6.	4,6325	1,6956	2,8270	2,9209	3,1913	3,0044	3,6637
	dec	Goldstein price	3,7634	1,5541	2,8642	2,9287	3,0773	2,9869	3,1396
	dec	hatpúpú teve	3,9168	1,7294	2,6675	2,7796	3,0307	2,6790	3,1455
	dec	Easom	4,3272	1,6837	3,0643	3,1990	3,4229	3,2181	3,8127
	dec	Himmelblau	4,7895	1,6783	3,4276	3,5464	3,6912	3,2515	4,0931
Tesztfeladatok	dec	10 város centrum keresés	7,2458	2,2475	4,4714	4,6515	5,0743	4,7076	4,6844
	dec	100 város centrum keresés	9,6514	3,7983	6,4729	6,6610	7,0019	6,1809	8,1458
	dec	30 város 3 centrum(multi) keresés	18,4768	10,7327	15,2171	-	-	-	18,8156
	dec	Esettanulmány: Integrált elosztási rendszer (7.3. fejezet)	54,2386	-	44,9427	-	-	-	54,0003
Átlag (dec feladatoknál)			5,3579	1,9890	3,5643	3,6806	3,9286	3,6071	4,1671

A 8. táblázat a hosszúsága miatt ketté lett vágva, de egytáblázatnak minősül, amelyben a felső A táblázat a pontosságot, az alsó B táblázat az időket tartalmazza. A 8. táblázatban láthatóak konkrét futtatási adatok az algoritmusok teljesítményéről a genetikus algoritmushoz mérve és első látásra azt mondhatjuk, hogy a MATLAB programba beépített genetikus algoritmus minden esetben jobban teljesít, mint a firefly algoritmusok. Azonban az is elmondható, hogy a Mósodított Firefly (M-FF), amely mindhárom módosítást egyszerre tartalmazza, szinte minden esetben jobban teljesít pontosság szempontjából, mint az eredeti verzió (O-FF), viszont átlagosan kb. 20%-al nagyobb a szükséges futásideje. Amit még érdemes megfigyelni a 8. táblázatban az a tesztfeladatoknál nagymértékben észrevehető pontosságnövekedés. Az eredetihez képest a módosított FF nagyjából kétszeres nagyságrendű javulást ért el.

Ezek mellett ennél az algoritmusnál is elkészítettem a szekvenciális feladatra megoldást nyújtó változatát, amelyet semmilyen irodalomban nem találtam, azonban nagyságrendekkel rosszabb futásteljesítménye volt, mint a genetikus algoritmusnak, ezért felhagytam a fejlesztésével. Mielőtt ez még kiderült volna, addig viszont sikeresen publikáltam három műben [S5] [S9] [S10].

DOI: 10.14750/ME.2020.018

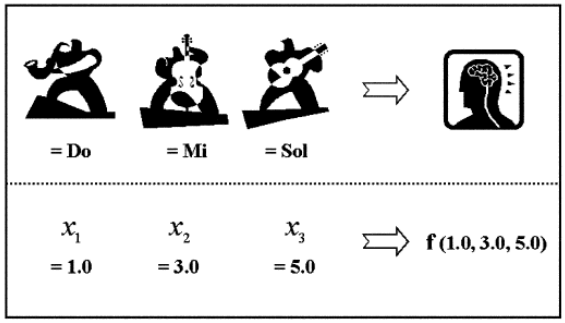
III. TÉZIS: Bevezettem a randomizált életciklus és az elhalálozási ráta fogalmát új egyedek generálására a Firefly algoritmus robusztusságnak növelése érdekében. Emellett bevezettem a legfényesebb memória alkalmazását Firefly algoritmusok hatékonyságának növelése céljából. Az elvégzett benchmark tesztek igazolták, hogy az általam kifejlesztett bővítmények révén az eredeti Firefly algoritmus hatékonysága növelhető.

A tézis állítását alátámasztó saját publikációk és előadások: [S5] [S9] [S10] [S11]

5.3. Harmony Search algoritmus

2001-ben Z.W. Geem, J.-H. Kim és G.V. Loganathan kifejlesztettek egy olyan metaheurisztikus algoritmust, amelyet a zenei életből vett tökéletesség vagy más néven harmónia megteremtése alapján készítettek el és a harmónia keresési (Harmony Search) algoritmus nevet adták neki [I23]. A harmonikus zene az optimalizált megoldás vektornak és a zenészek az improvizációikkal a lokális és globális kereső rendszereknek felelnek meg. Az algoritmus alapja a zenei harmónia keresés, amelyet sztochasztikus véletlenszerű elemekkel épít fel a megszokott gradiens elemek helyett, amivel a járulékos szükségtelen számítások is eltűnnek. Különféle mérnöki optimalizációs problémákra használható az eljárás [I22].

A harmóniakeresési algoritmus egy természeti jelenségen, még pedig a zene harmóniájának rendszerén nyugszik. A zenés előadások célja, hogy megtalálják azt a kellemes zenei harmóniát, ami örömöt nyújt a hallgatóknak; az optimalizálási folyamat célja, hogy megtalálja a globális megoldás által meghatározott célt. A zene harmóniája egy optimalizált megoldásvektor és a zenész improvizációs technikája egy lokális és globális keresési séma az optimalizálási technikában. Ez az algoritmus nagyon sok másik módszerrel ellentétben nem igényel kezdőértéket a döntési változóknak és nem szükséges keresési teret sem meghatározni. Egyedül az inicializáló lépéseknek kell megadni valamilyen kezdőértéket. Alappillérei a Harmónia Memória Figyelési Arány (Harmony Memory Considering Rate, röviden HMCR) és a Hangmagasság Szabályzó Arány (Pitch Adjust Rate, röviden PAR). Az esztétikai minőséget a különböző hangszerek hangmagassága és azok aránya határozza meg, ahogy egy eredményt a célfüggvény és változóinak értéke ad meg. A HS algoritmus alapját olyan zenék adják, ahol a zenei előadást a zenész improvizációval kívánja feldobni. Ilyen zenei irányzat például a Jazz [S12].



24. ábra Zenészek és azok megfeleltetése döntési változóknak [I23]

DOI: 10.14750/ME.2020.018

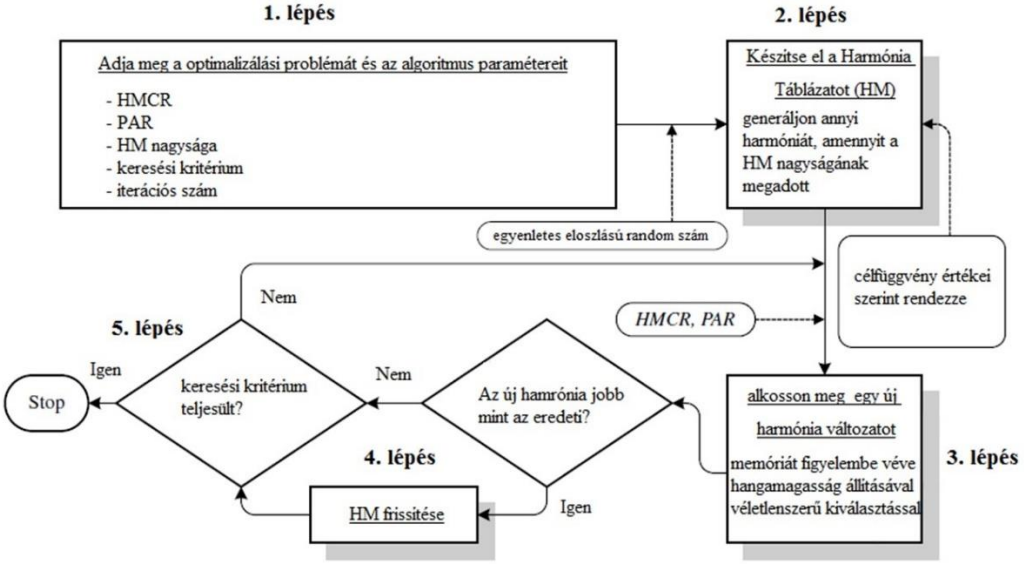
A 24. ábrán látszik, ahogy több zenész, más-más hangszerrel és hangmagassággal a tökéletes zenei harmóniára törekszik, úgy ahogy több változó is az optimális értékre törekszik.

Ha az összes hangmagasság közül a legjobbat, amivel a legszebben hangzott a zene, minden zenész megjegyzi, akkor a következő alkalommal, már azt fogja alkalmazni és egyre kimagaslóbb lesz a zenei élmény. Ugyanígy működik a harmónia keresés algoritmusa is. Ha megjegyezzük az eredeti változók értékeit és változtatunk valamin, ami ha jobb eredményt mutat, akkor az újat használjuk inkább. Így lépésenként egyre jobb megoldást kapunk. Ezeket a lépéseket pedig addig ismételjük, amíg meg nem találjuk a tökéletes harmóniát, az optimális megoldást [S12]. Az utóbbi állításokból látszik, hogy az algoritmus a memóriával ellátott algoritmusok közé tartozik és majd a későbbi állításokból kiderül, hogy populáció alapú ez is, mint az össze többi algoritmus, amellyel dolgozok.

A HS algoritmus 5 lépésből áll:

1. Meg kell adni az optimalizálási problémát és az algoritmus paramétereit.
2. El kell készíteni a Harmónia Memória mátrixot (HM).
3. Meg kell alkotni (improvizálni) egy új harmónia változatot.
4. Be kell illeszteni az új harmóniát a Harmónia Memória mátrixba.
5. Ismételni kell a 3. és 4. lépést, amíg el nem érjük a tökéletes harmóniát.

A fenti folyamatot mutatja a 24. ábra:



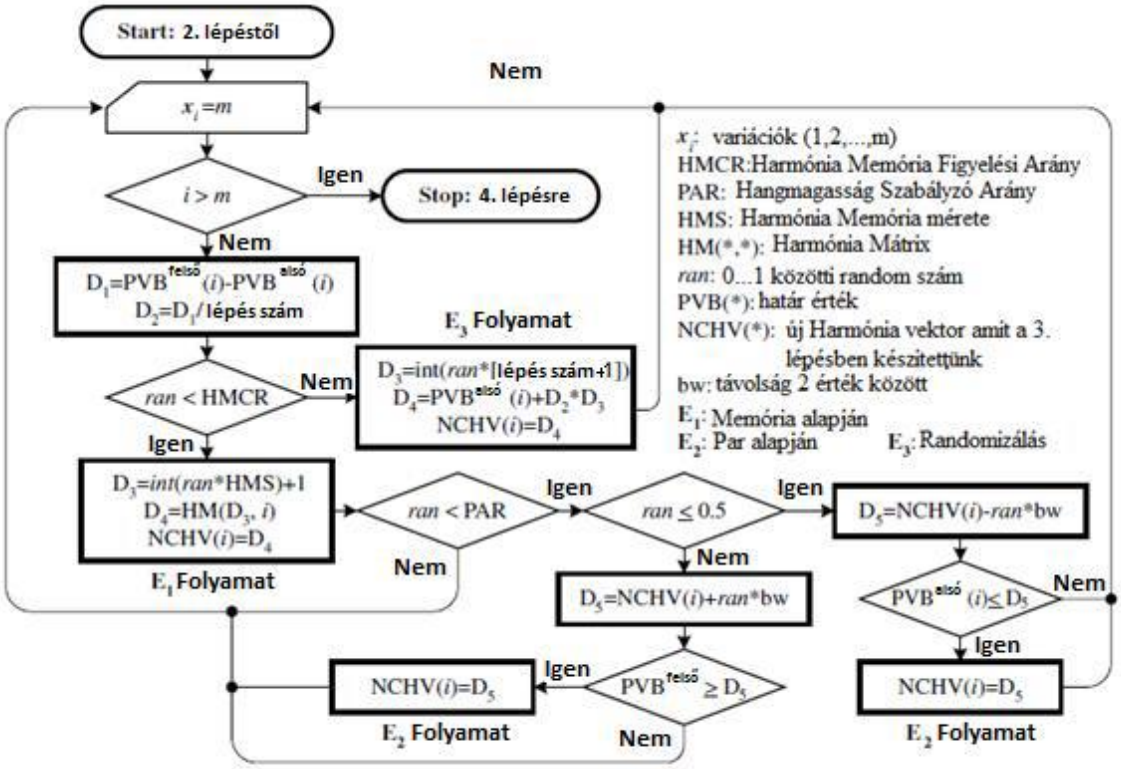
25. ábra Harmony search algoritmus leírása

A 25. ábrából nem derül ki, de a lelke az egész algoritmusnak a 3. és a 4. lépésben az új harmónia, azaz megoldás generálása [I23]. Az ellenőrzéskor megnézzük, hogy a legrosszabb megoldásunknál jobbat sikerült-e generálnunk és ha igen akkor megtartjuk és felülírjuk a legrosszabbat. Az új változat generálására pedig 3 módszert ismertet a Harmony Search algoritmus:

- Generáljunk egy új teljesen független random változatot.
- A már mátrixban lévő változatok értékeiből válogassunk össze egy új változatot.
- A 2. módszernél kapott értékeket változtassuk meg a „bw” arányt figyelembe véve.

DOI: 10.14750/ME.2020.018

Az alábbi ábrán a 3. lépés van kiemelve és ezen van bemutatva a 3 módszer, amellyel az algoritmus új értékeket generál.



26. ábra: Harmony Search algoritmus változó generálásának folyamata [I23]

A kutatómunkám és doktori tanulmányaim előtt a Harmony Search algoritmust tanulmányoztam és a logisztikai mesterképzésen diplomamunkámat is ebből a területből írtam. Egy viszonylag egyszerűen megérthető algoritmusról van szó, amely nagyon sok helyen könnyedén módosítható és új funkciókkal bővíthető. Korábban két módosítást és egy új irányító változót implementáltam, annak érdekében, hogy logisztikai feladatokat gyorsabban és pontosabban tudjon megoldani. Ebben a témában két publikációt jelentettem meg: [S2] [S13].

5.4. Genetikus algoritmus

A genetikus algoritmus (GA) egy természet inspirálta módszer, melynek az alapja a természetes kiválasztódás. John Holland [I42] mutatta be először a genetikus algoritmus egy kezdetleges formáját 1960-ban, amely a Darwini evolúcióelméletet vette alapul, később a tanítványa David E. Goldberg 1989-ben fejlesztette tovább a már ma is ismert és használt alakjára. Erősen populáció alapú módszer, melynél egy populáció egyedei, melyek a megoldásokat jelentik, küzdenek a túlélésért. Azonban a túlélést egy egyed számára nem a fizikai és szellemi fittsége jelenti, mint a természetben, hanem a megadott feladat megoldásában mutatkozó eredményessége határozza meg.

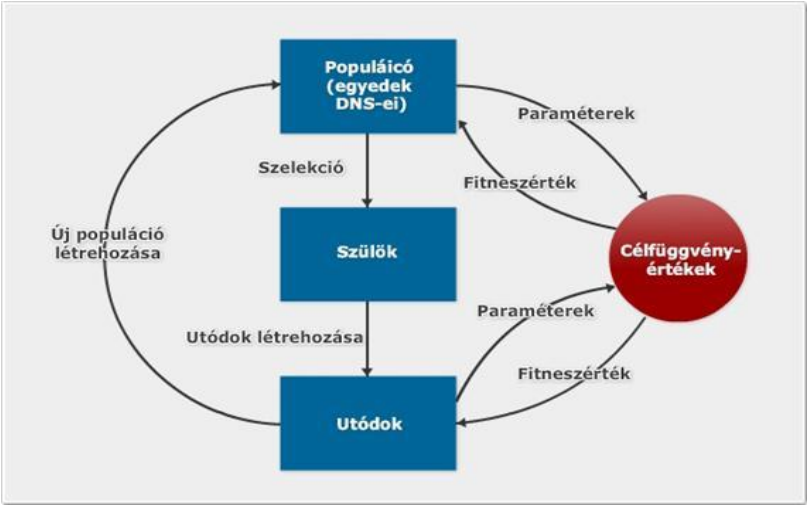
Minden egyed génekkel rendelkezik melyek a külső és belső tulajdonságaik határozzák meg, azonban számunkra a paramétereket és azok értékeit jellemzik. Az öröklődési folyamat során az utódok ezeket a géneket vagy génrészleteket fogják örökölni. Azon egyedek, melyek

DOI: 10.14750/ME.2020.018

jobban teljesítenek a feladat megoldásában, nagyobb eséllyel adják át a génjeiket. Ezáltal a jó képességű utódok aránya folyamatosan nőni fog a populáción belül és egyre jobban az optimális megoldás felé terelik a részeredményeket. Az egyedek közül gyengén teljesítők génállománya nem, vagy csak kis eséllyel öröklődik tovább, így előbb-utóbb a gyenge génállomány ki fog szelektálódni.

Genetikus algoritmus nem egy különálló metaheurisztikus algoritmus, hanem egy csoport olyan keresési technika, melyekkel optimumot vagy adott jellemzőjű, paraméterű elemet lehet keresni. A genetikus algoritmusok a speciális evolúciós algoritmusok csoportjába tartoznak, melyek módszerei a makro- és mikroevolúció alapjait használják fel [I45].

A populáció egyedeinek a paraméterei a keresési térben lehatárolt értékeket vehetnek fel, azonban vannak olyan GA változatok melyek képesek a keresési téren kívül is működni. Ezeket a paramétereket az öröklődési folyamat során lehet keresztezni, más szóval rekombinálni és mutáció is felléphet. Így képesek vagyunk új megoldásváltozatokat, azaz egyedeket létrehozni. A genetikus algoritmus célfüggvényének pontosan az a feladata, mint a már említett fittségnek, amely a valóságban olyan tulajdonságok összegét jelenti, melyek hozzájárulnak egy élőlény szaporodásához. Ezért is nevezik a GA célfüggvényét fitnesz függvénynek és az értékét fitnesz értéknek. Az összes egyed fitnesz értékét összehasonlítva határozza meg az algoritmus az egyedek szaporodási esélyét, amely annál nagyobb minél jobb a fitnesz értékük. Így az erősek nagy valószínűséggel továbbörökítik a jó tulajdonságaikat a gyengébben teljesítők génállománya pedig eltűnik a populációból. Ezzel a technikával oldja meg az optimumhoz való konvergálást a genetikus algoritmus.



