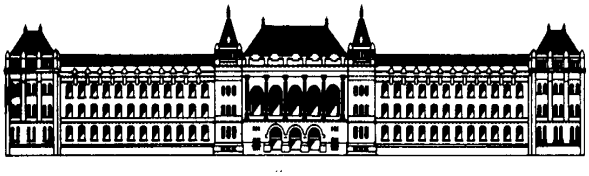


Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Elektronikai Technológiai Tanszék

Tudományos Diákköri Konferencia Dolgozat

Raktármodell fejlesztése genetikus és heurisztikus tárhelykiosztási algoritmusokkal

VONYÓ PÉTER ANDRÁS

HUOKFQ

Konzulensek:
Dr. Martinek Péter és Dr. Sztrapkovics Balázs

2023

Tartalom

Absztrakt 4

1. Bevezető 5

1.1. A dolgozat célkitűzései 6

1.2. Személyes motiváció 6

2. A téma háttere és elméleti alapjai 7

2.1. Az ellátási lánc 7

2.2. A raktározási rendszerek bemutatása 8

2.3. A raktározási folyamatok 9

2.3.1. A komissiózás 9

2.3.2. Komissiózást támogató fizikai eszközök 10

2.3.3. Komissiózást támogató folyamatszervezési megoldások 11

2.3.4. Betárolás 12

2.4. Útvonalkereső és tárhelykiosztó algoritmusok rövid ismertetése 13

2.4.1. Útvonalkereső algoritmusok 13

2.4.2. S-shape algoritmus 13

2.4.3. Largest gap heurisztika 15

2.4.4. Tárhelykiosztó algoritmusok – a tárhely kijelölési probléma 18

2.4.6. A véletlen és az ABC tárhelykiosztás 20

2.4.7. Az apriori datamining technikán alapuló heurisztika 21

2.4.8. Az genetikus tárhelykiosztó algoritmus működési elve 22

2.5. Algoritmusok összehasonlítása 24

3. A modellezési környezet tervezése és felépítése 26

3.1. A modellezési környezet felépítése 26

3.2. A modellező rendszer felépítése 27

3.2.1. Tárolási rendszerek és raktártopológiák 28

3.2.2. Raktármodell 29

3.2.3. Árucikkek modellezése 29

3.2.4. Komissiózó ágensek modellezése 30

4. Az adatok és a raktármodell bemutatása 31

4.1. A megrendelésadatok és kigyűjtési listák bemutatása 31

4.2. A raktártopológia bemutatása 32

4.2.1. „A” raktártopológia 33

4.2.2. „B” raktártopológia 34
4.2.3. A modell kikötései 34
4.2.4. A komissiózó ágens műszaki paraméterei 35
4.3. A számítási módszerek bemutatása 35
5. Tesztek és eredmények 38
5.1. Az algoritmusok tesztelése 38
5.1.1. A vizsgált algoritmuspárok 38
5.1.2. Keresztvalidáció 38
5.1.3. Futási idők 39
5.2. Az eredmények bemutatása 39
5.2.1. Az „A” raktáreset eredményeinek bemutatása 39
5.2.2. A „B” raktáreset eredményeinek bemutatása 41
5.2.3. A két raktár eredményei egymáshoz viszonyítva 42
6. Az eredmények elemzése és értékelése 45
6.1. Az algoritmuspárok hatékonysága 45
6.1.1. Eredményességük 45
6.1.2. Útvonalkeresés és tárhelykiosztás 48
6.1.3. Az „A” és a „B” eset közti eltérések elemzése 50
6.2. Szinergia, vagy tompítás 51
6.3. Összegzés 52
7. További kutatási kérdések 54
8. Irodalomjegyzék 55

Absztrakt

A logisztika már az ókorban is kulcsfontosságú volt, hiszen a hadi és kereskedelmi ellátási láncok zökkenőmentes működését biztosítani kellett. A raktározás, a komplex logisztikai rendszerek alapvető eleme. Különösen a komissiózás igényel kiemelt figyelmet, mivel költségei a raktári műveletek költségvetésének jelentős részét képezik. A komissiózási folyamat során két kritikus optimalizációs kihívás kerül előtérbe: a tárhely-kijelölés és az útvonal-választási probléma, melyek mindkettő NP-nehéz problémának minősülnek. Ezek hatékony kezelése jelentős erőforrás-megtakarítást eredményezhet.

Ebben a kutatásban tovább viszem egy korábbi projektemet, melyben genetikus algoritmusokon és heurisztikákon alapuló tárhely kijelölési módszereket (mint például, Pareto-elv alapján működő „ABC" algoritmus, valamint egy datamining módszereken alapuló heurisztikus algoritmus) vizsgáltam, keresztfolyosó nélküli, „egyblokkos" raktári modellekben. A jelenlegi munka során komplex, több blokkból álló raktári topológiákat modellezve teszteltem az útvonalválasztó és a tárhelykiosztó algoritmusok hatékonyságát, a komissiózás mozgásának időigénye alapján. Az előbbi esetében a Largest Gap és az S-shape algoritmusokat alkalmaztam.

A kutatásom középpontjában az a meghatározó kérdés állt, hogy melyik útvonalválasztó és tárhelykiosztó algoritmus kombináció bizonyul a leghatékonyabbnak, valamint hogy fennáll-e kimutatható kölcsönhatás ezen algoritmusok között. A vizsgálatok során egyértelmű válaszokat sikerült találnom az első kérdésre: minden vizsgált raktártopológia esetében a Largest Gap és a genetikus algoritmus párosa bizonyult a legelőnyösebbnek. Ezen algoritmusok 21%~24% - százlékponttal jobb eredményeket értek el, mint a legrosszabb eredményt elért algoritmuspár. Nem találtam egyértelmű bizonyítékot az algoritmusok közötti kölcsönhatásra. A kutatási folyamat során felmerült számos további izgalmas kérdés is, amelyek mélyebb elemzést és további vizsgálatokat igényelnek. Például, hogy az útvonalválasztási vagy a tárhelykijelölési probléma hatékonysága befolyásolja jobban a folyamat hatékonyságát, vagy hogy rosszabb topológiájú raktárakban valóban nagyobb megtakarítási potenciálokat nyújtanak a fejlettebb algoritmuspárok? Bizonyos feltételezéseket már e kutatás adatai alapján fel lehet vetni, ugyanakkor a teljes kép megalkotása és bizonyítása további kutatásokat igényel.

1. Bevezető

A logisztika szó a görög "logosz" és "istemi" szavakból származik, ami jelentése "gondolkodás és szervezés". Egyidős az emberi civilizációval. Mióta háborúk vannak és az emberek kereskednek, léteznek ellátási láncok is melyek hadseregeket, flottákat és városokat láttak el. Az ellátási láncok működtetéséhez pedig szükség volt raktárakra, átrakó pontokra kikötőkre, infrastruktúrára [1].

A római hadseregben logistáknak hívták, akik a légiók ellátásáról gondoskodtak. Feladatkörük komplex és jól meghatározott volt. Számos bizánci irat is kiemelten foglalkozott a hadtápellátás, a hadilogisztika feladatával. Nagy ugrást hozott, a világkereskedelem kialakulása és az ipari forradalom. Egyrészt, igény keletkezett az áruk nagy volumenben, nagy távolságra történő mozgatására. Másrészt a vasúthálózat kiépülése lehetővé tette az áruk hatékony szárazföldi szállítását, ami addig kizárólag vízi úton volt megoldható. A következő ugrás az 1950-es években történt. A fejlődés fő motorjai ezen időszakban is az igények globális növekedése továbbá az infrastruktúra és a technológia fejlődése volt. Egyrészt kiépültek az autópálya hálózatok, melyek a kisebb volumenű, nagyobb értéksűrűségű áruk szállítását és a terítő illetve gyűjtő hálózatokat forradalmasították, másrészt a második világháborúban az amerikai hadsereg kifejlesztette a raklapos és konténeres egységbe foglalási módszereit, melyekkel géppel kezelhetővé tették a legkülönbözőbb árucikkeket is [1]

Az elmúlt évtizedekben is zajlik egy logisztikai forradalom, mely oka, hogy jelentősen felértékelődött a logisztika szerepe. Ennek számos forrása van, többek között a kis csomagküldő szolgáltatások elterjedése, az e-kereskedelem, a szétterülő és globalizálódó ellátási láncok, továbbá azok a termelési koncepciók melyben a kis sorozatokra történő gyártásütemezés kap nagy hangsúlyt. Ezen okok volumenében és komplexitásában is komoly terhelést jelentettek a logisztikai rendszerek számára, valamint egyre inkább megkövetelik a logisztikáról, hogy a logisztikai rendszer a lehető legolcsóbban, legpontosabban, legmegbízhatóbban végezze el a feladatát. Ezt pedig csak nagyon komoly és sokrétegű informatikai támogatással lehet megvalósítani[2].

A logisztika feladatát úgyis összefoglalhatnánk, mint az ellátási lánc működtetése, akár globális szinten vállalatok között (extralogisztika), akár telephelyen belüli (intralogisztika) szinten. Ezen feladat ellátásában kulcsfontosságú szerepük van a raktárak ugyanis csak készletfelhalmozás segítségével lehet időbeli, illetve térbeli, különbségeket áthidalni. A raktárak ezen kitüntetett szerepe okozza azt, hogy folyamatainak működése jelentős befolyással bír az egész ellátási lánc működésére, hiszen az effektív raktározási megoldások lehetővé teszik a vállalkozások számára, hogy hogy optimalizálják az ellátási láncot, csökkentsék költségeket, növeljék a profitot és a versenyképességet.

Meg kell említeni, hogy a logisztikának súlyos felelőssége van a környezetszennyezésben és a klímaváltozási folyamatok felgyorsításában. Kritikusan fontos ezért, hogy a rendszerek hatékonyságát csökkentsük, hiszen ezzel a költségek mellett a környezetterhelés is csökkenthető.

1.1. A dolgozat célkitűzései

A dolgozatom célja a raktári folyamatok hatékonyabbá tétele optimalizáló algoritmusok segítségével. A dolgozat során bemutatok egy általam tervezett és fejlesztett rendszert mellyel tetszőleges topológiájú raktárt lehet lemodellezni. Implementálhatók benne a termékek és a hozzájuk tartozó megrendelési adatok. Lehetőséget nyújt kielemezni, hogy az adott paraméterekkel rendelkező raktári problémánál a folyamatokat támogató algoritmikus megoldások közül melyik milyen hatékonysággal támogatja a folyamatokat. Kiemelt figyelmet kapnak a tárhely-kijelölés és a komissiózási útvonal meghatározás problémái. A végső cél, egy olyan modellező rendszer létrehozása, mely használható ipari körülmények között, a raktári folyamatok fejlesztésénél.

1.2.Személyes motiváció

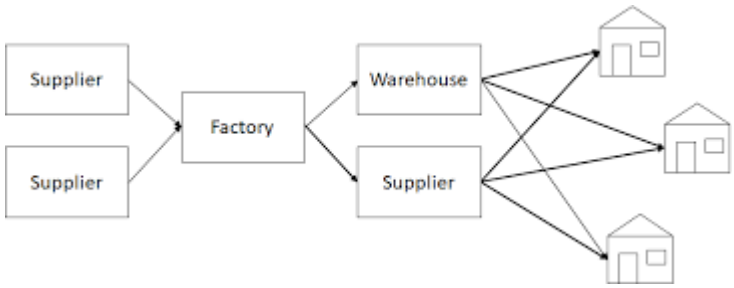
Abban, hogy a téma kutatásába kezdjek, két tényező játszott szerepet. Az egyik ok, hogy logisztikában dolgozó mérnökként az egyes folyamatok rendszerszintű optimalizálása a fő feladatom és kiemelten érdekelnek az informatikában továbbá a modern algoritmusokban rejlő lehetőségek. Azt tapasztalom, hogy az iparban, a termelési és logisztikai rendszerekben sok olyan fejlesztési lehetőség van, melyek nem igényelnek hatalmas tőkét vagy tudást, ugyanakkor komoly megtakarításokat lehet velük elérni. A másik ok, hogy egy külső cég felvetette számomra a tárhelykijelölési problémát, mint kutatandó kérdéskört, és minél inkább beleástam magam a témába, annál inkább megfogott.

2. A téma háttere és elméleti alapjai

A fejezetben bemutatom a raktárak folyamatait melyeket modellezni kell ahhoz, hogy leírhassuk egy raktár működését. Részletezem a folyamatok nehézségeit és kihívásait melyek hatékony megoldása kulcsfontosságú a raktár működése szempontjából.

2.1.Az ellátási lánc

Modern világunk gazdasági vérkeringését a logisztika által működtetett ellátási láncok alkotják. Az ellátási lánc (supply chain) „egy olyan hurok, mely a vevőnél kezdődik, és nála fejeződik be” [3]. A vevőnél kezdődik, hiszen ő határozza meg, hogy milyen termékekre/szolgáltatásokra van szükség. Ez az információ az adott termékek gyártóin majd azok beszállítóin keresztül lépcsőzetesen egyre „távolabb” kerül a primer vevőtől egészen a nyersanyagok szintjéig. Onnan aztán anyagáramlás indul el, mely folyamat során az anyagok számos cégen, műveleten keresztülmennek, mígnem végeredményként a vevő megkapja az általa kívánt terméket [3].



1. ábra: *Ellátási lánc*

Az ellátási láncban fontos szerepe van a raktáraknak. A raktárak két alapvető feladata, hogy áthidalják az igény és az előállítás közti időbeli és térbeli különbségeket. Hiszen, ha igény keletkezik valamire, nem valószínű, hogy akkor az a termék, azon a helyen, abban a pillanatban kerül legyártásra, amikor és ahol az igény felmerül, hanem egy raktárból elégítik ki az igényt mely raktárt ehhez üzemeltetni kell. Ez a folyamat épp úgy igaz lehet egy egyszerű vásárlóra, mint egy cégre [4].

A lean filozófia és a JIT elterjedésével az ellátási lánc szereplői törekednek egyre kisebb készleteket tartani, aminek hatására jelentősen csökkentek a raktárkészletek. Ugyanakkor semmilyen iparágban sem lehet teljesen raktármentes működést kialakítani, hiszen mindig lesznek ingadozások az egyes logisztikai rendszerekben, valamint mindig lesznek olyan kritikus fontosságú termékek, melyek rendelkezésre állása 100%-közeli kell, hogy legyen [4], [5].

A raktárkészletek méretének csökkenése nem jelenti a logisztikai rendszereken áramló áruk volumenének csökkenését. Épp ellenkezőleg, a rendszereken áramló áruk mennyisége nő, a készletek forgási sebessége pedig felgyorsul, így nincs szükség állandó nagy készletre. Ugyanakkor ez a felgyorsult forgási sebesség nagy plusz terhet jelent a raktári folyamatoknak, melyek így rákényszerülnek hatékonyságuk javítására két fő szempont, a költség és a kiszolgálási színvonal mentén. Utóbbit érdemes picit kifejteni. A kiszolgálási színvonal magában foglalja mindazokat az elvárásokat, melyeket a vevők támasztanak:

- Átfutási idő: kiemelten fontos, hogy egy megrendelés beérkezése és az áru kiszállítása között mennyi idő telik el. Amennyiben ezt az időt sikerül állandóan alacsonyan tartani, anélkül, hogy a többi paraméter kárt szenvedne, óriási előnyre tehet szert egy vállalat.
- Pontosság: kritikusan fontos, hogy az egyes megrendelések teljesítése során, a megfelelő anyagot, a megfelelő anyagi minőségében, minden szükséges információval ellátva szállítsanak ki. Természetesen még a legjobb rendszerekben is előfordulnak hibák, de ebben a paraméterben nagyságrendekkel kisebb azokban a rendszerekben a hiba, ahol megfelelő információs és identifikációs rendszer van kiépítve.
- Rugalmasság: a rugalmasság a raktárban végzett tevékenységek és folyamatok alkalmazkodó képességét jelenti a változó piaci igényekhez. Ha a raktár rugalmas, akkor jobban tud alkalmazkodni a piaci igényekhez. Ez a normál működés során is fontos, hiszen folyamatosan lehetnek ingadozások a vevői igényekben, ugyanakkor válsághelyzetekben még kiemelt előnyt tud jelenteni.
- Kiszámíthatóság: fontos a folyamatok működésének egyenletessége, hogy az előre meghatározott színvonalon mozogjon. Az ipari folyamatokat csak akkor lehet jól tervezni, ha kiszámítható folyamatok alkotják. Jó példa erre a JIT. Elképzelhetetlen lenne a működése, ha a beszállítók megrendelésátfutási idejében akárcsak néhány nap ingadozás lenne [3].

2.2.A raktározási rendszerek bemutatása

A darabárutárolási rendszereket három fő komponens befolyásolja:

- a tárolási egység,
- az anyagmozgatási technológia és
- a tárolási technológia

Ezek az elemek határozzák meg a raktár működését és erős kölcsönhatásban állnak egymással. A raktármodell tervezése előtt érdemes megvizsgálni ezt a három alrendszert, a kölcsönhatásaikat, valamint hogy az iparban mely megoldások fordulnak elő leggyakrabban [6].

A tárolási egységet darabárutárolási rendszerekben egységrakományok jelentik. A legelterjedtebbek a raklapos egységrakományok, ugyanakkor terjednek főként az automatizált rendszerek esetében az un. KLT-s rendszerek. Az ilyen rendszerek esetében a tárolási egységeket műanyag ládák alkotják. Ezekben a ládákban vannak elhelyezve az árucikkek [6]. Az árucikkek kapcsán érdemes megvizsgálni a készleteket és a megrendeléseik jellegét. A készletek szempontjából megkülönböztethetünk polistruktúrájú és monostruktúrájú raktárakat. Előbbi esetében a cikkszámféleségek aránya magas az egyes cikkszámokból tárolt mennyiségekhez képest, míg a monostruktúrájú készletek esetében aránylag kevés féle anyagból tárolnak jóval nagyobb volument. A készlet és a készletezési stratégia területén is számos vizsgálati szempont lehet. A dolgozat szempontjából releváns, hogy a szezonalitás, illetve a cikkszámok változása mennyire jellemzi a készletet [6].

Az anyagmozgatási technológiában megkülönböztetünk folyamatos és szakaszos működésű anyagmozgató rendszereket. Bár egyre több raktárban előfordulnak folyamatos működésű görgős pályák és egyéb szállító rendszerek, még mindig a legelterjedtebbek a szakaszos működésű targoncás rendszerek, raktári felrakótargoncákkal és homlokvillás targoncákkal [6].

A tárolási rendszer kapcsán is számos megoldás létezik. Megkülönböztethetünk statikus és dinamikus, állványos és állvány nélküli rendszereket. Minden rendszernek megvannak a saját előnyei és hátrányai, költség, tárhelykihasználás és folyamatidők szempontjából. A leguniverzálisabb eszköz a soros-állványos tárolási technológia. Ez az a klasszikus tárolási rendszer, amikor egy statikus állványrendszerben úgy vannak tárolva az áruk, hogy mindegyik raktárhelyhez, rakathoz hozzá lehet férni anélkül, hogy bármely más rakatot mozgatni kellene [4]

2. ábra: Sorosállvány, [7]

Ez a tárolási technológia különösen alkalmas polistruktúrájú raktárak esetében, ahol az áruféleségek aránya magas az egy áruból tárolt volumenhez képest. Előnye továbbá, hogy lehetővé teszi egyes raktári folyamatok automatizálását. Ugyanakkor hátránya a beruházási költség, továbbá a tárhelykihasználás szempontjából sem a legideálisabb, hiszen a sok közlekedő út helyigényessé teszi[6].

2.3. A raktározási folyamatok

2.3.1. A komissiózás

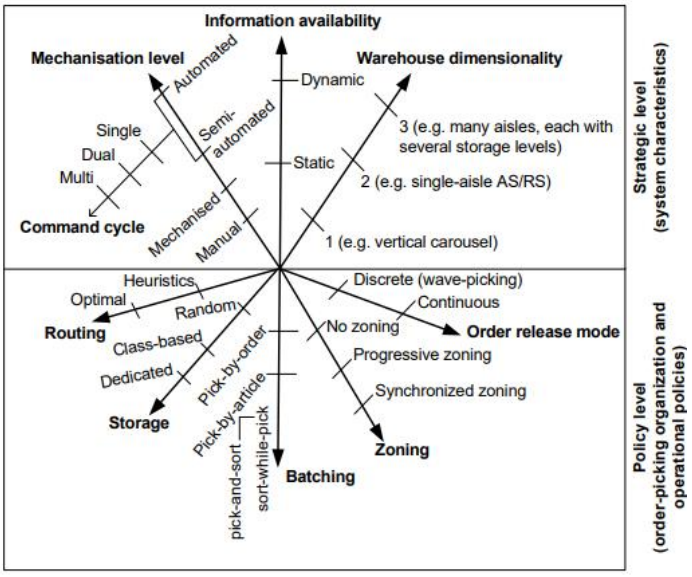
A raktári folyamatok közül a legfontosabb a komissiózás. A komissiózás az áruk kigyűjtése egy előre meghatározott lista alapján. A komissiózás során a beérkezett megrendelési listákhoz szükséges anyagokat kell valamilyen metodika szerint kigyűjteni. Jellemzően az egy anyagból érkezett megrendelésszám kisebb, mint a raktárban tárolt mennyiség, ez pedig a tárolási egységek megbontását, újra csomagolását jelentheti. A komissiózási folyamat jelentőségét mutatja, hogy a raktár költségeinek nagy részét-körülbelül ötvenöt százalékát jelenti. Ennek okai többek között, hogy nagy az élőmunka igénye, sok esetben bonyolult a folyamatszervezése és kiemelten fontos a munka pontos elvégzése, emiatt pedig szükséges sok beépített ellenőrző mechanizmus. A komissiózás sebessége, valamint pontossága döntően befolyásolja a logisztikai rendszer kiszolgálási színvonalát. Az átfutási időre is jelentős hatást gyakorol[6][8].

