Eötvös Loránd Tudományegyetem

Természettudományi Kar

Tóth Lilla

Az újpesti cérnagyár raktározásának optimalizálása

MSc Szakdolgozat

Témavezetők:

Horváth János, Coats Hungary Ltd.

Szabó Csaba, Algebra és Számelmélet Tanszék

Budapest, 2015

Tartalomjegyzék

1. Az újpesti cérnagyár **3**

1.1. Bevezető 3

1.2. A raktározás alapelvei 4

2. A lokációk meghatározása - helykiosztási feladat **6**

2.1. Raktározási folyamatok 6

2.2. A raktár elrendezésének megtervezése 8

2.3. Lokációk meghatározása 9

2.4. Zónák kialakítása 12

2.5. Megrendelések csoportosítása 12

2.6. Útvonal tervezési módszerek 12

2.7. Felhalmozás és válogatás 15

3. Megrendelések csoportosításának megoldási módszerei **16**

3.1. A feladat felírása egészértékű feladatként 16

3.2. Konstruktív megoldási módszerek 18

3.3. Megoldás metaheurisztikus módszerekkel 22

3.4. Kiszedés hullámokban 25

3.5. Pick listák ütemezése 26

3.6. Dinamikus csoportosítás 29

4. Eredmények a gyakorlatban **31**

4.1. Útvonal minimalizálása a pick listák bontásával 31

4.2. Útvonal minimalizálása a termékek optimális elhelyezésével 32

1. fejezet

Az újpesti cérnagyár

Az újpesti cérnagyár a Coats cégcsoport része. A Coats vállalat az 1750-es évektől gyárt cérnát, ma a világ piacvezető ipari és ruházati cérnagyártója. Több mint 20.000 alkalmazottat foglalkoztat 6 földrészen, több mint 70 országban. Termékeit a világ több mint 100 országában forgalmazzák, és a világon nagyjából minden ötödik ruhát a Coats vállalat cérnái tartanak egyben. A cérnák mellett fonalak, cipzárak és egyéb, a varráshoz és kézimunkázáshoz szükséges termékek széles skáláját gyártják és forgalmazzák világszerte.

A magyarországi gyárban a gyártás alapanyaga a nyers cérna, amelyet festés után kiszerelnek különböző méretű csévékre. Ezeket csomagolják, majd szállítják ki a megrendelőkhöz. Ezen felül elhelyezkedéséből adódóan az újpesti telephely nem csak gyártó funkciót lát el, hanem a nyugatra menő szállítmányokat itt csomagolják.

1.1. Bevezető

A cérnagyárral a kapcsolatot 2014 novemberében vettük fel annak reményében, hogy be tudok kapcsolódni egy olyan projektbe, amely a szakdolgozatom alapját képezheti, és a cérnagyár számára is hasznos lehetek.

Először Horváth Jánossal találkoztunk, aki a külső témavezetőm lett. Horváth János közbenjárására a Coats Hungary vezérigazgatója, Simona Sava engedélyezte, hogy a szakdolgozatomat a cérnagyárban írhassam és a számításaimhoz szükséges adatokat megkaphassam. A második alkalommal Táborszki Tamás tartott egy kis bevezetőt a logisztika életébe, és felvetett pár kérdést, mint például mit érdemes inkább készleten tartani, késztermékет vagy köztes terméket.

A számos megbeszélés eredményeképp végül a CIP, azaz Continuous Improvment csa-

pathoz csatlakoztam, ahol Bujtár Norbert és Józsa Imre bemutatta Boros Gergely pick listákra vonatkozó projektjét. A projekt célja annak megállapítása volt, hogy mennyi idő takarítható meg a pick listák bontásával, azaz mennyi idő takarítható meg, ha egy pick listán nem szerepel egyszerre ipari és ruházati termék is, mivel ezeket a raktár két végében tárolják.

1.2. A raktározás alapelvei

Az újpesti gyár EDC raktára több funkciót lát el. Itt tárolják a gyárban megfestett majd kiszerelt cérnákat, illetve a vállalat más gyáraiból érkező és tovább szállítandó cérnát és egyéb termékeket. Itt történik a megrendelések összeállítása, és csomagolása is.

A raktárban a termékeket polcrendszereken és úgynevezett palettákon tárolják. A polcrendszer oszlopokból áll, egy oszlopban 5 polc van, ezeket bineknek nevezik. A paletta egy olyan állványrendszer, melyen raklapnyi mennyiségű árut lehet tárolni. Azt, hogy a beérkező árut binekben vagy palettán tárolják, az áru mennyisége dönti el.

Amikor egy termék beérkezik a raktárba, kap egy úgynevezett lokációt, amely pontosan megmondja a helyét a raktárban. A lokációk elnevezése úgy van kialakítva, hogy amikor a megrendelések csomagolásához összegyűjtik a termékeket, az útvonal a lokációk alfabetikus rendezésével kapható. A legtöbb Coats raktárban a termék lokációjának meghatározása dinamikusan történik az alábbi szempontok alapján.

- Az ipari és a ruházati termékeket a raktár különböző részében tárolják.
- A megrendelésre gyártott termékek külön helyre kerülnek, függetlenül az egyéb tulajdonságaiktól.
- Palettára kerülő termék esetén az újonnan érkezett termék nem kap új palettát, ha a korábban elraktározott ugyanolyan termék mellé befér legalább az új mennyiség fele.
- Ha a beérkező termék mennyisége pont egy bin, akkor kap egy saját bint.
- Ha a termék mennyisége kevesebb, mint egy bin, akkor a rendszer megnézi, hogy van-e már ugyanilyen termék a raktáron, és ha igen, akkor az újonnan jött terméket is a mellé teszi be. Ha az eddig tárolt és újonnan jött termék mennyisége együttesen meghaladja a bin méretét, akkor is feltölti az adott bint, és a maradék mennyiségnek nyit egy újat.

- A gyorsan fogyó termékeket a csomagolási helyhez közelebb, a polcokon belül pedig kézmagasságban helyezik el.

Az újpesti raktárban az utóbbi szempontot jelenleg nem veszik figyelembe a lokáció meghatározásakor, de egy későbbi projektben tervezik ennek megvalósítását.

A megrendelések teljesítéséhez az áruk kiszedését a következőképpen végzik. Az adott műszakban teljesítendő megrendeléseket a rendszer összesíti és úgynevezett pick listákra bontja. Egy pick listán több megrendelő megrendelései is szerepelhetnek. A pick listákon szereplő termékeket az adott műszakban dolgozók egymástól függetlenül, egyszerre gyűjtik be. A dolgozók először felveszik a soron következő pick listát, azaz a PDA készülékükkel beolvassák a soron következő pick lista vonalkódját. Ekkor a PDA megmutatja a pick listában szereplő első termék lokációját. Ha a dolgozó begyűjtötte az első terméket, a PDA készülékében bejelöli, hogy a termék már az összeszedő kocsin van, mire az megmutatja a soron következő termék lokációját. Ha az utolsó terméket is begyűjtötte, akkor a PDA készülék segítségével kinyomtatja az általa összeszedett megrendelések vonalkódját, és leadja a termékeket a csomagoló részlegnél. Ezzel a pick listát teljesítette, így felveheti a következőt.