27. ábra: Genetikus algoritmus működése [K19]

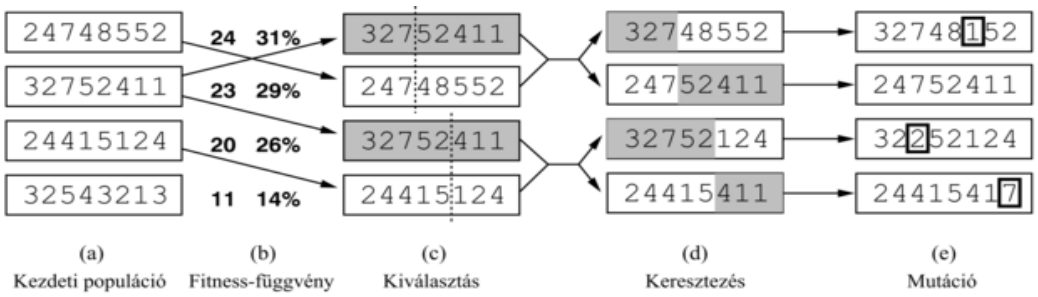
A genetikus algoritmusok működése (27.ábra):

1. **Kezdő lépés/Inicializáció**: A kezdeti populációt véletlenszerűen generáljuk a keresési tér határait figyelembe véve. A populáció mérete nagyban függ a probléma bonyolultságától és a paraméterek számától függ, de a legtöbben 100 és néhány 1000-es érték között szokták meghatározni.
2. **Szelekció/kiválasztás**: Minden generációban a populáció egy része, amely nagy részben a jobban teljesítő egyedekből áll, kiválasztásra kerül a szaporodásra. Legtöbbször a fitnesz érték alapján történik a besorolás, ahol a fittebb egyedek nagyobb

DOI: 10.14750/ME.2020.018

súlyokat vagy valószínűségi értékeket kapnak. Vannak olyan módszerek is, melyek minden egyedet felmérnek és kiválasztják a legjobbakat, de vannak olyanok is, amelyek véletlenszerűen kiválasztják a populáció egy részét és csak azokat nézik át. Az utóbbit akkor használják, ha túl sok időbe telni mindenkit megvizsgálni és a memóriaeffektust is szeretnének az algoritmusukba.

3. **Szaporítás/Örökítés**: Az egyedekből új (rész)generációnyi egyedet készítenek először a keresztezés műveletével, amelyhez két egyed genetikai tulajdonságai kellenek, majd ezekből készítik véletlenszerűen az új egyedet és mutációval, amely egyes tulajdonságokat véletlenszerűen kismértékben megváltozat. Ez utóbbi nem mindig működik és nem minden paraméteren, hanem véletlenszerű választás alapján. Ezt a 3. folyamatot mutatja be a 28. ábra egy példán keresztül.
4. **Leállás**: A metaheurisztikus algoritmusok, melyek iteratívak, mint a GA, addig futnak, míg a leállítási feltétel nem teljesül. A leggyakrabban egy bizonyos előre meghatározott generációszám után megállítják őket, vagy ha már több generáción keresztül nem javul az eredmény.



28. ábra Genetikus algoritmus bemutatása példán keresztül [K19]

A genetikus algoritmus egy nagyon sokrétűen felhasználható algoritmus. A legtöbb problémára gond nélkül alkalmazni lehet, többek között a logisztikában megjelenő problémákra, melyekről már szó esett. Ez az egyik leggyakrabban használt heurisztikus optimalizálási eljárás. Mind a kutatók, mind a programozók szívesen használják, mert viszonylag egyszerű vele dolgozni és gyorsan pontos eredményt ad. Én arra használom, hogy egy összehasonlítási alapot adjon a többi algoritmushoz, amelyet használok és kutatok.

DOI: 10.14750/ME.2020.018

6. HEURISZTIKUS ALGORITMUSOK TESZTELÉSE

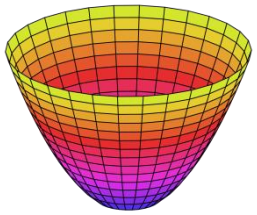
Az algoritmusok teszteléséhez egy háromlépcsős folyamatot használok:

1. Tesztfüggvényekkel való tesztelés
2. Logisztikai alapproblémákon való tesztelés
3. Újonnan megalkotott logisztika rendszert leíró modellen való tesztelés

A tesztfüggvényekkel való tesztelést nagyon sok kutató végzi és készít tanulmányokat róla. Emellett rengeteg kiváló tesztfüggvény létezik meghatározott optimumpontokkal. Ezek a függvények nem csak azért jók, mert könnyen lemérhető velük több algoritmus teljesítménye, amely jelen esetben, a gyorsaságban és pontosságban számít a legtöbbet, hanem mert megtudható, hogy mennyire egyszerű vagy bonyolult őket alkalmazni és programozni. Léteznek olyan tesztfüggvények is, melyek több lokális minimumponttal rendelkeznek egymáshoz nagyon közel. Ezekkel tesztelhető egy algoritmus robosztussága és lokális minimumpontba beragadási hajlama. Emellett még megemlíteném, hogy a legtöbb tesztfüggvény valós értékű paramétereket használ, amelyekkel a legegyszerűbb dolgozni.

Ezekből a tesztfüggvényekből szeretném bemutatni azt a párat, amelyet fel szoktam használni a legelső tesztelési fázisomhoz [K20]:

1. Az egyik legegyszerűbb tesztfüggvény a kétismeretlenes parabolikus függvény (44) melynek a minimumpontja f(0,0)=0 értékeknél van. A szöveg melletti ábra mutatja a 3D-s kinézetét.



29. ábra: Parabola függvény

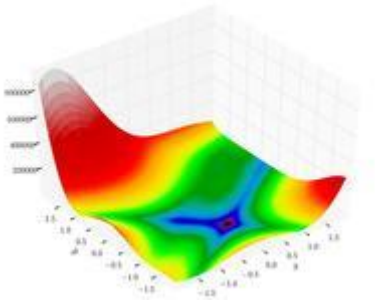
$$x^2 + y^2 \rightarrow min. \tag{44}$$

2. A következő tesztfüggvényt akkor alkalmazom, ha nagyszámú paraméterre akarom tesztelni egyszerűen az algoritmusokat. Ez a függvény az n-dimenziós parabolikus függvény. A képlete a következő:

$$\sum_{i=1}^{n} x_i^2 \rightarrow min. \tag{45}$$

DOI: 10.14750/ME.2020.018

3. A Goldstein-Price függvény egy globális minimumponttal f(0,-1)=3 ellátott függvény. A képlete a következő:

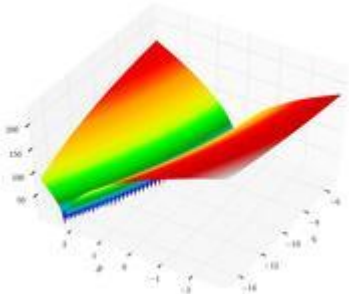


30. ábra: Goldstein-Price tesztfüggvény

$$\{1 + (x + y + 1)^2 * (19 - 14x + 3x^2 - 14y + 6xy + 3y^2)\} * \{30 + (2x - 3y)^2 * (18 - 32x + 12x^2 + 48y - 36xy + 27y^2\} \rightarrow min \quad (46)$$

.

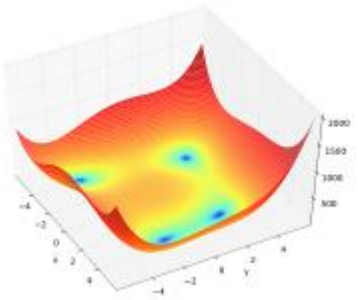
4. A következő tesztfüggvény lényege, hogy egy egyenes vonal mentén a függvény tele van lokális minimumpontokkal és van egy globális minimumpontja (-10,1)=0 –ban. A Bukin No.6 függvény képlete a következő:



31. ábra: Bukin No.6 tesztfüggvény

$$100 * \sqrt{|y - 0.01 * x^2|} + 0.01 * |x + 10| \rightarrow min. \quad (47)$$

5. Egy általam nagyon kedvelt tesztfüggvény a Himmelblau függvény, amelynek négy jól elkülöníthető globális minimumpontja van, amelyek 0 értéket vesznek fel. A képlete:

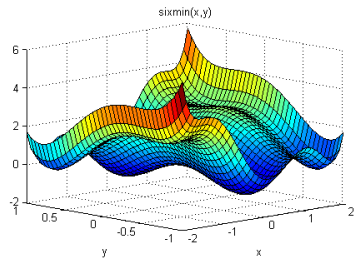


32. ábra: Himmelblau függvény

$$(x^2 + y - 11)^2 + (y^2 + x - 7)^2 \rightarrow min. \quad (48)$$

DOI: 10.14750/ME.2020.018

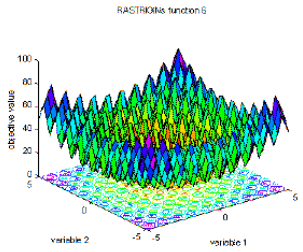
6. Egy matematikusok által eléggé ismert függvény következik, amely a hatpúpú teve függvény nevet viseli és a hárompúpú továbbfejlesztett változata, csak ennek 2 optimumpontja van.



33. ábra: Hatpúpú teve függvény

$$\left(4 - 2{,}1x^2 + \frac{x^4}{3}\right) * x^2 + xy + (-4 + 4y^2) * y^2 \rightarrow min. \quad (49)$$

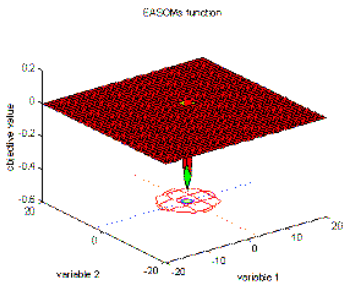
7. Rastrigin függvénye egy nagyon intenzív felületű függvény, amelyben sok egyszerű algoritmus nem képes megtalálni a minimumpontot.



34. ábra: Rastrigin függvény

$$f(x) = 10 * n + \sum_{i=1}^{n} x_i^2 - 10 * \cos(2 * \pi * x_i)) \; ; \; ahol$$
$$-5{,}12 \leq x_i \leq 5{,}12 \quad (50)$$

8. Easom tesztfüggvénye egyedi darab, ugyanis egy szinte teljesen síkfelületről van szó, viszont a minimumpontnál mintha egy ceruza nyomta volna be a síkfelületet. Az a függvény olyan algoritmusoknak lehet nagy kihívás, melyek nem képesek jól feltérképezni a területüket. A minimumpontja, mint szinte az összes többinél a (0,0)-ban található.



35. ábra: Easom tesztfüggvény

$$-\cos(x) * \cos(y) * e^{-((x-\pi)^2+(y-\pi)^2)} \rightarrow min \quad (51)$$

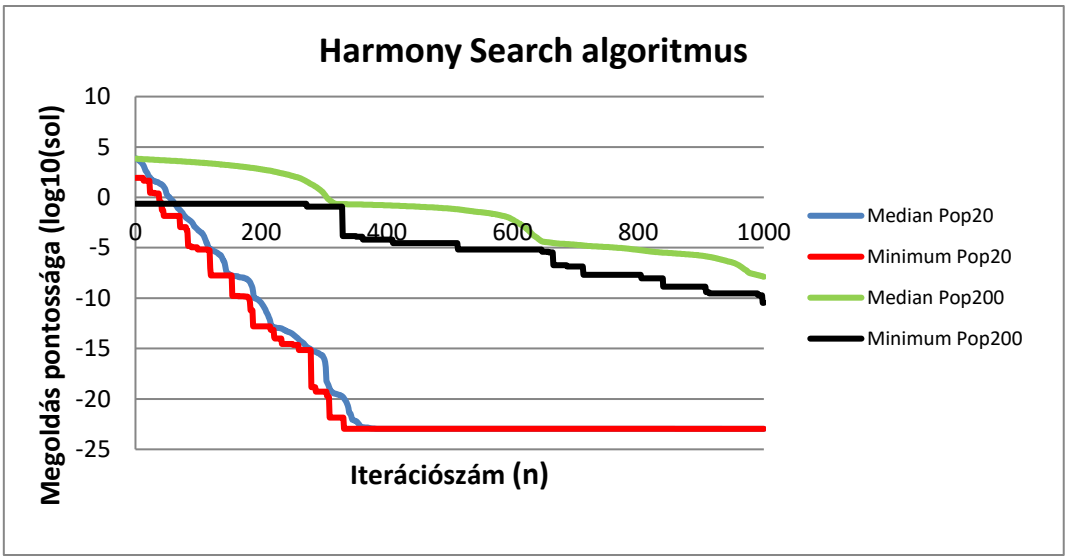
A tesztelés menete során többször lefuttatom az algoritmusokat és kiátlagolom az eredményt, ugyanis ezek az algoritmusok nagyon sokat dolgoznak véletlen értékekkel és random generátorral, amellyel minden egyes futtatáskor más értékeket ad ki.

Az alábbi futtatásokon (36. és 37. ábra) egy korábbi tanulmány eredményei láthatók, amelyben a Black hole algoritmust hasonlítottam össze a Harmony Search algoritmussal az előbb bemutatott tesztfüggvényeken. Az alsó két ábrán futtatások eredményei láthatók, amelyet a Goldstein-Price függvényen értek el. Jól látható, hogy a BH algoritmus nagyjából 400 iterációs szám alatt elérte a 10^{-320} –as maximális pontosságot 200-as populáció mellett, míg a

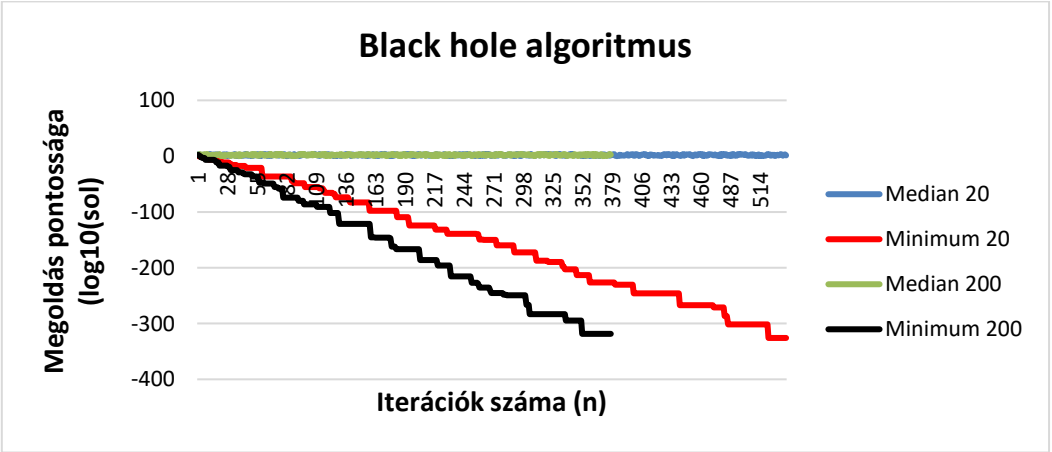
DOI: 10.14750/ME.2020.018

HS 20-as populációszámmal 350 iteráció után beállt a 10^{-23} körüli pontosságra. Csak szemléltetésképpen készítettem még egy futtatást a HS algoritmussal, 200-as populációszámmal, azonban az eredmény és a számítási idő is sokkal rosszabb lett, mivel egy nagyobb populáció egy egyszerű problémára nem tud gyorsan reagálni, idő kell míg minden egyed elkezd a jó irányba fejlődni.

Ami még megfigyelhető a két algoritmuson, az a medián. A Harmony Search algoritmusnál a medián szépen követi a legjobb értéket, ami itt a minimum. Ez annak köszönhető, hogy a HS egy iteráció új egyedeit a régiekből rakja össze, kevés változtatással. Azonban a Black hole algoritmusnál az átlag folyamatosan a kezdőérték közelében marad, mert amikor egy teljesen új egyed készül, akkor annak random értékei lesznek és ezek teljesen elviszik az átlagot.



36. ábra Harmony Search algoritmus tesztfüggvénnyel



37. ábra Black hole algoritmus tesztfüggvénnyel

A pontosságot az optimális megoldás és az ideális érték közötti különbség adja meg. és 10^{x} -en képlettel írható fel, ahol a 36. és 37. ábrán a függőleges tengelyről leolvasható általában negatív érték adja meg.

DOI: 10.14750/ME.2020.018

Logisztikai alapproblémákon való tesztelés

Logisztikai alapproblémának azokat a logisztikában előforduló feladatokat tekintjük, amelyeket a megoldáshoz nem tudunk további részfeladatokra bontani [K10].
Alapvető logisztikai problémák és feladatok:

- Járattervezés (utazóügynök probléma)
- Hátizsák feladat (kapacitás menedzsment)
- Centrumkeresés (lokáció meghatározás)
- Termelési mélység meghatározás (Make or Buy)
- Hozzárendelési probléma (szállítási feladat)
- Ütemezés

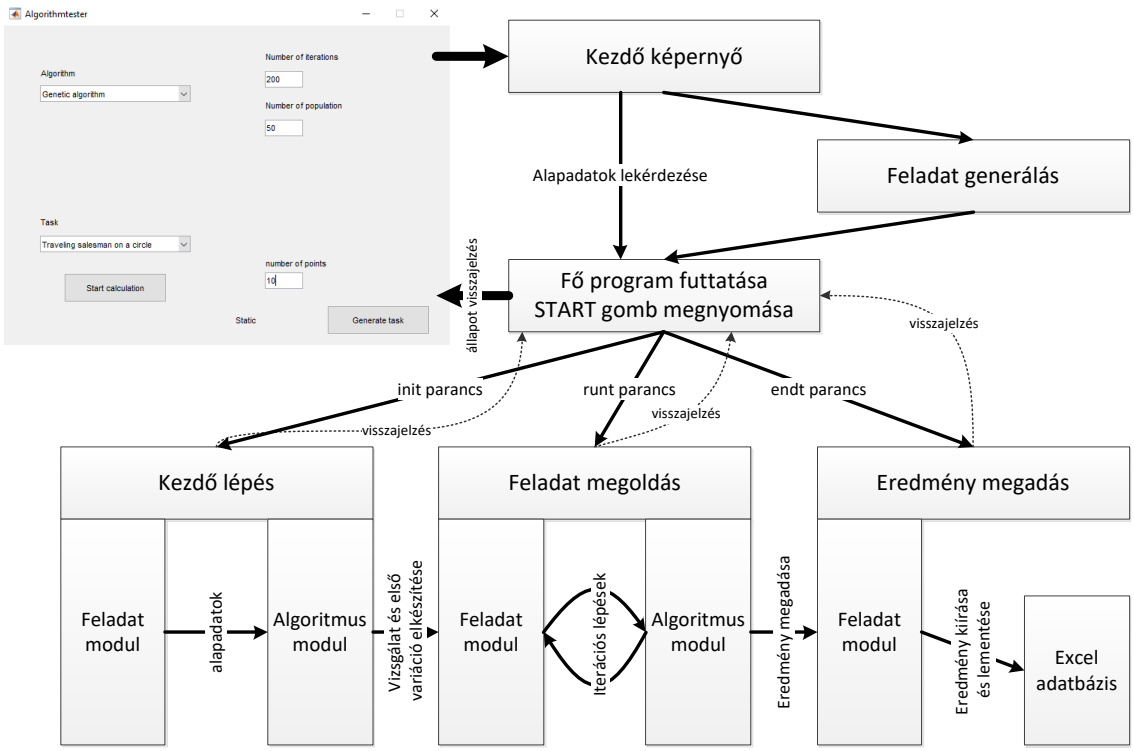
A legtöbb logisztikában fellelhető probléma ezekből az alapfeladatokból tevődik össze, néha többrétűen. Például a mai világban egy járattervezési feladatnál nem elég egy útvonalat meghatározni, amely a legrövidebb, hanem figyelembe kell venni az ütemezést és az egyes pontok időkapuit, amikor lehet azokhoz menni, a járművek mennyiségét és kapacitását, majd a feladatok szétosztását közöttük.