A raktári komissiózás kérdéskörével először T. Gudehus[9] foglalkozott a 70-es évektől kezdve. A komissiózási rendszerekben számos vizsgálandó kérdés felmerül, mint például:

- a komissiózó rendszer működési stratégiája,
- a komissiózási rendszer teljesítménye,
- a leadóhelyek pozíciója és a keresztfolyosók számának hatása a komissiózási időszükségletekre,
- a komissiózási útvonalak,
- a tárolási technológia,
- az anyagmozgatási technológia.

Magyar nyelven a témáról legátfogóbb mű Dr. Prezenski József Logisztika I-II című könyvei[4]. Továbbá Dr. Bóna Krisztián *Anyagmozgatási és raktározási folyamatok* című jegyzete is átfogó bepillantást nyújt a témába[6].

Ezen szakirodalmak megalapozták a téma tudományos vizsgálatát. Azóta a komissiózást számos szempont szerint vizsgálják. Az alábbi ***3. ábra*** a komissiózási feladat problémaköreit vizsgálja. A komissiózási folyamat fejlesztése több módon is történhet. Egyrészt a folyamat valamilyen szintű automatizálása, az információáramlás gyorsítása, un. komissiózási zónák kialakítása stb. Az egyes nyilak az egyes problémakörök, a nyilakon szereplő megoldások pedig a középponttól távolodva egyre növelik a folyamat komplexitását. Jól kialakított folyamatok esetében komplexitás mellett a hatékonyság is nő. A dolgozatban két fő irányt elemzek részletesebben, az útvonalválasztási, és a tárhelykijelölési problémák, hatékonyabb megoldását [10].



3. ábra: *a komissiózás fejlesztési lehetőségei* [10]

2.3.2. Komissiózást támogató fizikai eszközök

A komissiózás során számos eszköz segítheti a folyamatot. Érdemes a folyamatot részfolyamatokra bontani és úgy megvizsgálni:

- Információt kap a komissiózó ágens, hogy honnan és mennyi terméket kell kigyűjtenie.
- Mozog az adott lokációra.
- Kiveszi a tárhelyről a megfelelő mennyiségű terméket (közben ellenőrzi, hogy biztos az-e a termék és biztos jó-e a mennyiség).
- Adminisztrálja, hogy ki van véve az adott tétel.

Az információ-adás/vétel leginkább digitális eszközök segítségével wifin keresztül valósul meg. Gyakori a tabletek alkalmazása, ugyanakkor ez sok esetben körülményes, hiszen a komissiózás közben plusz mozdulatokat igényel. Hatékony és egyre inkább elterjedő megoldás az un. voice picking, mikor a komissiózó kolléga hang alapú utasítást kap és ad vissza. Ennek előnye, hogy ilyenkor a komissiózó munkatárs mindkét keze szabadon marad. Az un. pick by light rendszereknél az állványba LED-ek vannak beépítve, és egy fényjelzés, vagy az állványon lévő kijelző mutatja, hogy honnan és hány árut kell kigyűjteni. Ennek a rendszernek a kiépítése igencsak költséges, és a rendszer sok szempontból rugalmatlan, ugyanakkor igen jó teljesítményt lehet vele elérni[4], [6].

A mozgatás tekintetében számos megoldást használnak, amit egyrészt a komissiózás helyén lévő tárolási technológia másrészt az áruelőkészítés módja határoz meg. Dinamikus áruelőkészítés esetén az árut a rendszer odamozgatja a komissiózó kollégához (pl.: görgős pálya vagy körforgó páternoszter/shuttle rendszerek segítségével), míg statikus esetben a komissiózó kolléga megy az áruért, gyalog, komissió targoncával, homlokvillás targoncával stb.[4], [6].

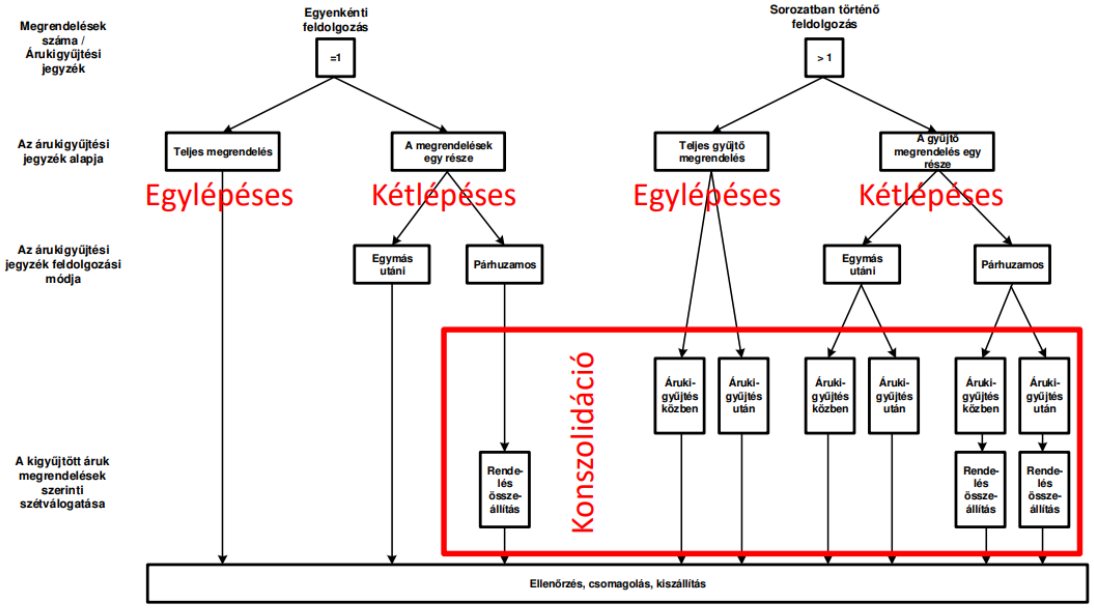
A mennyiségellenőrzés kritikusan fontos feladat, hiszen, ha a szükségesnél több vagy kevesebb áru lesz kigyűjtve, az számos problémát okoz a továbbiakban. Ugyanakkor a nem megfelelő ellenőrző részfolyamat lassíthatja is a komissiózást, ami hátrányos a folyamat egésze szempontjából. Az ellenőrzést leggyakrabban vagy technológiai támogatás nélkül végzi a komissiózó kolléga (ami gyakori hibához vezet), vagy vonalkód olvasós technikát alkalmaznak, ez azonban jelentősen lassítja a komissiózást. Modern rendszereknél már RFID-s technológiával vagy tömegméréssel támogatják a folyamatot, ezek a technológiák azonban drágák[6].

2.3.3. Komissiózást támogató folyamatszervezési megoldások

Az egymástól messze lévő tételek kiszedésének problémájára kézenfekvő módszer lehet, a több lépésben történő komissiózás. Ilyenkor több megrendelést összefognak, majd először kigyűjtik az egymáshoz közeli termékeket, utána pedig ezeket szétválogatják külön-külön az egyes listáknak megfelelően. Ez a módszer nagyobb komissiózási teljesítményt jelent, de a két lépés miatt, nagyobb és bonyolultabb a szervezési feladat, továbbá az átfutási időket is növeli [6].

A folyamat bővíthető a több megrendelés összefogásával, ami még tovább növelheti a komissiózási teljesítményt, de az előbbiekben leírt hátrányok is fokozódnak[4], [6].

A komissiózás hatékonyságát befolyásolja, hogy honnan történik az áruk kigyűjtése. illetve, hogy a komissiózás, a tárolótér és a betárolás helye elkülönül vagy megegyezik. A komissiózás történhet a tárolótérből, az áruelőkészítő téren és komissiózó raktárban. Ezt minden esetben egyedileg kell meghatározni, a forgalom, a hely és az elvárások függvényében[6][4].



4. ábra: *a komissiózás munkaszervezési lehetőségei* [6]

A ***4. ábra*** is jól szemléteti, hogy milyen sok féle folyamatszervezési megoldás kínálkozik a komissiózás feladatának megoldására. Azonban egyáltalán nem triviális kérdés, hogy milyen esetben melyik szervezési megoldás a leghatékonyabb. Ennek kiválasztásához számos mérés és elemzés szükséges[4]

2.3.4. Betárolás

A betárolás feladata a beérkezett áruk elhelyezése a raktárban. Érdemes a betárolást részfolyamatai mentén megvizsgálni:

- áru tárhelyhez rendelése,
- áru tárhelyre mozgatása,
- áru elhelyezése és adminisztrálása a tárhelyen.

Az első pont az áru tárhelyhez rendelése az, ahol a tárhelykiosztási probléma felmerül, ugyanakkor ennek következménye itt még nem jelenik meg, hiszen a betárolási folyamat hatékonyságát döntően nem befolyásolja (betárolásnál nagyobb volumeneket mozgatnak, és szabadon lehet összefogni a betárolási tételeket, így a tételre vetített hatékonysága sokszorosa a komissiózásénak). A tárhely kiválasztása sok esetben még mindig a betáras kolléga egyéni döntése alapján történik, azonban a modern vállalatirányítási rendszerek esetében már kijelölik a kolléga számára a tárhelyt. Ez a SLAP megoldó algoritmusok számára kulcskérdés, hiszen ott mindenképp a rendszernek kell előírnia, hogy az áru milyen tárhelyre kerüljön. [4], [6]

Az áru tárhelyre történő mozgatása a tárolási technológiától függ. Raklapos, állványos technológiánál a targoncák a legelterjedtebbek, de a kézikocsitól az automata felrakógépig számos megoldás létezik[4], [6]

Az adminisztrálás a vállalatirányítási rendszertől és az alkalmazott azonosítási technikától függ. Jellemzően az adott információs rendszerben történik, manuálisan, egy vagy kétdimenziós vonalkódok beolvasásával, esetenként RFID technológiával[6].

Egy raktározási rendszerben számos más folyamat is jelen lehet a komissiózás és a betárolás mellett. A leggyakoribbak a csomagolás, a kamionrakodás és a minőség-ellenőrzés azonban ezen folyamatok költsége lényegesen kisebb így az optimalizálásnak is kisebb megtakarítási lehetőségei vannak[4]

2.4. Útvonalkereső és tárhelykiosztó algoritmusok rövid ismertetése

A következő fejezetekben olyan algoritmusokat mutatok be, melyek igazoltan használhatóak a különböző raktári problémák során. Ezen algoritmusok közül fogok többet is implementálni a tervezett rendszerben.

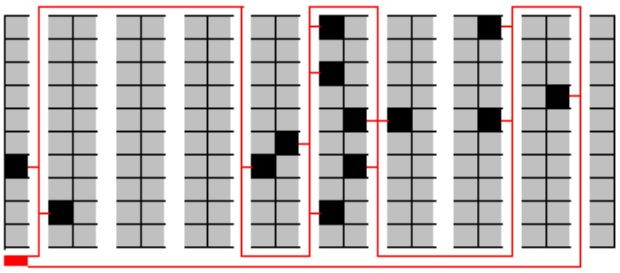
2.4.1. Útvonalkereső algoritmusok

Az útvonalkereső algoritmusok feladata a komissiózó ágens számára meghatározni az érintendő tárhelyek felkeresési sorrendjét. Ez a probléma az utazó ügynök probléma egy speciális esete melyről bizonyították, hogy egy NP nehéz probléma, vagyis az egzakt megoldása nem polinom időben számolható. Azonban számos heurisztikus, metaheurisztikus és egyéb megoldási módszer alkalmazható a probléma megoldására. Az útvonal kereső algoritmusok szakirodalma jelentős. Számos kutató foglalkozott már a problémával, melynek oka az, hogy az utazó ügynök probléma igen gyakran fordul elő a különböző optimalizáló feladatok között [8]

A problémára a gyakorlatban számos megoldást alkalmaznak, kezdve a teljesen szabályozatlan folyamattól, ahol a dolgozók döntései határozzák meg az útvonalat, a különböző heurisztikákon és metaheurisztikákon át, a bonyolult szimulációkon alapuló optimalizáló módszerekig [8]. Én a dolgozatomban két gyakran használt heurisztikus megoldást mutatok be az S-shape és a Largest gap algoritmust.

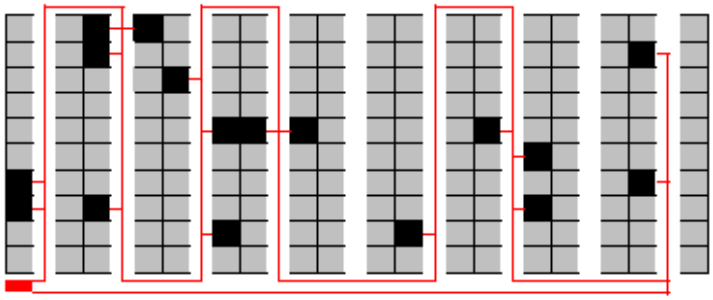
2.4.2. S-shape algoritmus

Az S-shape heurisztika az egyik legegyszerűbb és leggyakoribb módszer az komissiózási útvonal meghatározás során. A módszer neve utal a működési elvére: a komissiózó végighalad az első árutároló soron. A hátsó keresztfolyosónál elfordul és abba a legközelebbi sorba megy be, ahol a következő kigyűjtendő cikk van. A sorokon melyekbe befordult mindig végig halad (kivétel lehet az utolsó kigyűjtendő cikk), majd a sor végén ismét elfordul és halad tovább a következő sorig, ahol van kigyűjtendő árucikk. Az út, amelyen a munkavállaló halad, az "S" betűre emlékeztető alakot ölt [11].



5. ábra: *Az S-Shape algoritmus használata egy tárhelyblokk esetében*

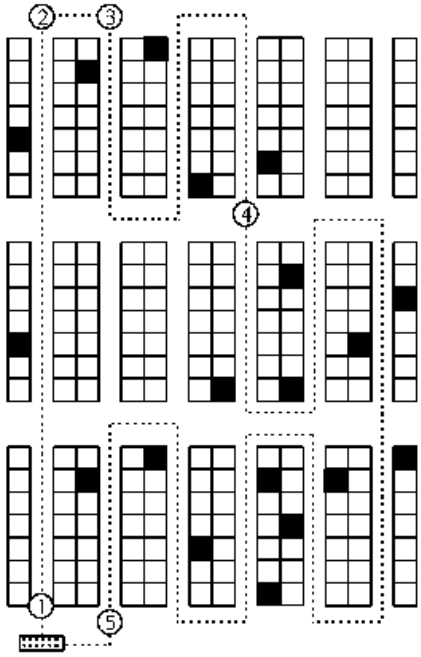
Az ***5. ábra*** és a ***6. ábra*** példákat mutatnak az S-shape heurisztika használatára a komissiózás során (fekete színnel jelölt négyzetek a rendelések). A ***6. ábra*** által szemléltetett példában a komissiózó kiindulópontja megegyezik ahol a kész rendelést átadja (piros téglalap a bal alsó sarokban). A munkavállaló az komissiózás során az első sorban kezdi az áruk összegyűjtését, ahol két árucikk található, majd tovább halad a sor végéig. Ezután kihagy két olyan sort, ahol nincs felvenni való árucikk, majd folytatja a munkát az áruk felvételével kapcsolatos tovább sorok mentén (amelyekben vannak felvenni való árucikkek) azokat a sorokat kihagyva, ahol nincs felvenni való árucikk. Miután az utolsó árutároló soron áthaladt, amelyben a rendelt árucikkek találhatók, a munkavállalót a teljes rendelés felvételi pontjára irányítják. A ***5. ábra*** az S-shape heurisztika (1) módszerét mutatja be [11]



6. ábra: *Az S-shape algoritmus, egy blokkos raktáresetben b-verzió*

A ***6. ábra*** azt a helyzetet szemlélteti, amikor a komissiózó ágensnek nem kell az árutároló sor végéig mennie, hogy az árucikket felvegye. Ilyen helyzet csak az utolsó árutároló sorban fordul elő (ahonnan az árucikkeket fel kell venni), amikor a felvevő pont azon az oldalon van, ahová a munkavállaló belép a sorba [11].

Érdemes megvizsgálni az S-shape heurisztika működését olyan esetben amikor nem egy blokkból áll a raktár (kettőnél több keresztfolyosó van). A heurisztika működését ebben az esetben a ***7. ábra*** mutatja.



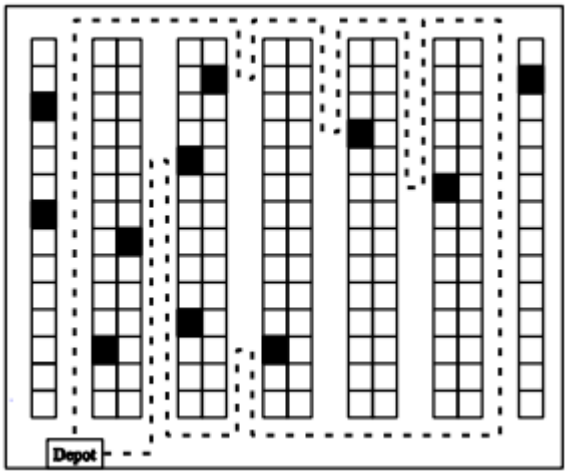
7. ábra: *Az S-shape algoritmus többblokkos raktártopológiánál*

A rendelések kiválasztásának útvonala ilyenkor is egy I/0 pontból kezdődik. Nulladik lépésben az első olyan átjáróhoz megy a komissiózó ágens, ahol legalább egy termék található és az a legközelebb van a kezdőponthoz [12].

1. Ezt az átjárón végig halat, egészen azon blokkig, amely a legtávolabbi a raktártól és amelyben van legalább egy termék
2. Ha a jelenlegi blokkban van legalább egy termék, akkor el kell jutni az olyan átjáróhoz, amelyen a legközelebb található termékek vannak, legyen az balra vagy jobbra (jelen esetben jobbra megyünk tovább)
3. Az összes olyan átjárón végig halad, ahol vannak termékek, majd visszatér a blokk elejére
4. Ha a blokkban nincs termék, akkor azt az átjárót választja, amely a jelenlegi pozíciójához a legközelebb van. Ezt a folyamatot minden blokkra elvégezi, amíg el nem érjük a raktárhoz legközelebbi blokkot
5. Végül visszatér a kezdőpontra[12].

2.4.3. Largest gap heurisztika

A largest gap heurisztika egy, az S-shape heurisztikánál jóval kifinomultabb algoritmus, ugyanakkor ez nem jelenti azt, hogy minden esetben, hasznosabb is. Alap ötlete, hogy megkeresi a sorokban a legnagyobb szakaszokat a kigyűjtési pontok között és ezeket a szakaszokat kihagyjuk az útvonalból [13].



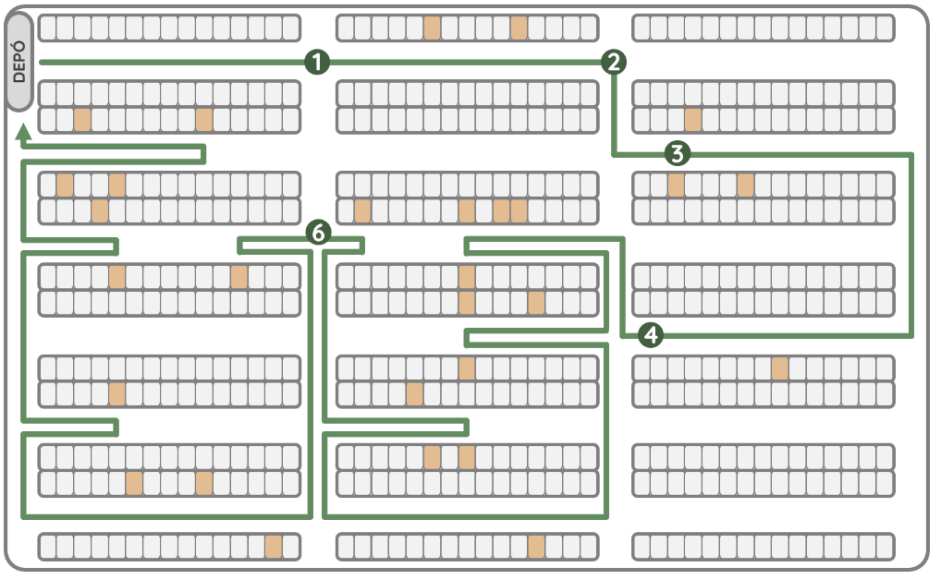
***8. ábra**: Largest gap heurisztika egy blokk esetében* [13]

A ***8. ábra*** és az alább szereplő pszeudokód segítségével könnyen értelmezhető az algoritmus működése. Az algoritmus általában átlagosan 7-8 százalékponttal jobban teljesít az S-shape algoritmusnál. Előnye továbbá a modell egyszerűsége, ezáltal könnyen implementálható és a futásideje is alacsony. Ugyanakkor ez az algoritmus sem garantálja, hogy megtalálja az optimális megoldást, továbbá lehetnek olyan egyedi jellegű esetek, amikor a heurisztika kifejezetten rosszul teljesít [13].