2. fejezet

A lokációk meghatározása - helykiosztási feladat

A lokációk meghatározásához az úgynevezett helykiosztási feladatot (storage assignment) kell megoldanunk. A feladat az, hogy határozzuk meg a termékek helyét a raktárban úgy, hogy a raktár bizonyos teljesítménymutatói optimálisak legyenek. A következő néhány alfejezetben áttekintjük a raktározás folyamatait, a felmerülő célfüggvényeket, és a gyakorlati megoldásokat a helykiosztási problémára.

2.1. Raktározási folyamatok

Általában a raktáraknak a következő fő funkciókat kell ellátniuk: beérkező áruk fogadása, átszállítása, szortírozása, elraktározása, kiszedése, csomagolása, majd kiszállítása. Ezek közül az áruk kiszedése (order picking) teszi ki a raktározási költségek több, mint 55%-át. Ebbe a munkafolyamatba beletartozik a megrendelések csoportosítása és ütemezése, kiszedési listák összeállítása, a listán szereplő tételek kiszedése és az árukészlet változásának dokumentálása, nyomon követése.

Megrendelések kiszedése

A legtöbb raktárban a kiszedés nem automatizált, hanem kézi erővel történik. Ekkor a dolgozók a sorok között haladva veszik le a polcokról a kiszedési listákon lévő tételeket. Előfordulhat, hogy a raktárban nem csak kézmagasságban tárolnak termékeket. Ekkor a dolgozók targoncával vagy daruval közlekednek, és annak segítségével gyűjtik be a meg-

rendelések teljesítéséhez szükséges termékeket.

Kézi kiszedés esetén érdemes lehet a megrendeléseket csoportosítani, és azokat egyszerre kiszedni. Ha a megrendelések önmagukban is elég nagyok, akkor erre nem feltétlen van szükség. Egy másik megközelítés a kiszedési idő rövidítésére az, hogy a raktárat zónákra bontjuk, és minden dolgozó csak egy zónában szedi össze kiszedési listán szereplő tételeket. Mindkét megközelítéssel a későbbiekben részletesebben is foglalkozunk.

Egy automatizált raktárban minden sorban egy daru mozog. Ezek a daruk szállítják a termékeket a dolgozókhoz, akik leveszik róla a megfelelő mennyiséget, ezután a daruk visszaviszik a maradékot a raktárba. A daruk többféleképpen működtethetők. Az egyszeres vezérlésű daru egyszerre egy terméket visz a leszedési pontra, majd azt rögtön a leszedés után visszaviszi a helyére. A párhuzamos vezérlésű daru két terméket is kezelhet párhuzamosan, míg a többszörös vezérlésű több terméket is vihet a leszedési pontra, illetve vissza a raktárba.

A továbbiakban feltesszük, hogy a kiszedés nem automatizált, és minden termék kézmagasságban van, így nem kell targoncát vagy darut használni a kiszedéséhez. A tapasztalatok alapján Nyugat-Európában ez a legjellemzőbb módszer, a raktárak nagyjából 80%-ában ezt alkalmazzák.

Ha nagyon rövid átfutási idővel kell teljesíteni megrendeléseket, akkor a megrendeléseket csoportosítva vagy zónás módszerrel szokták kiszedni. Ekkor viszont a kiszedés után ki kell válogatni, hogy melyik tételek tartoznak egy megrendeléshez. Ezért a kiszedett tételek egy elosztó központba kerülnek, ahol az ottani dolgozók szétválogatják a megrendeléseknek megfelelően. Ilyen elosztó rendszert használ például az Amazon németországi raktára is.

A valóságban megtervezni egy áru kiszedési rendszert meglehetősen bonyolult feladat, mivel a tervezést külső- és belső tényezők is befolyásolják. Külső tényezők lehetnek: a marketing csatornák, a vásárlók igényének változása, a beszállítók, a készletszintek, az összes igény egy termékre, a gazdaság állapota. Belső tényezők például a rendszer jellemzői, vagy a szervezeti- és működési szabályzatok.

Megrendelések kiszedésének célfüggvényei

A legtöbb vállalatnál a cél a kiszolgálás színvonalának maximalizálása, ha adott a munkaerő- és gépkapacitás, illetve a költség. A kiszolgálás színvonalát több tényező határozza meg, mint például az átlagos kiszállítási idő, az ettől való eltérés, és a megrendelés teljesítésének pontossága. A raktár ehhez azzal tud hozzájárulni, ha a megrendelési lis-

tákon szereplő tételeket a lehető leggyorsabban szedik ki, ezzel csökkentve a megrendelés átfutási idejét. A kiszedési idő minimalizálása a következő tényezőkön múlik: utazási idő (50%), keresés (20%), áru áthelyezés (15%), adminisztráció (10%), egyéb (5%). Mivel a legtöbb időt az utazás veszi igénybe, a kiszedést ennek minimalizálásával tudjuk elsődlegesen csökkenteni.

További gyakori célfüggvények:

- Összes költség minimalizálása
- Megrendelések átfutási idejének minimalizálása
- A teljes átfutási idő minimalizálása (nem csak egy megrendelés)
- A helykihasználtság maximalizálása
- A berendezések kihasználtságának maximalizálása
- A munkaerő kihasználtságának maximalizálása
- Az áruk megközelíthetőségének maximalizálása

Ezek eléréséhez az alábbi kérdésekben kell dönteni a vállalat különböző szintjein:

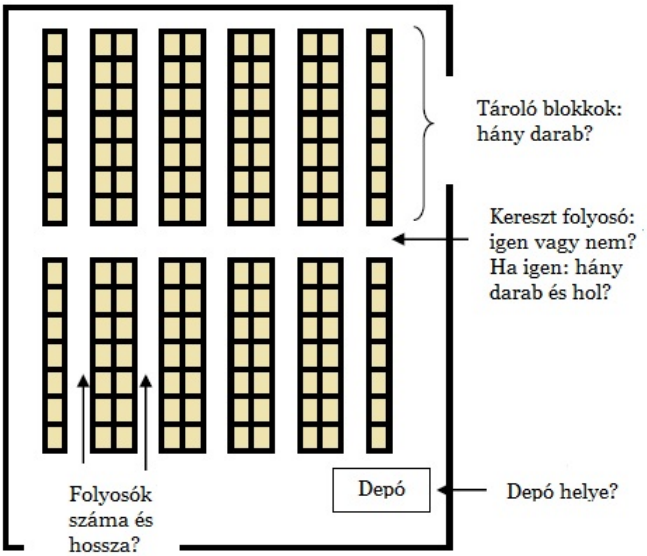
- A raktár elrendezése, és a tárolók dimenzionálása (taktikai szint)
- A termékek lokációkhoz való rendelése (taktikai és üzemeltetési szint)
- A megrendelések csoportosítása és a raktár zónákra osztása (taktikai és üzemeltetési szint)
- Útvonalak kialakítása (üzemeltetési szint)

Ezek a döntési tényezők szorosan összefüggnek, azonban egy modellen belül kezelhetetlenek. Ezért a tanulmányok is egyszerre csak néhányat vesznek belőlük figyelembe.