6.1. Algoritmustesztelő alkalmazás

Létrehoztam egy algoritmustesztelő alkalmazást, amely nagyban megkönnyíti a különböző függvények és egyszerűbb problémák tesztelését különböző algoritmusokon.

Az algoritmus tesztelő alkalmazás MATLAB programozási nyelven készült, amelynek több oka is van. Mivel ez nem egy piacra készült alkalmazás, ezért feleslegesnek tartottam, hogy olyan programozási nyelven írjam, amelyet a jelenlegi programozók használnak, mint a C#, Java vagy Python. Emellett ezeknek a nyelveknek és a fordítójuknak megvan az a sajátossága, hogy elkészítés után nagyon nehéz a forráskódjukba belenyúlni, márpedig egy olyan alkalmazásban, ahol modulok vannak és folyamatosan újabbakkal lehet bővíteni, a folyamatos felülírás elengedhetetlen. Ezenfelül a program elkészítéséhez nem volt szükség semmilyen különleges bővítményre vagy objektumorientált programozásra ahhoz, hogy elkészítsem az alkalmazást, nem is említve, hogy ezen elemek használatára a MATLAB is képes. Az egyetlen dolog, amit a választott nyelv megkövetel, hogy a MATLAB-ban készített alkalmazások csak és kizárólag a saját programozási környezetükben képesek futni optimálisan, amelyhez szükséges a MATLAB program [S8].

DOI: 10.14750/ME.2020.018



38. ábra Az algoritmus tesztelő alkalmazás működése

Az alkalmazás elindítása azonnal a grafikus kezdő képernyőt tölti be, amely a 38. ábra bal felső sarkában látható. A bal oldalt felső legördülő menüsorból lehet kiválasztani az algoritmust, amelyet vizsgálni akarunk, az alsóból pedig a feladatot, amelyen a vizsgálatot elvégezzük. A kezdő képernyő jobb oldalán lehet megadni az algoritmus leállításának kritériumát, jelen esetben egy iterációs számot, és alatta pedig a populáció méretét. Emellett látható még a kezdő képernyőn egy Generate task gomb és felette egy érték. Ez azért kell, mert nem minden esetben dolgozunk előre meghatározott feladatokkal. Ezzel a kis funkcióval elő tudunk állítani véletlenszerű vagy módosítható feladatokat. Utóbbira példa az utazóügynök feladat azon verziója, ahol egy körön egymástól egyenlő távolságra helyezkednek el a pontok és meg kell határozni a legrövidebb utat, amellyel minden pontot érintünk. Számunkra egyértelmű, hogy a legrövidebb út a kör kerülete mentén található, viszont az algoritmusok ezt nem ismerik fel és a pontok számának növelésével számítási teljesítményigény is exponenciálisan nő [S8] [S14].

Algoritmus tesztelő alkalmazással történő tesztelés

A fent említett alkalmazást azért készítettem, hogy egyszerűbben és több mindent tudjak tesztelni különösebb átállási idő nélkül. Ennek néhány eredménye a 6. 7. és 8. táblázat és a hozzá kötődő ábrák és grafikonok, amelyek a 5. heurisztikus algoritmusok fejezetben találhatók.

DOI: 10.14750/ME.2020.018

9. táblázat: Teszttáblázat egy része

			Populáció 300					
	változó típus	Algoritmus	**Pontosság (log10)**; Iteráció 1000			**Idő (s)**		
Test			GA	O-FF	M-FF	GA	O-FF	M-FF
	dec	2 változós parabolikus	-143	-97	-104	4,5683	2,9003	3,1336
	dec	10 változós parabolikus	-104	-69	-82	5,3259	3,4074	3,6481

Az 5. fejezetben található táblázatokból, amelynek egy szeletét beillesztettem ide 9. táblázatnak, jól látszik, hogy az algoritmusok pontosságára és futás idejére voltam kíváncsi, amelyeket általában két nagyságrendű populációmennyiségen is leteszteltem. A pontosság, úgy, ahogy korábban is 10-es alapú logaritmussal van kiírva, ami azt mondja meg (abszolút értékben) hogy a tényleges és a kiszámolt megoldás között hány nagyságrendi különbség van. Minél nagyobb a negatív szám annál jobb. A futásidő sokkal egyszerűbben meghatározható és értelmezhető, ugyanis másodpercben van megadva, hogy mennyi idő alatt végezte el az 1000 iterációt az alkalmazás a megadott beállításokkal. A táblázatokba már nem csak egyszerű tesztfüggvények kerültek, hanem esettanulmányok is, amelyek egyikét integráltam a tesztelő alkalmazásba. A többihez és azokhoz, amelyek nem részei ennek a disszertációnak teljesen különálló programokat kellett írnom.

DOI: 10.14750/ME.2020.018

7. LOGISZTIKAI MODELLEK

Ebben a fejezetben három egymástól teljes mértékben eltérő esettanulmányt mutatok be, melynek nem csak az a célja, hogy az előző fejezetekben átalakított algoritmusokat és kifejlesztett módszereket megvizsgálja, hanem hogy új perspektívát és alternatív megoldást mutassanak be bizonyos logisztikai feladatok elvégzésére. A három esettanulmányban olyan alapvető logisztikai folyamatok optimalizálásán keresztül mutatom be az előző fejezetekben tárgyalt algoritmusok hatékonyságát, mint a vállalaton belüli és kívüli járattervezés, a többcsoportos centrumkeresés, vagy a kapacitástervezés. Ezen modellek mindegyike NP-hard feladatként jellemezhető. A bemutatásra kerülő modellekhez kidolgozott megoldási módszerek már létező és általam átalakított algoritmusok. Elsőként bemutatásra kerülnek azon szabályok, melyek alapján a különböző modellek kialakításra kerülnek.

7.1. Általános modellalkotás

Az első és talán legfontosabb elv a kimeneti érték egységesítése. Ugyan nem SI mértékegység, mégis logisztikai modellekben talán a leggyakrabban alkalmazott a költség. A mai világban mindennek van értéke, azonban nagyon sok dolognak relatív. Erre egy kiváló példa az idő:

Általános mondás, hogy az idő pénz. Azonban a konvertáláshoz ismernünk kell a váltót, azaz azt, hogy egységnyi idő mennyi „jószágra” cserélhető. Számításához többféle modell és elv is létezik [K21]:

1. Munkával arányos juttatás, amibe beletartozik a havi fizetés alkalmazottként, vagy az elvégzett munka mennyisége utáni kereset (fix konvertálási érték).
2. Piaci vagy egyéb környezeti behatásra érzékeny pénzmozgás (változó konvertálási érték).
3. Nem, vagy nehezen mérhető fizikai vagy szellemi termékek értéke (becsléssel és spekulációval előállított érték, amely minimálisan alapszik paramétereken).
4. Az elhalasztott lehetőségek által vesztett javak (itt mindig költségről, vagy negatív pénzmozgásról beszélünk, célunk a lehető legkevesebbet veszteni).

A dolgozatomban a legkönnyebben és legtudományosabban alkalmazható fix vagy paraméterek által irányított konvertálási értékekkel dolgozok, amelyeket minden esetben reálisan megválasztok és kifejtek. A valóságban ezektől teljesen eltérő értékekkel is dolgozhatunk és dolgoznak is, ezért a modelleket úgy alkottam meg a különböző feladatokhoz, hogy a konvertálási értékek cserélhetők legyenek szükség esetén.

A kutatásaim nagy részében, amikor egy modellt alkottam vagy használtam fel egy probléma megoldására, a két logisztikai anyagmozgatási modell valamelyikéből indultam ki. A két modell a logisztika két nagy csoportját reprezentálja vállalati környezetben: a vállalaton belüli és vállalaton kívüli logisztika.

A belső logisztika, mint azt a neve is mutatja, a vállalaton belüli anyag, információ, eszköz és ember mozgását jelenti, amelybe eseteként beletartozhat maga az épület vagy terület logisztikai vonzáskörzete is, mint az épületenergetika vagy layout-tervezés.

DOI: 10.14750/ME.2020.018

A külső vagy external logisztika, a vállalaton kívüli anyagmozgatási tevékenységeket írja le, amelybe beletartoznak a megrendelések és feldolgozások, a szállítási távolságok, útvonalak tervezése, szállítási idők és időablakok meghatározása, gépjárműpark fenntartás vagy bérlés, kapacitáskihasználások és üresjáratok megléte. Ezeken felül ide tartozik a be- és kilépési pontok megtervezése és koordinálása, az intermodális szállítások megvizsgálása, jogi háttér tisztázása, áruvédelem és minőségellenőrzés.

Mindkét modellt úgy alkottam meg, hogy a végeredménye egy közös paraméter legyen, valamilyen költség vagy nyereség formájában.

7.1.1 Belső logisztika tulajdonságai

A belső anyagmozgatás sok apró lépésből és ezek koordinálásából tevődik össze. Egy gyártó vállalat nagy összegeket képes veszteni egy rossz belső logisztikai rendszer miatt, amelyet az esetek többségében nem is érzékel. A lépések átszervezésével, módosításával, összevonásával vagy kihagyásával egyre jobb rendszereket kaphatunk, melyek tesztelésére elsősorban szimulációt szoktak alkalmazni. Mivel itt a szállítási útvonalak kicsik, de rengeteg van belőlük és a kiegészítő tevékenységekből, ezért nem csupán a szállítás költsége lesz a mérvadó ebben a helyzetben, hanem a rakodás, manipulálás, raktározás és ellenőrzés költsége. A belső logisztikai modellnél az alábbi paramétereket kell figyelembe venni [K7]:

- szállítási távolságok,
- szállítási mennyiség,
- idő:
    - szállítási idő,
    - tárolási idő,
    - rakodási,
    - átállási idő,
    - komissiózási idő,
    - várakozási idő,
    - termékátfutási idő,
- belső hálózati csomópontok, és azok terhelése,
- raktár mennyisége, kihasználtsága, készlet forgási sebessége,
- termékféleségek és azok tulajdonságai,
- tárolóhelységek helye, kialakítása és kapacitása,
- munkahelyek elrendezése, megközelíthetősége,
- dolgozók létszáma és eloszlása,
- szállítóeszközök mennyisége, kapacitása és rendelkezésre állása.

Ezek a feltételek még az objektív kategóriába tartoznak, azaz viszonylag könnyen mérhetők vagy meghatározhatók. Ha tudunk minden egyes tulajdonságra egy egységárat mondani, akkor ki tudjuk számolni, hogy a teljes logisztikai rendszerünk mennyi összköltséggel üzemel.

DOI: 10.14750/ME.2020.018

7.1.2 Külső logisztikai tulajdonságok

A külső logisztika keretében a vállalat elsősorban a szállításra és annak megszervezésére koncentrál. A költségek nagy része a szállításból plusz az azt kiegészítő műveletekből és díjakból (adminisztráció, fuvarszervezés, díjak és tarifák) tevődik össze és sokkal könnyebb becsülni, mint a belső logisztikai tulajdonságokat.
A külső logisztikai modellnél a következő paramétereket kell figyelembe venni [K7]:

- szállítási távolságok,
- szállítási mennyiségek,
- jelenlegi szállítási költségtényezők (útdíjak, adó-vám, üzemanyag, stb…),
- idő:
    - szállítási idő,
    - tárolási idő,
    - időablak,
    - rakodási idő,
    - várakozási idő,
- sofőrök és szállítóeszközök mennyisége és rendelkezésre állása,
- lehetséges kombinált szállítások,
- üresjáratok elkerülése.

Ezek figyelembevételével általánosságba a következő nagyon egyszerű összefüggés fogalmazható meg, ahol a C_K a külső vállalati költségek összessége és a C_B a belső vállalati költségek összessége:

$$\sum C_K + \sum C_B \rightarrow min \tag{52}$$

A C_K és a C_B alatt a fejezetben felsorolt összes tényező költségvonzatát kell venni, mint például szállításnál:

$$C_K = \sum L_i * C_{sz} + \sum C_A + \sum C_{UD} + \sum C_{SF} * t_i + \sum C_E \tag{53}$$

ahol a L_i szállítási távolságokat, a C_{sz} átlagos szállítási költséget, a C_A adminisztrációs költséget, a C_{UD} útdíjakat és a C_{SF} a sofőr óradíját és t_i azállítási időt, majd a C_E az egyéb költségeket jelenti.

DOI: 10.14750/ME.2020.018

7.2. Integrált járattervezési feladat megoldása

A legtöbb kis- és középvállalat, amely rendelkezik valamilyen járműflottával, nem tudja teljes mértékben kihasználni szállítóeszközeit. Ez a fajta probléma, az erre specializálódott vállalatoknál is jelen van, ahol a járművek a megtett útjuk 20-30%-ban üresen, az esetek újabb 25-35%-ban pedig részben megrakottan futnak [K22]. A kisebb vállalatoknál ez még nagyobb problémát jelent, ahol csak minimális személyzet látja el a szállítmányozási feladatokat és járatok megszervezését, ezért rengeteg olyan járatot indítanak, amely részben vagy teljesen üresen indul el, vagy tér vissza telephelyére. Ha használnak is szoftvereket a járatok megszervezésére, azok csak alapszinten tudják lekezelni a járműfutásokat, mivel a legtöbb ilyen szoftver csak a járat indítása utáni nyomkövetésre képes, és néhány paramétert vizsgál. Ezek a szoftverek nem képesek beavatkozni és a kommunikáció is csak egyirányú. Léteznek komoly és speciális járattervező és kezelő szoftverek is, amelyeket elsősorban szállítmányozásra szakosodott vállalatok használnak, azonban ezek nagyon drágák. Ezenkívül még nem találkoztam olyannal, amelyik automatikusan tudná felvenni a megrendelést és útvonalba integrálni vagy újratervezni a már mozgásban lévő járművek útvonalait. Ehhez a megoldáshoz legjobban az internetes kapcsolattal ellátott GPS-alapú szoftverek hasonlíthatók, amelyek probléma esetén, mint pl: útépítés, forgalmi dugó, vagy megváltozott útszakaszok, képesek az útvonalat módosítani a lehető legkisebb változtatás alkalmazásával. Ennek oka, hogy egy szállító járműnek az előre kijelölt útvonalon kell közlekednie hatósági engedéllyel. Kisebb vállalatoknál bármilyen változtatás csak manuális beavatkozással lehetséges.

Már léteznek olyan megoldások hálózat-alapú alkalmazások formájában, melyek több vállalkozás, virtuális vállalat vagy egy klaszter teljes járműparkját foglalják magukba, viszont ezek nagy részénél a megbízó maga választja ki, hogy ki és mikor fogja a szállítást végezni, a rendszer pedig csak lehetőségeket kínál fel.

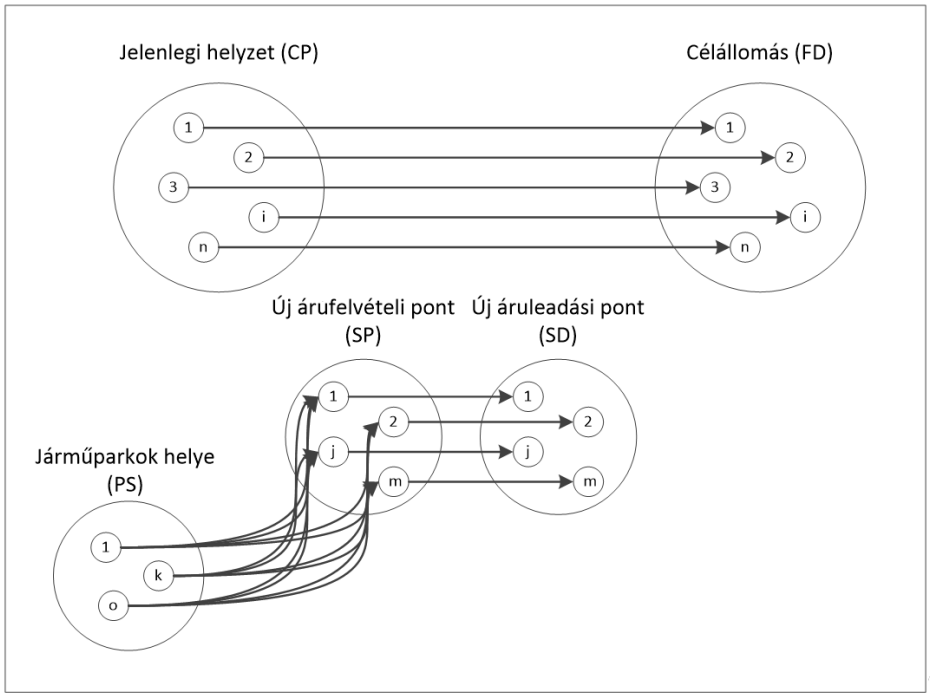
Ma a legnagyobb elektronikus piactér a szállítmányozóknak a TIMOCOM. Ez egy olyan webes felülettel ellátott adatbázis, melyre közel 40.000 vállalkozás adja fel a szállítási feladatait, mint az elszállítandó árumennyiséget és helyszínt, majd árajánlattal lehet licitálni arra, hogy ki és mennyiért szállítja el azokat, az adott helyről, helyre és időpontra. A piactérre elsősorban olyan feladatok és áruk kerülnek fel, amelyek speciális szállítást igényelnek, vagy nem rendelkeznek szerződött partnerekkel azért, mert változékony az elszállítandó mennyiségük vagy keveset gyártanak.