Largest gap algoritmus

```
Input: raktártopológia, kigyűjtendő termékek száma, kigyűjtendő
termékek helye

végig menni az első olyan soron, ahol van kigyűjtendő anyag
       kigyűjteni a sorban lévő összes anyagot
ciklus <nem-e ez az utolsó terméket tartalmazó sor?>
       haladni a következő sorhoz
       ha <van a sorban kigyűjtendő termék>
            kiszámolni, hogy hol van a legnagyobb hézag
            a hézag határáig kigyűjteni az összes terméket
      elágazás vége
ciklus vége

végig haladni a soron és kigyűjteni az összes terméket

ciklus <van-e még kigyűjtendő termék?>
       haladni a következő sorhoz
       ha <van a sorban kigyűjtendő termék>
            kiszámolni, hogy hol van a legnagyobb hézag
            a hézag határáig kigyűjteni az összes terméket
      elágazás vége
ciklus vége
vissza a I/O pontra

eljárás vége

Output: bejárási útvonal
```

Érdemes megvizsgálni a heurisztika működését több blokk esetében is. Ilyen esetben az algoritmus működése bonyolódik. Ezt mutatja be a ***9. ábra***.



9. ábra: *Largest gap heurisztika alkalmazása több blokkra*[14]

1., 2., 3. lépés: meg kell keresni a legtávolabbi blokk legelső kiszedési pontját tartalmazó sorát, annak bejáratáig el kell menni olyan kiszedési soron keresztül, amely tartalmaz kiszedési pontot és a depóhoz a legközelebb van [14].

4. lépés: végig kell csinálni a fentebb már leírt blokkon belüli Largest gap algoritmust:

- Ha adott blokkban csak egyetlen - kiszedési pontot is tartalmazó - alfolyosó van, akkor abba be kell menni, a kigyűjtést el kell végezni, a folyosón vissza kell fordulni és vissza kell térni a tárolóblokk elülső keresztfolyosójára.
- Ha több ilyen alfolyosó van, akkor az elsőn teljesen át kell haladni (közben a kiszedéseket elvégezni), majd a largest gap módszer szerint először a blokk hátsó keresztfolyosójáról, majd a blokk elülső keresztfolyosójáról kiindulva kell a kigyűjtéseket elvégezni. Az utolsó folyosón teljesen végig kell haladni[14].

5. lépés: a legtávolabbi blokkal végezett az algoritmus, most az egyel korábbi blokk hátsó keresztfolyosójánál tart. Meg kell határozni, hogy merre induljon tovább:

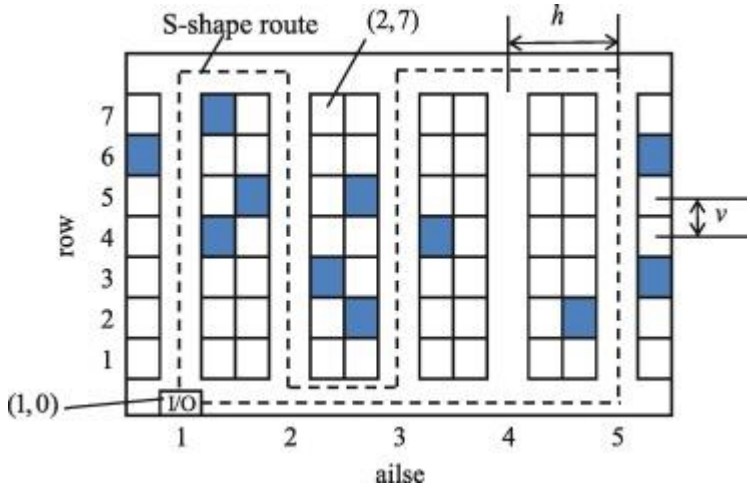
- Ha az egyel korábbi blokkban nincs kiszedés, akkor csak haladjunk keresztül a legközelebbi alfolyosón.
- Ha csak egy alfolyosó tartalmaz kiszedési pontot, akkor menjünk oda és azon menjünk keresztül.
- Ha több - kiszedést tartalmazó - folyosó is van, akkor a tőlünk legtávolabbi (két szélső) kiszedést tartalmazó folyosók közül a közelebbit válasszuk, menjünk el annak bejáratáig [14].

6. lépés: az algoritmus elvégzi ismét a largest gap módszert: először az új blokk hátsó keresztfolyosójáról, majd az elülső folyosójáról végezve a kiszedéseket.

- Ha van még felkeresendő tárolótéri blokk, akkor ezután újfent az 5. lépés következik.
- Ha nincs több tárolótéri blokk, akkor pedig térjünk vissza a depóba [14].

2.4.4. Tárhelykiosztó algoritmusok – a tárhely kijelölési probléma

A raktári tárhelyallokációs probléma (storage location assignment problem, ezentúl SLAP) alapja az a kérdés, hogy „hogyan rendezzem el a raktárban a termékeimet úgy, hogy komissiózásnál a lehető legközelebb legyenek egymáshoz a kigyűjtendő termékek. Tételezzük fel, hogy van x_1, x_2, ...x_k termékünk, illetve van egy háromdimenziós raktárunk tárhelyekkel $y_{i,j,z}$ ahol j a sorok száma i az oszlopok száma és z az oszlopok magassága. (Olyan eseteket vizsgálok majd, amelyekben a tárhelyek száma meghaladja az elhelyezendő termékek számát.) Továbbá vannak L listák, melyekben x_i termékek vannak $L=\{ x_a, x_b, ...x_r \}$. L tetszőleges számú terméket tartalmazhat x-ből. A SLAP fő kérdése az, hogy hogyan helyezzem el x_1, x_2, ...x_k termékeket $y_{i,j,z}$ állványrendszerben, hogy az L listák kigyűjtési ideje/távolsága (lehet más paraméter is, de ez a legjellemzőbb) kedvező legyen. A ***10. ábra*** szemléltet egy komissiózást, mely során egy valamilyen meghatározott útvonalbejárási algoritmus alapján a komissiózó ágens felkeresi azokat a tárhelyeket, ahonnan a listán szereplő adott mennyiségű elemet ki kell gyűjtenie [15].



10. ábra: *lista komissiózás raktárból S-shape útvonal alapján* [16]

Az általam talált legrégebi forrás, ami a tárhelykiosztási problémával foglalkozik, W. H. Hausmann: *Optimal storage assignment in automatic warehousing system* című 1976-ban közölt cikke, melyben részletesen definiálja a problémát és hatékony metódusokat ír le annak megoldására [14]. Az egyik ilyen megoldásnak az osztályalapú tárolást írja le, melynek egyik továbbfejlesztett fajtáját, a pareto elvre épülő ABC tárhelykiosztást én is modelleztem egy a továbbiakban bemutatott tanulmány által leírt módon[17] .

A SLAP-ot E. H. Frazelle 1989-ben egy NP nehéz problémának minősítette a termékek száma és a raktár tárolási jellemzői miatt. A probléma tovább bonyolódik, ha a termékek és a raktárhelyek száma megegyezik. Ebben az esetben kvadratikus allokációs problémaként definiálták[18].

Leginkább az osztályalapú, a dedikált és a véletlenszerű tárhelykiosztás elterjedt,[4] ugyanakkor számos modernebb megoldás is született a problémára az elmúlt évtizedben. Az *International Journal of Industrial Engineering Computations* című folyóiratban 2019-ben megjelent egy cikk, amely feldolgozza a téma addig megjelent szakirodalmát, a SLAP

problémára alkotott modern algoritmusok tekintetében. [15] A cikk szerzői 2005 és 2017 között megjelent 71 tanulmány eredményét dolgozták fel. Több szempont szerint csoportosították a módszereket, mint például az optimalizálni kívánt paraméter (távolság, idő, infrastruktúraterhelés, emberi tényezők stb.), továbbá az alkalmazott módszer alapján (egzakt, heurisztikus, metaheurisztikus, szimulációalapú, stb…). [15] E tanulmány alapján vizsgáltam meg több módszert, melyekből kiindulva alkottam meg a saját algoritmusomat. Továbbá kiválasztottam pár algoritmust, melyekkel össze is tudtam hasonlítani a módszeremet.

Több kutatást is találtam, melyek használtak a SLAP megoldására genetikus algoritmusokat. Az Atlantis Press folyóiratban 2017-ben megjelent egy tanulmány „Optimization of Automated Warehouse Location Based on Genetic Algorithm" címmel. A tanulmány sok fontos észrevételt tesz a genetikus algoritmusokkal és azok hatékonyságával kapcsolatban, ugyanakkor mivel az optimalizálandó cél a jobb tárhelykihasználtság és csak speciális automata raktárakat vizsgál a modelljeiben, ezért a kutatása jelentősen eltér az általam vizsgálandó általánosabb esetektől[19]. Egy 2019-es amerikai tanulmány hatékonyan alkalmazta a genetikus algoritmusokat a komissiózás támogatására speciális gördülő állványos automata rendszerekben. A tanulmány tapasztalatai sokat segítettek abban, hogy kialakítsak egy általánosabb (nem feltétlen automata) rendszert, melyben sorosállványos tárolási technológiát alkalmazok. (A sorosállványos tárolási technológia jóval elterjedtebb, mint a gördülőállványos.) [20]

Két, a sajátomtól meglehetősen eltérő módszer keltette fel leginkább érdeklődésemet. Ezeket a módszereket választottam ki arra, hogy összehasonlítsam őket a saját algoritmusommal. Az egyik egy 2011-ben megjelent tanulmány, amely részletesen leírja, hogyan alkalmazzák a pareto elvre épülő ABC-analízist osztályalapú tárhelykiosztásra, amivel jelentős eredményeket érnek el[17]. A másik tanulmány *Data mining based storage assignment heuristics for travel distance reduction* címmel jelent meg 2012-ben. Egy apriori algoritmusra épülő heurisztikus megközelítést mutat be, mellyel szintén a SLAP probléma megoldására nyújt egy módszert[21]. Ez utóbbit azért választottam ki, mert merőben más oldalról próbálja megközelíteni a problémát, így új szemszögből is megvizsgálhatom azt.

2.4.5. Genetikus algoritmusok

Az optimalizálás a logisztika alapvető feladata. Több paraméter változtatása által kínált számos lehetőség közül ki kell választani a legmegfelelőbbet. A lehetséges megoldások alkotják az un., keresési teret (search space). A célállapotot pedig a célfüggvény felírásával és megoldásával próbáljuk meg meghatározni [22]

Az 1950-es 60-as években matematikusok kezdték el kutatni az evolúciót úgy, mint egy nagyon sok paraméterrel rendelkező problémára való optimalizációs folyamatot. Az 1990-es évek elején pedig kialakult az evolúciós számítás tudományterülete, melynek három fő változata van: [23]

- Genetikus algoritmusok
- Evolúciós stratégiák
- Evolúciós programozás

Ezek közül, a genetikus algoritmusok terjedtek el a legszélesebb körben. A GA-k működése a darwini evolúciós elméleten és a genetika alapjain nyugszik. [24] A genetikus algoritmusok

alapjait először John Holland publikálta 1975-ben, utána pedig sorra jelentek meg a tanulmányok a GA-k tulajdonságairól és felhasználási lehetőségeiről[8]. Azóta számos optimalizációs problémánál kipróbálták, például a B. Molnár PhD dolgozatában a tárhelyfelkeresési problémára alkalmazza, de a mérnöki tudományok más területein is alkalmazzák[8].

Ahogy a szakirodalom bemutatásakor is látható, a problémát számos módon próbálják megoldani. Az ipari alkalmazásban Magyarországon az alábbi három módszer a legelterjedtebb:

- osztály alapú tárhelykiosztás,
- a dedikált tárhelykiosztás és
- a véletlenszerű tárhelykiosztás.

Ezek közül az osztályalapú tárhelykiosztásnak egy modern – a 2.4.6-os fejezetben már bemutatott– 2011-es tanulmányra alapuló pareto elv szerinti un. ABC tárhelykiosztási logikát fogok implementálni, [25] valamint a véletlen kiosztást fogom vizsgálni, mivel az ABC tárhelykiosztás bizonyítottan hatékony módszer, a véletlen tárhelykiosztás pedig jó kontrollmérésnek. A kutatásaim során olvastam több más algoritmusról is, amelyek a SLAP-ra nyújtottak megoldást. Ezek közül David Ming–Huang Chiang, Chia-Ping Lin és Mu-Chen Chen 2012-es apriori datamining algoritmusra épülő heurisztikáján alapuló metódusát implementálom[21], mivel így egy a többitől alapjaiban eltérő algoritmust és gondolatmenetet is tudok tesztelni.

2.4.6. A véletlen és az ABC tárhelykiosztás

A véletlen tárhelykiosztás során a betároláskor a beérkezett termék (tfh. még nincs a raktárban olyan termékből) véletlenszerűen kap egy tárhelyet, ahova be kell tárolni. A rendszer előnye, hogy ezáltal a termékek egyenletesen lesznek elosztva a raktárban.

Az ABC tárhelykiosztási stratégiánál a termékeket szortimentanalitikai eszközökkel ABC csoportokra osztjuk a Pareto-elv alapján. A Pareto-elv azt mondja ki, hogy általában az áruk 20 %-ka adja a forgalom 80%-át. Fontos, hogy ennél a problémánál felkeresés alapú ABC analízist kell végezni, vagyis azt kell vizsgálnunk, hogy az egyes termékek hány kigyűjtési listán szerepelnek, függetlenül a kigyűjtési volumenektől és az áruk értékétől. A felkeresések 80%át adó termékek lesznek A-kategóriájúak, a 15%-át adó termékek B-k, és az 5%-ot adók pedig C-s termékek. [25]Az algoritmust pszeudokódja:

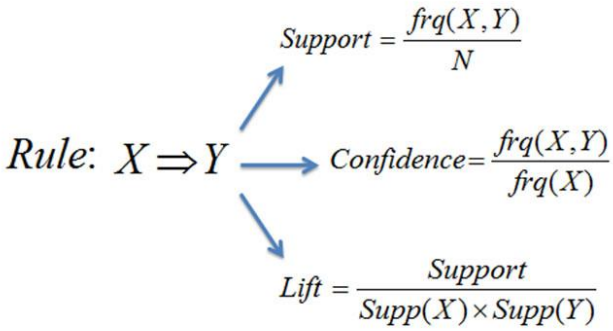
ABC tárhelykiosztó algoritmus

```
Input: tárhelyek_A, tárhelyek_B, tárhelyek_A, termékek
<i> = <0>

ciklus <0-tól termékek számáig>
        ha < termékek[i].kategóriája=A >
            <r> = <random 0-tól tárhelyek_A hosszáig>
                ha<tárhelyek_A[r]= üres>
                   termékek[i]betárolás tárhelyek_A[r] tárhelyre
                elágazás vége
            elágazás vége
        ha < termékek[i].kategóriája=B >
        …
        ha < termékek[i].kategóriája=C >
        …

ciklus vége
eljárás vége

Output: összerendelt termékek-tárhelyek
```

Az áruk mellett a tárhelyeknek is kategóriákat kell adni. A legkönnyebben elérhető tárhelyek lesznek A-s kategóriájúak, a kevésbé a B-sek és a legnehezebben elérhetőek a C-sek. Az áruk a betárolásnál csak a saját kategóriájuknak megfelelő tárhelyre kerülhetnek, azon belül viszont véletlen, hogy pontosan melyikre.

2.4.7. Az apriori datamining technikán alapuló heurisztika

Ennél a technikánál először elvégzem a tanítóhalmazon az apriori algoritmust, melynek segítségével minden termékpárhoz (amely definiálható mértékű) meghatározom az algoritmus un. „confidence” értéket. Ez az érték bemutatja, hogy milyen szoros a kapcsolat a kéttermék között, „mennyire valószínű, hogy szükségem van B-termékre is, ha A-termékre szükségem van”. Ezt az apriori algoritmust minősített internetes kódok és az R szoftver beépített algoritmusai segítségével hoztam létre R-ben. [26]



***11. ábra**: apriori algoritmus eredményei* [25]

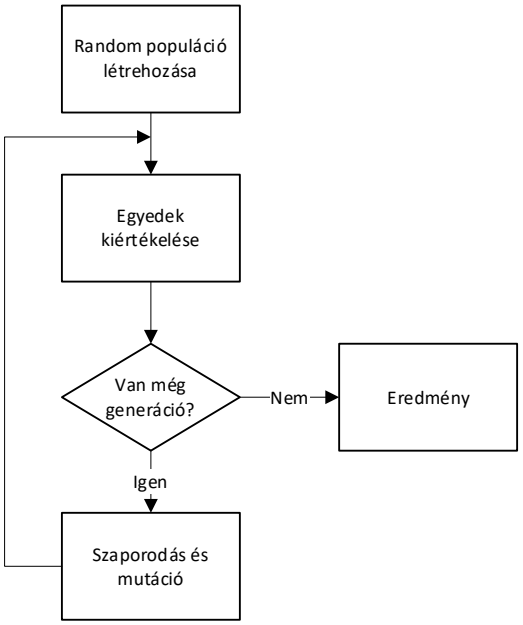
A **11. ábra** mutatja az apriori algoritmus által számolt legfontosabb értékeket.

- A „support” egy termék alapértelmezett népszerűségére utal, úgy számítható ki, hogy az adott terméket tartalmazó tranzakciók számát elosztjuk a tranzakciók teljes számával.
- A „confidence” annak valószínűségére utal, hogy egy B terméket is kigyűjtök, ha A terméket is. Ez úgy számítható ki, hogy azon tranzakciók számát, ahol A-t és B-t is kigyűjtötték, elosztjuk azon tranzakciók teljes számával, ahol A-t kigyűjtötték.
- A Lift alapvetően azt mondja, hogy a A és a B együttes megvásárlásának valószínűsége x-szer nagyobb, mint annak, hogy csak a B-t vásároljuk meg.

Miután meghatároztam a termékek „confidence” értékeit, szükséges, hogy a raktárban már legyenek áruk, úgyhogy betárolok pár árucikket. (Érdemes ABC logika alapján betárolni, mert a véletlen betárolásnál gondot okozhat, hogy nagy forgalmú, fontos termékek, amelyek számos más termékkel szoros kapcsolatban vannak előnytelen helyre kerülnek, és mivel szoros kapcsolatban vannak más adott esetben szintén fontos termékkel, azokat is „magukhoz” vonzzák). Utána pedig, amikor egy újabb terméket akarok betárolni, meghatározom, hogy az adott sorban már bent lévő áruk, ehhez a most betárolandó áruhoz kapcsolódva milyen „confidence” értékekkel rendelkeznek. Ezt összegzem soronként, és ha van szabad tárhely abban a sorban, ahol ezen értékek a legmagasabbak, akkor oda teszem a termékemet. Amennyiben nincs szabad tárhely, egy szomszédos sorba igyekszem elhelyezni az árut. Előfordul olyan eset, amikor nincs definiálva „confidence” érték a betárolandó termék és a sorokban lévő termékek között (mert például kimutathatatlanul gyenge a kapcsolat). Ilyen esetekben véletlenül helyezem el a terméket a raktárban [21].

2.4.8. Az genetikus tárhelykiosztó algoritmus működési elve

A genetikus algoritmus, amelyet írtam, alapvetően nem különbözik az általános genetikus algoritmusoktól, csupán a problémára szabtam. Az algoritmusnál egy egyed paraméterezhetően sok „kromoszómát tartalmaz”. Minden kromoszóma egy adott termékhez egy tárhelyet rendel hozzá. Így a kromoszómák száma megegyezik az elhelyezni kívánt árucikkek számával; esetünkben. Az algoritmusnál paraméterezhető az egyedszám, a generációk mértéke, a mutáció mértéke és az algoritmus elitizmusa.



12. ábra: *A genetikus algoritmus folyamatábrája*

A genetikus algoritmus lefutásának folyamatát az **12. ábra** mutatja be. Az első generáció előtt véletlenszerűen létrejön az összes egyed. Egy generáció lefolyása három részből áll: szelekció, szaporodás, mutáció. A szelekció során, minden egyed „minőségét” kiértékelem a fitneszfüggvény segítségével. Kétféle fitnesz-függvénnyel dolgoztam, mivel egyes tesztesetekben a kigyűjtési távolságra optimalizáltam más esetekben a kigyűjtési időre. A fitneszfüggvény mindkét esetben a múltbeli, már ismert megrendelések alapján számolt. A fitneszfüggvény kiértékelése során megrendelések halmazából választott tanítóhalmaz megrendeléseit gyűjtötte ki minden egyednél a program, és meghatározta az azokhoz szükséges utat/időt. Miután a fitneszfüggvény kiértékelte az egyedeket (meghatározta a kigyűjtési utakat/időlet), azok egy része „elpusztul” (hogy mekkora részük azt az elitizmus paramétere határozza meg, ami a modellem esetében 50%ra volt beállítva). Ezzel lezárul a szelekció.