2.2. A raktár elrendezésének megtervezése

Amikor a raktár elrendezését tervezzük, akkor először azt kell eldöntenünk, hogy a különböző funkciókat (beérkező áruk fogadása, raktározása, csomagolása, kiszállítása) ellátó részlegeket egymáshoz képest hogyan helyezzük el, hogy a bejárásuk optimális legyen. Ezután pedig a tároló részleg kialakításáról kell döntenünk, azaz arról, hogy a tárolókat hogyan rendezzük sorokba és oszlopokba. A leggyakrabban az alábbi ábrán látható módon

szokták a tároló rendszereket kialakítani.



2.3. Lokációk meghatározása

A feladat az, hogy a termékeket betároljuk a raktárba, azaz minden termékhez rendeljük hozzá egy tároló lokációját. Ehhez azonban előbb el kell döntenünk, hogy milyen tárolási és kiszedési rendszert alkalmazunk.

Gyorsan forgó/ tartalék felosztás

Sok esetben csökkentheti a kiszedési időt, ha a gyakran és kis mennyiségben kért termékeket egy kisebb, úgynevezett gyorsan forgó területen is tároljuk. Míg az olyan termékeket, amelyekre nagy mennyiségben vagy ritkán van igény, azokat csak a nagyobb, úgynevezett tartalék területen. Ennek működtetéséhez azonban szükség van a gyorsan forgó terület folyamatos feltöltésre a tartalék területről, ami viszont csökkentheti a hatékonyságot.

Lokációk meghatározásának irányelvei

Számos módon lehet a lokációkat kiosztani a gyorsan forgó és tartalék területeken belül, így ezek közül csak párat említünk, melyeket a leggyakrabban szoktak alkalmazni. Az egyik lehetséges módszer a *random raktározás*, ahol minden egységnyi mennyiségű beérkező termék egyenlő valószínűséggel kap egyet a lehetséges lokációk közül. Ezzel a módszerrel a távolságok rovására növelhetjük a helykihasználtságot. Egy másik módszer

a *legközelebbi nyitott lokáció*, ahol a beérkező árut a legközelebbi üres lokációra tárolják be. Ha ezzel a módszerrel töltjük fel a raktárat, akkor a depó közelében az állványok tele lesznek, majd fokozatosan üresednek a raktár hátulja felé. Hausman 1979-ben megmutatta, hogy az említett két módszer hatékonysága megegyezik, hogyha minden alkalommal egységnyi mennyiségű árut mozgatunk.

Egy harmadik kiosztási lehetőség, ha minden terméknek fix helye van. Ez az úgynevezett *dedikált raktározás*. Ennek az a hátránya, hogy esetleg olyan terméknek foglaljuk a helyet, amiből jelenleg nincs a raktáron. Továbbá minden terméknek elegendő helyet kell biztosítani, így a maximális készletszintek nagyon kötöttek. Ezekből adódóan ennél a változatnál a legrosszabb a raktár helykihasználtsága. A dedikált raktározásnak az lehet az előnye, hogy a dolgozóknak nem kell állandóan keresniük a termékeket, mert azok mindig ugyanazon a helyen vannak. Ezért a kiskereskedelmi raktárakban gyakran ezt a módszert alkalmazzák. Nagyobb raktárakban ezt csak a gyorsan forgó területen érdemes alkalmazni, amit például egy véletlen módon kiosztott tartalék területtel lehet kombinálni. Így megmarad a dedikált raktározás előnye, de a hátránya csak egy kis területen fejti ki hatását.

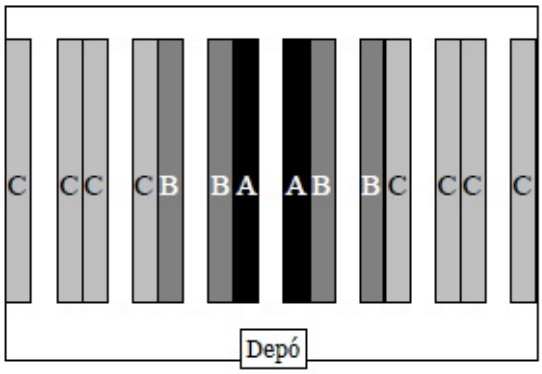
A negyedik módszer az úgynevezett *teljes forgalmon alapuló raktározás*. Ekkor úgy osztják ki a helyeket a raktárban, hogy a nagyobb forgalmú termék kerüljön a depóhoz közelebb, a kisebb forgalmú pedig távolabb. A termékek forgalmát szokás az úgynevezett COI indexükkel jellemezni (cube-per-order index). Ez úgy kapjuk, ha a termék teljes térfogatát elosztjuk azon utak számával, amennyit meg kell tenni az igények kielégítéséhez egy adott időszakban. Ez alapján azok a termékek kerülnek közelebb a depóhoz, melyeknek alacsonyabb a COI indexük. A gyakorlatban célszerű ezt a kiosztást a dedikált raktározással kombinálni. A módszer hátránya, hogy a kereslet állandóan változik, így ha mindig jó kiosztást szeretnénk a raktárban, akkor annak minden változásánál újra kéne rendezni a raktárat.

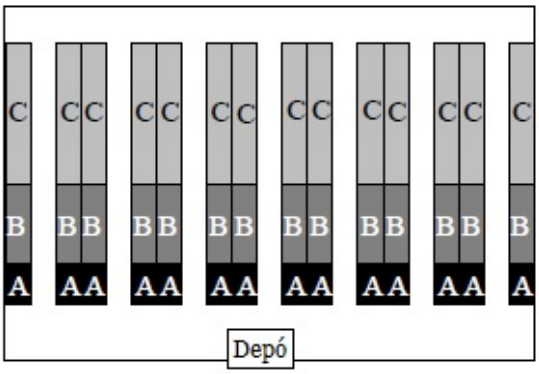
Osztályozás alapú raktározás

Az *osztályozás alapú raktározás* az eddig említett módszereket kombinálja. Az osztályokat a termékek forgalma alapján határozzuk meg úgy, hogy a legnagyobb forgalmú osztály a termékek legfeljebb 15%-át tartalmazza, de lefedje a teljes forgalom legalább 85%-át. Ezután minden osztály kap egy dedikált területet a raktárban, amin belül az osztály termékei véletlenszerűen kapnak lokációt. Általában három osztályba sorolják a termékeket: A, B és C.

Petersen 2004-ben szimulációkkal megmutatta, hogy a távolságok szempontjából a teljes forgalmon alapuló raktározás jobb eredményt ad, mint az osztályozás alapú raktározást. A kettő közötti különbség mértéke függ az osztályok számától, az osztályok által meghatározott százalékos eloszlástól és az útvonal tervezésétől. A gyakorlatban azonban az osztályozás alapú raktározás implementálása sokkal egyszerűbb és nem ad sokkal rosszabb eredményt a teljes forgalmon alapuló raktározásnál, így mégis inkább ezt érdemes alkalmazni.