Ezen fejezet célja bemutatni a fent említett új funkciót ezeknek a rendszereknek a kibővítéséhez, mellyel hatékonyabbá és kifizetődőbbé lehetne tenni ezeket a szállításokat.

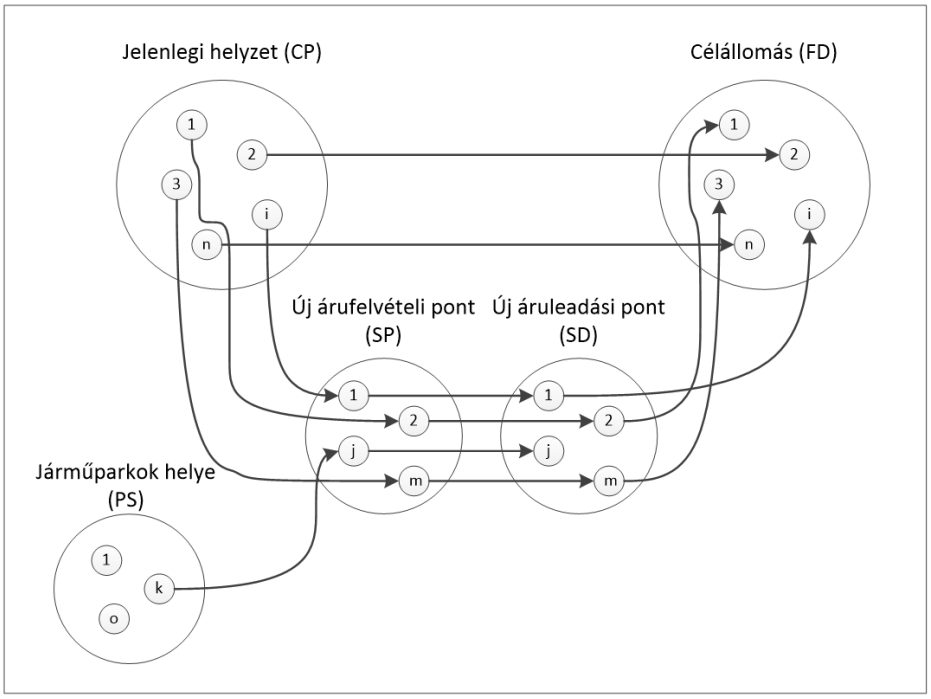
7.2.1. Új járattervezési és szervezési irány

Ahhoz, hogy egy ilyen funkciót lehessen beépíteni, először szükség van egy adatbázisra, mely képes kell legyen kezelni minden jármű álló vagy mozgás közbeni paraméterét. Jelenleg ezek a rendszerek csak olyan adatokat tartalmaznak, mint a rendelkezésre állás, minimális követés, kapacitás és technikai tulajdonságok. Annak érdekében, hogy működjön a rendszer, a járművekbe épített helymeghatározó rendszernek folyamatos kapcsolatban kell állnia a rendszerrel kétirányú kommunikációs lehetőséggel. El kell tudnunk érni a járművek pontos helyét és legfőképpen szabad kapacitását. Utóbbi azért szükséges, mert egy megfelelő szabad kapacitással rendelkező jármű, amely egy lehetséges árufeladó pont közelében jár (40. ábra), nagy valószínűséggel sokkal jobb áron el tudja szállítani az árut, mint egy olyan jármű, amit ennek az árunak a felvevésére küldenének ki (39. ábra) [S5]. Ez azonban nem minden esetben tud teljesülni, mivel rengeteg feltételhez van kötve, melyek később kerülnek bemutatásra.

DOI: 10.14750/ME.2020.018



39. ábra: Jelenlegi járatszervezési módszer új áruk felvételére



40. ábra: Javasolt járatszervezési módszer új áruk felvételére

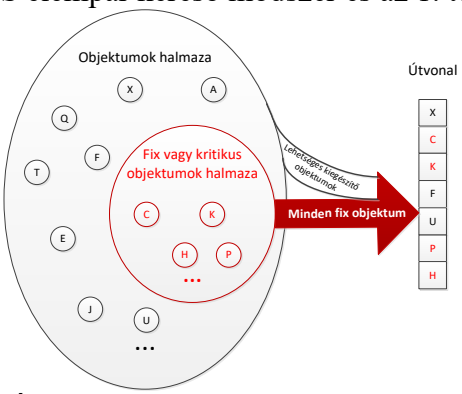
A logisztikai folyamatokban költségmegtakarítás érhető el abban az esetben, ha olyan algoritmusok és bővítmények integrálhatók ezekbe a rendszerekbe, melyek révén lehetséges a járművek követése és a megrendelések feldolgozása, majd a járművek és a megrendelések összerendelése úgy végezhető el, hogy a szállítási útvonalakra vonatkozó előírások nem sérülnek.

DOI: 10.14750/ME.2020.018

7.2.2. Az új módszer matematikai modellje

Ebben az alfejezetben a már említett célfüggvények és korlátozások bemutatása következik. Ezen függvények és vonatkozó korlátozások esetében számítható az optimális megoldás az optimalizáló algoritmus segítségével a járatok módosítására vonatkozóan.

A 41. ábrán jól látszik, hogy a lehetséges járatokat, hogyan generálja az algoritmus fix (már megtervezett és elindult járat) pontokból és opcionális (lehetséges szállítási) pontokból. Ezek mérete és felépítése eltérhet. Amikor összehasonlítunk két útvonalat, csak a végeredményeiket tudjuk. Ez nem elég egy algoritmusnak, ugyanis a legtöbbnek szükségük van arra is, hogy a két megoldás milyen távol áll egymástól „fizikailag". Ezt a távolságot sorba rendezett elemhalmazokban (szekvenciák) cseretávolságnak hívjuk, amelyet a 4. fejezetben tárgyal az értekezés. Ezt a távolságot nehéz meghatározni, főleg akkor, ha nem ugyanazokból az elemekből épül két szekvencia, ezért létrehoztam egy új módszert ezen cseretávolság mérésére, melynek neve: SPS elempár kereső módszer és az 1. tézis tartalmazza[S5].



41. ábra: Útvonal felépítése fix és opcionális pontokból

A matematikai leíráshoz egy heurisztikus algoritmusnál szükség van célfüggvényre és korlátozásokra. A matematikai modellben a 10. táblázatban szereplő jelöléseket használom.

10. táblázat: Jelmagyarázat

CT_1	eredeti szállítási költség
CT_2	új jármű szállítási költsége
CO_s	s. rendelés-feldolgozás költsége az új járműnek
CP_t	t. szállítás utáni profit
L_{FDi}^{CPi}	távolság az i. jármű és az i. célállomás között
L_{SPj}^{PSk}	távolság a k. járműpark és a j. új felvevőpont között
L_{SDj}^{SPj}	távolság a j. felvevőpont és a j. leadópont között
L_{SPj}^{CPi}	távolság az i. jármű és a j. felvevőpont között
L_{FDi}^{SDj}	távolság a j. leadópont és az i. célállomás között
n	eredetileg elindult és úton lévő járművek száma
m	útközben beérkezett új megrendelések száma
o	járműparkok száma
q	új megrendelések kielégítésére indított új járművek száma
θ_i	azon járműveket halmaza, amelyet el tudják látni az i. szállítási feladatot
$CAP_{Ti}^{V}/CAP_{Ti}^{W}$	i. jármű térfogat- vagy tömegkapacitása
V_{TGi}/W_{TGi}	i. járműben már elfoglalt térfogat vagy tömeg
V_{TDj}/W_{TDj}	j. megrendelés térfogat- vagy tömegigénye
T_B^A	A-ból B-be való eljutásidőszükséglete
TC_i	aktuális idő az i. járműnél
TR_i	i-edik jármű célállomásának határideje

DOI: 10.14750/ME.2020.018

A célfüggvény (54) megadja, hogy mennyi lesz a megtakarításunk a különböző variációk generálásakor. Mivel mindig a legnagyobb megtakarítás a cél, ezért azt a lehetőséget keressük, ahol a szállítási költségek különbsége az 39. és 40. ábrán felvázolt módszer között a legnagyobb. Ezekhez még hozzáadódik a rendelés-feldolgozási költség és az extra profit lehetősége. A (55) és (56) képlet definiálja a járatok és járművek mennyiségét és szabályozza a változók értékkészletét.

$$CS = \sum_{l=1}^{p}\left[CT_1 * L_{FDi}^{CPi} + CT_2 * \left(L_{SPj}^{PSk} + L_{SDj}^{SPj}\right)\right] - \\ -CT_1 * \left(L_{SPj}^{CPi} + L_{SDj}^{SPj} + L_{FDi}^{SDj}\right) + \sum_{s=1}^{q} CO_s + \sum_{t=1}^{m-q} CP_t \rightarrow max. \tag{54}$$

$$i = 1 \dots n\,; j = 1 \dots m\,; k = 1 \dots o \tag{55}$$

$$0 \leq q \leq m\,;\; n \leq p = n + q \leq n + m \tag{56}$$

A (57) és (58) képlet két nagyon fontos szabályzás, amelyet még a változatkészítés előtt meg kell ejteni, ugyanis ez határozza meg, hogy egy járműbe több árut nem lehet belepakolni, mint amennyi belefér (kapacitáskorlát). Erre azért van szükség még a változatkészítés előtt, hogy azok a járművek, amelyekbe nem fér bele az árumennyiség, azok kiessenek a választhatók közül és sokkal kevesebb lehetőségből kelljen az algoritmusnak döntenie.

$$CAP_{Ti}^{V} - V_{TGi} \geq \sum_{r \in \theta_i} V_{TDrj} \tag{57}$$

$$CAP_{Ti}^{W} - W_{TGi} \geq \sum_{r \in \theta_i} W_{TDrj} \tag{58}$$

A (59) képlet szintén szabályzásra való, de csak a variációkészítés után tudjuk megvizsgálni. Ez a képlet a határidőket veszi figyelembe, miszerint a már elindult járműnek időre kell a célállomásához érnie, viszont, ha kitérőt tesz a jármű és elszállít egy másik árut egy másik helyre, úgy sem lépheti át az eredeti cél határidejét (időkorlát). Ha ez a lehetőség fennáll, a jármű nem viheti el azt az árut.

$$TC_i + T_{SPj}^{CPi} + T_{SDj}^{SPj} + T_{FDi}^{SDj} \leq TR_i \tag{59}$$

A járattervezési feladatnál a három legfontosabb tényező az idő, a tömeg- és térfogatkapacitás. A valóságban ennél sokkal több paraméter és korlátozó tényező befolyásolja a járattervezés feladatát, azonban az algoritmus futását ezek jelentősen nem befolyásolnák, csak a modellt tennék komplexebbé és nehezebben érthetővé.

DOI: 10.14750/ME.2020.018

7.2.3. Tesztprobléma

A fentiekben bemutatott modell validálását az alábbi tesztfeladaton keresztül mutatom be. Legyen adott egy olyan tervezési probléma, amely 9 éppen haladó járművet és 5 új megrendelést tartalmaz. A járműveknél a jelenlegi koordináta, a cél-koordináta és a szabad hely az alapadat. Az új megrendeléseknél a felvételi és leadási pont koordinátája, a várható plusz nyereség és a szükséges kapacitás meg van adva; azonban az időkorlátokat jelenleg nem számoljuk, de azok kritériumként bármikor implementálhatóak a modellbe. Az útvonalakat az egyszerűbb számítás és a modell áttekinthetősége érdekében nem térképekről kérdeztük le, hanem légvonalbeli távolságként értelmezzük, azonban a valós szállítási távolságok implementálása szintén egyszerűen elvégezhető. Ezt a valóságban úgy lehet orvosolni, hogy egy útvonalkészítő szoftverrel kombináljuk, amely minden járattervező szoftverben integrálva van, vagy valamilyen online térképet használunk lekérdezésre, amely szintén lehet automatikus. Az OpenStreetMap erre kiváló lehetőségeket kínál. Emellett még tudnunk kell, hogy a szállítási költség kilométerenként, beleszámítva az üzemanyagot, a sofőr fizetését, az útdíjakat és adminisztrációs költségeket 10 EUR/km. Az alábbi táblázat tartalmazza az alapadatokat:

11. táblázat: Jármű alapadatai

Járműszám	Jelenlegi járműkoordináta		Jármű célkoordinátája		Rendelkezésre álló kapacitás (t)
	Szélességi	Hosszúsági	Szélességi	Hosszúsági	
1	45,706	20,165	45,863	22,028	3
2	46,128	20,009	47,208	20,017	8
3	45,264	21,548	48,821	20,435	3
4	45,701	21,684	46,478	20,581	2
5	46,444	20,153	46,327	22,202	9
6	48,255	20,645	48,833	20,141	8
7	46,133	22,369	45,924	20,337	7
8	46,562	20,098	46,778	20,367	8
9	48,885	20,061	46,956	20,183	6

12. táblázat: Új szállítási feladatok adatai

Szállítási feladat száma	Áru felvételi koordinátája		Áru leadási koordinátája		Szállítási mennyiség súlya (t)	Szállításból várható profit (EUR)
	Szélességi	Hosszúsági	Szélességi	Hosszúsági		
1	48,103	20,142	47,722	20,238	5	1000
2	48,943	20,005	48,459	20,315	4	2000
3	48,356	20,629	48,707	21,049	7	1800
4	48,676	21,403	48,829	21,758	8	1500
5	47,723	21,998	47,908	22,022	2	1200

DOI: 10.14750/ME.2020.018

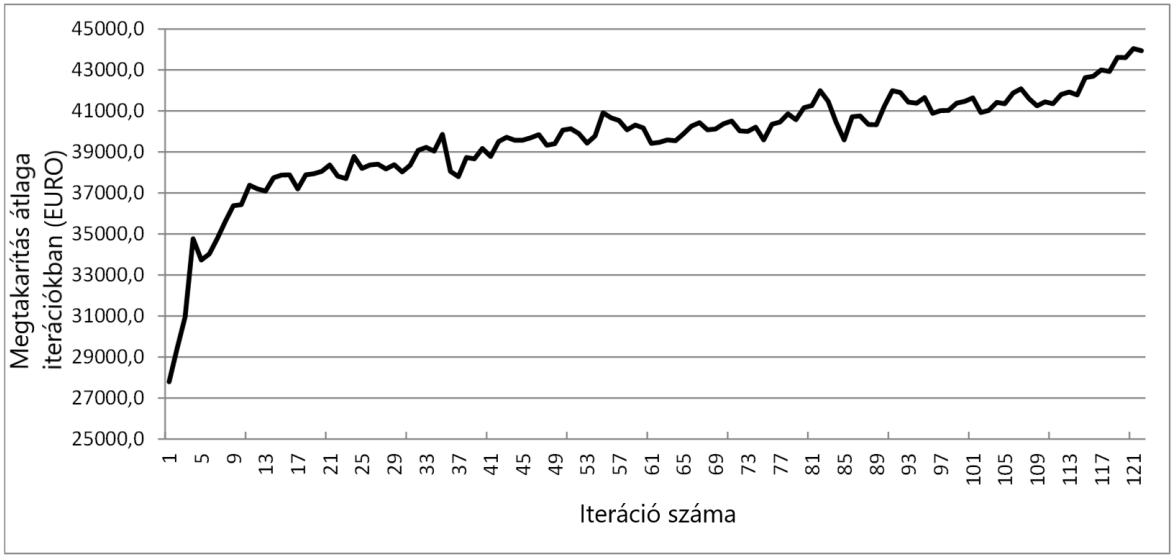
A modell alapján elkészített alkalmazást a Black-hole algoritmus MSEQ-BH (szekvenciális) módosított változatával oldottam meg, azonban elkészítettem a program genetikus algoritmussal integrált változatát is. Mivel a két algoritmus összevetése már be lett mutatva az 5.2.1.-es fejezetben, valamit a 6. és 7. táblázatban, így itt csak annyit említek meg, hogy az általam módosított algoritmus működik, viszont a genetikus algoritmus jobban teljesített nála.

A feladat megoldása a módosított BH algoritmussal a 13.táblázatban tekinthető meg.

13. táblázat: Feladat megoldása

Járműszám	Szállítási kapacitás (t)	Rendelkezésre álló kapacitás (t)	Hozzáadott szállítási mennyiség (t)	Kihasználtság (%)
1	20	3	-	85
2	25	8	-	68
3	20	3	-	85
4	30	2	-	93
5	25	9	-	64
6	15	8	7	100
7	20	7	-	65
8	20	8	-	60
9	25	6	4	40

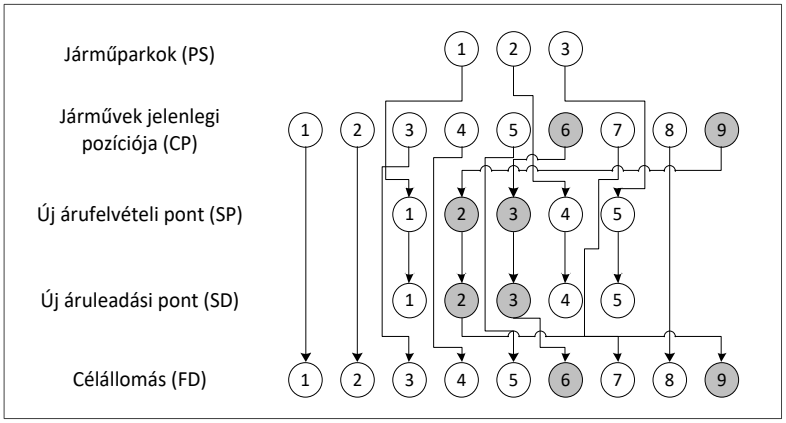
A szoftver és algoritmus a feladatot 4,3 másodperc alatt oldotta meg. Amint azt a 42. ábra is demonstrálja, az algoritmus 121 iterációból egy közel optimális megoldást talált a problémára és nagyjából 44000 Eurós megtakarítást eredményezett.



42. ábra: Optimalizálási folyamat konvergenciagörbéje

DOI: 10.14750/ME.2020.018

A 43. ábra a tesztfeladat eredményét mutatja. A 6. és 9. járművön kívül mindegyik tart a saját eredeti célja felé, a 6. és 9. jármű pedig a 2. és 3. új megrendelést viszi el még az eredeti megrendelésen kívül. A többi új megrendeléshez új járműveket rendelt a program.