A szaporodás fázisában a populáció „életképesebb” feléből újra létrejön a populáció elpusztult egyedei helyére egy-egy egyed. Miden új egyed miden kromoszómáját véletlenszerűen egy, a populáció életben maradt egyedétől kapja. Miután létrejöttek az új egyedek, a mutáció fázisába lépünk, mely során az új egyedek egy kis százalékánál véletlenszerűen megcserélődnek kromoszómák (hogy melyik termékhez melyik tárhely tartozzon). A mutáció lényege, hogy elkerülje a lokális minimumokban való beragadást. A mutáció után egy újabb generáció következik, mely során újból szelektálják az egyedeket és így fut tovább annyi generáción át amennyit előre megadtunk. A végeredmény az utolsó generáció legjobb egyede lesz. Alább látható egy generáció egyszerűsített pszeudokódja:

GA tárhelykiosztó algoritmus egy generációjának egyszerűsített pszeudokódja

```
Input: egyedek

szelekció
      <i> = <0>
      ciklus <0-tól egyedek számáig>
            egyedek[i] fitneszértéke=kigyűjtési erőforrás
            i++
      ciklus vége
      egyedek növekvő sorba rendezése
eljárás vége;

szaporodás
      <i> = <egyedek száma/elitizmus >
      <r> = random 0-egyedekszáma/elitizmus között
      ciklus <egyedek száma/elitizmus mértéke-től egyedek száma-ig>
            <j> = <0>
            ciklus <kromoszómák száma>
                  egyedek[i] kromoszóma[j]= egyedek[r].kromoszóma[j]
                  j++
            ciklus vége
      ciklus vége
eljárás vége;

mutáció
      ciklus <0-tól mutációs paraméterieg>
            egyedek[rand] kromoszóma [rand2]=egyedek[rand3]
            kromoszóma [rand4]
            i++
      ciklus vége
eljárás vége;
Output: összerendelt termékek-tárhelyek
```

2.5. Algoritmusok összehasonlítása

Érdemes összevetni az előző fejezetekben említett algoritmusok előnyeit és hátrányait előnyeik és hátrányaik, illetve hatékonyságuk és bonyolultságuk alapján. Ezt az összevetést az alábbi táblázatok mutatják be:

1. táblázat: Útvonaltervező algoritmusok

	Útvonaltervező algotirmusok	
	S-shape	Largest gap
Átlagos hatékonyság	Közepes	Jó
Bonyolultság	Egyszerű	Közepes
Fő erőssége	Gyorsan implementálható	Magas hatékonyságú heurisztika
Fő gyengesége	Bizonyos esetben rossz a hatékonysága	Komolyabb informatikai támogatást igényel

Mindkét általam kiválasztott heurisztika már bizonyított az iparban. A tapasztalatok alapján az ilyen egyszerűnek tűnő algoritmusok bevezetése sem egy könnyű feladat a raktározási rendszerek esetében. Sok esetben érdemes lépcsőzetesen bevezetni ezeket a megoldásokat, az egyszerűbbel kezdve és annak elsajátítása után haladni a következő lépcsőfok felé. A bevezetés

előtt mindenképp meg kell vizsgálni, hogy az adott algoritmus bevezetése számszerűen milyen előnyökkel jár és milyen nehézségeket okoz. Például abban az esetben ha az adott rendszeren mindkét algoritmus közel ugyan olyan jól teljesít és a rendszer még fejletlen nem érdemes a Largest gap algoritmust használni, ellenben ha nagy plusz megtakarításokkal járna érdemes lehet egyből abban gondolkodni. A döntéshez szükséges számszerű paraméterek meghatározását pedig nagyban megkönnyíteni egy modellező rendszer [4], [6], [8], [13], [22]

A következő táblázatban a tárhelykiosztó algoritmusokat hasonlítom össze:

2. táblázat: Tárhelykiosztó algoritmusok

	Tárhelykiosztó algoritmusok	
	ABC	Genetikus
Átlagos hatékonyság	Jó	Kimagasló
Bonyolultság	Egyszerű	Bonyolult
Fő erőssége	Gyorsan implementálható, általános esetben hatékony	Egyedi esetekben is nagyon hatékony
Fő gyengesége	Speciális esetekben, alacsony lehet a hatékonysága	Komoly informatikai támogatást igényel

A tárhelykiosztó algoritmusokkal már korábbi munkáim során is sokat foglalkoztam. Saját tapasztalataim és a szakirodalom is alátámasztja, hogy az ABC algoritmus egy igen hatékony és nagyon egyszerű algoritmus. Ugyanakkor azt tapasztaltam, hogy minél bonyolultabb egy modell, egy feladat annál inkább lemarad a genetikus algoritmussal szemben, hiszen egyszerűségéből kifolyólag nem képes egyedi szempontokat figyelembe venni[6], [8].

3. A modellezési környezet tervezése és felépítése

A modellezési környezet tervezése és fejlesztése során igyekeztem megfelelni az ipar és a tudományos világ által támasztott követelményeknek. A legnagyobb hangsúlyt a felcserélhetőség, változtathatóság, a modularitás kapta, hiszen az iparban fontos, hogy a lehető legtöbb helyzetet képesek legyünk lemodellezni, a kutatások során pedig előnyös, ha egy-egy problémára egy-egy modellhez akár több megoldó algoritmust is tudunk kapcsolni.

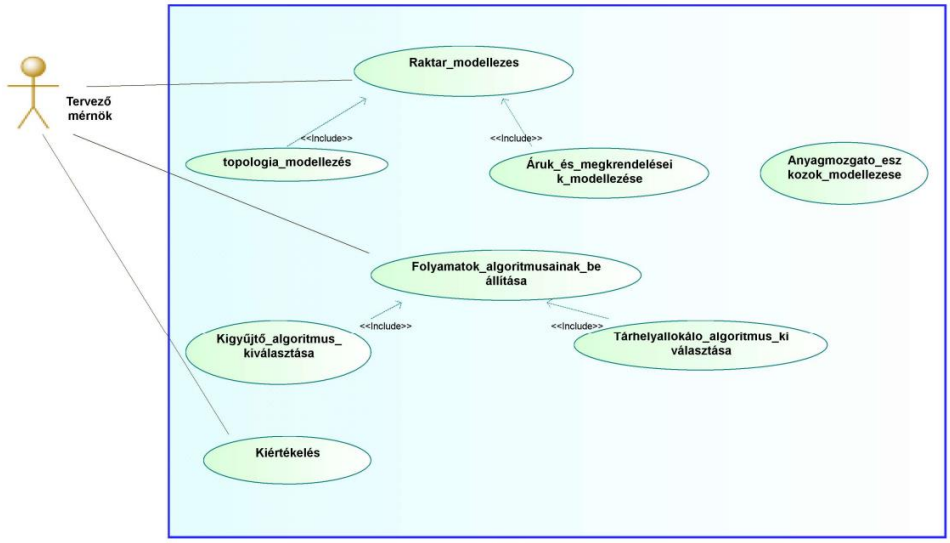
A modellezési környezetet objektum orientált szemléletmóddal C# nyelven fejlesztettem, ugyanis ez az a nyelv, amelyben a leginkább teret kap az objektumorientált szemléletmód, továbbá igen hatékony és jól használható fejlesztői környezet áll rendelkezésre. A fejlesztés során a logisztikai szakmai szempontok mellett, figyeltem a szoftvertervezés legfontosabb elveire, mint például:

- Open/Closed Principle, vagyis igyekeztem a rendszert könnyen bővíthetővé tenni, úgy, hogy ne kelljen módosítani az eredeti kódot.
- Do not repeat yourself, vagyis kerültem a kódduplikációkat, az ismétlődéseket kiszerveztem mindig például ős osztályokba.

3.1. A modellezési környezet felépítése

A modellezési környezet felépítésekor fontos szempont volt, hogy modulárisan lehessen hozzácsatolni a különböző problémák megoldó algoritmusait, „motorjait. Fő erőssége is ez a modularitás, hiszen ez teszi lehetővé, hogy sok valós helyzetben, több megoldási lehetőséget lehessen vele vizsgálni. A modellezési rendszer úgy lett kialakítva, hogy az ipari környezetben legjellemzőbb raktárfelépítéseket le lehessen vele másolni, a megoldó motoroknál pedig, tetszőleges számú opció implementálható legyen.

Az ***13. ábra*** mutatja a szofver use case diagrammját. Az első fontos lépés a modellalkotás. Itt az ipari alkalmazhatóság miatt a fő elvárás, hogy minél gyorsabban és egyszerűbben ugyanakkor minél pontosabban implementálható legyen a lehető legtöbbféle raktár, vagy raktáregység.



13. ábra: *A rendszer use case diagrammja*

A raktár modellezésnél három részre kell bontani a feladatot:

- a topológia modellezése,
- az áruk modellezése és
- az anyagmozgató rendszer modellezése.

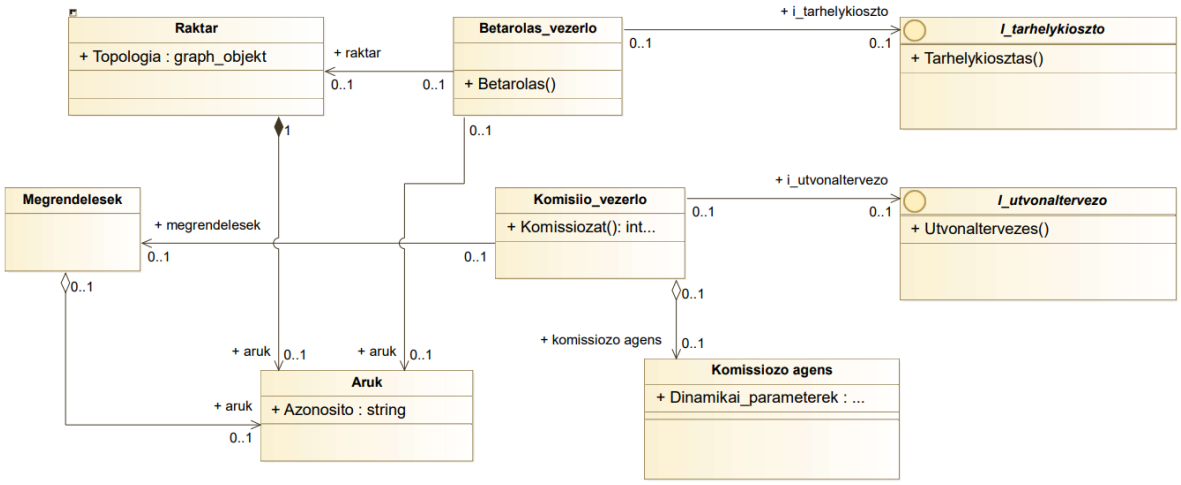
A topológia modellezés során meg kell határozni a tárolási technológiát, majd a tárolási helyek egymáshoz való elrendezkedését, a raktár valós elrendezésének megfelelően. Az áruk modellezése során számos fontos paraméter merülhet fel. Az áru adatai, fizikai paramétereire (súly, térfogat, típus), forgalmuk és a megrendelések összetétele. A modellezési környezet első változatában az anyagok fizikai paramétereivel nem foglalkozunk, de a modell felépítésekor szempont hogy bővíthető legyen velük majd a későbbiek folyamán. Az anyagmozgató rendszer modellezésénél paraméter lehet a gépek mozgási paraméterei úgy mint: gyorsulás, lassulás, maximum megengedett sebesség, vertikális és horizontális irányban egyaránt. Továbbá fontos lehet az anyagmozgató gépek száma és a teherbíró képességük. Ezt a két paramétert azonban a modellezési környezet első változatában elhagyom.

Miután felépült a modell, fel van töltve adatokkal és ki vannak választva a szükséges megoldó motorok egy kiértékelés során, a tárhelyallokáló motor elhelyezi valamilyen metodika szerint az árukat, utána pedig a kigyűjtő algoritmus kigyűjti az egyes létrehozott komissiózási listákat. Végül megkapjuk, hogy az egyes megoldó algoritmusok támogatásával milyen erőforrásigénnyel tud működni az adott rendszer. Természetesen a listagenerálásnál, illetve az input megrendelési adatok felhasználásánál érdemes keresztvalidációt alkalmazni, hogy ezáltal a véletlennek kisebb jelentősége legyen és pontosabb képet kapjunk az egyes támogató algoritmusok a modellhez vonatkozó hatékonyságáról.

3.2. A modellező rendszer felépítése

A modellező rendszer kialakításánál, fontos szempont volt, hogy számos raktártopológia modellezhető legyen vele. Ehhez először megvizsgáltam a legtipikusabb tárolásirendszereket, raktártopológiákat, továbbá a szakirodalomban erre vonatkozó tanulmányokat. Ezek után fogalmaztam meg egy irányt, mely szerint a raktármodellező rendszert fel kell építeni. A tervezési folyamat végén pedig, megvizsgáltam a lehetőségeket, hogy milyen környezetben milyen hatékonysággal lehet a feladatot implementálni.

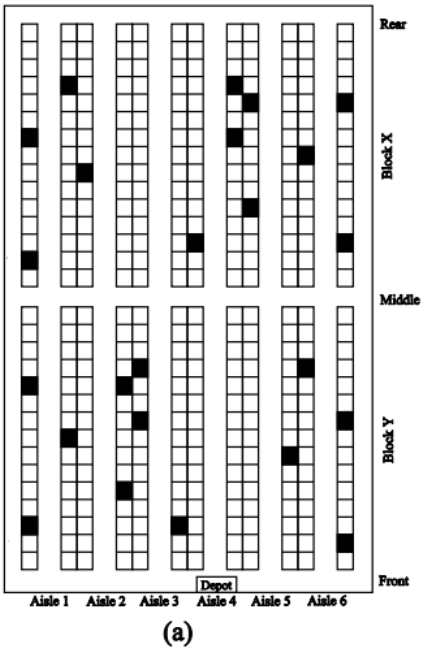
A következő ***14. ábra*** mutatja a tervezett rendszer egyszerűsített osztálydiagrammját. A diagrammon látszanak a legfontosabb tervezési szempontok. Egyrészt, hogy a tárolási, anyagmozgatási technológia és a tárolási egységek jól elkülöníthetők. Másrészt, hogy az interfaceken keresztük könnyen implementálható számos megoldó algoritmus.



14. ábra: *a rendszer egyszerűsített osztálydiagrammja*

3.2.1. Tárolási rendszerek és raktártopológiák

A raktározásban az egyik legelterjedtebb tárolási rendszer a soros állványos tárolás. A sorosállványos tárolási technológiát alkalmazó raktárak topológiájának vizsgálata során kirajzolódik egy egyértelmű mintázat, melyet a szakirodalom is alátámaszt. A sorosállványos rendszerek ún. blokkokat hoznak létre. Egy blokk az két keresztfolyosó között elhelyezkedő állvány rendszer.



15. ábra*: Raktártopológia*[10]

Ezek a blokkok persze tetszőleges elrendezésben lehetnek, több csarnokban elhelyezve. A blokkok közötti távolság változó lehet. Egyszerű keresztfolyosók esetében ez a távolság kisebb, ugyanakkor gyakran előfordulnak nagyobb árumanipulációs terek (pl.: csomagolóállomás), melyeknél a távolság jelentős is lehet. [10][4]

Látható, hogy a topológiában egyértelmű mintázatok vannak, ugyanakkor ezekből végtelen számú opciót ki lehet alakítani a blokkok méretének és egymáshoz való elhelyezkedésének szempontjából, így a tervezési folyamat során mindenképp egy rugalmas megoldásra lesz szükség.

3.2.2. Raktármodell

Kutatásom során először egy koordinátarendszer alapú modell építésén gondolkodtam. Ennek nagy előnye az egyszerűsége volt, azonban a több blokkos tárolási rendszer kidolgozásakor számos problémába ütköztem. Túl bonyolulttá vált a raktár topológiájának a modellezése koordinátarendszer alapú környezetben.

A szakirodalom a probléma kapcsán számos esetben gráfokat használ. Ennek számos előnye van. Egyrészt a gráfokkal könnyedén és rugalmasan létre lehet hozni bármilyen elrendezést, másrészt a különböző bejárási algoritmusok egyszerűen implementálhatóak gráfokon.

A modellem alapját a blokkok képezik. A csarnokok és keresztfolyosók alapján blokkokra osztom fel a raktárt. A blokkok paraméterei egymástól különbözhetnek. Más oszlop, sor, magasságszám. A modellépítés folyamatának elején, meg kell határozni a blokkokat, utána pedig, fel kell építeni az egyes blokkok gráfhálóit a megfelelő fizikai paraméterekkel együtt. A gráf csomópontjai azok a pontok, ahol a komissiózó ágens megáll kigyűjteni az egyes anyagokat. A gráf élei pedig a sorok közti folyosó azon pontjai, amik összekötik a kigyűjtési pontokat. Az élek súlyai pedig, az egyes pontok közötti távolságok. A blokkok felépítése után, meg kell határozni a blokkok egymáshoz való elhelyezkedését. Ehhez un. áthidaló blokkokat használok, melyekkel a keresztfolyosókat, illetve a csarnokok közötti területeket modellezem le. Itt az egyes csomópontok olyan kereszteződések, ahol releváns lehet a komissiózó ágens irányváltása. Az élek ebben az esetben is az egyes csomópontok közötti utak, a súlyuk pedig a távolságokat jelentik.

3.2.3. Árucikkek modellezése

Az árucikkek modellezésénél, számos egyszerűsítéssel éltem. Az áruk fizikai paramétereit (tömeg, kiterjedés, stb), valamint az áruk értékeit figyelmen kívül hagytam, az első modell felépítésekor, ugyanakkor bővíthető vele a modell. Az első modellben úgy vettem, hogy egy tárhelyen, továbbá egy komissiózó ágensen tetszőleges számú áru elhelyezhető.

Az árukkal kapcsolatos egyetlen, és a folyamatok szempontjából legfontosabb paraméter, az árukhoz kapcsolódó megrendelési adatok, hiszen ezek generálják a komissiózó listákat és a komissiózás folyamatát, melyek optimalizálására törekszünk. A cikkszámok és a hozzájuk kapcsolódó megrendelési adatok, egy vállalat valós adatai. A cikkszámok, tárolt volumenek és megrendelések alapján különböző csoportokba sorolhatjuk a raktárak árukészletét. A munkám során polistruktúrájú raktárat vettem alapul olyan megrendelésállománnyal melyben nem tapasztalható szezonalitás.

3.2.4. Komissiózó ágensek modellezése

A komissiózó ágensek modellezésénél szakaszos működésű szállítóeszközök modellezésére alakítottam ki a modellt. Ez alatt főként targoncákat és automata targoncákat un. AGV-ket kell érteni. Ezen eszközöknél alapvetően a fizikai paramétereik és a mozgási paramétereik meghatározóak, azonban a modell első változatában a fizikai paraméterekkel nem foglalkoztam. Ez azt jelenti, hogy a komissiózó ágensek terhelhetőségét „végtelennek” vettem, tehát minden a listán szereplő tételt ki tud gyűjteni. A továbbiakban azonban a modell ezen területen fejleszthető.

Az ágensek mozgási paraméterei: gyorsulás, lassulás, maximum megengedett sebesség stb. azonban szerepelnek a modellben. Ezek a paraméterek igen meghatározóak, hiszen döntően befolyásolják a komissiózó időszükségletet, ami az optimalizáció fő célpontja.

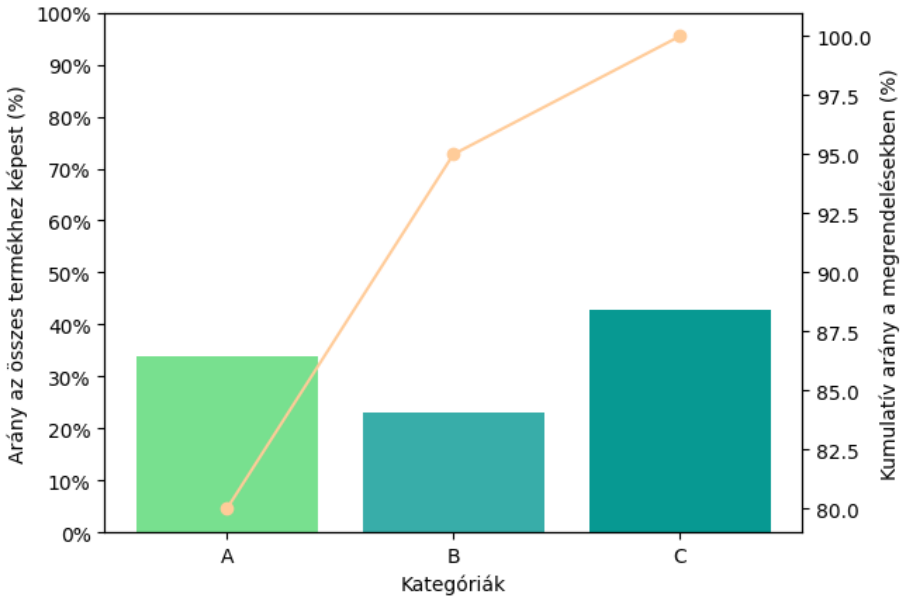
4. Az adatok és a raktármodell bemutatása

Ebben a fejezetben ismertetem a vizsgálat során alkalmazott adatokat és a raktár topológia jellemzőit. Emellett bemutatom a tesztelési eredmények kiértékeléséhez alkalmazott számítási eljárásokat, valamint részletezem a vizsgálat során használt genetikus algoritmus konfigurációs paramétereit.

4.1.A megrendelésadatok és kigyűjtési listák bemutatása

A teszteléshez használt adatok egy valós magyarországi kiskereskedelmi lánc több hónapnyi megrendelési adataiból származnak. Az adatok több mint 1,300 különböző terméket és több mint 600 000 megrendelést tartalmaznak. Ez az adatmennyiség nemcsak a tesztelési folyamat pontosságát növeli, hanem lehetővé teszi az algoritmusok teljesítményének átfogó értékelését is.