Az osztályokat többféleképpen is elhelyezhetjük a raktárban. Hogy ezek közül melyik az optimális, az attól függ, hogy a dolgozók milyen útvonalon járják be a raktárat.





Korrelált csoportosítás

Az eddig felsorolt módszereknél nem vettük figyelembe, hogy bizonyos termékeket a megrendelők sokszor rendelnek egyszerre, így ezeket érdemes lenne egymás mellett tárolni. Természetesen a korreláció alapú csoportosítást lehet egyszerre alkalmazni az előbb említett módszerekkel, mint például az osztályozáson alapuló raktározással. Ugyanakkor az, hogy melyik osztályba rakjuk a csoportosított termékeket, függ az osztály többi elemének tulajdonságától is. Ahhoz, hogy ezt a módszert alkalmazni tudjuk, ismernünk kell a korrelációt a termékek között, vagy legalábbis meg kell tudnunk becsülni.

Az egyik legegyszerűbb korrelált csoportosítási módszer a *komplementer alapú módszer*, melynek két fázisa van. Az első fázisban a termékeket csoportokba rendezi a közös igény mértéke alapján, a második fázisban pedig a kialakított csoportokat rendeli lokációkhoz úgy, hogy a csoport tagjai olyan közel legyenek egymáshoz, amennyire csak lehetséges.

2.4. Zónák kialakítása

Ahogy korábban említettük, ha a raktárat zónákra bontjuk, azzal rövidíthetjük a kiszedési időt, továbbá elkerülhetők a "közlekezdési dugók" a raktárban. A zónázás legnagyobb hátránya, hogy a megrendeléseket szét kell bontani, majd újra összerakni a kiszállítás előtt. Ezt kétféle megközelítéssel is megpróbálhatjuk orvosolni, az egyik a *progresszív összeszerelés*. Ennek során a megrendelést úgy szedik össze a dolgozók, hogy miután az első érintett zónából minden termék a kiszedő kocsira került, a kocsit a rendelési listával együtt a dolgozó odaadja a következő érintett zónában dolgozónak, és így tovább. Így a megrendelés akkor lesz teljesen kiszedve, ha a kiszedő kocsi végigjárta az összes érintett zónát. A másik megközelítés a *paralel/szinkronizált kiszedés*, ahol minden érintett zónában egyszerre kezdik el kiszedni a megrendelés zónába eső részeit, és amint mindenki végzett a saját zónájában, rögtön összeteszik a megrendelés részeit. A gyakorlatban a zónákat a termékek bizonyos tulajdonságai alapján szokták kialakítani, például a tárolandó hőmérsékletük, súlyuk, méretük vagy biztonsági követelményük alapján.

2.5. Megrendelések csoportosítása

A szintén korábban említett, kiszedési időt csökkentő módszer a megrendelések csoportosítása. Vagyis ha a megrendelések csak kevés tételből állnak, akkor érdemes lehet egy túra során több megrendelést is kiszedni. Choe and Sharp (1991) szerint a jó csoportosításnak két alapvető feltétele van: a csoportosított megrendelések nagyjából azonos helyen lévő tételeket tartalmazzanak és nagyjából azonos időben érkezzenek be.

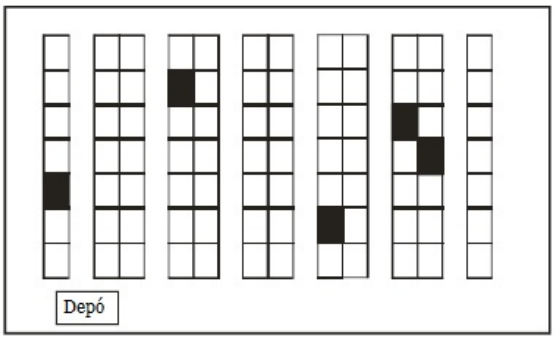
Gademann & Van de Velde (2005) megmutatta, hogy a csoportosítás kézi kiszedés mellett $\mathcal{NP}$-nehéz, ha minden csoportban legalább két megrendelés van és az összes megtett utat szeretnénk minimalizálni.

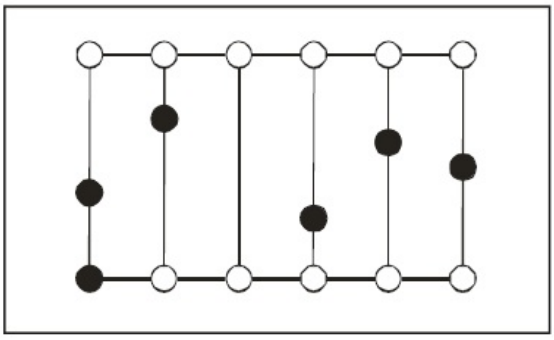
A megrendelések csoportosítását a következő fejezetben tárgyaljuk részletesen.

Az összes eddig említett módszer hiányossága, hogy nem veszi figyelembe a megrendelések határidejeit, és nem bünteti azok be nem tartását.

2.6. Útvonal tervezési módszerek

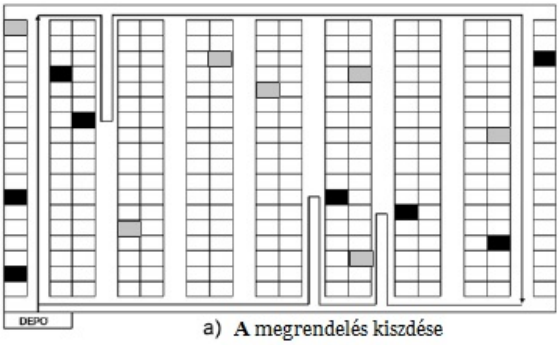
A kiszedés útvonalát a megrendelési listán szereplő tételek sorba rendezésével kapjuk. A raktár alaprajzát könnyen reprezentálhatjuk egy gráffal. Ez látható az alábbi ábrán.



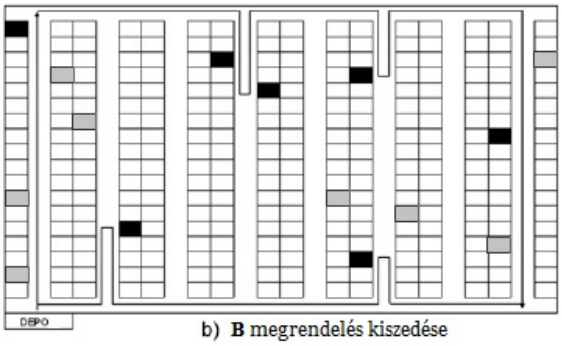


Ebben a gráfban a fekete csúcsokat kell mindenképpen meglátogatnunk, a fehér csúcsokra nincs megkötés (meglátogathatjuk őket többször, de akár ki is hagyhatjuk a túrából), továbbá a depót reprezentáló csúcsot is meglátogathatjuk többször is, mint 2. (Feltettük, hogy a kiszedés a depóból indul és ott is ér véget, ezért kell legalább kétszer meglátogatnunk.) Ezt a feladatot Steiner utazó ügynök feladatnak szokás hívni. Ennek megoldása általában nem polinomiális idejű, de Ratliff and Rosenthal (1983) megmutatta, hogy az olyan típusú gráfokon, mint amit az előbb mutattunk, van olyan algoritmus, amely lineáris a sorok és felvevési pontok számában.