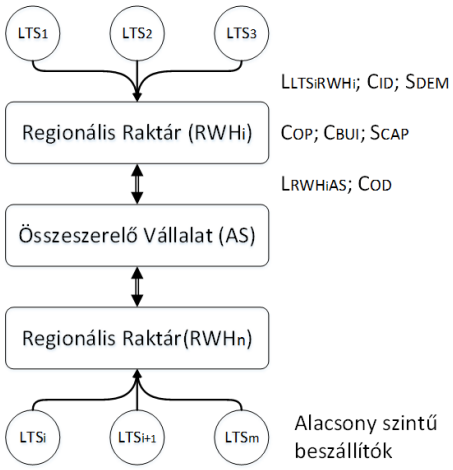


43. ábra: A feladat vizuális megoldása

DOI: 10.14750/ME.2020.018

7.3. Elosztási rendszer integrált tervezése

A negyedik ipari forradalom jelentős mértékben felgyorsította a beszállítói folyamatok fejlődését, hiszen a kiber-fizikai rendszerek megjelenése új perspektívát nyitott az együttműködések koordinálására. A beszállítói rendszerek fejlődése különösen az autóiparban és a mechatronikai összeszerelőiparban figyelhető meg, ahol a már gyakorlatnak számító just-in-time beszállítás mellett egyre nagyobb jelentőségre tesznek szert a just-in-sequence beszállítói megoldások. A stratégiai beszállítóknak általában megvan a jól kidolgozott rendszerük a beszállítás lebonyolításához. Azonban csak néhány nagyobb és fontosabb vállalat mondhatja el magáról, hogy stratégiai beszállító, az összes többi alacsonyabb szintűnek mindent maguknak kell megszervezni és lebonyolítani [I6]. A mai világban hatalmas előnyre tehetünk szert, ha másokkal kooperálva vagy csoportba összeállva együttes erővel tudjuk leküzdeni az akadályokat, jelen esetben a beszállítási problémákat. Olyan egymáshoz fizikailag közel elhelyezkedő vállalatokat kell összefogni, akiknek közös érdekeltségük van, így csökkenthetik a költségeiket és stabilabbá válhatnak a piacon. Ezek lehetnek klaszterek, virtuális vállalatok vagy keiretsuk, amelyekről már az 1. fejezetben szó esett.



44. ábra: Alacsony szintű beszállítók gyűjtőraktárakkal való összekapcsolása

7.3.1. Beszállító diszponálási rendszer célfüggvényei és paraméterei

Megbízható források kialakítása érdekében regionális raktára(ka)t hozhatunk létre a közeli beszállítók között, ahol a beszállítást, raktározást és időben való kiszállítást könnyebb szervezni. Ebben az esetben a szállítási költségek csökkenhetnek, nőhet a járatok kihasználtsága, szabályossá és megbízhatóvá válhatnak az összeszerelő vállalathoz beérkező szállítmányok. Másfelől viszont bérelni (konszignációs raktárak, logisztikai központok) vagy építeni kell ilyen raktárakat, amelyeknek van beszerzési és működtetési költsége is. Ezek miatt fontos meghatározni, hogy megéri-e létrehozni ezeket a raktárakat, és ha igen, hány darabot kell telepíteni és hova. A számítás költségalapú. A következő táblázat és célfüggvény tartalmazza és határozza meg azokat a paramétereket, amelyek alapján a későbbiekben bemutatott algoritmus és alkalmazás kiszámolja az optimális raktármennyiséget és helyeket.

DOI: 10.14750/ME.2020.018

$$TC_{TF} = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ID} \cdot 2 \cdot L_{LTS_i RWH_j} \cdot \frac{TF}{F_{LTS_i}} + \sum_{j=1}^{m} C_{OD} \cdot 2 \cdot L_{RWH_j AP} \cdot \frac{TF}{F_{RWH_j}} + j \cdot C_{Oper} \cdot TF + j \cdot C_{build} \rightarrow min \quad (60)$$

14. táblázat: Jelölések

Jelölés	Magyarázat
$\mathrm{LTS_i}$	i. alacsony rangú beszállító
$\mathrm{TC_{TF}}$	A futamidő alatti teljes költség
$\mathrm{C_{ID}}$	Költség az egyéni raktárba való beszállításokhoz
$\mathrm{C_{OD}}$	Szervezett szállítás költsége a raktár és az összeszerlő vállalat között
$\mathrm{C_{Oper}}$	Raktár működtetésének költsége
$\mathrm{C_{build}}$	Raktár építésének költsége
$\mathrm{L_{LTS_i RWH_j}}$	Úthossz a raktár és beszállítók között
$\mathrm{L_{RWH_j AP}}$	Úthossz a raktár és vállalat között
TF	Futásidő
$\mathrm{F_{LTS_i}}$	i. beszállító járatainak rendszeressége
$\mathrm{F_{RWH_j}}$	j. raktár járatainak rendszeressége
$SCAP_{RWH_i}$	Hozzárendelt raktár kapacitása
$SDEM_{LTS_i(arranged)}$	Beszállító raktári kapacitás szüksége
U	Termékek fontossága
LR	Úthossz
n	Beszállítók száma; i=1…n
m	Regionális raktárok száma; j=1…m

A célfüggvényt 4 részre tudjuk bontani. Az első rész számítja ki a beszállítóknak a raktárba való szállításhoz tartozó költségét a szállítás gyakorisága és a távolság alapján:

$$TC_{TF}^{1} = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ID} \cdot 2 \cdot L_{LTS_i RWH_j} \cdot \frac{TF}{F_{LTS_i}} \quad (61)$$

A második rész számítja a szervezett szállítás költségét a raktárból az összeszerelő vállalathoz:

$$TC_{TF}^{2} = \sum_{j=1}^{m} C_{OD} \cdot 2 \cdot L_{RWH_j AP} \cdot \frac{TF}{F_{RWH_j}} \quad (62)$$

A harmadik rész számítja ki a raktárak működtetési költségét. Jelenleg egy átlagos értékkel számoltam, amely tartalmaz minden általánosan felmerülő költséget pl.: személyzet, hűtés, fűtés, energiaellátás, adminisztráció, járművek költsége, biztonság, biztosítás és

DOI: 10.14750/ME.2020.018

legfőképpen a tárolt anyagok költsége. Ezt a költséget jelen modellben, mint lineáris függvényt veszem figyelembe a beszállítók és raktárak számának növekedése kapcsán:

$$TC_{TF}^{3} = j \cdot C_{Oper} \cdot TF \tag{63}$$

Az utolsó rész számítja a raktárak építésének költségét, ami egy egyszeri viszonylag nagy költség. Az utóbbit a megtérülés miatt kell belekalkulálni. Ezen kívül nem árt figyelembe venni a készletezési és kapacitás felügyeleti feladatokat is, de jelen esetben ettől eltekintünk:

$$TC_{TF}^{4} = j \cdot C_{build} \tag{64}$$

A beszállítók hozzárendelését a raktárakhoz a legközelebbi szomszéd módszerrel oldottam meg, amely ha más tulajdonságot nem veszünk figyelembe, a lehető legegyszerűbb és leghatásosabb módnak bizonyult. A módszer lényege, hogy megvizsgál minden hozzárendelendő elemet, hogy milyen távolságban van minden olyan objektumtól, amihez hozzá lehet rendelni. Ennél az esettanulmánynál nincs olyan tulajdonság, amely zavarná a módszer használhatóságát, így a távolságot a legegyszerűbben légvonalbeli távolságként kapjuk meg.

Minden újabb vizsgálandó tulajdonságnál azonban oda kell figyelni, hogy alkalmas-e ez a módszer a hozzárendeléshez. Az egyik ilyen kivételt erősíti, ha a kapacitást is figyelembe veszük. Ilyenkor az alacsony szintű beszállítóknak (gyártóegységeknek) van egy bizonyos termékkibocsátási időszakos mennyisége, amely lehet, hogy meghaladja a közelében lévő telepítendő raktár befogadó kapacitását. Ezt a problémát szakirodalomban csak hátizsák problémának szokták hívni. Ezzel a problémával bővebben a következő esettanulmány foglalkozik.

A túlságosan nagy kapacitásigény problémáját azonban meglehet oldani egyszerűen is házon belül a raktározási technika megváltoztatásával vagy átszervezésével. Erre a problémára két egyszerűen leírható, de sokkal nehezebben kivitelezhető megoldást is alkalmazni lehet:

- A raktár egy dinamikusabb raktározási technikát használ kezelésre és átfutásra, amihez egy fejlettebb rendszer is szükséges.
- Megosztja tárolandó termékeit egy másik közeli raktárral.

A második opció már a hátizsák optimalizálás feladat felső határait súrolja, amely egy újabb NP-hard probléma. Ennek a megoldásához van szükség a termékek fontosságára (prioritás), mennyiségére és kapacitások ismeretére, hogy egy optimális megoldást tudjunk számítani:

$$SCAP_{RWH_i} \geq \sum SDEM_{LTS_i(arranged)} \tag{65}$$

$$\sum \frac{U}{LR} \rightarrow max. \tag{66}$$

Ezt a feltételt és célfüggvényt felhasználva képesek vagyunk meghatározni a raktárak helyzetét. A feladat megoldására az eredeti Black hole algoritmust használtam. Később

DOI: 10.14750/ME.2020.018

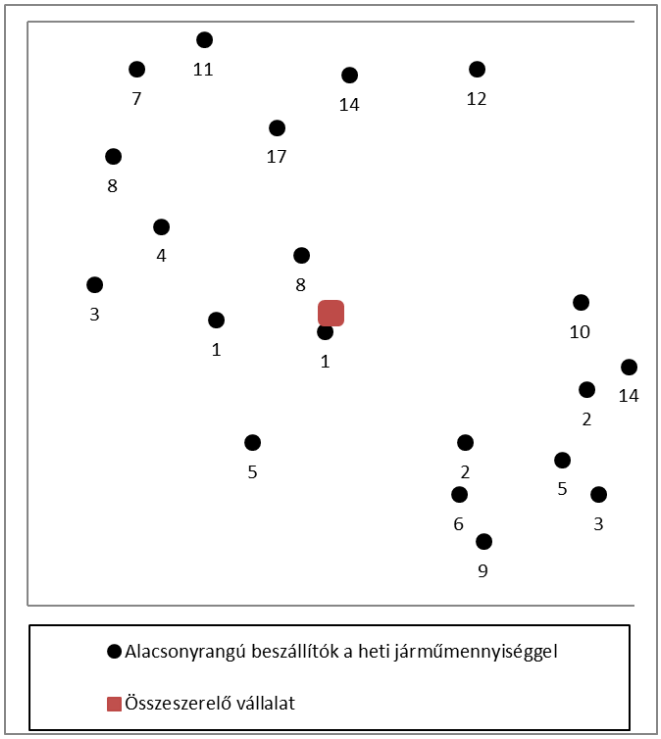
tesztelés céljából implementálva lett az eredeti genetikus algoritmus és a Black hole algoritmus általam továbbfejlesztett változata is, az MBS-BH.

7.3.2. Esettanulmány

A fent kidolgozott modellt és képleteket egy alkalmazásba foglalva teszteltem. Ennek két oka van. Egyrészt megvizsgálhatom, hogy a készített modell és képletek alkalmazhatók-e. Másrészt az algoritmusokat olyan feladaton validálhattam, amely nagy mennyiségű adat feldolgozását igényli és alapjában használható lenne ipari tervezésre.

Az esettanulmány során felvettem 20 beszállítót az összeszerelő vállalattól 50 km sugarú körön belül (45. ábra). Lefuttattam egy olyan szimulációt, amely megvizsgálta, hogy 1 és 20 közötti raktár optimális telepítése után milyen költségekkel üzemeltethető a teljes rendszer. Ebben az esetben a kapacitásokat, mint felső limitet nem vettem figyelembe. Minden beszállítónak megadtam, hogy hetente hányszor szállít és milyen mennyiséget (db teherautó/hét) és a szimulációt 3 évnyi időtartamig futtattam. Utóbbit azért választottam, mert ezt szokták megtérülési határértéknek tekinteni felfutóban lévő vállalatoknál. A további statikus paraméterek a következők voltak:

- Egyedi beszállítások átlagos költsége: 50 USD/szállítás
- Szervezett beszállítás átlagos költsége: 150 USD/komissiózott szállítás raktárból
- Raktár heti működési költsége: 3000 USD
- Raktár építésének költsége: 200000 USD



45. ábra: Összeszerelővállalat és beszállítók helyei, felettük a heti járműszám

DOI: 10.14750/ME.2020.018

15. táblázat: 1-10 raktár pozíciójának meghatározása

Raktárak száma		1	2	3	4	5	6	7	8	9	10
Teljes költség (M USD)		1,4484	1,4130	1,3014	1,4916	1,7209	1,9161	2,1778	2,3965	2,6190	2,8001
Raktár 1	X Koord	50,00	24,41	86,65	33,27	20,03	84,30	77,39	77,48	36,88	34,84
	Y Koord	50,00	80,68	26,77	49,02	73,93	28,73	26,89	38,70	28,26	45,31
Raktár 2	X Koord	-	50,00	31,10	38,86	45,60	18,42	31,21	77,82	75,90	24,98
	Y Koord	-	50,01	82,01	83,06	70,60	69,57	48,88	60,46	25,98	57,44
Raktár 3	X Koord	-	-	50,00	86,38	34,41	48,45	41,34	39,63	64,70	55,18
	Y Koord	-	-	50,00	28,90	34,28	83,51	81,37	37,12	47,44	83,18
Raktár 4	X Koord	-	-	-	49,99	80,68	32,45	27,46	73,87	77,80	22,63
	Y Koord	-	-	-	49,95	28,31	46,78	64,85	19,62	45,05	84,17
Raktár 5	X Koord	-	-	-	-	50,00	52,26	44,99	28,66	41,80	40,86
	Y Koord	-	-	-	-	50,00	52,60	59,97	73,72	81,27	81,78
Raktár 6	X Koord	-	-	-	-	-	50,03	43,15	40,93	31,05	59,69
	Y Koord	-	-	-	-	-	50,02	40,10	79,95	49,00	59,30
Raktár 7	X Koord	-	-	-	-	-	-	50,00	29,05	22,82	71,90
	Y Koord	-	-	-	-	-	-	50,00	49,91	64,62	27,91
Raktár 8	X Koord	-	-	-	-	-	-	-	50,00	51,73	49,69
	Y Koord	-	-	-	-	-	-	-	50,01	39,39	58,15
Raktár 9	X Koord	-	-	-	-	-	-	-	-	50,02	89,46
	Y Koord	-	-	-	-	-	-	-	-	50,00	35,80
Raktár10	X Koord	-	-	-	-	-	-	-	-	-	49,84
	Y Koord	-	-	-	-	-	-	-	-	-	49,46

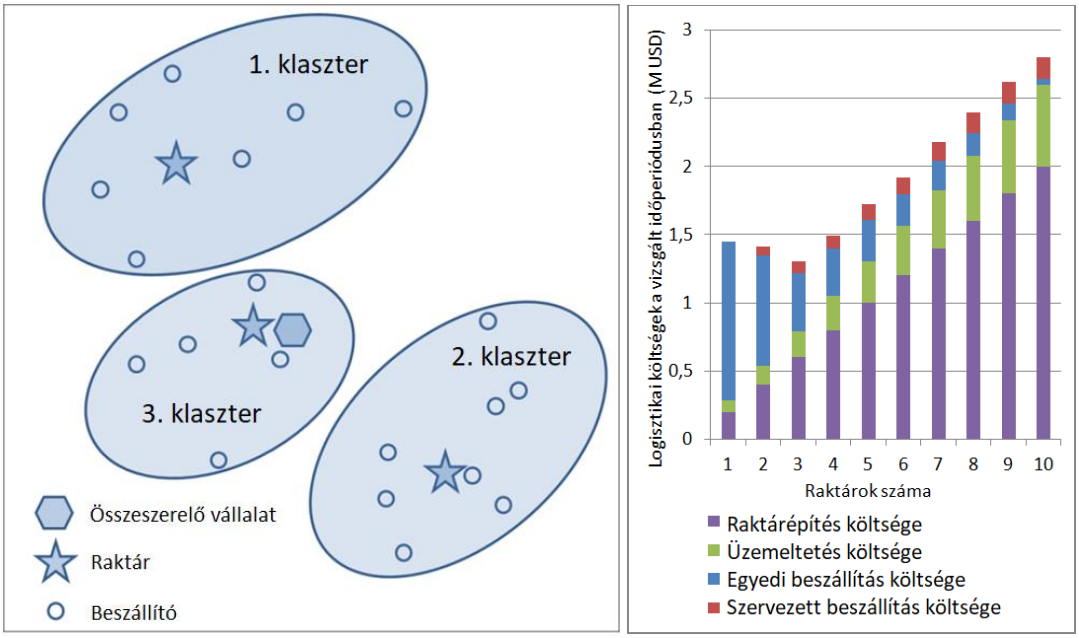
Néhány futtatás után észrevettem egy furcsa jelenséget, amelyet a 15. táblázatban is meg lehet figyelni: mindegy mennyi raktárat vettünk fel, mindig rakott az algoritmus egyet az összeszerelő vállalat közvetlen közelébe (50,50 pozíció). Ennek az az oka, hogy a gyár közelében lévő beszállítók lényegében a gyárba szállítsanak közvetlenül vagy egy gyár mellett lévő raktárba és oda nem kell külön gyűjtő raktár. Azt a határtávolságot, amelynél inkább megéri közvetlenül bevinni az összeszerelő vállalathoz, úgy számíthatjuk ki, hogy a gyár és a legközelebbi raktár távolságának a felét vesszük.

Az első felvetésre, miszerint képesek vagyunk valamilyen szinten szimulálni a raktártelepítést és megoldást keresni, a 46.ábra mutatja be. Ahogy a jobb ábráról leolvasható, ennek az esetnek a legköltséghatékonyabb megoldása 3 db raktár elhelyezése esetén történik, 1,3 millió USD-ből megoldható a teljes anyagmozgatási munka. Fontos megjegyezni, hogy rengeteg fontos tényezőt hagytunk ki a számításból, amely a számítási módszert minimálisan azonban a végeredményt nagyon is befolyásolhatja. Ilyen kihagyott lényeges tényezők: a regionálisan eltérő árak és költségek, mind a szállításban, mind a raktár építésében; kapacitások figyelmen kívül hagyása; szállítójárművek töltöttségének vizsgálata, egyéb szállítási módok alkalmazása, szállítási távolságok leegyszerűsített számítása. Ezek mind könnyedén beilleszthetők az alkalmazásba, ha szükség van rá és van elég adatunk.