Az adatok olyan piaci szegmensből származnak, ahol a termékkínálat viszonylag stabil. Ez kulcsfontosságú volt a kutatás szempontjából, mivel lehetővé tette, hogy a tesztelés alatt kiinduljak abból az alapfeltevésből, hogy minden termék folyamatosan elérhető a vizsgált raktárkészletben.



16. ábra: *A megrendelési adatok Pareto elemzése*

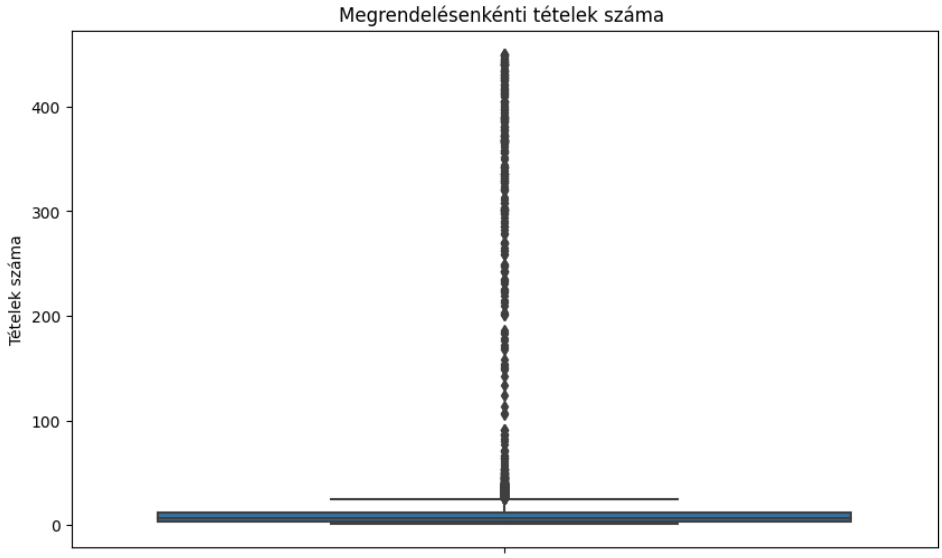
A ***16. ábra*** alapján látható, hogy a megrendelési adatokra jelen esetben részlegesen igaz csak a Pareto elv. Az A-kategóriájú termékek aránya meglehetősen magas ~35% míg a B-s termékeké mindössze 25% körüli.

A kigyűjtési listák adatait az alábbi **3. táblázat** tartalmazza:

3. táblázat: A komissiózó listák adatainak statisztikái

Statisztika	Érték:
Átlag:	12,9
Szórás	40
min:	1
25. percentilis	3
medián	7
75. percentilis	12
maximum	446

A táblázat alapján látható, hogy a listák döntő többségében csupán néhány termék van, ugyanakkor az adatok között számos kilógó adat van. A ***17. ábra*** alapján látható, hogy arányag kis részre sűrűsödik az adatok döntő többsége.



17. ábra *Megrendelésenkénti tételek száma*

Fontos megemlíteni, hogy a komissiózó listák alapjául ezen listák szolgáltak. Ez azért fontos, mert így a listák nagyrésze viszonylag kevés megrendelést fog össze, ami befolyásolhatja az optimalizációs paramétereket. Lehetséges, hogy a listák összefogása során bővülnének az optimalizációs algoritmusok lehetőségei.

4.2.A raktártopológia bemutatása

Az algoritmusok hatékonyságának értékeléséhez nemcsak a termék- és megrendelési adatokra volt szükség, hanem megfelelő raktárak vizsgálatára is. A kutatás során soros állványos tárolási technológiát alkalmazó raktárakat elemeztem. Célom az volt, hogy valós raktárak fizikai adottságain végezzem el a tesztelést, de nem találtam olyan vállalatot, amely megfelelő paraméterekkel rendelkezett volna és hajlandó lett volna adatait megosztani velem. Ennek eredményeképpen, saját magam által létrehozott raktártopológiákon végeztem a teszteket. Ki kell emelni, hogy a raktártervezés önmagában egy komplex szakterület, amely jóval többet igényel, mint csupán néhány terület és méretvonal megrajzolása folyosókkal és

keresztfolyosókkal. Bár nem rendelkezem mélyreható szakmai ismeretekkel ezen a területen, a raktártopológiák kialakításakor igyekeztem a szakirodalomban fellelhető tudásra és példákra alapozni.[26]

Két raktártopológiát terveztem, melyeken az algoritmusokat lefuttattam. Az első topológiát A-nak a második topológiát B-nek neveztem el. Mindkét topológiánál közös, hogy alapvetően raklapos áru tárolására lettek tervezve és két szint magasságban van tárolva az áru. Ez azért fontos mert a modellben a komissiózó ágens csak kétdimenziós mozgást vesz figyelembe, függőleges mozgásokkal nem foglalkozik. A valóságban a komissiózás sok esetben úgynevezett komissiózó zónákból történik, ami jellemzően a soros állványos tárolási rendszereknél az állványrendszer alsó, vagy alsó néhány szintjéről történik.[6]

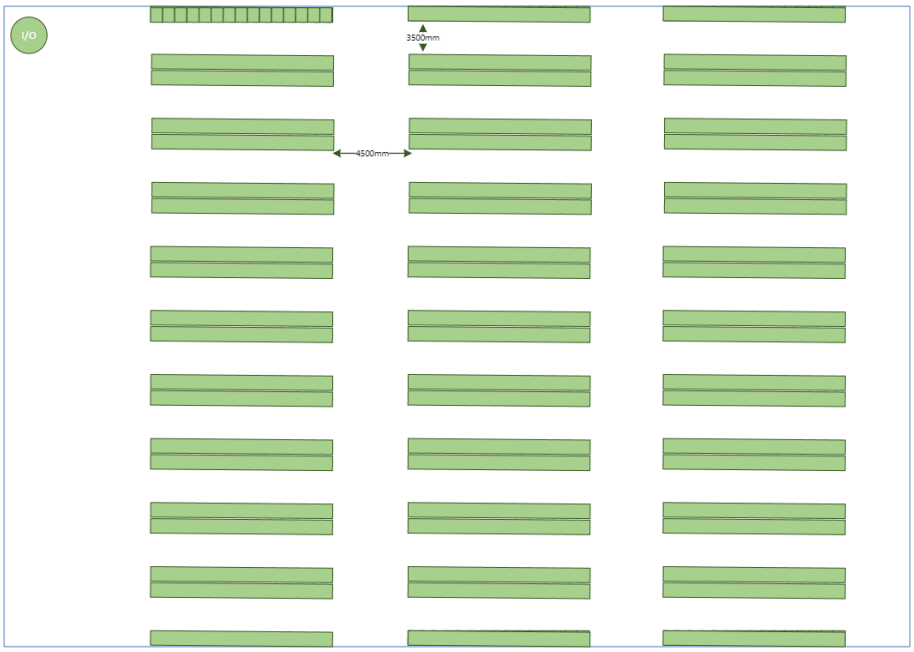
4.2.1. „A” raktártopológia

Az A raktártopológia kialakításánál azt határoztam meg, hogy az 1354 terméket 75%os raktárkihasználtság mellett 1800 tárhelyen tárolom. Három blokkban hossz irányú állványelhelyezéssel egy fej elrendezéssel. Blokkonként húsz állvánnyal állványonként öt mezővel.

Standard méretű állványra és raklapra terveztem a modellt. Keresztirányú raklapelhelyezéssel így egy mezőben, két szinten tárolva összesen hat raklap fér el, ami hat tárhelyt jelent. A mezőszélesség 2700mm az állvány mélysége pedig 1100mm. Továbbá az állványpárok között mindenhol szükség van egy 100mm-es hézagra.

A raktár fej elrendezésű, hosszirányú állványelrendezéssel. A folyosók szélessége 3500mm a keresztfolyosók szélessége pedig 4500mm-re lett tervezve.

A ***18. ábra*** bal felső sarkában látható az az I/O pont, amely a modell esetében minden komissiózó útvonal kezdő és végpontja.

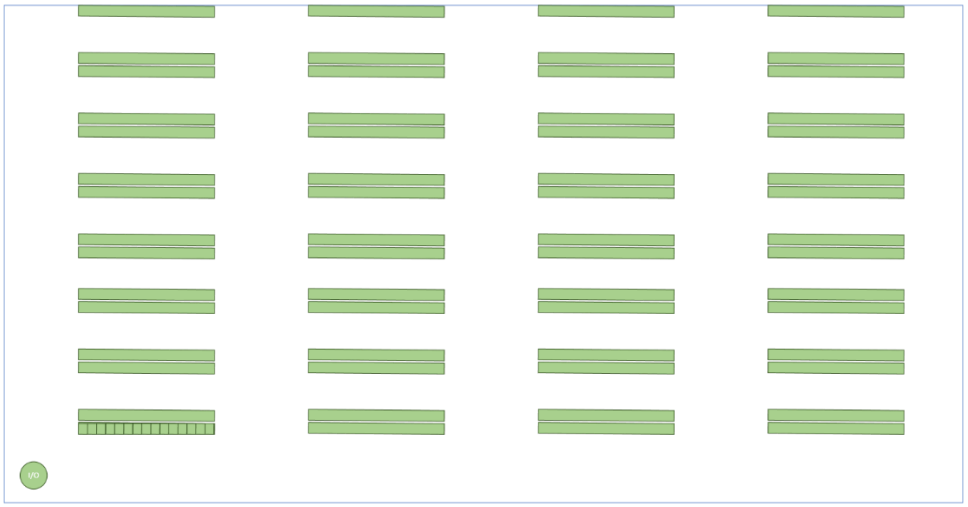


18. ábra: *Az „A” raktártopológia*

4.2.2. „B” raktártopológia

A „B” raktártopológia esetében ugyancsak 1800 tárhelyt terveztem, ami 75%os raktárkihasználtságot jelent. A „B” esetnél keresztirányú állványelrendezést választottam, de megmaradt a raktár fej elrendezése.

A „B” raktártopológiát négy blokk alkotja, mindegyik blokkban tizenöt állvánnyal. Az állványok ez esetben is standard 2700mm mezőszélesség 1100mm mélységűek 100mm-es közökkel. A raklapelhelyezés az állványon ezen raktáresetnél is keresztirányú. A folyosó szélesség 3500mm a keresztfolyosók szélessége 4500mm.



19. ábra: *A "B" raktártopológia*

4.2.3. A modell kikötései

A modellalkotás során bizonyos egyszerűsítéseket vezettem be, amelyek eltérhetnek a valós rendszerek sajátosságaitól. Azonban a modellezés bizonyos aspektusai megkövetelik ezeket a leegyszerűsítéseket. A modell a következő feltételezéseken alapul:

- Minden áru egy tárhelyen van tárolva és egy tárhelyen csak egy áru van tárolva.
- Minden tárhelyen adott árucikkből mindig rendelkezésre áll a szükséges készlet.
- A komissiózó eszköz minden esetben be tudja fogadni a kigyűjtési lista teljes tartalmát.
- A komissiózó eszköz kanyarban lelassít 0-ra megfordul és 0-ról kezd el gyorsítani.
- A raktárban minden komissiózó eszköz ugyanolyan gyorsulási és sebességparaméterekkel rendelkezik.
- Függőleges irányban ugyan van távolságkülönbség az egyes tárhelyek között, de a kigyűjtési időnél nincs időkülönbség.
- A tárhelyen kigyűjtés közben eltöltött időt nullának veszem és így csak a komissiózási mozgási időket vizsgálom.

4.2.4. A komissiózó ágens műszaki paraméterei

A komissiózó ágens jelentős hatást gyakorol a folyamatok hatékonyságára, mivel ez határozza meg a számítások során kulcsfontosságú gyorsulási és sebesség paramétereket. A modellben a Jungheinrich DFG/TFG 316 típusú homlokvillás targonca paramétereit alkalmaztam, mivel ezek a targoncák gyakoriak az ilyen típusú soros állványos raktárakban. Természetesen lehetne más típusokra is tesztelni, például kimondottan komissiózásra tervezett komissiózó targoncákra.

A fent említett típus legfontosabb paramétereit a következő **4. táblázat** tartalmazza.

4. táblázat: Targonca műszaki paraméterek (forrás:[27])

Paraméter:	Érték:
Maximális sebesség (teherrel):	5,27 m/s
Gyorsulás (teherrel):	1,17 m/s²
Fordulási sugár:	1975mm

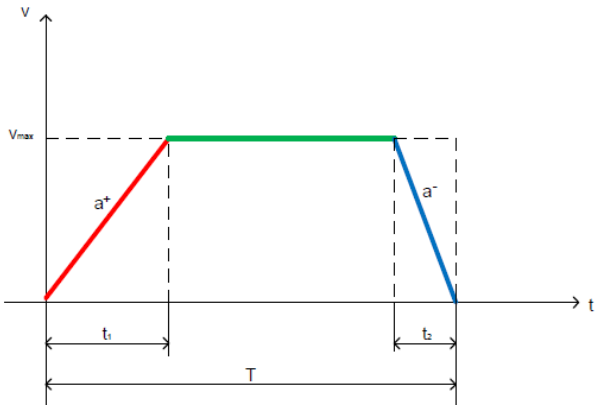
4.3.A számítási módszerek bemutatása

Az időalapú optimalizálás során, nem az egyes listák kigyűjtése folyamán bejárt távolságot, hanem az ahhoz szükséges időt veszem alapul. Ez az idő lesz az egyes lefutások utáni eredmény, továbbá a GA fitnesz függvénye is az időértékek minimalizálására fog törekedni.

A kigyűjtési idő optimalizálásának paraméterei: az x és y távolságadatok az egyes pontok (nullpont-tárhely, tárhely-tárhely) között, továbbá a komissiózó gép gyorsulása és a megengedett maximum sebesség. Fontos paraméter az is, hogy hány mozgásszakasz van, hányszor kell a gépnek kanyarodnia, ugyanis olyankor le kell lassítania nullára. A modellben a gyorsulási és a lassulási tényezőt ugyanannyinak vettem.

A számítás során megkülönböztettem gyorsuló, lassuló és egyenletes sebességgel haladó szakaszokat. A számítást főként az teszi bonyolulttá, hogy bizonytalan tényező az, hogy a mozgás során a gép eléri-e a maximum sebességét. A ***20. ábra*** mutatja ezt az esetet. Látható, hogy egy gyorsuló és egy lassuló szakasz, illetve egy a kettő közötti egyenletes mozgás alkotja ezt az esetet. Sebesség-idő diagramok esetén a görbe alatti terület adja meg a megtett út hosszát. Ebben az esetben visszafelé számolunk az (1)-es képlet alapján.

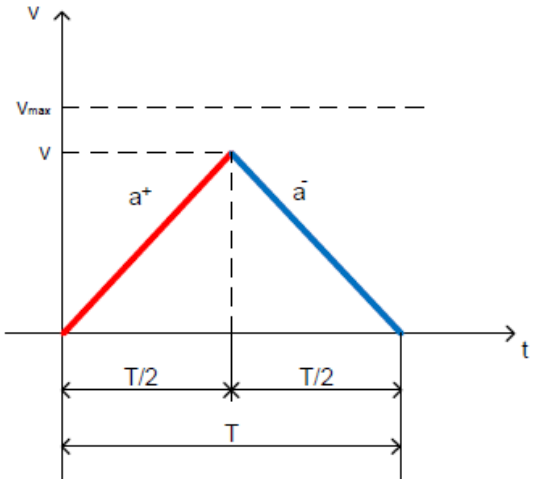
$$(1) \quad T = \frac{s}{v_{max}} + \frac{v_{max}}{a}$$



20. ábra: *haladás maximális sebesség elérésével. Forrás: [3]*

A ***21. ábra20. ábra*** mutatja azt az esetet, amikor a komissiózó gép nem éri el, vagy csak éppen egy pillanatra éri el a maximális sebességet. Ilyen esetben a mozgás csak egy gyorsul és egy lassúló szakaszból áll. Ilyenkor a (2)-es képlet segítségével számoljuk a szükséges időt.

$$(2) \quad T = \sqrt{\frac{s}{a}}$$



21. ábra: *Gyorsuló és lassuló szakasz. Forrás: [3]*

A program miután kiszámolja az egyes szakaszok időszükségletét, összegzi annak adatait.

4.4.A genetikus algoritmus paraméterei

A genetikus algoritmus teljesítménye és hatékonysága nagymértékben függ a különböző paraméterek beállításaitól. Ezen fontos paraméterek:

- Az egyedszám, amely esetünkben az egyszerre létrejött raktári tárhelykiosztás opciókat tartalmazza,
- az elitizmus, amely meghatározza, hogy az egyes generációk során a populáció mekkora része cserélődik,
- a mutációs ráta, amely arány megmutatja, hogy hány egyednél történik egy véletlenszerű random változás a generáció végén, továbbá
- a generációk száma, mely azon ciklusnak a száma, hogy hányszor ismétlődjön meg a fejlesztési folyamat.

Ebben a fejezetben ezen kulcsfontosságú paramétereket vizsgálom meg részletesebben.

A generációk számának meghatározása alapvetően határozza meg az algoritmus futási idejét és a konvergencia gyorsaságát. Korábbi kutatásaim során egyenes arányosságot tapasztaltam a generációk számának növelésével a genetikus algoritmus teljesítményében. Elemzésem során azt vizsgáltam, hogy a generációk számának változása milyen mértékben befolyásolja az elérhető eredményeket. Ezt szem előtt tartva a generációk számát 300-ra állapítottam meg, hogy optimális egyensúlyt találjak a számítási idő és a konvergencia sebessége között.

Az egyedszám a futási időt lineárisan érinti, és az algoritmus diverzitását befolyásolja. Az előző megfontolásokat követve az egyedszámot 3000-re állapítottam be. Ez a mennyiség elegendő változatosságot biztosít az algoritmus számára a potenciális megoldások tekintetében, anélkül, hogy aránytalanul megnövelné a számítási időt.

A mutációs rátát két százalékban határoztam meg, korábbi kutatásaim tapasztalataira alapozva. A mutációs rátával körültekintően kell eljárni: túlzottan magas érték esetén az evolúciós folyamat kiszámíthatatlanná és instabillá válhat, míg alacsony rátánál az algoritmus hajlamos lehet lokális optimumokba szorulni, ami gátolja a globális optimum közelítését.

Az algoritmus elitizmusának mértékét ötven százalékban állapítottam meg, ami egy gyakran alkalmazott érték. A szakirodalom számos pontján találkozhatunk ezzel a paraméterrel, és korábbi kutatásaim során is ezt a beállítást preferáltam.

Az algoritmus konvergenciájának jellemzői kulcsfontosságúak. Egy nagyobb egyedszámú populáció lassabban konvergál, viszont nagyobb esélye van arra, hogy fejlettebb egyedeket hozzon létre. Ugyanakkor megfigyeltem, hogy egy bizonyos generációszám elérése után a populációban egyfajta "genetikai homogenitás" alakul ki, ami azzal jár, hogy az egyedek túlzottan hasonlóvá válnak, és ez komplikációkat okoz. Ennek a tendenciának a mérséklésére törekedtem azzal, hogy a túl hasonló egyedek közül csak egyet engedtem tovább, de ez az intézkedés sem volt képes teljes mértékben kiküszöbölni a problémát ezért minden egyedszám értékhez megvolt az a generációszám, amit még hibamentesen elvisel a modell. Ez a 300-generáció 3000 egyeddel még ezen határétéken belül található.

A generációk számának és az egyedszámának meghatározását korábbi kutatásaim és tapasztalataim alapján végeztem el. A 300 generáció és a 3000 egyedszám egy kiegyensúlyozott választásnak bizonyult a futási idő, a konvergencia gyorsasága és az algoritmus hatékonysága szempontjából. Ezek a beállítások biztosítják az algoritmus zökkenőmentes működését anélkül, hogy indokolatlanul nagy erőforrásokat igényelne.

5. Tesztek és eredmények

5.1. Az algoritmusok tesztelése

Az algoritmusok tesztelését az előző fejezetekben bemutatott „A” és „B” raktárakon hajtottam végre. A vizsgálat során hat különböző algoritmus-kombinációt értékeltem, egy vállalat több hónapnyi megrendelési adatát felhasználva. A célom az volt, hogy megvizsgáljam az egyes algoritmus párok hatékonyságát egymáshoz képest, a különbségeket a két raktáresetben, továbbá az egyes szinergiákat, vagy épp tompító tényezőket.

5.1.1. A vizsgált algoritmuspárok

A tesztelés során különböző algoritmusok hatékonyságát vizsgáltam meg, hogy megértsük, melyik kombináció nyújtja a legjobb teljesítményt a raktári műveletek során. Az algoritmusok két fő kategóriába sorolhatók: betároló algoritmusok és komissiózó algoritmusok. Mindkét kategória algoritmusai kritikus szerepet játszanak a raktári rendszer hatékonyságának növelésében, de különböző aspektusokra fókuszálnak.

Vizsgált betároló algoritmusok:

- Véletlenszerű: Az algoritmus véletlenszerűen választja ki a termékek tárolási helyét.
- ABC algoritmus: Az algoritmus az ABC kategóriák alapján választja ki a termékek tárolási helyét.
- Genetikus algoritmus: az általam a problémára írt evolúciós algoritmus, amely optimalizálja a termékek tárolási helyét.