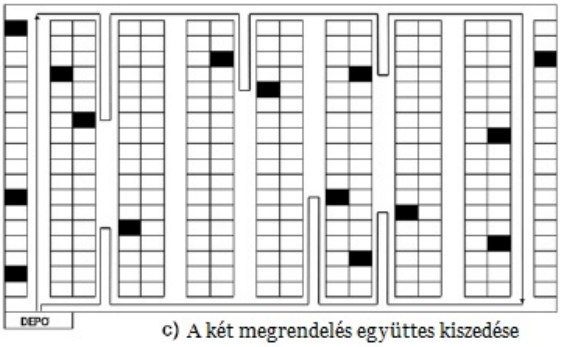
A következő egyszerű példán is láthatjuk, hogy jó csoportosítással csökkenthetünk az összesen megtett távolságot.



a) **A** megrendelés kiszdése

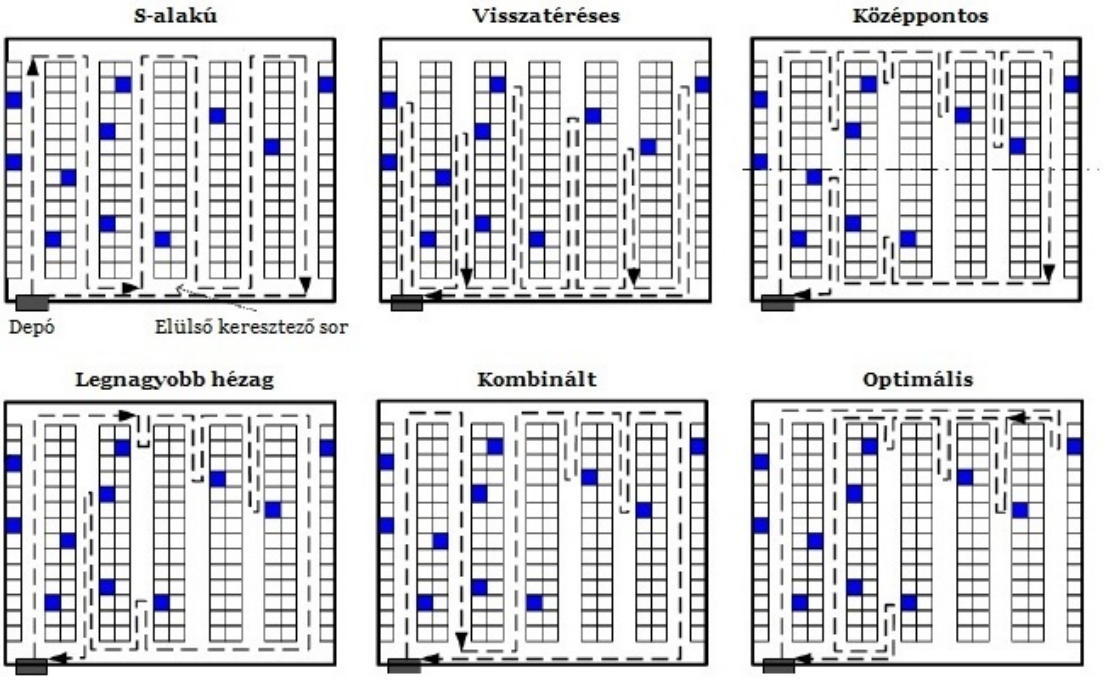


b) **B** megrendelés kiszedése



c) A két megrendelés együttes kiszedése

Útvonal tervezési heurisztikák



Különböző heurisztikus módszerek megoldásai ugyanazon a feladaton

Az egyik legegyszerűbb heurisztika az úgynevezett *S-alakú heurisztika*, amely pontosan azokon a sorokon megy teljesen végig, amelyekből ki kell szedni árut. Ha az utolsó ilyen soron is végigment, akkor annak végéből visszamegy a depóba. A *visszatéréses módszerrel*, a meglátogatandó sorokon a dolgozó nem megy végig, csak a kiszedendő tételig, majd onnan visszatér a sornak ugyanahhoz a végéhez, ahonnan jött. A *középpontos módszer* lényegében két részre osztja a raktárat, az elülső részen elhelyezkedő tételeket a sorok elejéről indulva, a hátsó részen lévő tételeket a sorok hátsó végéből indulva gyűjti be a dolgozó. Így csak az első és utolsó soron kell teljes egészében végig menni. Hall (1993) szerint ez a módszer jobb, mint az S-alakú, ha a tételeket a sorok számával osztva kis számot kapunk.

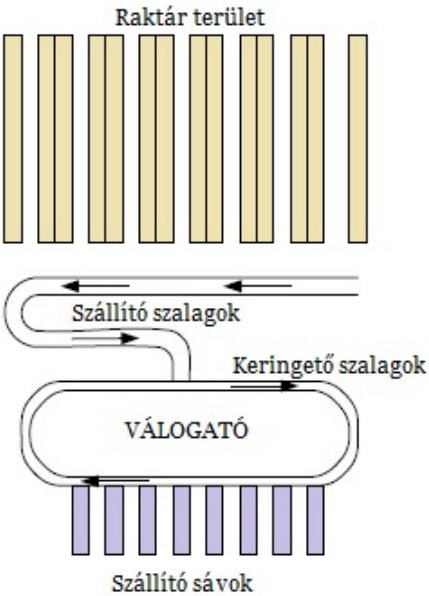
A *legnagyobb hézag módszerét* alkalmazva kiszámoljuk a legnagyobb távolságot a sorok széle és a sorban a kiszedendő termékek között, és a sornak arról a végéről megyünk be, amelyik végétől vett távolság nagyobb. Ha a termékek közti távolság nagyobb, mint a sor két végétől vett távolság összesen, akkor mindkét végéről bemegyünk a sorba. Tehát a legnagyobb hézag az a távolság, amit nem kell megtenni egy sorban. Ez a módszer mindig jobb megoldást ad a középpontos módszernél, bár azt egyszerűbb implementálni. A

kombinált heurisztikus módszernél egy soron vagy teljesen végig kell menni, vagy vissza kell térni abba a végébe, ahonnan bementünk.

Petersen (1997) számos numerikus kísérletet végzett az előbb felsorolt heurisztikákon, és arra a következtetésre jutott, hogy a legjobb heurisztikus megoldás átlagosan 5%-kal tér el az optimálistól.

2.7. Felhalmozás és válogatás

Ha csoportosítottuk a megrendeléseket, vagy zónákra osztottuk a raktárat, akkor a kiszedés után még ki kell válogatnunk az egy megrendeléshez tartozó tételeket. Ezt a műveletet sokszor úgy hívják, hogy felhalmozás és válogatás, aminek egy tipikus megvalósítását láthatjuk a következő képen.