A végleges elhelyezkedést 46. ábra bal oldalán lehet megtekinteni és a raktárakhoz kapcsolódó beszállítókat klaszterekkel csoportba rendezve lehet megtekinteni.

Az indok, amiért 3 csoport az ideális kialakítás, a 46. ábra jobb oldalából derül ki. Ugyanis jól látszik, hogy ha megvizsgáljuk a raktárak számának és az összes többi költségnek a szerepét, akkor 3 raktár kialakításával a 3 év alatt csak 1,3 millió USD költségünk keletkezik. Magáról a futtatásról nehéz ábrát készíteni, ugyanis egy számításnál nem csak egyszer fut le a raktárak helyeinek meghatározására szolgáló optimalizáló algoritmus, hanem minden raktármennyiség változatra külön-külön, és akkor áll meg, ha egy bizonyos pontosságnál nem tud tovább finomítani. Minden raktár két koordinátával rendelkezik, amelyet folyamatosan finomít az algoritmus, ezek az algoritmus számára az optimalizálandó paraméterek. Azaz 1 raktárnál csak 2 paramétert kell finomítania és 20 raktárnál 40-et.

DOI: 10.14750/ME.2020.018



46. ábra: Raktárak helyei és beszállítók csoportosítása. (balra),
Költségek alakulása raktárak számának függvényében (jobbra)

A raktárak helyzetének meghatározása után a program hozzárendeli a beszállítókat a raktárakhoz, ami után minden verzióban egy segéd optimalizáló algoritmus gyűjtőjáratot tervez közéjük.

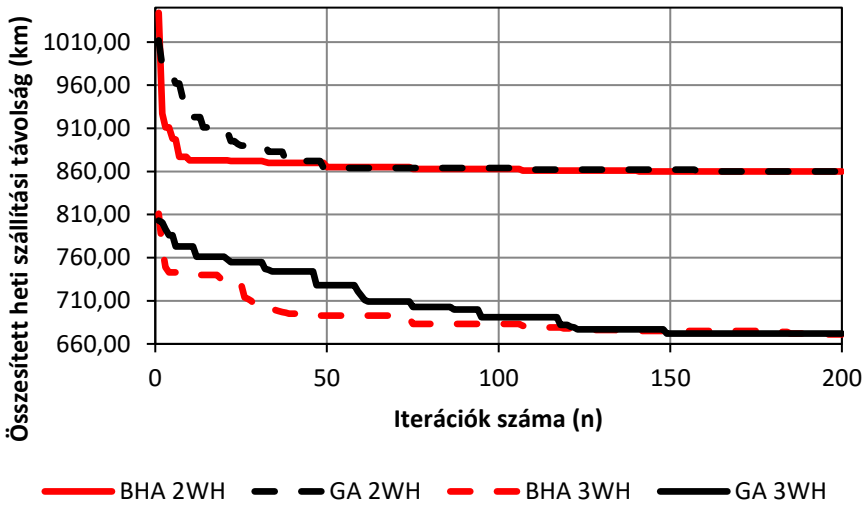
7.3.3. Algoritmusok tesztelése a feladaton

A fent vázolt alkalmazás alkalmas a tesztelésre is, azonban csak egy konfigurációval, ez pedig akkor, ha a telepítendő raktárak száma megegyezik a beszállítók számával, hiszen ilyenkor az alkalmazás rá van kényszerítve, hogy olyan raktári helyeket adjon meg, amely megegyezik a beszállítók helyével.
A tesztelésben az alábbi hat algoritmust használtam fel:

- Genetikus algoritmus (GA)
- Harmony Search algoritmus (HSA)
- Eredeti Black hole algoritmus (O-BH)
- Módosított Black hole algoritmus (MBS-BH)
- Eredeti Firefly algoritmus (O-FF)
- Módosított Firefly algoritmus (M-FF)

Ebben a fejezetben csak egy összehasonlító ábrát (47. ábra) fogok bemutatni, amelyben a genetikus algoritmust hasonlítom össze a módosított Black hole algoritmussal, két futtatásban. A felső két vonal (BHA 2WH és GA 2WH) bemutatja, hogy alakult a szállítási távolság akkor, ha 2 raktárt próbálunk lehelyezni, míg az alsó két vonal (BHA 3WH és GA 3WH) ugyanezt mutatja be 3 raktár esetén. Ebből az ábrából is jól látszik, hogy a módosított BH (piros) sokkal hamarabb jobb eredményeket generál, mint a GA (fekete). A további összehasonlításról és tesztekről szóló részt az 5.1. és 5.2-es fejezetben lehet megtalálni, amelyek részletesen leírják a tesztek eredményét.

DOI: 10.14750/ME.2020.018



47. ábra: Szállítási távolság két algoritmus és két raktármennyiség függvényében

DOI: 10.14750/ME.2020.018

7.4. Milkrun ellátási rendszer tervezése

A JIT (Just in Time) és JIS (Just in Sequence) ellátás tervezését és vezérlését széles körben kutatják, különösen a toló (push) elvű gyártás területén. A gyártási teljesítmények mérése és a JIT-kanban alkalmazása egyre fontosabbá válik [I6]. A gyártási és szerelési folyamatok összetettsége miatt az általános modellek és módszerek sajnos nem alkalmazhatók minden helyzetben. Az elmúlt években széles körben vizsgálták a milkrun-alapú alkatrész-ellátást és annak alkalmazását elsősorban az autóiparban. Az üzemen belüli ellátás szakirodalmában a milkrun használatának két gyakorlati oka van. Az első ok a belső ellátási oldal rendszerparamétereinek optimalizálása különösen logisztikai szempontból, míg a második ok az, hogy az anyagmozgatás folyamatai a gyártási és összeszerelési folyamatokhoz szervesen kapcsolódnak, ahol a két oldal szinkronizálása NP-nehéz optimalizálási problémának tekinthető. A szinkronizálási problémát nem csak belső, de akár külső folyamatoknál is figyelembe lehet venni [S18].

A gyáron belüli mikrunt a gyártás minden területén használhatják, de a legfontosabb terület a lean-elvű gyártás, ahol a „karcsúsított” anyagmozgató rendszereket a gyártási és összeszerelő állomások alapanyaggal való ellátására használják időszakos vagy kvázi-időszakosan mozgó járműveken keresztül [I13]. A milkrun járművek útvonalválasztási problémáinak megoldására szolgáló módszerek speciális modelleket tartalmaznak: a létesítmény vázlatát vagy tervrajzát megadják, amiből az anyagáramlás lehetséges útvonalait, a raktárok és tárolók elhelyezkedését meg tudjuk határozni. Ezekből el tudjuk készíteni a forrásokat és nyelőket és definiálni tudjuk azok anyagátvételi stratégiáját, amiket felhasználva egy egészértékű programozással képesek vagyunk megoldást, azaz útvonalakat és szállítási stratégiát találni.

A milkrun szállítás koncepcióját általában a gyáron belüli ellátás keretében használják annak érdekében, hogy a gyártó- vagy összeszerelő állomások alapanyag-ellátását kezeljék, és általában az ehhez kapcsolódó kutatások középpontjában a jármű útválasztási problémája (Vehicle Routing Problem, VRP) áll.

Komplex gyártási vagy összeszerelési folyamatokban a valós idejű járműmenedzselés lehetővé teszi az üzemen belüli ellátás vezérlését. A vizsgált moduláris megoldások a dinamikus járművezetési problémákon alapulnak. Egyes esetekben a kutatások motivációja nem a logisztikára összpontosít, hanem a gyártási vagy összeszerelési folyamatok optimalizálására vagy ütemezésére irányul.

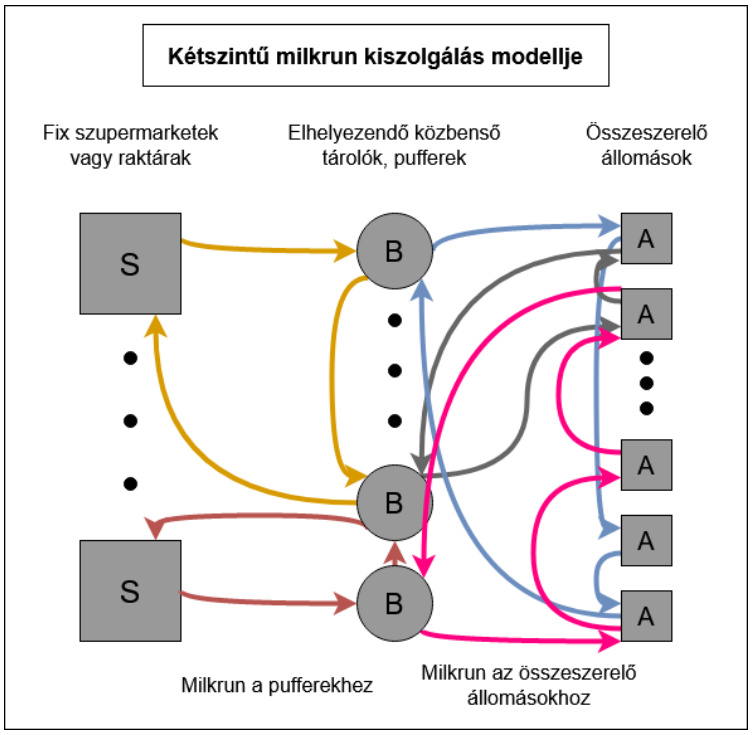
A fejezet célja, hogy megvizsgáljam a gyártási és összeszerelési állomások ellátásának integrált modelljét, amelyre egy heurisztikus algoritmus alapú applikációt készítettem, mely képes megoldást generálni. Először bemutatom egy egyedi gyáron belüli milkrun hálózat felépítésének modelljét, ahol két szintű milkrun rendszert alkalmazok. A továbbiakban ismertetem az applikáció működését és egy példán keresztül egy megoldást is bemutatok.

7.4.1. A kétszintű milkrun modellje

Amikor ellátási láncról beszélünk, főként a gyártó és szolgáltató vállalatok közötti kapcsolatokra gondolunk, ahol az összeszerelésre összpontosító fő vállalat szinte minden alkatrészt beszerez más cégektől. Az ellátási lánc alapelvei azonban sokkal alacsonyabb szinten

DOI: 10.14750/ME.2020.018

is megtalálhatók. Egy gyártó cégen belül ugyanolyan fontos, hogy az egyes gyártási állomásokat is ellássuk. Az anyagáramlás a gyártási ellátási láncban pontosan az ellenkezője a normál ellátási láncnak. Láthatjuk, hogy az anyagok és alkatrészek elhagyják a raktár vagy szupermarket területét milkrun formában és feltöltik a puffertárakat. Minden puffer legalább egy összeszerelő sorhoz vagy állomáshoz csatlakozik, így egy vállalat helyett több helyszínt kell kiszolgálni. A milkrun-járatoknak több különböző típusát is ismerjük, de mindegyik legfontosabb feladata, hogy alapanyaggal ellássuk a gyártógépeket, összeszerelő állomásokat vagy sorokat. E fejezet keretein belül szeretnénk ismertetni egy speciálisabb modellt, amely kifejezetten a milkrun alapú, üzemen belüli kiszolgálásokra, a létesítmény és objektum elhelyezésekre és a hozzárendelési problémákra összpontosít.



48. ábra: Kétszintű milkrun modellje

Ezek a kapcsolatok számos problémát felvetnek. A szupermarketből meg kell határozni egy optimális útvonalat a puffertárak milkrunnal való ellátására. Ha több alapanyag-raktárunk van, akkor a probléma nehezebb, mert a járműveket feladatokhoz kell rendelnünk, több útvonal-változatot kell létrehoznunk és meg kell határoznunk a mennyiségeket. Ebbe a modellbe az útvonaltervezést nem vesszük bele.

Integrált modellünk döntési változói a következők: a milkrun járművek felvevőállomásainak koordinátái; az összeszerelő állomás hozzárendelése a milkrun járművekhez és a felvevőállomásokhoz. Itt elsősorban arra összpontosít, hogy hogyan válasszuk ki a felvevőállomások legjobb helyét, ha a szupermarketek és az összeszerelő állomások állandó helyen vannak. Ennek érdekében ki kell számítanunk az anyagmozgatási munkát:

$$E_D \sum_{i=1}^{n} \sum_{j=1}^{m} Q_{S_i B_j} L_{S_i B_j} + (1 - E_D) \sum_{j=1}^{m} \sum_{k=1}^{p} Q_{B_j A_k} L_{B_j A_k} \rightarrow min. \qquad (67)$$

DOI: 10.14750/ME.2020.018

Koordinátákkal nagyon könnyű kiszámítani az útvonal hosszát (L). Az "E_d" szállítási energia-tényező a végeredmény gyakorlati irányba való eltolására szolgál. Ha nem használjuk ezt a tényezőt, a pufferek helye közelebb lesz a szupermarketekhez, mint az összeszerelő sorhoz, ugyanis a szupermarket önmagában olyan mennyiségű anyagot továbbít, mint az összeszerelő állomások anyagáramlásának kombinációja (67). Ez a tényező segít a puffertárolók elhelyezésében, hogy azok minél közelebb legyenek az összeszerelő sorokhoz vagy állomásokhoz. Ha nem akarunk beavatkozni az alaphelyzetbe, meg kell adnunk az "E_d=0,5". értéket. A gyakorlatban a tényező értéke "$E_d \ll 0{,}5$, kell, hogy legyen, így a szupermarketben az anyagmozgatás aránya csökken, és az összeszerelő sor melletti munka növekedni fog.

$$\sum_{i=1}^{n} Q_{S_i}^t \cong \sum_{k=1}^{p} Q_{A_k}^t \tag{68}$$

Van egy másik probléma ebben az ellátási láncban. Minden puffertároló rendelkezik maximális kapacitással, amit nem lehet túllépni (68). Ha van olyan puffertárolónk, amely nem rendelkezik elegendő kapacitással a környező állomások kiszolgálásához, három megoldásra is lehetőségünk van. A legegyszerűbb az, ha több puffertárolót vásárolunk vagy hozunk létre. Ha nincs pénzünk vagy szabad helyünk, akkor a milkrun-járatok mennyiségi növelésével vagy sűrűbb indításával tudjuk megoldani. Az utolsó módszer az aktuális állapot átrendezése. A távolságot és a kapacitást figyelembe véve ki kell választanunk, hogy melyik összeszerelő állomás csatlakozik az adott pufferhez.

$$C_{B_j} \geq \sum_{l=1}^{q} Q_{A_l} \text{ és } q \leq p \tag{69}$$

16. táblázat: Jelmagyarázat

Változó	**Jelentés**
$Q_{S_iB_j}$	anyagmennyiség i. szupermarketből j. tárolóba
$Q_{B_jA_k}$	anyagmennyiség j. tárolóból k. szerelőállomásra
$L_{S_iB_j}$	milkrun úthossz i. szupermarketből j. tárolóba
$L_{B_jA_k}$	milkrun úthossz j. tárolóból k. szerlőállomásra
E_D	szállítási energia állandó (0…1)
C_{B_j}	j. tároló kapacitása
n	szupermarketek száma
m	tárolók száma
p	összeszerelő állomások száma
q	j. tárolóhoz rendelt szerelőállomások száma

DOI: 10.14750/ME.2020.018

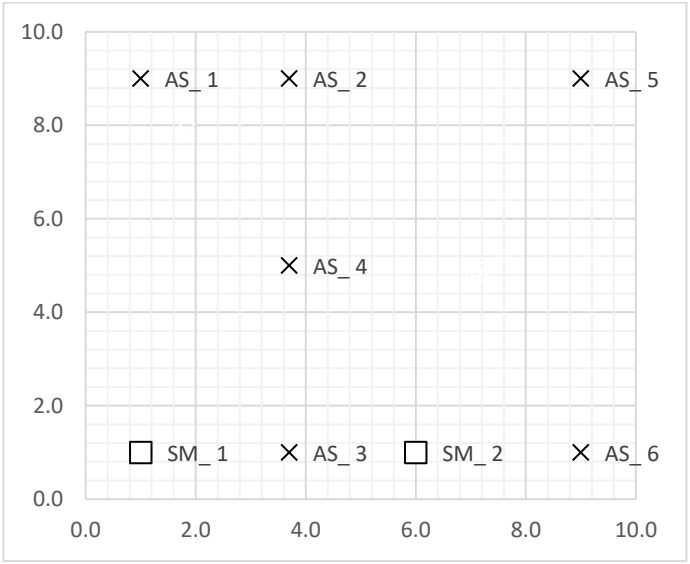
7.4.2. Tesztfeladat

A 49. ábra egy esettanulmányt mutat, ahol hat összeszerelő állomás (AS_ 1…6), két tároló (puffer) (BU_1&2) helyezkedik el (a 49. ábrán látható pozíciókban). Emellett a milkrun járművek két szupermarketet (SM_1&2) is útjukba ejtenek. A táblázatban található adatok segítségével mondjuk meg hova kerüljön a két tároló, amelyek maximális kapacitása 150 egység.

17. táblázat: Milkrun feladat alapadatai

		SM1	SM2	AS1	AS2	AS3	AS4	AS5	AS6
Koordináták	**X**	1	6	1	3,7	3,7	3,7	9	1
	Y	1	1	9	9	1	5	9	9
Anyagmennyiség		-	-	100	20	20	20	20	20

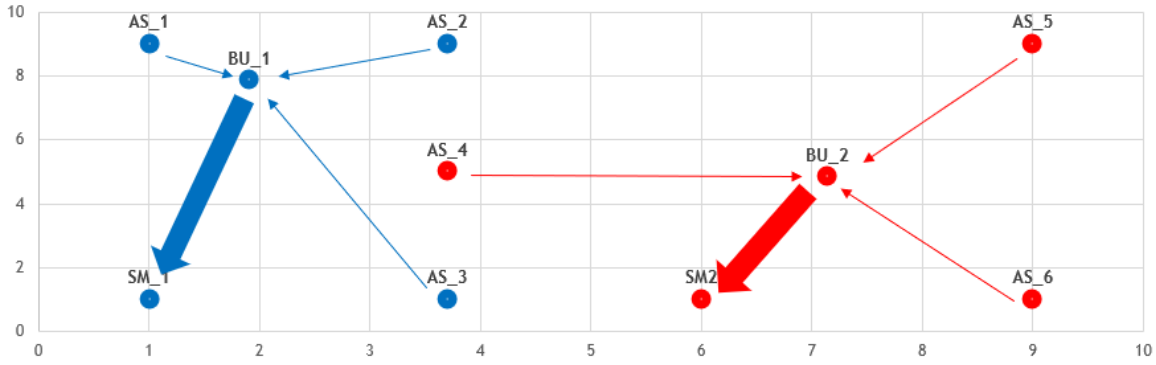
A feladat megoldását a következő két ábrán láthatjuk, amelyet ebben az esetben a Black hole algoritmus szolgáltatott a dinamikus (eseményhorizont) Schwarzschild sugárzás bevezetésével. Ezzel kicsit felgyorsítottuk a konvergencia sebességét, de nagyban megnőtt a számítási idő is.