Vizsgált komissiózó algoritmusok

- S-shape algoritmus: Az algoritmus az S alakú útvonal mentén határozza meg a komissiózási útvonalat.
- Largest Gap heurisztika: Az algoritmus a legnagyobb szabad közöket kihagyva határozza meg a komissiózási útvonalat.

5.1.2. Keresztvalidáció

A keresztvalidáció kritikus fontosságú a tesztelési folyamatban, mivel lehetővé teszi az algoritmusok teljesítményének objektívabb értékelését. Ez az eljárás elősegíti, hogy a modell ne csupán a tanítóadatokra legyen finomhangolva, hanem új, eddig ismeretlen adatokon is kiválóan működjön. Ez különösen lényeges olyan összetett rendszerek esetében, mint amilyen egy raktározási rendszer, ahol az algoritmusoknak nagyszámú és igen különféle változók és körülmények között is hatásosan kell teljesíteniük. A keresztvalidáció ezáltal mérsékli a túltanulás kockázatát, így általánosabb, szélesebb körben alkalmazható megoldásokat eredményezhet.

A tesztelés során "one-leave-out" keresztvalidációs technikát alkalmaztam. A meglévő, több mint 600 000 megrendelési listát véletlenszerűen tíz különböző részre osztottam. Minden tesztelési iterációban kilenc részt a modell tanítására, míg a tizedik részt a tesztelésére

használtam. Ezt az eljárást minden lehetséges tíz részre osztott kombinációra alkalmaztam, és az algoritmusokat minden variációban többször is lefuttattam.

5.1.3. Futási idők

A futási idők elemzése fontos lehet algoritmusok értékelésében, mivel a sebesség jelentősen befolyásolja a rendszer használhatóságát és költségeit; a túlzottan lassú algoritmusok alkalmazhatatlanná válhatnak. Ebben a szakaszban a különféle algoritmusok és algoritmus-kombinációk átlagos futási idejét tárgyalom. A táblázatban szereplő eredmények a két raktártopológiai környezetben végzett futtatások átlagát tükrözik. (A két eltérő raktártopológia között a futási idők tekintetében nem volt észlelhető jelentős eltérés.)

5. táblázat: az algoritmuspárok átlagos futási idejei

	Átlagos futásidő
S-shape+ véletlenszerű	1,8 s
S-shape+ ABC	1,9s
S-shape+ Genetikus	186,6 s
Largest Gap+ véletlenszerű	2,3 s
Largest Gap+ ABC	2,5 s
Largest Gap+ Genetikus	187,1 s

Ahogy az **5. táblázat** mutatja, az S-shape és Largest Gap algoritmusok futási ideje hasonló. Ugyancsak a véletlenszerű és az ABC algoritmus is. Kivételt egyedül a genetikus algoritmusok képeznek, ahol a futási idő drasztikusan megnő, két nagyságrenddel nagyobbak a többinél.

Amennyiben a genetikus algoritmusok kiemelkedően magas teljesítményt nyújtanak, megfontolandó azok alkalmazása, különösen ha az erőforrások rendelkezésre állnak és a futási idő nem jelent kritikus szempontot. Ha viszont a teljesítménybeli különbség elhanyagolható, érdemes lehet más, például az ABC algoritmusok felé fordulni, amelyek gyorsabbak és kevesebb erőforrást igényelnek.

5.2. Az eredmények bemutatása

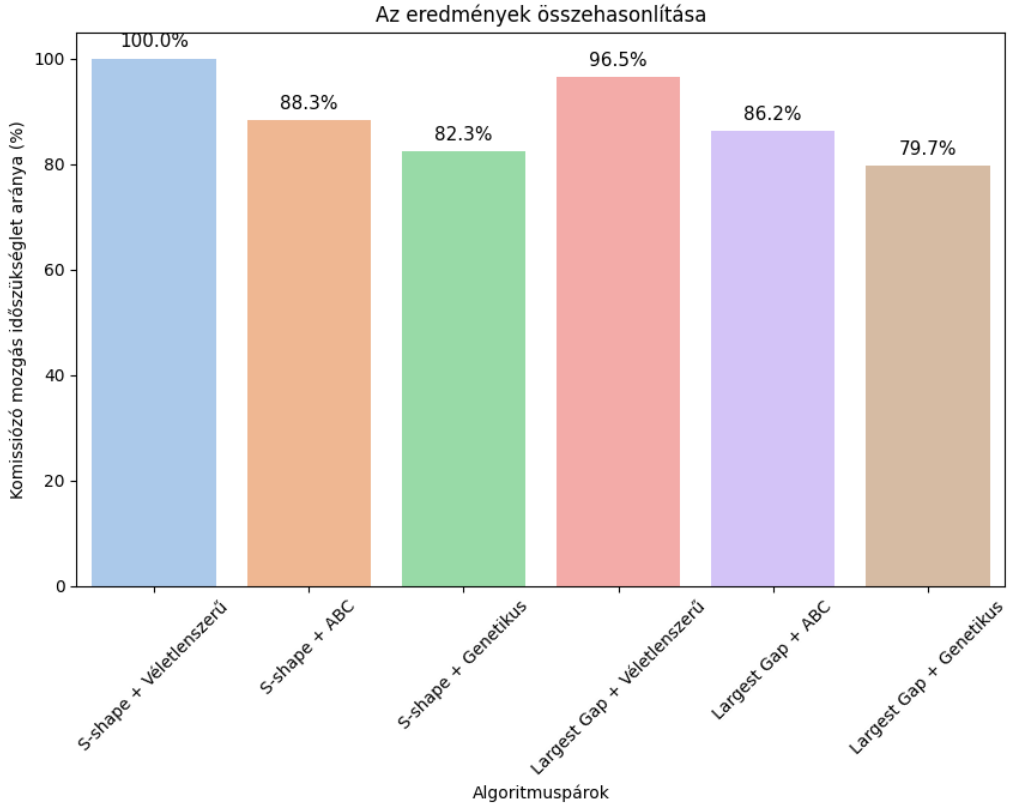
5.2.1. Az „A” raktáreset eredményeinek bemutatása

Az algoritmuspárok értékeléséhez a listák kigyűjtéséhez használt komissiózás közben megtett út időszükségleteit használtam, nem vettem figyelembe az egyes tárhelyeken eltöltött időt. Az egyes algoritmuspárokra az előző fejezetben említett keresztvalidációs technikával többször is vizsgáltam végül a kapott eredmények átlagát vettem. Az eredményeket százalékos formában mutatom be, ahol az S-shape algoritmus és a véletlenszerű tárhely-kiosztás kombinációja szolgált az 100%-os referenciapontként. Vizsgáltam továbbá az eredmények szórását is.

6. táblázat: Az algoritmusok eredményei az „A” raktártopológia esetén

	Arány	Szórás
S-shape+ véletlenszerű	100%	1,7%
S-shape+ ABC	88,3%	2,3%
S-shape+ Genetikus	82,3%	2,5%
Largest Gap+ véletlenszerű	96,5%	1,9%
Largest Gap+ ABC	86,2%	2,6%
Largest Gap+ Genetikus	79,7%	3,0%

Az **6. táblázat:** alapján láthatjuk, hogy a kapott eredmények szórása kicsi, úgyhogy az eredmények konzisztensek és elegendő a keresztvalidációnál a tíz részre való osztás. Érdemes megfigyelni azt a mintázatot, hogy az algoritmusok bonyolódásával egyre nő a kapott szórások értéke. Ennek oka az lehet, hogy létezhetnek kiugróbb esetek mikor az adott bonyolultabb algoritmus jelentősen jobb, vagy gyengébb eredményt ér el, mint az egyszerű, nagyon általános megoldások.



22. ábra: *Az algoritmusok eredményei az „A” raktártopológia esetén*

A ***22. ábra*** mutatja a kapott eredményeket. Láthatjuk, hogy a bonyolultabb algoritmuspárok jobb eredményt adnak.

A **7. táblázat** a korábban tárgyalt eredmények 95%-os megbízhatósági szintű konfidenciaintervallumait mutatja be. Megfigyelhető, hogy az egyes esetek közötti átfedés elenyésző. A teszteredmények azt sugallják, hogy az átlagok által meghatározott hatékonysági sorrendtől való eltérés ritkának mondható.

7. táblázat: Az "A" raktártopológia eredményeinek konfidenciaintervalluma

	Konf. intervallum alsó határa	Konf. felső határa
S-shape+ véletlenszerű	99%	101,05%
S-shape+ ABC	86,87%	89,73%
S-shape+ Genetikus	80,75%	83,85%
Largest Gap+ véletlenszerű	95,32%	97,68%
Largest Gap+ ABC	84,59%	87,81%
Largest Gap+ Genetikus	77,84%	81,56%

5.2.2. A „B” raktáreset eredményeinek bemutatása

Az algoritmuspárokat ezen esetben is a komissiózási időszükséglet alapján értékeltem ki. Az eredményeket százalékos formában mutatom be, ahol az S-shape algoritmus és a véletlenszerű tárhely-kiosztás kombinációja szolgált az 100%-os referenciapontként. Vizsgáltam továbbá az eredmények szórását is. Az eredményeket a **8. táblázat** mutatja be.

8. táblázat: Az algoritmusok eredményei a „B” raktártopológia esetén

	Arány	Szórás
S-shape+ véletlenszerű	100%	2,8%
S-shape+ ABC	86,5%	3,1%
S-shape+ Genetikus	80,3%	3,5%
Largest Gap+ véletlenszerű	95,5%	1,8%
Largest Gap+ ABC	83,1%	2,9%
Largest Gap+ Genetikus	75,2%	3,8%

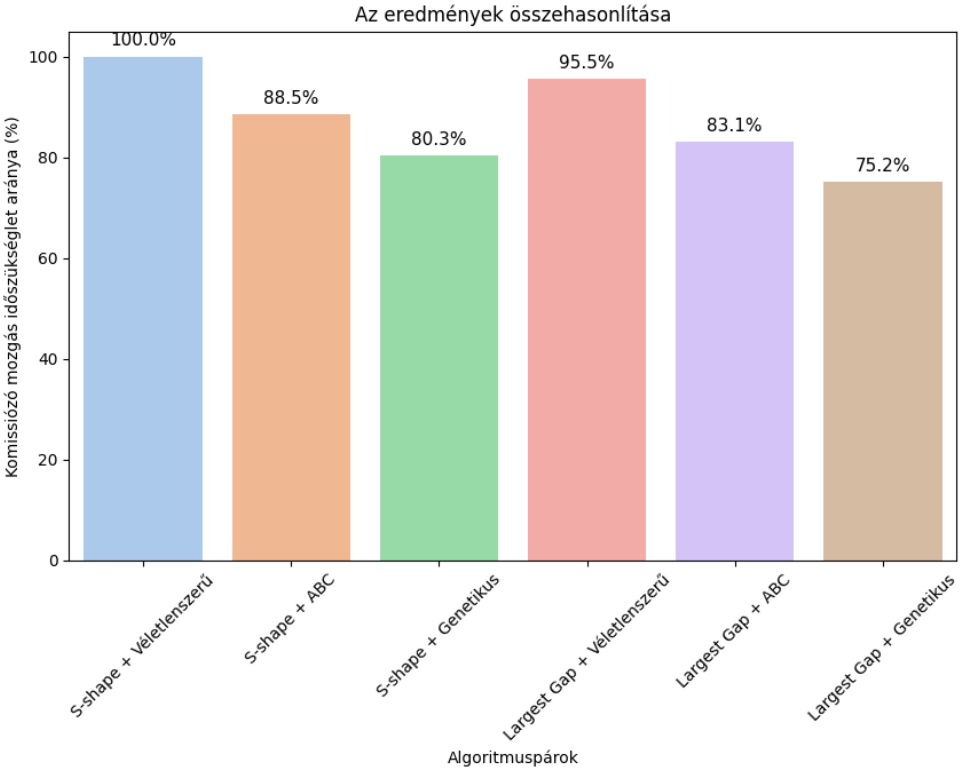
Az **8. táblázat** által bemutatott eredmények alapján megállapítható, hogy a Largest Gap és a Genetikus algoritmus kombinációja bizonyult ez esetben is a leghatékonyabbnak, továbbá észrevehető, hogy a különböző esetek szórása szinte mindenhol nagyobb, mint az „A” raktártopológia vizsgálatakor.

A **9. táblázat** a „B” raktártopológiához tartozó eredmények konfidenciaintervallumait mutatja be. Hasonlóan az előző esethez, itt is csak elenyésző mértékű átfedést tapasztalhatunk, ami megerősíti azon sorrendet, amelyet az algoritmusok teljesítményének átlagértékei alapján határoztam meg.

9. táblázat: A „B” raktártopológia eredményeinek konfidenciaintervalluma

Konf. intervallum alsó határa	Konf. intervallum alsó határa	Konf. Felső határa
S-shape+ véletlenszerű	98%	101,74%
S-shape+ ABC	84,58%	88,42%
S-shape+ Genetikus	78,13%	82,47%
Largest Gap+ véletlenszerű	94,38%	96,62%
Largest Gap+ ABC	81,30%	84,90%
Largest Gap+ Genetikus	72,84%	77,56%

Ezen eredmények figyelhetők meg a ***23*. ábra** alapján is.

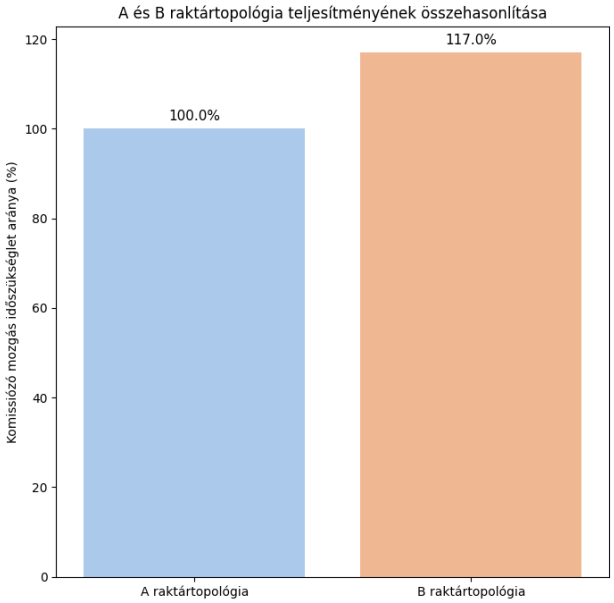


23. ábra: *Az algoritmusok eredményei a „B” raktártopológia esetén*

5.2.3. A két raktár eredményei egymáshoz viszonyítva

Érdemes összehasonlítani a két raktár esetében kapott eredményeket több szempont alapján. Egyrészt, hogy összességében, melyik raktár hogyan szerepelt, másrészt pedig, hogy melyik raktár esetében melyik algoritmuspár, milyen eredményt ért el.

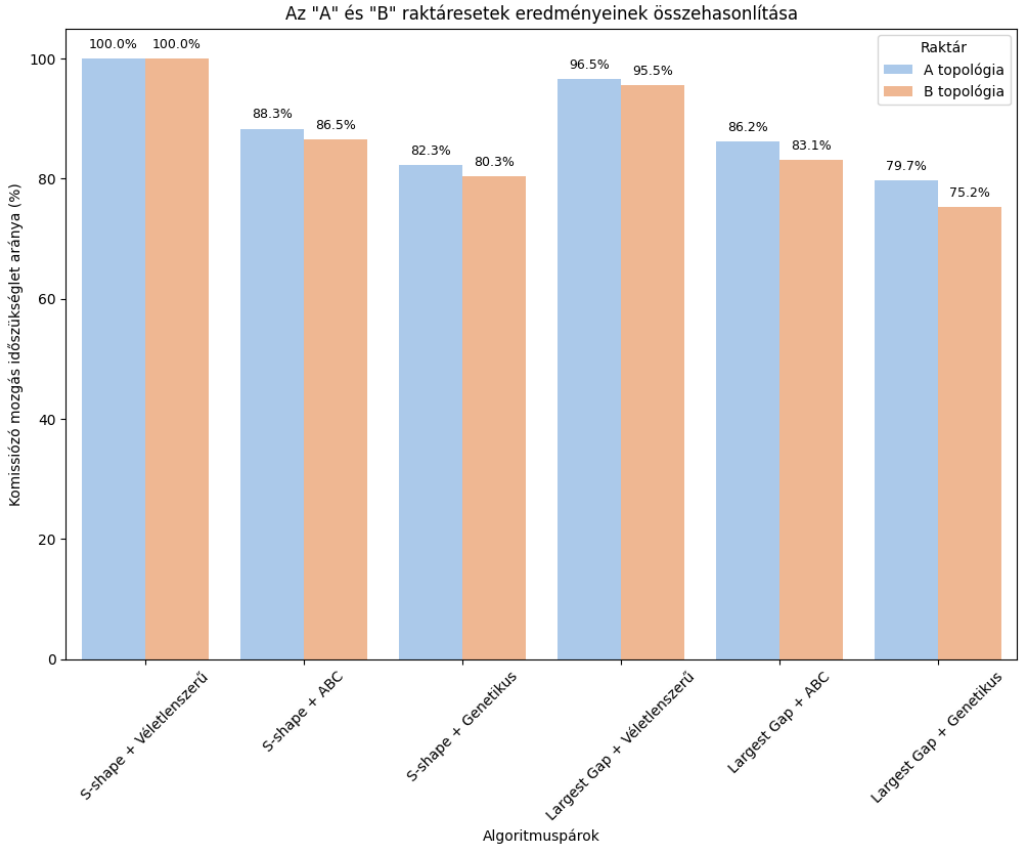
Az ***24. ábra*** mutatja be, hogy a két raktáreseten összességében milyen teljesítménnyel futottak le az algoritmusok. Ez a két érték az összes futás eredményét tartalmazza minden algoritmuspár minden keresztvalidációs futására. Az értékeket arányosan mutatom be, ahol a 100% az „A” raktáreset teljesítménye volt.



24. ábra: *Az "A" és a "B" raktártopológi egymáshoz viszonított eredményei*

A ***24. ábra*** alapján látható, hogy az „A” raktáresetben átlagosan jelentősen kisebbek voltak a komissiózási időszükségletek, mint a „B” raktáresetben.

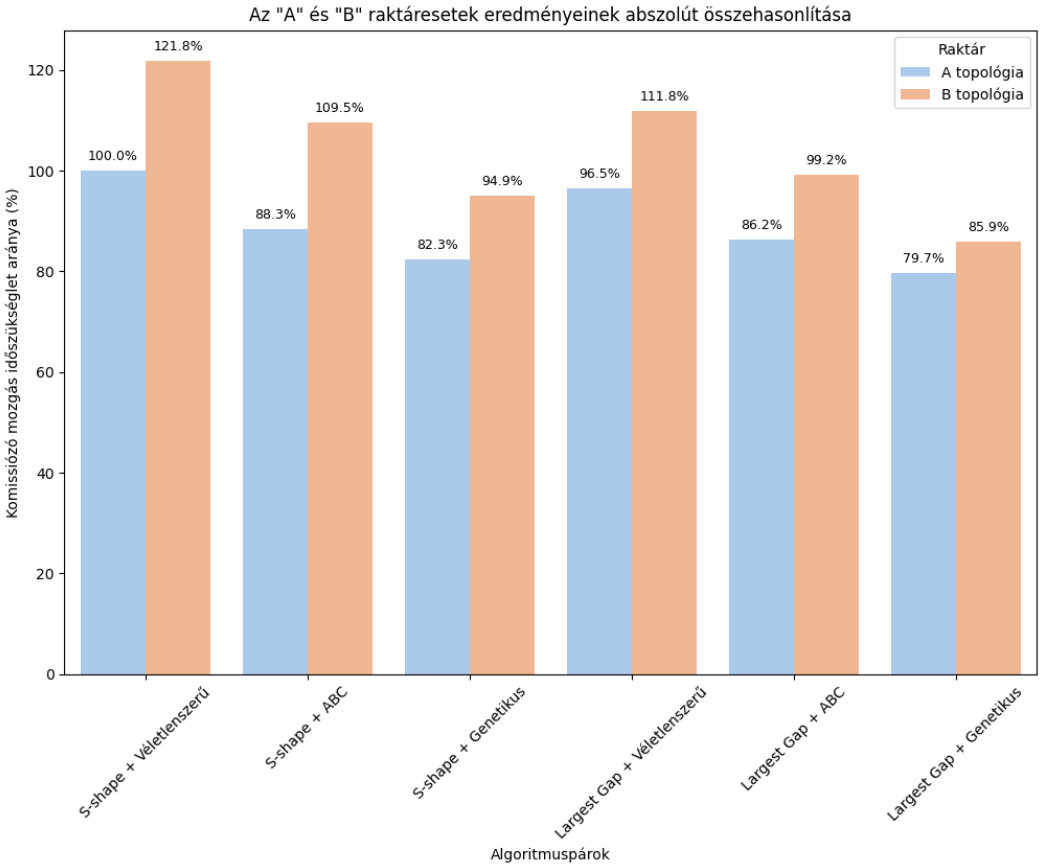
Az ***25. ábra*** azt mutatja, hogy melyik raktártopológia esetében milyen eredményeket értek el az egyes algoritmuspárok. Mindkét esetben a 100% az azonos raktártopológia az S-shape algoritmus - véletlenszerű tárhelykiosztó algoritmuspár volt.