A kiszedett tételeket a dolgozók a szállítószalagra teszik, így random sorrendben jutnak el a keringető részbe. Innen a szállító sávokra rendezik a megrendeléseket, akár többet is, ha azokat ugyanarra a teherautóra kerülnek. Ha egy termék korábban kerül a keringető szalagra, mint hogy a szállító sávra lehetne tenni, mert még nem érkezett meg a sávon lévő megrendelésnek minden tétele, akkor addig kering körbe, amíg már rá lehet tenni a szállító sávra. A rendszer áteresztőképessége nem csak a válogatás és a keringető szalag sebességétől függ, hanem az olyan működési szabályzatoktól is, mint a megrendelések szállítósávokhoz való rendelése.

3. fejezet

Megrendelések csoportosításának megoldási módszerei

Ebben a fejezetben a kiszedés gyorsítását egy másik oldalról közelítjük meg, azaz ahelyett, hogy a termékek optimális elrendezését vizsgálnánk, a megrendelések optimális csoportosításával foglalkozunk.

Az előző fejezethez hasonlóan, továbbra is kézi kiszedésről lesz szó, tehát feltesszük, hogy a kiszedés során a dolgozók túrákat tesznek a raktárban, melyek során egy kiszedő kocsira gyűjtik a pick listán szereplő tételeket. Egy pick listán nem szerepelhet több tétel, mint amekkora a kiszedő kocsi kapacitása. A továbbiakban néhol ezt a pick lista kapacitásának fogjuk nevezni. Emellett ebben a fejezetben feltesszük, hogy a megrendeléseket nem lehet több részre bontani, azaz minden megrendelés pontosan egy pick listán szerepel. Ezzel a módszerrel a kiszedés utáni válogatási időt meg lehet spórolni.

Tehát a *megrendelések (statikus) csoportosításakor* az a feladat, hogy adott kiszedési útvonal, kapacitás és ismert lokációk mellett úgy bontsuk pick listákra a megrendeléseket, hogy a kiszedéskor összesen megtett út minimális legyen.

3.1. A feladat felírása egészértékű feladatként

A következő egészértékű program minden megengedett megoldása a megrendelések egy csoportosításának felel meg. A modellt Gademann és Van de Velde publikálta 2005-ben. A feladat felírásához vezessük be a következő jelöléseket:

- J: a megrendelések halmaza, $J = \{1, \ldots, n\}$;

- C: a kiszedő kocsi kapacitása;
- c_j: a j. megrendelés által igényelt kapacitás, $j \in J$;
- I: a megengedett csoportosítások halmaza;
- d_i: az i. csoport által meghatározott túra hossza, $i \in I$;
- $a_i = (a_{i1}, \ldots, a_{in})$: bináris változó - a j. megrendelés benne van-e az i. csoportban;
- x_i: bináris változó - az i. csoportot kiválasztjuk-e.

Ekkor a feladat a következőképp írható fel.

$$\min \sum_{i \in I} d_i x_i \tag{3.1}$$

$$\sum_{j \in J} c_j a_{ij} \leq C \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i \in I} a_{ij} x_i = 1 \qquad \forall j \in J \tag{3.3}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{3.4}$$

Ahol a (3.2) korlát biztosítja, hogy egy pick lista se lépje túl a kiszedő kocsi kapacitását, (3.3) és (3.4) pedig azt fogalmazza meg, hogy egy megrendelés csak egy pick listára kerüljön.

Az egészértékű feladat megoldása

Az előbbi egészértékű feladatban az x döntési változó a megrendelések számában exponenciálisan változik, így ezzel a módszerrel csak kis méretű problémákat lehet megoldani. Nagyméretű példákra az LP/IP megoldó csak megengedett megoldást tud adni, aminek az optimalitását nem tudja megmutatni, mivel kifut a memóriából.

Ezért Gademann és Van de Velde (2005) oszlopgenerálással oldotta meg a feladatot. A kezdeti bázisban lévő csoportokat **iterated descent** algoritmussal határozzák meg, amire megoldják a feladat relaxáltját. Ezután minden iterációban árazó algoritmussal hozzávesznek csoportokat az eddigiekhez, melyeknek redukált költsége minimális, és azokkal együtt oldják meg újra a relaxált feladatot. Ha a megoldás nem egész, akkor branch-and-price algoritmussal generálják le a feladat egész megoldását.

Gademann és Van de Velde (2005) széleskörű numerikus vizsgálatokat végeztek 400 lokációval, 30 megrendeléssel úgy, hogy legfeljebb 10 megrendelés kerülhetett egy pick listára. Az így generált feladatok mindegyike pár percen belül lefutott, azonban a megrendelések számának növelésével a futási idő radikálisan nőtt. Ebből adódóan valós feladatokra még oszlop generálással sem alkalmazható jól ez a megközelítés, így a továbbiakban heurisztikus és metaheurisztikus módszereket fogunk vizsgálni.

3.2. Konstruktív megoldási módszerek

Megoldás prioritási listával

A módszer első lépésként a megrendelésekhez prioritási értéket rendel, ami alapján sorba rendezi őket. Ezt a sorrendet hívjuk prioritási listának. Az eljárás ezután a prioritási lista által meghatározott sorrendben alakítja ki a pick listákat.

A megrendelések prioritásának meghatározására számos módszer létezik. Az egyik legegyszerűbb, ha a megrendelés beérkezésének időpontja szerint állítjuk sorrendbe a megrendeléseket, tehát azt soroljuk előrébb, amelyik hamarabb érkezett. Ez az úgynevezett *FCFS szabály (First Come First Served)*. Az ezzel a szabállyal kapott sorrendben a megrendelések tekinthetők véletlenszerűnek, így ettől nem várhatunk igazán jó csoportosítást, de viszonyítási alapnak használható az így kapott eredmény. Egy másik módszer aszerint rendezi sorba a megrendeléseket, hogy a kiszedendő tételek sorai között mekkora a legnagyobb távolság. (Ruben & Jacobs 1999)

A megrendeléseket a pick listákhoz rendelhetjük egyszerre vagy egymás után is. A *Next-Fit szabály* szerint egymás után rendeljük a megrendeléseket a pick listához, és csak akkor nyitunk új pick listát, ha a következő megrendelés hozzávételével már megsértenénk a kapacitási korlátot. A *First-Fit szabály* szimultán kezeli a pick listákat, és a következő megrendelést arra a nyitott pick listára teszi, amelyiken a legkevesebb megrendelés van és a kapacitás korlát az új megrendeléssel együtt nem sérül (Ruben & Jacobs 1999). A *Best-Fit szabály* is szimultán kezeli a pick listákat, de a következő megrendelést arra a pick listára teszi, ahol a fennmaradó kapacitás a legkisebb, de az új megrendeléssel együtt sem lépi át a maximumot.