49. ábra: Hat összeszerelő állomás (jele:X) és két szupermarket (jele: □) helyzete gyáron belül

A feladat megoldása a 50. ábrán látható, ahol az első három összeszerelő állomás (AS_1…3) az első felvevőállomáshoz (BU_1) van rendelve, amit az 1. mikrun járat elégíti ki, míg a második három összeszerelő állomás (AS_4…6) a második felvevőállomásról (BU_2) és a 2. szupermarketből (SM_2) kapja az alapanyagokat.

DOI: 10.14750/ME.2020.018



50. ábra: Hat összeszerelő állomás kétszintű milkrun ellátórendszerrel való kiszolgálása: hozzárendelés elvégzése

A további teszteredményeket az algoritmusokkal az 5.1. és 5.2.-es fejezetek tartalmazzák, ahol részletesen ki van emelve a hátizsák probléma is.

IV. TÉZIS: Megalkottam egy olyan általános keretrendszert, mely alkalmas speciális logisztikai modellek leszármaztatására. Ezen keretrendszer alapján megalkottam az integrált járattervezés, az elosztás és a milkrun anyagellátás egy-egy speciális modelljét. Ezen modellek által definiált NP-nehéz optimalizálási feladatok megoldására megoldásokat kerestem és egyedi módon átalakítottam. A problémák megoldásra egyedi alkalmazásokat készítettem, melyekkel a kidolgozott modellek és módszerek hatékonyságát validáltam. Mindegyik probléma megoldható volt lágyszámítási módszerekkel.

A tézis állítását alátámasztó saját publikációk és előadások: [S3-S11], [S14] [S15] [S17] [S18]

DOI: 10.14750/ME.2020.018

8. TÉZISEK ÖSSZEFOGLALÁSA

I. TÉZIS: Kidolgoztam egy új, szekvenciák közötti távolságok mérésére alkalmas módszert, bizonyítottam annak alkalmasságát különböző hosszúságú és felépítésű szekvenciák közötti távolságok mérésére vonatkozóan az egyezőség, a szimmetria és a távolság nem-negativitás szempontjából. A kidolgozott módszer alkalmas különböző járattervezési feladatokban az egyes megoldásváltozatokat reprezentáló permutációs egyedek közötti távolságok meghatározására.
A tézissel a 4.2. fejezet foglalkozik. A tézis állítását alátámasztó saját publikációk és előadások: [S5] [S8] [S9] [S10]

II. TÉZIS: Kidolgoztam egy olyan új paramétergenerálási módszert a hagyományos black hole heurisztikára, melynek révén annak konvergenciasebessége – különösen a keresési fázis elején – szignifikáns mértékben javítható. Kifejlesztettem a black hole heurisztikák egy olyan változatát, mely alkalmas szekvencia-problémák, például járattervezési feladatok megoldására.
A tézissel a 5.1. fejezet foglalkozik. A tézis állítását alátámasztó saját publikációk és előadások: [S3] [S6] [S7]

III. TÉZIS: Bevezettem a randomizált életciklus és az elhalálozási ráta fogalmát új egyedek generálására az algoritmus robusztusságnak növelése érdekében. Emellett bevezettem a legfényesebb memória alkalmazását Firefly algoritmusok hatékonyságának növelése céljából. Az elvégzett benchmark tesztek igazolták, hogy az általam kifejlesztett bővítmények révén az eredeti Firefly algoritmus hatékonysága növelhető.
A tézissel a 5.2. fejezet foglalkozik. A tézis állítását alátámasztó saját publikációk és előadások: [S5] [S9] [S10] [S11]

IV. TÉZIS: Megalkottam egy olyan általános keretrendszert, mely alkalmas speciális logisztikai modellek leszármaztatására. Ezen keretrendszer alapján megalkottam az integrált járattervezés, az elosztás és a milkrun anyagellátás egy-egy speciális modelljét. Ezen modellek által definiált NP-nehéz optimalizálási feladatok megoldására megoldásokat kerestem és egyedi módon átalakítottam. A problémák megoldására egyedi alkalmazásokat készítettem, melyekkel a kidolgozott modellek és módszerek hatékonyságát validáltam. Mindegyik probléma megoldható volt lágyszámítási módszerekkel.
A tézissel a 7. fejezet foglalkozik. A tézis állítását alátámasztó saját publikációk és előadások: [S3] [S4] [S5] [S6] [S7] [S8] [S9] [S10] [S11] [S14] [S15] [S17] [S18]

DOI: 10.14750/ME.2020.018

8.1. Summary of theses

I: THESIS: I worked out a new method for measuring distances between sequences, and proved its fitness for measuring distances between sequences of different length and build in aspect of equality, symmetry, and distance non-negativity. The finished method can determine the distances between the solution variations that are represented by the permutations in different route planning tasks.
The thesis is explained in chapter 4.2. My publications and presentations that support the thesis: [S5] [S8] [S9] [S10]

II. THESIS: I worked out a new method for generating parameters for the conventional black hole heuristic, which enables that its convergency speed - especially in the beginning of the search phase - can be improved significantly. I developed such version of the black hole heuristics, which is suitable for solving sequencing problems, such as route planning.
The thesis is explained in chapter 5.1. My publications and presentations that support the thesis: [S3] [S6] [S7]

III. THESIS: I introduced the concept of randomized lifecycle and death rate for generating new specimens to make the Firefly algorithm more robust. Furthermore, I introduced the usage of brightest memory for improving the efficiency of the Firefly algorithm. The results of the benchmark tests proved that the extensions can improve the effectiveness of the original Firefly algorithm.
The thesis is explained in chapter 5.2. My publications and presentations that support the thesis: [S5] [S9] [S10] [S11]

IV. THESIS: I created a general framework, which is capable of deriving special logistic models. Based on this framework, I created a special model for integrated route planning, distribution and milkrun material supply. For solving the NP-hard optimization tasks defined by these models I was looking for solutions and modified them in an unique way. I created individual applications for solving these tasks that I used for the validation of the effectiveness of the finished models and methods. Every problem was solvable using soft calculation methods.
The thesis is explained in chapter 7. My publications and presentations that support the thesis: [S3] [S4] [S5] [S6] [S7] [S8] [S9] [S10] [S11] [S14] [S15] [S17] [S18]

DOI: 10.14750/ME.2020.018

9. ÖSSZEFOGLALÁS

Kutatómunkám kitűzött célja olyan módszerek és megoldások keresése volt logisztikai feladatok megoldására, amelyre hagyományos analitikus módszerek nem adnak jó eredményt vagy nem alkalmasak a kezelésükre. Dolgozatomban több módszert és több esettanulmányt is készítettem, amely ezek megértésére, tesztelésére és felhasználására szolgál ipari és azon belül logisztikai feladatoknál.

Munkám első részében egy strukturált szakirodalmi feldolgozás keretében megismertem és ismertettem a múltbeli és jelenlegi vállalati trendeket, mint a globalizáció, digitalizáció, vevői igény és minőségnövekedés, és Ipar 4.0. Ezek és egyéb kisebb elvek együttesen határozzák meg a világ jelenlegi fejlettségét és igényét. Ezeket az igényeket csak akkor tudjuk effektíven kielégíteni, ha van egy jól működő hálózatunk, amelynek kiépítése és irányítása nagy feladat emberek számára, ezért amiben csak lehet szoftveres segítséget igényelnek. Egy nagy- vagy multinacionális vállalat a mai világban nem képes komplex vállalatirányítási rendszer nélkül effektíven működni és döntéseket hozni. Ezeknek az összekapcsolt rendszereknek és adatbankoknak a hálózatában normál matematikai eszközökkel szinte lehetetlen bármit is csinálni, magasabb rangú algoritmusok működnek alattuk, melyek magukba integrálnak egyszerű matematikai, gyakorlati és heurisztikus eszközöket. A heurisztikus eszközök nem csak a feladatok megoldásában, hanem a teljes rendszer irányításában is részt vesznek. Azonban vannak olyan vállalatirányítási és logisztikai területek, ahol a heurisztikák alkalmazása nagyon gyerekcipőben jár. Ezeknek a területeknek a felderítésére szolgált az irodalomkutatás fázisa, amelyből a kutatási cél is meg lett határozva.

A bevezető és irodalomkutatási rész után a heurisztikák ismertetése következett. Bemutatásra kerültek azok előnyei és hátrányai, amelyekhez példa is készült. Ezek után leszűkítettem a heurisztikákat az általam választott kutatási területre: a természet és populáció alapú metaheurisztikus algoritmusokra. Négy algoritmusfajtát mutatok be részletesen:

- a *Genetikus algoritmust*, mint összehasonlítási alapot,
- a *Harmony Search algoritmust*, mint előzetes kutatási témát és hasonlítási alapot,
- a *Firefly algoritmust*, mint a kutatási időszak elején használt és fejlesztett alapot,
- és a *Black hole algoritmust,* mint az utóbbi idők fő kutatási és fejlesztési alapját.

Ezekkel az algoritmusokkal dolgoztam a kutatási idő nagy részében és mutattam be olyan változataikat, amelyek képesek voltak megoldani nem nekik szánt feladatot, vagy amelyek különböző kiegészítésekkel gyorsabbak és pontosabbak lettek, mint eredeti verzióik. A 2. és 3. tézis ezeket foglalja magába.

Emellett bemutatásra került egy matematikailag bizonyított szekvenciák cseretávolságának meghatározására készített módszer, amely képes különböző mennyiségű és felépítésű elemekből álló szekvenciát kiértékelni és a távolságukat meghatározni. Erre a módszerre járattervezéseknél volt nagy szükségem és az 1. tézis foglalja magába a jelentőségét.

A dolgozat második nagy részében bemutatásra kerülnek tesztfeladatok és ipari gyakorlatból vett vagy ipari alkalmazhatóságú esettanulmányok. A tesztfeladatoknak és az esettanulmányoknak is az elsődleges szerepe a különböző algoritmusok teljesítményeinek egymáshoz mérése volt. Kifejlesztettem egy moduláris algoritmustesztelő alkalmazást, amelyben az algoritmusok és a feladatok úgy vannak elkészítve, hogy bármikor cserélhetők legyenek.

Az esettanulmányok a tesztelés mellett alkalmasak valós ipari körülmények szimulálására is. Olyan modelleket alkottam, amelyek képesek leírni vállalaton belüli vagy kívüli logisztikai feladatokat és megoldani azokat heurisztikák segítségével. A három esettanulmány közül az első a járattervezési feladatok komplexitását és módosítását mutatja be. A modell és az elkészült alkalmazás segítségével lehetséges automatikusan feladatokat kiadni, útvonalakat tervezni és

DOI: 10.14750/ME.2020.018

már éppen szállítás közben lévő járatokat módosítani, hogy akár új szállítási feladatot is elvégezhessen az adott járat.

A második esettanulmány az elosztási rendszer integrált tervezése nevet kapta, ahol gyűjtő- és elosztóraktárak helyeinek és hozzájuk rendelt beszállítóknak a mennyiségét kell meghatározni, a szállítási, beruházási, üresjárati és adminisztrációs költségek figyelembevételével. Ezt az esettanulmányt és modellt ipari munkák felhasználásával készítettem, ahol a raktárak számának és méretének meghatározása volt az elsődleges cél, emellett a modell segítségével járműflotta és személyzet is méretezhető.

A harmadik esettanulmány a belső logisztikai járattervezés modelljét mutatja be, amelynek egy és kétszintű milkrun rendszerű elosztás az alapja. A szakirodalomban fellelhető kétszintű milkrun rendszer modelljét fejlesztettem tovább. A rendszeren belül, szintén költség alapon határozzuk meg a szállítási utakat, tárolók és szupermarketek helyeit és méreteit. A modell ötletét a Robert Bosch Kft. egyik szerelőcsarnokában látottak alapján készítettem el. A 4. tézis foglalja magába az általános modellt és a belőle leszármaztatott feladatspecifikus megoldásokat az esettanulmányokra.

A Logisztikai Intézet 2019-ben egy olyan járattervezési K+F feladat elvégzésére tett ajánlatot, melyben jól alkalmazhatók az általam kifejlesztett heurisztikus megoldások. Általam kifejlesztett algoritmusokat használnak számos komplex vállalati feladat megoldása során, például nagyméretű üzemen belüli milkrun alapú ellátási folyamat optimalizálására, vagy komplex komissiózási feladatok idő- és energiahatékony megoldására.

Összefoglalásként a kutatómunkámban bemutattam olyan feladatokat, amelyekre az egyik, ha nem az egyetlen megoldás heurisztikák alkalmazása. Elkészítettem ezekhez alkalmazásokat és olyan heurisztikus módszereket, amelyek képesek a jelenlegi módszerekkel felvenni a versenyt és több esetben meghaladni azt gyorsaságban és pontosságban.

9.1. Summary

The goal of my research was to find such methods and solutions for solving logistical problems, for which traditional analytical methods couldn't provide satisfactory results or wouldn't be suitable for handling the task. In my paper I created several methods and case-studies, which can help understand, test and use these methods for industrial, more specifically logistical tasks.

In the first part of my work, while processing structured academic literature I reviewed and presented the past and present company trends, like the globalization, digitisation, increase in buyer demand and quality, and Industry 4.0. These and other smaller principles together determine the present advancements and demands of the world. These demands can only be effectively satisfied, if we have a network that works well, and to build and control it, which is an enormous task, people have resorted to the assistance of software. A big or multinational company in this day and age couldn't operate and make decisions efficiently without a complex enterprise management system. In the network of these connected systems and databanks it is almost impossible to do anything with normal mathematical tools, they are based on more complex algorithms, which can incorporate simple mathematical, practical and heuristical tools. These heuristical tools are not just there to solve tasks, they participate in controlling the whole system as well. There are some enterprise management systems and logistical fields where the use of these heuristics is not widespread yet. To explore these topics literature research was used, from which the research goal has also been determined.

The review of heuristics followed the intro and the literature research chapters. Its advantages and disadvantages have been demonstrated through examples. After that I narrowed

DOI: 10.14750/ME.2020.018

the heuristics down to one, I have chosen: the nature and population-based metaheuristic algorithms. I will present four types of algorithms in detail:

- the *Genetic algorithm*, as a frame of reference,
- the *Harmony Search algorithm*, as a previous research subject and reference,
- the *Firefly algorithm*, used and developed at the beginning of the research period,
- and the *Black hole algorithm*, as a base recently used in research and development.

During my research I mainly worked with these algorithms, and presented such versions of these, which were able to solve problems not meant for them, or with modifications were become faster and more precise, than their original versions. The 2nd and 3rd thesis includes these topics.

In addition, a method for determining the swap distance of the mathematically proven sequences has also been presented, which can evaluate and determine the distances of sequences that consists of elements of different quantity and structure. This method was really necessary for route planning, the 1st thesis covers its importance.

In the second main part of the study, test exercises and case studies of industrial practices has been presented. The main goal of these test exercises and case studies was to compare the performance of the different algorithms. I developed a modular algorithm testing application, in which the algorithms and tasks can be switched out at any time.

The case studies besides testing can be used for simulating real industrial conditions as well. I created such models, which can demonstrate the inside and outside logistic tasks of a company, and solve them with the help of heuristics. From the three case studies the first presents the complexity and modifications of the route planning tasks. With the help of the model and the finished application automatic assignments can be given, it is possible to plan or even modify routes during delivery, so a new delivery task can be executed.

The second case study got the title of "Integrated planning of distributed systems", which describes that the location of the collector and distributor warehouses and the quantity of their assigned subcontractors has to be determined based on the transportation, investment, unladen and administration costs. I created this case study and model based on industrial works, where the number and the size of the warehouses can be determined easily, the vehicle fleet and the number of personnel is also scalable.

The third case study demonstrates the model of the inner logistic route planning, which is based on the one and two level milkrun system. I improved upon the two level milkrun system model which can be found in the academic literature. This system also determines the delivery routes and the locations of warehouses and supermarkets based on the cost. The idea of the model is based on what I saw in one of the erecting shops of Robert Bosch Ltd. The general modell and the above mentioned three modells and solutions of industrial problems are involved in the 4th thesis.

In 2019 the Logistics Institution proposed a route planning R&D task, in which the heuristic solutions developed by me can be used efficiently. Algorithms that were developed by me are being used for solving several complex company problems, like optimizing the milkrun based supply process in a large plant, or solving complex commissioning tasks time and energy efficiently.

In summary, in my research I introduced such problems for which a solution, if not the only solution would be the use of heuristics. I created applications and such heuristic methods, that can compete with already existing methods, or even exceed them in speed and accuracy.

DOI: 10.14750/ME.2020.018

10. FELHASZNÁLT IRODALOM

10.1. Irodalomkutatáshoz feldolgozott irodalom

[I1] Cronin, P.; Ryan, F.; Coughlan, M. Undertaking a literature review: A step-by-step approach. British Journal of Nursing. 2008, Volume:17, pp. 38–43.

[I2] Fung, R.Y.K.;Chen, T.: A multiagent supply chain planning and coordination architecture, International Journal of Advanced Manufacturing Technology, Volume 25, Issue 7-8, April 2005, pp. 811-819.

[I3] Tiwari, A.K. Tiwari, A., Samuel, C.: Supply chain flexibility: A comprehensive review, Management Research Review, Volume 38, Issue 7, 20 July 2015, pp. 767-792.

[I4] Fosso Wamba, S.: Achieving supply chain integration using RFID technology: The case of emerging intelligent B-to-B e-commerce processes in a living laboratory, Business Process Management Journal, Volume 18, Issue 1, February 2012, pp. 58-81.

[I5] Bányai T., Bányai, Á., Illés, B., Tamás, P.: Ipar 4.0 és logisztika, Miskolc, 2019, 1. kiadás, ISBN: 978-963-358-182-7.