25. ábra: *Az algoritmuspárok teljesítményei mindkét raktáresetnél*

A ***25. ábra*** alapján megfigyelhető egyrészt, hogy az algoritmusok hatékonyságának a sorrendje mindkét raktártopológia esetében ugyan az. Ugyanakkor a B topológia esetében minden algoritmuspár jobban teljesít az első algoritmuspárhoz képest, mint az A topológia esetében.

Érdemes abszolút mértékben is vizsgálni, hogy összességében mely algoritmuspár és raktártopológia, milyen hatékonysággal tudta elvégezni az adott feladatot. Ezt az összehasonlítást mutatja a ***26. ábra***. Ebben az esetben a 100%-ot az „A” raktártopológia S-Shape +véletlenszerű tárhelykiosztás algoritmuspár jelentette. Ehhez viszonyítottam az összes többi algoritmuspár és raktártopológia teljesítményét. Látható, hogy ebben az esetben, algoritmuspáronként mindig az „A” raktártopológia esetei nyújtottak jobb megoldást, ugyanakkor érdemes lesz majd vizsgálni ezen eltérések mértékét.



26. ábra: *Az algoritmuspárok és raktártopológiák abszolút összehasonlítása*

6. Az eredmények elemzése és értékelése

Ebben a fejezetben alaposan elemezem azokat az eredményeket és adatokat, amelyeket a kísérletek során gyűjtöttem, és amelyeket az előző fejezetben részleteztem. Ezen adatok és elemzések alapján igyekszem konklúziókat és megállapításokat alkotni. Ugyanakkor felmerülhetnek olyan következtetések, amelyeket az adatok alapján nem lehet maradéktalanul megerősíteni, mégis fontosak lehetnek további gondolatindítónak és későbbi részletes kutatások kiindulópontjának.

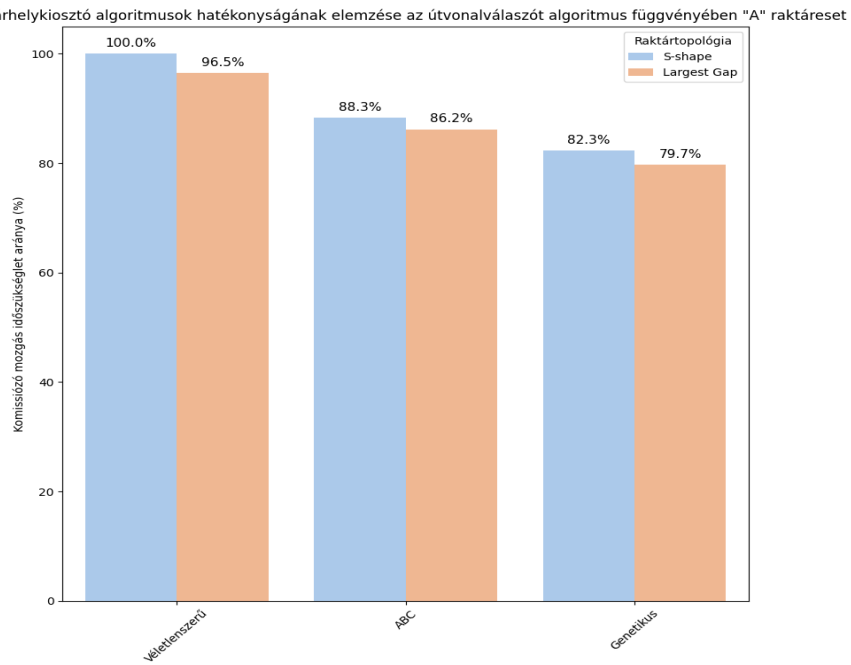
6.1.Az algoritmuspárok hatékonysága

6.1.1. Eredményességük

Mindkét raktár topológia esetében az algoritmus párok hatékonyságuk alapján ugyanazt a sorrendet adták:

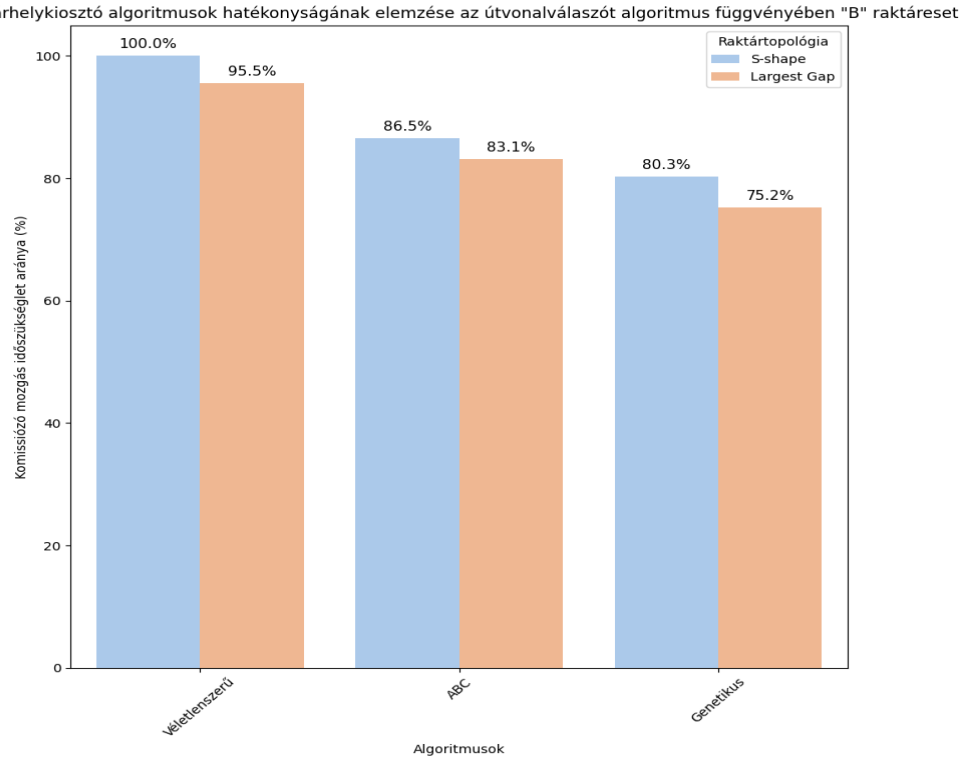
1. Largest gap + Genetikus
2. S-shape + Genetikus
3. Largest gap + ABC
4. S-shape + ABC
5. Largest gap + véletlenszerű
6. S-shape + véletlenszerű

Az algoritmusok sorrendje mellett érdemes megvizsgálni a **7. táblázat** és a **9. táblázat** adatait. A konfidenciaintervallumok csak minimális átfedést mutatnak, ami arra utal, hogy ritkán fordul elő eltérés az algoritmuspárok hatékonysági sorrendjében. Továbbá, figyelembe véve, hogy mindkét, egymástól lényegesen különböző raktártopológia esetén azonos az algoritmusok sorrendje és hasonlóak az értékeik, valószínűsíthető, hogy ez a sorrend nem csupán egyedi eset, hanem általános tendencia.



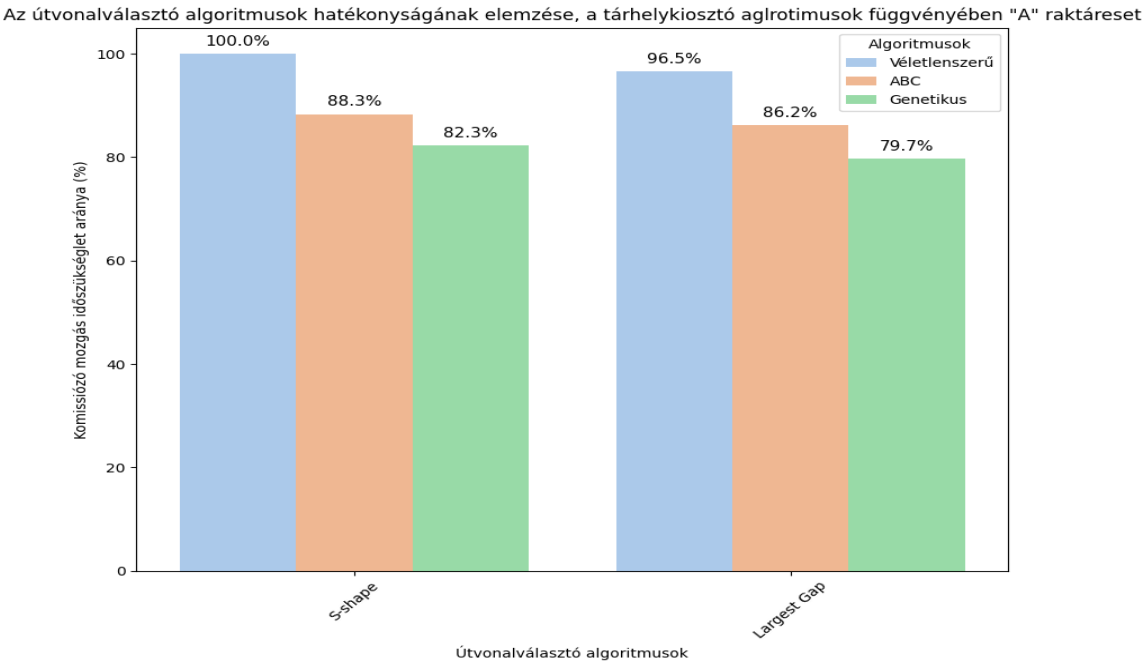
27. ábra: *Az tárhelykiosztó algoritmusok hatékonyságának elemzése az útvonalválaszót algoritmus függvényében "A" raktáreset*

A ***27. ábra*** és a ***28. ábra*** a két különböző raktártopológia tárhelykiosztási algoritmusainak teljesítményét szemléltetik az útvonalválasztó algoritmusoktól függően. Mindkét esetben világosan látható, hogy a komplexebb algoritmusok jobb eredményeket érnek el.



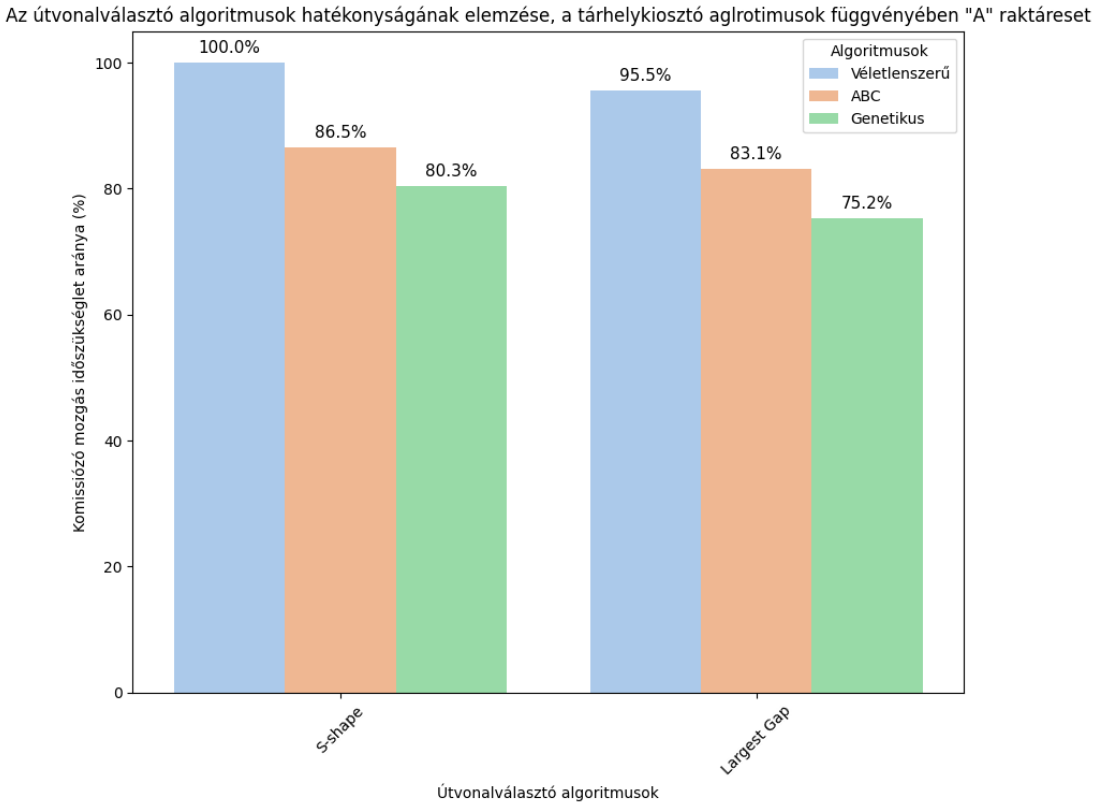
28. ábra: *Az tárhelykiosztó algoritmusok hatékonyságának elemzése az útvonalválaszót algoritmus függvényében "B" raktáreset*

Érdemes megvizsgálni, hogy a tárhelykijelölési algoritmusoknál milyen tendencia figyelhető meg.



29. ábra: *Az útvonalválasztó algoritmusok hatékonyságának elemzése, a tárhelykiosztó algoritmusok függvényében "A" raktáreset*

A ***29. ábra*** valamint a ***30. ábra*** egyes tárhelykiosztási algoritmusok hatékonyságát szemléltetik. Ebben az esetben is észlelhető, hogy a komplexebb algoritmusok mindkét raktárkonfigurációban jobb teljesítményt nyújtanak, mint az egyszerűbb megoldások.

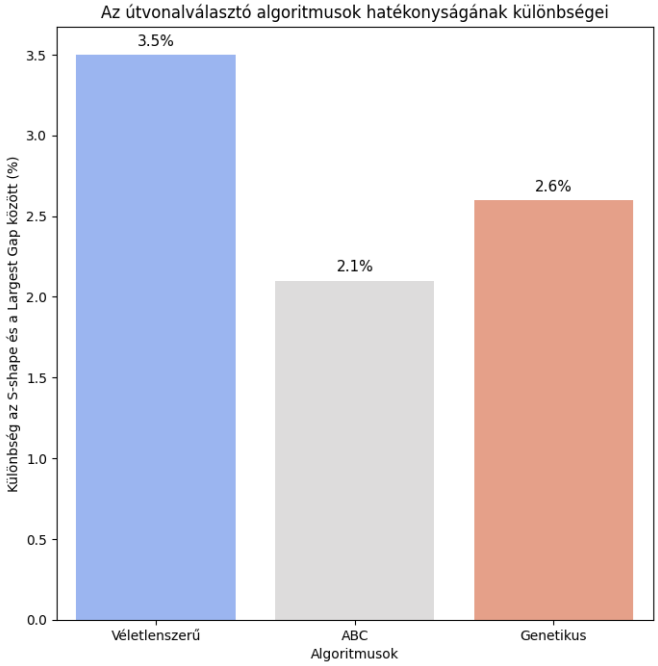


30. ábra: *Az útvonalválasztó algoritmusok hatékonyságának elemzése, a tárhelykiosztó algoritmusok függvényében "B" raktáreset*

Ugyanakkor biztosra kimondani, hogy a bonyolultabb algoritmus javítja az eredményt csak további raktártopológiákon való tesztelés után lehet, de az adatok e feltételezés irányába mutatnak.

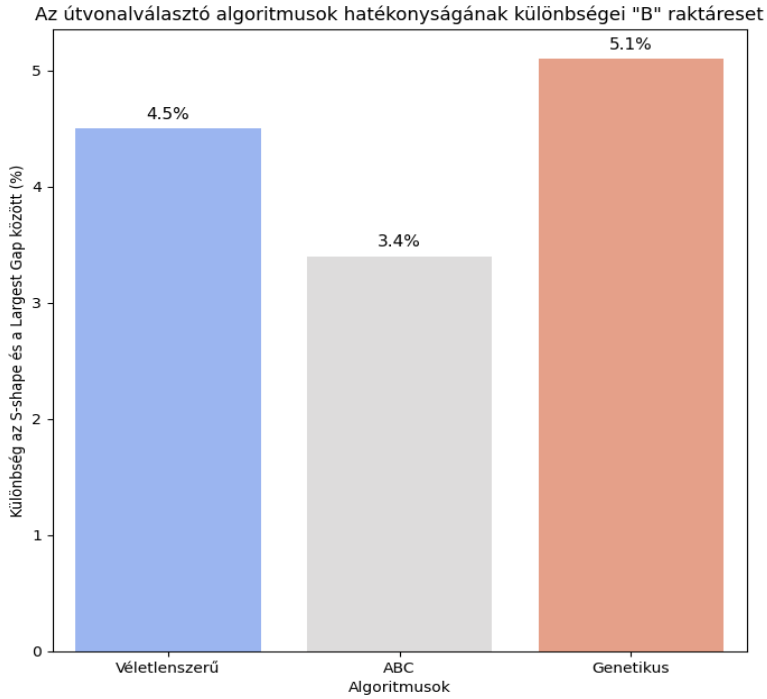
6.1.2. Útvonalkeresés és tárhelykiosztás

Az adatok elemzése során, felmerült bennem az a kérdés, hogy a komissiózási folyamat hatékonysága szempontjából, a tárhelykiosztási, vagy pedig az útvonal keresési probléma hatékony megoldása okozz-e nagyobb hatékonyságjavulást.



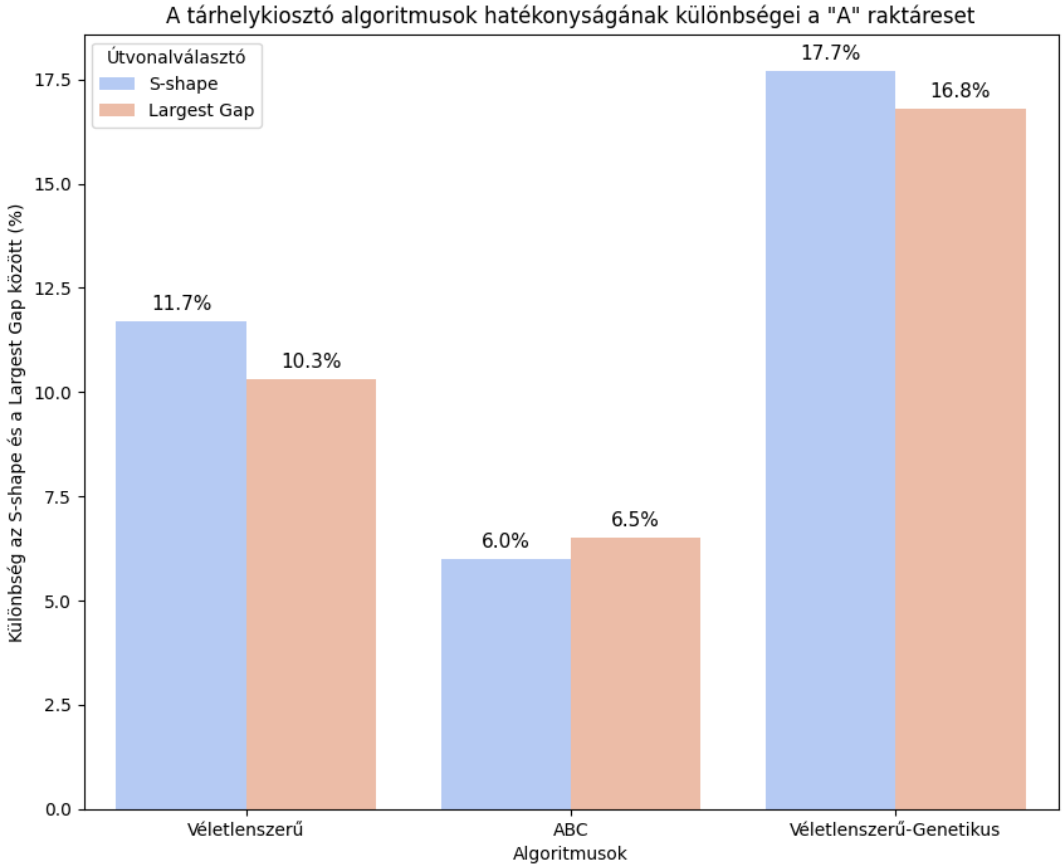
31. ábra: *Az útvonalválasztó algoritmusok hatékonyságának különbségei "A" raktáreset*

A ***31. ábra*** és a ***32. ábra*** az egyes útvonalválasztó algoritmusok által okozott változásokat mutatják „A” és „B” raktáresetekben. Mindkét esetben az S-shape algoritmus értékéből kivontam a Largest Gap ugyanazon tárhelykiosztó algoritmusához tartozó százalékos értékeket így kaptam meg az alábbi eredményeket melyek százalékpontban mutatják a különbségeket.