A prioritási listát használó algoritmus pszeudokódja a következő.

```
Minden megrendelés kap egy prioritási értéket;
Rendezzük a megrendeléseket a prioritási értékük alapján nem-növekvő sorrendbe;
Nyissuk meg az első pick listát, és rendeljük hozzá az első megrendelést;
    While Van szabad megrendelés do
        Válasszuk ki a következő megrendelést;
    If A megrendelés hozzárendelhető nyitott pick listához then
        Válasszunk ki egy nyitott pick listát;
        Ehhez rendeljük hozzá a megrendelést;
    Else
        Nyissunk meg egy új pick listát;
        Ehhez rendeljük hozzá a megrendelést;
    Endif;
Endwhile;
```

Megoldás mag alapú algoritmussal

Ez a megoldási módszer Elsayed (1981) nevéhez fűződik. Az algoritmus két fázisban csoportosítja a megrendeléseket. Először az újonnan megnyitott pick listának kiválaszt egy úgynevezett mag megrendelést. Majd a további megrendeléseket úgy válogatja hozzá a mag megrendeléshez, hogy a megrendelések "távolsága" minimális legyen.

Az alábbi módszerek arra vonatkoznak, hogy hogyan választhatjuk ki a kezdeti mag megrendelést.

- Random választással (Gibson & Sharp 1992).
- Legkevesebb vagy legtöbb tételt tartalmazó megrendelést választva (Elsayed & Stern 1983).
- Legkevesebb vagy legtöbb különböző lokáción lévő tételt tartalmazó megrendelést választva (Elsayed & Stern 1983).
- Legkevesebb vagy legtöbb sorban lévő tételt tartalmazó megrendelést választva (Ho & Tseng 2006).
- Legkisebb vagy legnagyobb lokáció-sor aránnyal rendelkező megrendelést választva (Ho & Tseng 2006).
- A sorok súlyának összegét minimalizáló vagy maximalizáló megrendelést választva, ahol egy sor súlya az a távolság, amennyivel nő a túra hossza, ha meg kell látogatni az adott sort (Ho 2008).

- Minimális vagy maximális téglalappal lefedhető megrendelést választva (Ho 2008).
- Azt a megrendelést választva, melynek tételei és a depó közötti átlagos euklideszi távolság minimális (Ho 2008).
- Leggyorsabban vagy leglassabban kiszedhető megrendelést választva (De Koster 1999).
- Legnagyobb sorkülönbséggel rendelkező megrendelést választva (De Koster 1999).

A következő módszerek/szabályok adnak megoldást arra, hogy hogyan válasszuk ki a további megrendeléseket a mag megrendelés mellé.

- A hozzáadott megrendelés a lehető legkevesebb további meglátogatandó sort adja hozzá a túrához (Rosenwein 1996).
- Az új megrendelést lefedő téglalap és az eddigi megrendeléseket lefedő téglalap közös területe legyen minimális (maximális) (Ho 2008).
- A hozzávett megrendeléssel minimálisan (maximálisan) nőjön az összes tételt lefedő téglalap mérete (Ho 2008).
- A következő megrendelés hozzávételével a megrendelés tételeinek átlagos távolsága a pick lista tételeitől, plusz a pick lista tételeinek átlagos távolsága a megrendelés tételeitől legyen minimális; ahol egy tétel távolsága tételek egy halmazától az a távolság, amennyit a tételtől meg kell tenni a halmaz legközelebbi eleméig (Ho 2008).
- A megrendelésen szereplő tételek távolságösszege a pick listától legyen minimális (Pan & Liu 1995).
- A hozzávett megrendelés súlypontja a legközelebb legyen az eddigi pick lista súlypontjához (De Koster 1999).
- Az új megrendelésnek a legtöbb közös sora legyen az eddigi pick listával (Ho & Tseng 2006).
- A következő megrendeléssel közös sorok száma osztva az összes sor számával (eddigi plusz új megrendelés) legyen maximális (Ho & Tseng 2006).
- Az új megrendeléssel közös sorok száma osztva az új megrendelés sorainak számával legyen maximális (Ho & Tseng 2006).

A mag megrendelést használó algoritmus pszeudokódja a következő.

```
1. While Van szabad megrendelés do
2.     Nyissunk meg egy új pick listát;
3.     Válasszunk egy megrendelést a pick lista magjának;
4.     While Van olyan megrendelés, melyet hozzá lehet venni a pick listához do
5.         Válasszunk ki egy megrendelést a hozzávehetők közül;
6.         Vegyük hozzá a pick listához;
7.     Endwhile;
8.     Zárjuk le a pick listát;
9. Endwhile;
```

Megoldás megtakarításos algoritmussal

A megtakarításos algoritmus az utazó ügynök feladatra adott megoldáson alapszik, melyet Clarke és Wright adott 1964-ben. A megrendelések csoportosítására alkalmazott első változat a következő képen működik. Az algoritmus először kiszámolja minden megrendelés párra, hogy mennyi utat lehet megtakarítani hogyha a két megrendelést együtt szedjük ki, azzal szemben mintha külön-külön tennénk ugyanezt. Ezután a megtakarítás alapján nem növekvő sorrendbe rendezi a párokat, és ezek közül először nézi az elsőt. Ha a pár egyik tagja sincs még pick listához rendelve, akkor nyit nekik egy új pick listát, és ahhoz rendeli őket. Ha az egyik megrendelés már hozzá van rendelve egy pick listához, akkor ha a kapacitás korlát megengedi, a másik megrendelést is hozzá veszi a pick listához, különben nézi a következő megrendelés párt. Abban az esetben, ha mindkét megrendelés hozzá van már rendelve pick listához, szintén veszi a következő párt. Ha már nincs több pár, de van olyan megrendelés, amelyet még nem rendelt pick listához, akkor az a megrendelés egyedül alkot egy pick listát. (Elsayed & Unal 1989). Ennek az a hátránya, hogy adott esetben túl sok pick listát generál.

Az algoritmus második változata annyiban tér el az elsőtől, hogy minden olyan esetben, amikor egy megrendelést hozzárendel egy pick listához, ismételten kiszámítja a megtakarításokat úgy, hogy a már kialakított pick listákon szereplő tételeket egy megrendelésnek tekinti. (Elsayed & Unal 1989). Ezt tovább lehet javítani, ha az abszolút megtakarítások helyett úgynevezett normált megtakarítások szerint rendezzük sorba a párokat. A normált megtakarítást úgy kapjuk, ha az együttes kiszedéssel megtakarított időt elosztjuk azzal az összidővel, amennyi akkor szükséges, ha a két megrendelést külön szedjük ki (Bozer és

Kile 2008).

A harmadik változat felhasználja a mag alapú algoritmust is. Ez az algoritmus a legnagyobb megtakarítással rendelkező párnak nyit egy új pick listát, és ezt a párt egy megrendelésként kezelve a pick lista magjának tekinti. Ezután ehhez a listához egyesével veszi hozzá a megrendeléseket úgy, hogy a hozzávett megrendeléssel a megtakarítás maximális legyen, de a kapacitás korlátot ne lépje át. (Elsayed & Unal 1989)

3.3. Megoldás metaheurisztikus módszerekkel

Lokális keresés

A lokális keresés alapötlete, hogy a megoldások között definiálunk egy szomszédsági relációt valamilyen transzformáció segítségével, majd úgy vesszük a megoldások sorozatát, hogy abban minden rákövetkező tag az előző szomszédja, de a megoldás célfüggvény értéke jobb, mint az előzőé. Ha valamelyik megoldásnak nincsen nála jobb célfüggvénnyel rendelkező szomszédja, akkor egy lokális optimumot kaptunk. A módszer hátránya, hogy előfordulhat olyan eset, amikor a lokális optimum, amit megtaláltunk, messze van a globális optimumtól. Ennek elkerülése érdekében a lokális optimum megtalálása után szokás keresni egy újabb megoldást, aminek a célfüggvénye rosszabb, és ebből keresni újabb lokális optimumot.