[I6] Galasso, F., Merce, C., Grabot, B.: Decision support framework for supply chain planning with flexible demand, International Journal of Production Research, Volume 47, Issue 2, January 2009, pp. 455-478.

[I7] Lima, R.M.; Sousa, R.M.; Martins, P.J.: Distributed production planning and control agent-based system, International Journal of Production Research, Volume 44, Issue 18-19, 15 September 2006, pp. 3693-3709.

[I8] Makatsoris, H.C., Chang, Y.S.: Design of a demand-driven collaborative supply-chain planning and fulfilment system for distributed enterprises, Production Planning and Control, Volume 15, Issue 3, April 2004, pp. 256-269.

[I9] Singh, H., Garg, R.K., Sachdeva, A.: Supply chain collaboration: A state-of-the-art literature review, Uncertain Supply Chain Management, Volume 6, Issue 2, 2018, pp. 149-180.

[I10] Gaspari, L. [et.al.]: Modularization in material flow simulation for managing production releases in remanufacturing, Journal of Remanufacturing, Volume 7, Issue 2-3, 1 December 2017, pp. 139-157.

[I11] Bányai, T.; Bányai, A.: Modelling of just-in-sequence supply of manufacturing processes, MATEC Web of Conferences, Volume 112, Innovative Manufacturing Engineering and Energy International Conference, IManE and E, Iasi, 3 July 2017; Article number: 0602521.

[I12] Belmecheri, F. [et. al.]: Particle swarm optimization algorithm for a vehicle routing problem with heterogeneous fleet, mixed backhauls, and time windows, Journal of Intelligent Manufacturing, Volume 24, Issue 4, August 2013, pp. 775-789.

[I13] Jharkharia, S., Shankar, R.: Selection of logistics service provider: An analytic network process (ANP) approach, Omega, Volume 35, Issue 3, June 2007, pp. 274-289.

[I14] Liou, J.J.H., Tamošaitiene, J., Zavadskas, E.K., Tzeng, G.-H.: New hybrid COPRAS-G MADM Model for improving and selecting suppliers in green supply chain management, International Journal of Production Research, Volume 54, Issue 1, 2 January 2016, pp. 114-134.

[I15] Nagy, J. (és társai): The role and impact of industry 4.0 and the internet of things on the business strategy of the value chain-the case of hungary, Sustainability, Volume 10, Issue 10, Switzerland, 29 September 2018, Article number 3491.

[I16] Gašová, M., Gašo, M., Štefánik, A.: Advanced Industrial Tools of Ergonomics Based on Industry 4.0 Concept, Procedia EngineeringVolume 192, 2017, 12th International Scientific Conference Of Young Scientists On Sustainable, Modern and Safe Transport, TRANSCOM 2017; High Tatras Grand Hotel Bellevue; Slovakia; 31 May 2017- 2 June 2017; Code 136438, pp. 219-224.

[I17] Kuby, M.J., Gray, R.G.: The hub network design problem with stopovers and feeders: The case of Federal Express, Transportation Research Part A, Volume 27, Issue 1, January 1993, pp. 1-12.

[I18] Romer, M. [et.al.]: The internet of load carriers design of a cloud-based service system for smart and connected load carriers, ICETE 2018 - Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, Volume 1, 2018, 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018; Porto; Portugal; 26 July 2018-28 July 2018; Code 150360, pp. 166-173.

[I19] Lang, W. [et. al.]: The "intelligent container" - A cognitive sensor network for transport management, IEEE Sensors Journal, Volume 11, Issue 3, 2011, Article number 5582153, pp. 688-698.

[I20] Elhedhli S., Merrick R. :Green supply chain network design to reduce carbon emissions, Transportation Research Part D: Transport and Environment, Volume 17, Issue 5, July 2012, pp. 370-379.

[I21] Figliozzi M.A.: The impacts of congestion on time-definitive urban freight distribution networks CO2 emission levels: Results from a case study in Portland, Oregon, Transportation Research Part C: Emerging Technologies, Volume 19, Issue 5, August 2011, pp. 766-778.

[I22] Hosseini S.D., Shirazi M.A., Ghomi S.M.T.F.: Harmony search optimization algorithm for a novel transportation problem in a consolidation network Engineering Optimization Volume 46, Issue 11, 2 November 2014, pp. 1538-1552.

[I23] Geem Z.W., Kim J.H., Loganathan G.V.: A New Heuristic Optimization Algorithm: Harmony Search, Simulation, Volume 76, Issue 2, February 2001, pp. 60-68.

[I24] Omran M.G.H, Mahdavi M.: Global-best harmony search, Applied Mathematics and Computation, Volume 198, Issue 2, 1 May 2008, pp. 643-656.

[I25] Karaboga D., Gorkemli B., Ozturk C., Karaboga N.: A comprehensive survey: Artificial bee colony (ABC) algorithm and applications, Artificial Intelligence Review, Volume 42, Issue 1, June 2014, Pages 21-57.

[I26] Saida I.B., Nadjet K., Omar B.: A new algorithm for data clustering based on cuckoo search optimization, Advances in Intelligent Systems and Computing, Volume 238, 2014, Pages 55-64.

[I27] Li X.L., Shao Z.J., Qian J.X.: Optimizing method based on autonomous animats: Fish-swarm Algorithm, Xitong Gongcheng Lilun yu Shijian/System Engineering Theory and Practice, Volume 22, Issue 11, November 2002, Page 32.

[I28] Eberhart R., Kennedy J.: Particle swarm optimization, IEEE International Conference on Neural Networks - Conference Proceedings, Volume 4, 1995, , Proceedings of the 1995 IEEE International Conference on Neural Networks.; 27 November 1995, pp. 1942-1948.

[I29] Rahmani A., Mirhassani S.A.: A hybrid Firefly-Genetic Algorithm for the capacitated facility location problem, Information Sciences, Volume 283, 1 November 2014, pp. 70-78.

[I30] Yang X.S.: Firefly algorithms for multimodal optimization, Lecture Notes in Computer Science, Volume 5792 LNCS, 2009, 5th Symposium on Stochastic Algorithms, Foundations and Applications, SAGA, 2009; pp. 169-178.

[I31] Fister I., Jr.a Perc, M., Kamal S.M., Fister I. : A review of chaos-based firefly algorithms: Perspectives and research challenges, Applied Mathematics and Computation, Volume 252, 1 February 2015, pp. 155-165.

[I32] Dorigo M., Gambardella L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation, Volume 1, Issue 1, 1997, pp. 53-66.

[I33] Dorigo M., Di Caro G., Gambardella, L.M.: Ant algorithms for discrete optimization, Artificial Life, Volume 5, Issue 2, 1999, pp. 137-172.

[I34] Wu Z., Choy K.L., Chan H.K., Lee, C.K.M., Zhang, S.: Swarm intelligence applied in green logistics: A literature review, Engineering Applications of Artificial Intelligence, Volume 37, 1 January 2015, pp. 154-169.

[I35] Ting C.J., Chen C.H.: A multiple ant colony optimization algorithm for the capacitated location routing problem, International Journal of Production Economics,Volume 141, Issue 1, January 2013, pp. 34-44.

[I36] Costa D., Hertz A. : Ants can colour graphs, Journal of the Operational Research Society, Volume 48, Issue 3, March 1997, pp. 295-305.

[I37] Hatamlou A.: Black hole: A new heuristic optimization approach for data clustering, Information Sciences, Volume 222, 10 February 2013, pp. 175-184.

[I38] Ricardo S., Broderick C., Rodrigo O.: Solving the non-unicost set covering problem by using cuckoo search and black hole optimization, Natural Computing 16(2), 2017.

[I39] Piotrowski A.P., Napiorkowski J.J., Rowinski P.M.: How novel is the "novel" black hole optimization approach?, Information Sciences, Volume 267, 20 May 2014, pp. 191-200.

[I40] Hasan Z., El-Hawary M.E. Optimal power flow by black hole optimization algorithm, Proceedings - 2014 Electrical Power and Energy Conference, EPEC 201427 February 2014, , pp. 134-141.

[I41] Azizipanah-Abarghooee R., Niknam T., Malekpour M., Bavafa F., Kaji M.: Optimal power flow based TU/CHP/PV/WPP coordination in view of wind speed, solar irradiance and load correlations, Energy Conversion and Management, Volume 96, 15 May 2015, pp. 131-145.

[I42] Grefenstette J.J.: Optimization of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man and Cybernetics, Volume 16, Issue 1, JANUARY/FEBRUARY 1986, pp. 122-128.

[I43] Konak A., Coit D.W., Smith A.E. Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engineering and System Safety, Volume 91, Issue 9, September 2006, pp. 992-1007.

[I44] Tamura K., Peterson D., Peterson N., Stecher G., Nei M., Kumar S.: MEGA5: Molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods, Molecular Biology and Evolution, Volume 28, Issue 10, October 2011, pp. 2731-2739.

[I45] Eberhart R.C., Shi Y.: Comparison between genetic algorithms and particle swarm optimization, Lecture Notes in Computer Science, Volume 1447, 1998, Pages 611-616, 7th Annual Conference on Evolutionary Programming, EP 1998; San Diego; United States; 25 March 1998.

[I46] Takagi T., Sugeno M.: Fuzzy Identification of Systems and Its Applications to Modeling and Control, IEEE Transactions on Systems, Man and Cybernetics, Volume SMC-15, Issue 1, January 1985, Pages 116-132.

[I47] Chiu S.L.: Fuzzy model identification based on cluster estimation, Journal of Intelligent and Fuzzy Systems, Volume 2, Issue 3, 1994, Pages 267-278.

10.2. Kutatómunkához felhasznált irodalom

[K1] Nagy M. Ipar 4.0: mindent átformál. Piac & Profit, 2018. szeptember 03., https://piacesprofit.hu/kkv_cegblog/ipar-4-0-az-uj-szabvany/ Letöltve: 2018.11.25.

[K2] The Human Memory, BRAIN NEURONS & SYNAPSES, http://www.human-memory.net/brain_neurons.html, Letöltve: 2018.10.25.

[K3] Tamás P., Illés B.: Gyártórendszerek folyamatfejlesztési lehetőségei a negyedik ipari forradalomban, Műszaki szemle, 67. kötet, 2016.

[K4] Nagy G, Illés B, Bányai Á.: Impact of Industry 4.0 on production logistics, IOP Conference Series: Materials Science And Engineering, Volume:448, 2018, Paper: 012013.

[K5] Tisztázzuk az Ipar 4.0 alapfogalmait! https://autopro.hu/trend/Tisztazzuk-az-Ipar-4-0-alapfogalmait/18073/ Letöltve: 2018.09.13.

[K6] Cisco Visual Networking Index: Háromszorosára nő az internet adatforgalma 2018-ra, http://www.hirado.hu/2014/06/13/haromszorosara-no-az-internet-adatforgalma-2018-ra/ Letöltve: 2018.10.20.

[K7] Cselényi J., Illés B.: Logisztikai rendszerek I., 2005, Kiadó: Miskolci Egyetemi Kiadó

[K8] Keiretsu, http://whatis.techtarget.com/definition/keiretsu Letöltve: 2018.10.09.

[K9] Lengyelné Molnár, T., Tóvári, J.: Kutatásmódszertan, 2001, Kiadó: Eszterházy Károly Főiskola.

[K10] Cselényi J., Illés B.: Anyagáramlási rendszerek tervezése és irányítása I. 2006, Kiadó: Miskolci Egyetemi Kiadó.

[K11] Newell, A., Simon, H.A.: Human problem solving, 1972, Kiadó: sci.brooklyn.

[K12] Pearl, J.: Heuristics: intelligent search strategies for computer problem solving, 1984, Kiadó: OSTI.

[K13] Dreo, J.P. Aumasson, W. Tfaili, P. Siarry: Adaptive Learning Search, a new tool to help comprehending metaheuristics, International Journal on Artificial Intelligence Tools, Vol. 16, No. 3., 1 June 2007.

[K14] Sain, M.: Matematikatörténeti ABC, 1978, Kiadó: Tankönyvkiadó.

[K15] Heeringa, W.J.: Measuring dialect pronunciation differences using Levenshtein distance, 2004, Kiadó:Citeseer.

[K16] Zhang, J., Liu, K., Tan, Y., He, X.: Random black hole particle swarm optimization and its application, in: 2008 IEEE International Conference Neural Networks and Signal Processing, ICNNSP, 2008, pp. 359–365.

[K17] Big Crunch Universe: DARK ENERGY AND THE RED SHIFT IN A CONTRACTING UNIVERSE, http://bigcrunchuniverse.com/home/big-crunch-universe/, Letöltve: 2017.06.25.

[K18] Chen, Q.(et.al.): Problem Definitions and Evaluation Criteria for CEC 2015 Special Session on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization, Computer Science, 2015.

[K19] Mitchell, M.: An Introduction to Genetic Algorithms. Cambridge, 1996, Kiadó: MIT Press.

[K20] BenchmarkFcns: Benchmark functions, Letöltve: 2018.07.10. http://benchmarkfcns.xyz/fcns

[K21] Hirschleifer J., Glazer A., Hirschleifer D.: Mikroökonómia, 2009, Kiadó: Osiris Kiadó.

[K22] Ibrahim, I.M., Crolla, D.A., Barton, D.C.: The impact of the dynamic tractor-semitrailer interaction on the ride behaviour of fully-laden and unladen trucks, SAE transactions, JSTOR, 2004.

[K23] Tannenbaum, A.S.: Számítógép-architektúrák, 2006, Kiadó: Panem Kiadó.

[K24] Demetrovics J., Katona Gy.: Az algoritmusok bonyolultsága, Letöltve: 2019.12.10.

http://www.termeszetvilaga.hu/kulonsz/k002/algoritmus.html

10.3. Saját irodalom

[S1] Veres P., Bányai T., Illés B.: Make-or-Buy optimisation with Harmony Search Algorithm, ASCONIKK 2014: Extended Abstracts: II. Future Internet Services. 2014.12.14-2014.12.17. Veszprém: University of Pannonia, 2014. pp. 59-66.

[S2] Bányai T., Veres P.: Optimisation of knapsack problem with matlab, based on harmony search algorithm, ADVANCED LOGISTIC SYSTEMS: THEORY AND PRACTICE 7:(1), 2013, pp. 13-20.

[S3] Bányai T., Veres P., Illés B.: Heuristic Supply Chain Optimization of Networked Maintenance Companies, PROCEDIA ENGINEERING 100, 25th DAAAM, 2014. Bécs, Ausztria: 2014.11.26 -2014.11.29., pp. 46-55.

[S4] Veres P., Bányai T., Illés B.: Optimization of in-plant production supply with black hole algorithm, 9th International Congress on Precision Machining, Key Engineering Materials, Görögország, Athén, 2017.09.06-2017.09.09.

[S5] Veres P., Bányai T., Illés B.: Route planning among non-predefined objects; 9th International Doctoral Students Workshop on Logistics. 104 p.;2016.06.22 Magdeburg: Otto von Guericke University Magdeburg, 2016. pp. 65-70.

[S6] Veres, P.: Heurisztikus algoritmusok alkalmazása a logisztikában, Műszaki tudomány az észak-kelet magyarországi régióban, 2019, Debrecen, Magyarország: Debreceni Akadémiai Bizottság Műszaki Szakbizottsága, 2019, pp. 440-443.

[S7] Veres P., Bányai T., Illés B.: Optimization of Inverse Supply with Black Hole Algorithm, 10th International Doctoral Students Workshop on Logistics, Németország, Magdeburg Otto von Guericke University, 2017.06.20.

[S8] Veres P., Illés B, Bányai T.: Software development for performance validation of heuristic algorithms, 10th International Conference of Mechanical Engineering COMEC 2019: 5th Symposium of Quality Management & Logistics, 2019, pp. 9-17.

[S9] Veres P., Bányai T., Illés B.: Automatikus járatmódosítás megoldása intelligens szállítási rendszerekben, Doktoranduszok Fóruma 2016, Gépészmérnöki és Informatikai Kar szekciókiadványa, Miskolc-Egyetemváros, Magyarország: Miskolci Egyetem, 2017, pp. 106-111.

[S10] Veres P., Bányai T., Illés B.: Járattervezés előre nem definiált objektumok között, Alkalmazott Tudományok III. Fóruma: Konferenciakötet, Budapest, Magyarország: Budapesti Gazdasági Egyetem, 2016, pp. 880-890.

[S11] Veres P, Bányai T, Illés B: Optimisation problems of networking manufacturing processes, 21st Innovative Manufacturing Engineering and Energy International Conference, IManE and E, EDP Sciences, 2017, Paper: 06026.

[S12] Veres P.: Harmony Search algoritmus alkalmazása anyagáramlási feladatok tervezésénél (előadás: Miskolci Egyetem, Logisztikai Intézet, 2014.10.07.).

[S13] Bányai, T.; Veres, P.: Optimisation of production depth, ADVANCED LOGISTIC SYSTEMS: THEORY AND PRACTICE, Volume 7: No.2, 2013, pp. 85-94.

[S14] Veres, P.; Illés, B.: Algoritmusok összehasonlító elemzése moduláris alkalmazással logisztikai problémákon, XXVII. Nemzetközi Gépészeti Konferencia OGÉT 2019, Nagyvárad, Románia: Erdélyi Magyar Tudományos Társaság, 2019, pp. 617-620.

[S15] Veres, P.; Bányai, T.; Illés, B.: Hálózatszerűen működő ellátási lánc modelljének és alkalmazásának továbbfejlesztése, Doktoranduszok Fóruma 2015: Gépészmérnöki és Informatikai Kar szekciókiadványa, Miskolc, Magyarország: Miskolci Egyetem Gépészmérnöki Kar, 2016, pp. 23-28.

[S16] Veres, P.; Illés, B.; Bányai, T.: Integrated Assignment and Facility Location Approach Based on Black Hole Optimization, ACADEMIC JOURNAL OF MANUFACTURING ENGINEERING 17: No.2, 2019, pp. 66-71.

[S17] P., Veres; B., Illés; C., Landschützer: Supply Chain Optimization in Automotive Industry: A Comparative Analysis of Evolutionary and Swarming Heuristics, LECTURE NOTES IN MECHANICAL ENGINEERING, 2018, pp. 666-676.

[S18] Veres, P.: Létesítmény telepítési és hozzárendelési feladat megoldása Blackhole optimálással, GÉPGYÁRTÁS 57: 1-2, 2018, pp. 79-83.