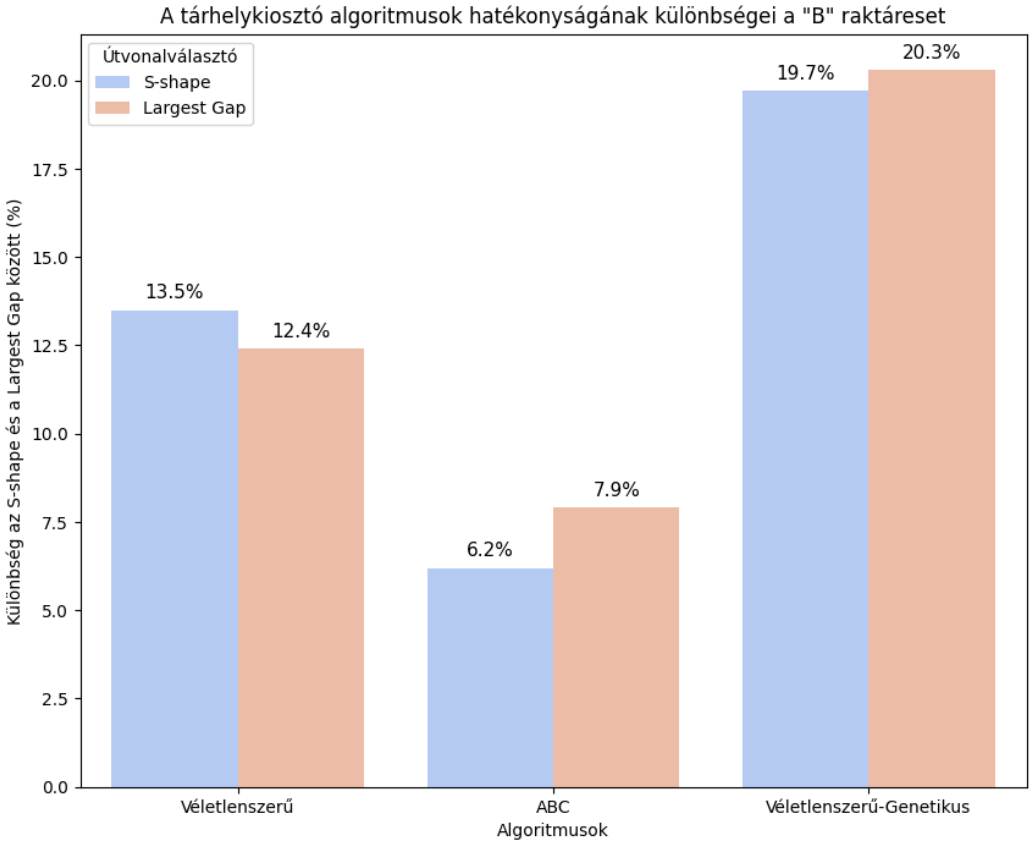
32. ábra: *Az útvonalválasztó algoritmusok hatékonyságának különbségei "B" raktáreset*

Megfigyelhető, hogy mindkét esetben az ABC algoritmusnál a legkisebb a változás és a genetikus algoritmusnál a legnagyobb.



33. ábra: *A tárhelykiosztó algoritmusok hatékonyságának különbségei a "A" raktáreset*

A ***33. ábra*** és ***34. ábra*** az egyes tárhelykiosztó algoritmusok által okozott különbségeket mutatja az „A” és a „B” raktáresetben. Megfigyelhető, hogy a legnagyobb eltérés a véletlenszerű és a genetikus algoritmus között van, a legkisebb pedig az ABC és a genetikus algoritmus között. Ez az eredmény egybe vág egy korábbi kutatásom eredményével, melyben ezen algoritmusok egymás közti eredményessége volt a vizsgálat középpontjában.

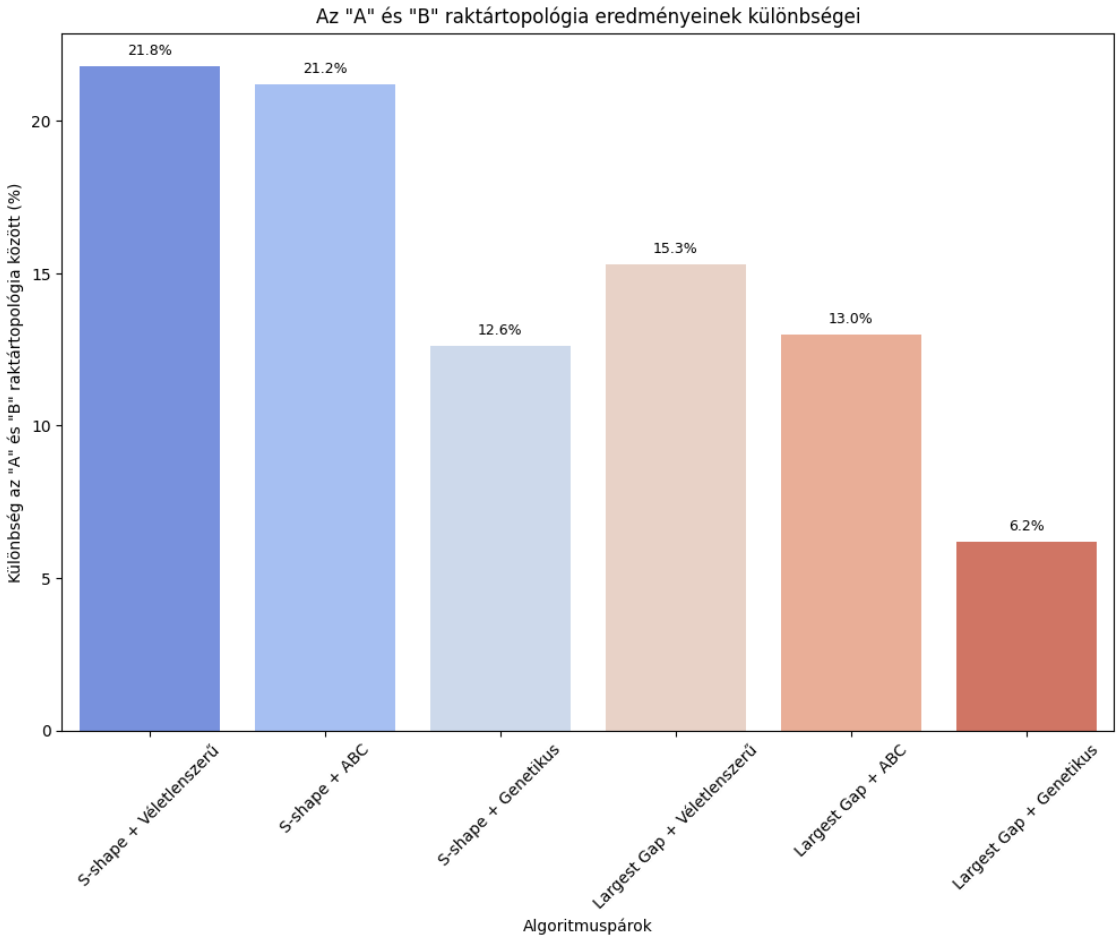


34. ábra: *A tárhelykiosztó algoritmusok hatékonyságának különbségei a "B" raktáreset*

Az adatokat megvizsgálva ezen tesztek alapján úgy tűnik, hogy a tárhelykiosztási probléma hatékonyabb megoldása okozza összességében a nagyobb hatékonyságnövekedést. Fontos azonban megjegyezni, hogy a tárhelykiosztási probléma esetében három szélsőségesen különböző hatékonyságú algoritmust teszteltem, (melyekkel már régebben is dolgoztam és voltak tapasztalataim a hatékonyságukkal kapcsolatban), míg a komissiózási útvonal meghatározása során két nem drasztikusan különböző heurisztikát hasonlítottam össze. Az előbb említett kérdés pontos megválaszolására, miszerint melyik probléma hatékony megoldása okozza a nagyobb hatékonyságnövekedést érdemes lenne a továbbiakban a legkevésbé és a leginkább hatékony útvonalmeghatározási algoritmusokkal is megvizsgálni az egyeseseteket.

6.1.3. Az „A” és a „B” eset közti eltérések elemzése

Érdemes megvizsgálni a két raktáreset közti különbségeket, melyet a ***35. ábra*** jelenít meg. Az ábrán minden esetben a „B” raktártopológiánál kapott értékekből vontam ki az „A” raktártopológia értékeit és így százalékpontban ábrázolom a kettő közti eltérést.



35. ábra: *"A" és "B" raktártopológia közti különbségek*

Az ábrán megfigyelhető, hogy minél komplexebb algoritmuspárról beszélünk, annál kisebb az eltérés a két eset között. Ebből két következtetést lehet levonni. Egyrészt, minél optimálisabban van megtervezve egy raktár, annál kevesebb javítást tudnak okozni a hatékony algoritmusok. Másrészt, ha a raktár maga nem optimális, a megfelelő folyamatirányító algoritmusokkal, igen jó eredményeket lehet elérni a működés hatékonyságának tekintetében.

6.2. Szinergia, vagy tompítás

A kutatás során felmerült kérdés, hogy a különböző problémamegoldó algoritmusok közötti kölcsönhatás hogyan befolyásolja az egyes algoritmusok hatékonyságát. A tesztek egyik célja az volt, hogy megállapítsam, van-e szinergia vagy tompítás a komissiózó útvonalválasztó és tárhelykijelölő algoritmusok között. Amennyiben szinergia fedezhető fel az algoritmusok együttes használata esetén az eredmények jobbak, mint az egyes algoritmusok külön-külön történő alkalmazásának összegénél várható lenne. Tompítás esetén az együttes hatás kisebb, mint az egyes algoritmusok hatásainak összege.

A statisztikai elemzéshez az ANOVA (analízis a varianciák között) módszert választottam, amely lehetővé teszi több csoport közötti átlagok összehasonlítását és megmutatja, hogy a különböző faktorok milyen mértékben befolyásolják a változókat. Az ANOVA modellben két fő faktort (a komissiózó útvonalválasztó és a tárhelykijelölő algoritmusokat) és azok interakcióját vizsgáltam.

Az ANOVA táblázat a következő eredményeket mutatta:

10. táblázat: ANOVA elemzés

	SS	*Df*	*F-érték*	*p-érték*
C (útvonalválasztó alg.)	37,45	1,00	30,03	0,00
C (tárhelykijelölő alg.)	712,69	2,00	285,68	0,00
C (útvonal választó alg.):C(tárhelykijelölő alg.)	0,93	2,00	0,37	0,71

Az eredmények alapján megállapítható, hogy mind az útvonalválasztó, mind a tárhelykijelölő algoritmusok jelentős hatással vannak az eredményekre, azonban az interakciós tag p-értéke (0.706051) azt mutatja, hogy nincs statisztikailag szignifikáns kölcsönhatás a két algoritmus között. Ez azt jelenti, hogy nincs bizonyíték arra, hogy az algoritmusok együttes alkalmazása erősíti, vagy gyengíti az egyes algoritmusok hatékonyságát.

Ez a függetlenség azt sugallja, hogy a komissiózó útvonalválasztó és tárhelykijelölő algoritmusok hatékonysága additív módon kombinálódik, amikor együtt alkalmazzák őket. A gyakorlatban ez azt jelenti, hogy a rendszer teljesítményének javításához az egyes algoritmusokat külön-külön kell optimalizálni, mivel azok együttes hatása nem eredményez jelentős változást a teljesítményben.

A fenti eredmények azt mutatják, hogy a választott algoritmusok hatékonyan működnek az adott feladatokon, de a hatékonyságuk nem függ össze a másik algoritmus választásával. Ez az információ hasznos lehet a rendszerek tervezésekor, ahol több algoritmust is alkalmaznak, mivel az egyes algoritmusok kiválasztása és finomhangolása függetlenül történhet meg anélkül, hogy figyelembe kellene venni a másik algoritmus hatását.

Összegzésképpen, az ANOVA elemzésünk alapján nem találtam bizonyítékot arra, hogy a komissiózó útvonalválasztó és tárhelykijelölő algoritmusok közötti kölcsönhatás szinergiát vagy tompítást eredményezne. Az eredményeket további kutatásokkal kell alátámasztani, különösen olyan környezetekben, ahol az algoritmusokat további változatos és ugyancsak komplex feladatokra alkalmazzák.

6.3. Összegzés

A 6. fejezetben tárgyaltam a különböző raktári problémamegoldó algoritmusok hatékonyságait, az általuk mutatott mintázattokat és azok kölcsönhatásait. A komissiózási időszükséglet alapján egyértelmű sorrend állítható fel az algoritmuspárok tekintetében. Ezen sorrendet mindkét raktáreset alátámasztja, továbbá a konfidenciaintervallumaik is csak minimális átfedést mutatnak.

A ***27. ábra*** és a ***28. ábra*** valamint a ***29. ábra*** és a ***30. ábra*** adatai is alátámasztják azt a felvetést, miszerint a bonyolultabb algoritmusok hatékonyabb megoldást adnak. Ugyanakkor én csak két raktáresetre vizsgáltam a kérdést. Ahhoz, hogy általános szabályt lehessen alkotni érdemesebb több esetet megvizsgálni. Külön érdekes kérdés lehet, hogy milyen olyan esetek vannak (ha vannak) ahol az egyszerűbb útvonalmeghatározó, illetve tárhelykiosztó algoritmusok jobban tudnak teljesíteni.

A 6.1.2-es fejezet adatai azt mutatják, hogy a tárhelykiosztó algoritmusok jobban befolyásolják a hatékonyságot, mint az útvonalmeghatározó algoritmusok. Ezt támasztja alá a **10. táblázat** is. Ugyanakkor ezt a felvetést erősen befolyásolhatja, hogy míg a tárhelykiosztásnál nagyon különböző hatékonyságú módszereket vizsgáltam, addig az útvonalmeghatározásnál egymástól nem radikálisan különböző heurisztikákat teszteltem. Ezért ez a felvetés további vizsgálatokra szorul.

A ***32. ábra*** és a ***33. ábra*** rávilágít, hogy az algoritmusok hatékonysága hogyan változik a raktári környezet függvényében. Minél kevésbé optimális egy raktár az adott elvárásokra, annál nagyobb megtakarítások érhetőek el a hatékony szervező algoritmusokkal. Ezt a felvetést is érdemes további raktáreseteken is tesztelni, ugyanis ez számos ipari esetben fontos kérdés lehet. Gondolva itt például olyan működő raktárakra, ahol a raktár kialakítása nem megfelelő és túl költséges lenne az átalakítás, de adott esetben a probléma kellőképpen orvosolható hatékony szervező algoritmusokkal.

Az 6.2-es alfejezetben a szinergia és tompítás témakörét vizsgáltam, ahol ANOVA elemzéssel megvizsgáltam, hogy az algoritmusok együttes használata erősíti vagy gyengíti-e azok hatékonyságát. Az elemzés nem talált bizonyítékot, hogy az algoritmusok között statisztikailag szignifikáns kölcsönhatás lenne, ami azt sugallja, hogy az algoritmusok hatékonysága additív módon kombinálódik és az együttes használatuk nem hoz létre szinergiát vagy tompítást. Ez azt jelenti, hogy a rendszer teljesítményének javítása érdekében a fejlesztő és logisztikaimérnököknek külön-külön kell megvizsgálniuk és finomhangolniuk az egyes algoritmusokat.

7. További kutatási kérdések

A dolgozat során bemutatott vizsgálatok és elemzések számos új kérdést vetettek fel, amelyek további kutatásokat igényelnek a raktári folyamatok problémáinak algoritmikus megoldásában. Az alábbiakban vázolom a legfontosabb kérdéseket és irányokat, amelyeket a jövőben szívesen vizsgálnék.

7.1. A dolgozatban felmerült kérdések és felvetések

A dolgozatban, különösen a 6. fejezetben, számos olyan felvetést fogalmaztam meg, melyek további vizsgálatra lehetnek érdemesek:

- Léteznek-e olyan esetek, raktártopológiák, megrendelési adathalmazok, ahol az egyszerűbb algoritmusok (mint például az S-shape), hatékonyabb és jobb eredményeket érnek el, mint a bonyolultabbak? Ha léteznek, akkor melyek ezek és miért történhet meg?
- Hogyan befolyásolják az algoritmusok működését a különböző listaösszefogási és csoportosítási logikák?
- Összességében a tárhelykiosztási, vagy az útvonalkijelölési probléma hatékonyabb megoldása eredményez-e nagyobb hatékonyságnövekedést?
- Biztos-e hogy az optimumtól távolibb raktártopológiák esetében nagyobb arányú fejlesztési potenciál van az útvonalkijelölő és az tárhelykiosztó algoritmusok számára?

7.2. A módszerek még általánosabb használata és fizikai környezetben történő tesztelése

Az általam alkotott rendszer tartalmaz a 4.2.3-as fejezetben meghatározott kikötéseket. Érdemes lehet a rendszert tovább fejleszteni és csökkenteni ezen kikötések számát. Valamint hasznos lehet a modell tesztelése valós fizikai környezetben, hiszen számos egyéb kérdés felmerülhet a valós alkalmazás során, mely kérdésekre nem feltétlenül derül fény a téma elméleti vizsgálata során.

A legfontosabb megoldandó feladatok azok lennének, hogy a rendszer képes legyen olyan szituációkat is kezelni, mikor egy termék több tárhelyen van, vagy éppen egy tárhelyen több termék is megtalálható. Továbbá a függőleges irányú mozgás komponenst is érdemes lenne implementálni.

7.3. AI és a genetikus optimalizáló algoritmusok

A mesterséges intelligencia (AI) és a genetikus algoritmusok gyors fejlődése új lehetőségeket nyit a raktárlogisztika területén. Érdemes lenne további kutatásokat végezni az AI-alapú technikák raktári problémákra való alkalmazhatóságáról. A genetikus algoritmusok továbbfejlesztése, különösen az adaptív és hibrid megközelítések felé való elmozdulás, szintén reménykeltő területnek ígérkezik, amely elősegítheti a raktári műveletek hatékonyságának növelését.

8. Irodalomjegyzék

[1] "A Brief History of Logistics - Universal Cargo." Accessed: May 18, 2023. [Online]. Available: https://www.universalcargo.com/a-brief-history-of-logistics/

[2] "Supply Chain Management (SCM): How It Works and Why It Is Important." Accessed: May 18, 2023. [Online]. Available: https://www.investopedia.com/terms/s/scm.asp

[3] Dr. Bóna Krisztián, *Ellátási-elosztási rendszerek, előadásjegyzet*. 2019.

[4] Dr. Prezenszki József, *Logisztika I.II.* 1999.

[5] Dévai Zoltán Fésüs Norbert Kosztolányi János Kővári Róbert Nika Tamás, *Lean szolgáltatásfejlesztés 1.* 2020.

[6] Dr. Bóna Krisztián, *Anyagmozgatási és raktározási folyamatok* . 2020.

[7] "Tárolási technológiák 3. – Soros állványos tárolás - INNOLOG SOLUTIONS Kft." Accessed: May 13, 2023. [Online]. Available: https://innolog.hu/tarolasi-technologiak-soros-allvanyos-tarolas/

[8] Molnár Balázs, "Raktári komissiózási folyamatok tervezése és irányítása".

[9] Timm Gudehus, *Grundlagen der Kommissioniertechnik*. 1999.

[10] R. de Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *Eur J Oper Res*, vol. 182, no. 2, pp. 481–501, Oct. 2007, doi: 10.1016/j.ejor.2006.07.009.

[11] R. Puka, J. Duda, and M. Karkula, "IMPROVING S-SHAPE ROUTING STRATEGY FOR ORDER PICKING Order picking optimization for huge warehouses View project Application of genetic programming to management problems View project IMPROVING S-SHAPE ROUTING STRATEGY FOR ORDER PICKING." [Online]. Available: https://www.researchgate.net/publication/343524829

[12] "Website of Kees Jan Roodbergen." Accessed: Apr. 11, 2023. [Online]. Available: https://roodbergen.com/warehouse/background.php

[13] KJ Roodbergen, *Layout and Routing Methods for Warehouses*. 2001.

[14] Bertalan Marcell, *Anyagmozgatási és raktározási rendszerek, gyakorlat jegyzet*. 2021.

[15] J. J. R. Reyes, E. L. Solano-Charris, and J. R. Montoya-Torres, "The storage location assignment problem: A literature review," *International Journal of Industrial Engineering Computations*, vol. 10, no. 2. Growing Science, pp. 199–224, Apr. 01, 2019. doi: 10.5267/j.ijiec.2018.8.001.

[16] Ren-Qian Zhanga, "New model of the storage location assignment problem considering demand correlation pattern," *Science Direc*, 2019.

[17] W. H. Hausmann, "Optimal storage assignment in automatic warehousing system".

[18] E. H. Frazelle, "Stock location assignment and order picking productivity".

[19] W. Wang, J. Gao, T. Gao, and H. Zhao, “Optimization of Automated Warehouse Location Based on Genetic Algorithm,” 2017.

[20] D. Zhang, Y. Si, Z. Tian, L. Yin, J. Qiu, and X. Du, “A genetic-algorithm based method for storage location assignments in mobile rack warehouses,” in *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Dec. 2019. doi: 10.1109/GLOBECOM38437.2019.9013447.

[21] D. Ming-Huang Chiang, C. P. Lin, and M. C. Chen, “Data mining based storage assignment heuristics for travel distance reduction,” *Expert Syst*, vol. 31, no. 1, pp. 81–90, Feb. 2014, doi: 10.1111/exsy.12006.

[22] Dr. Bóna Krisztián, “Simulation and Optimization of Logistic Systems with Genetic Algorithms”.

[23] Genetikus algoritmusok, *Genetikus algoritmusok*. 2003.

[24] D. Quammen, “Was Darwin Wrong? @ National Geographic Magazine Was Darwin Wrong?,” 2004. [Online]. Available: http://magma.nationalgeographic.com/ngm/0411/feature1/fulltext.html

[25] C. Battista, “Storage Location Assignment Problem: implementation in a warehouse design optimization tool,” 2011.

[26] Dr. Bóna Krisztián, *A raktár-logisztikai rendszerek tervezése*. 2022.

[27] “https://docplayer.hu/6282753-Dfg-tfg-316-320-425-430-435.html,” 2023.10.03.