Az első lokális keresésen alapuló módszert a megrendelések csoportosítására Gademann és Van de Velde dolgozta ki 2005-ben. Az általuk adott algoritmus kezdeti megoldását az FCFS szabály adja. A megoldás egy szomszédját úgynevezett *cserével* kapjuk, azaz két pick lista egy-egy megrendelését kicseréljük a másikra. Mivel Gademann és Van de Velde a pick lista kapacitásának a hozzárendelt megrendelések számát tekinti, így a cserével a megoldás megengedettsége nem romlik el. Ez alapján az algoritmus a megoldás szomszédságából mindig az első olyan megoldást választja, amelyiknek jobb (kisebb) a célfüggvényértéke. Ha az algoritmus lokális optimumot talált, akkor annak érdekében, hogy ne akadjon el egy rossz lokális optimumban, három megrendelést cserél föl véletlenszerűen három pick lista között. A hármas felcserélések számának maximuma az algoritmus bementi paraméterei között meg van adva.

Henn 2010-ben egy úgynevezett *iterált lokális keresési* algoritmust javasolt, amely két fázisból áll, egy perturbációs és egy keresési fázisból. A perturbációs fázisban az aktuális

megoldásból megrendelések cseréjével kap egy új induló megoldást. Ebből indítja el a keresési fázist, azaz egy lokális keresést. A kereséssel kapott legjobb megoldásra csak akkor cseréli le az induló megoldást, ha bizonyos elfogadási kritériumoknak megfelel. Ellenkező esetben az eddigi induló megoldásra alkalmaz ismét perturbációt. Ez a két fázis addig folytatódik, amíg az előre meghatározott leállási kritériumok nem teljesülnek.

Henn kezdeti megoldásnak az FCFS szabállyal kapott megoldást veszi, a lokális keresés során pedig az előbb definiált cseréket és úgynevezett váltásokat. A váltás egy olyan lépés, melynek során egy megrendelést egy másik pick listához rendel hozzá. A keresésnél először csak cserékkel keresünk jobb megoldást, majd ha ezzel a lépéssel már nem tudunk javítani, akkor váltásokkal próbálunk az eddiginél jobb megoldást találni. Ha a váltásokkal elakadtunk, akkor megint cserékkel keresünk jobb megoldást, és így tovább, amíg végleg el nem akadunk. A perturbációra Henn a következőt javasolja: válasszunk ki két megrendelést véletlenszerűen, és azok között véletlen számú megrendelést cseréljünk ki. Ha bármelyik lépés során olyan megrendeléseket adnánk valamelyik listához, amelyek sértik a kapacitás korlátokat, akkor a sértő megrendeléseknek nyissunk új pick listát. Egy megoldás csak akkor lehet új induló megoldás, ha annak célfüggvénye kisebb, mint az eddig ismert legjobb megoldásé.

Az Albareda-Sambola és csapata (2009) által megadott *változó lokális keresés* a következő három lépés alapján három osztályba sorolja egy megoldás szomszédságát: (*i*) egy megoldást át lehet rendelni egy másikhoz (váltás) (*ii*) legfeljebb két megoldást egy pick listáról át lehet rendelni egy vagy két másik listára (*iii*) legfeljebb két listáról legfeljebb két megrendelést másik listához lehet rendelni. A szomszédságba nem vesszük bele a nem megengedett, azaz a kapacitást sértő szomszédokat. Figyeljük meg, hogy az ezekkel a lépésekkel kapott szomszédságok tartalmazzák egymást, és a harmadik a legbővebb. Így az algoritmus egy megoldásból először az első lépéssel keres jobb megoldást, ha azzal már nem talál, akkor a másodikkal, végül a harmadikkal. Az így kapott legjobb megoldást tekinti az előző szomszédjának, és erre keresi a következő megoldást. Az algoritmus akkor áll le, ha egyik lépéssel sem kapunk jobb megoldást az aktuálisnál.

Tabu keresés

A tabu keresés alapötlete Glovertől (1986) származik, és a lokális keresés egy speciális változata. Ez a módszer a keresés során különböző lépéseket alkalmaz, de a nemrég használt lépések felkerülnek az úgynevezett tabu listára, és bizonyos számú iterációig nem

használhatók újra. Az új megoldást a jelenlegiből a nem tiltott lépésekkel kapható legjobb megoldásnak választva kapjuk, akkor is ha esetleg annak célfüggvény értéke nem kisebb, mint a jelenleginek.

Henn és Wächser a tabu keresés számos variációját dolgozta ki (2010). A legegyszerűbb változat a kezdeti megoldást FCFS szabállyal vagy a megtakarításos algoritmus második változatával generálja. A megoldás szomszédait (1) cserével (2) váltással vagy (3) cserével és váltással keresi meg.

Szintén Henn és Wächser (2010) dolgozta ki a következő heurisztikát. Először meghatározzuk az aktuális probléma megoldásainak speciális jellemzőit, majd a lokális keresés során egy újabb megoldástól azt várjuk el, hogy legalább egy jellemzőben legyen jobb, mint az aktuális megoldás. Az algoritmus akkor áll le, ha nem talál olyan szomszédos megoldást, amelyik jobb lenne az eddig talált legjobbnál valamelyik jellemzőjében. Például a megrendelések csoportosításakor egy megoldását jellemezhetünk azzal, hogy mely megrendelés párokat rendeltük ugyanahhoz a pick listához, illetve azzal is, ha egyesével figyeljük, hogy melyik megrendelés melyik pick listára került.

Populáció alapú megközelítések

A *rang alapú hangya rendszer* egy olyan populáció alapú megközelítés, ahol minden hangya a probléma egy megoldását reprezentálja (Bullnheimer 1999). Henn (2010) alkalmazta először ezt a megközelítést a megoldások csoportosítására. Az általa javasolt eljárásban adott számú hangyát figyelünk meg számos iteráción keresztül. Minden hangya esetén az algoritmus abból a megoldásból indul, amelyben minden megrendelés külön pick listát alkot. Egy iterációban kiszámoljuk a megtakarítást két pick lista csoportosításával, illetve a pick listák közötti "feromonok intenzitását", azaz a két pick lista relatív hatását egymásra. A feromonok intenzitása az olyan megrendeléspárok közötti feromonok átlagos száma, melyeknél az egyik megrendelés az egyik listán szerepel, a másik a másikon. A megtakarítás és a feromonok intenzitása adja meg annak a valószínűségét, hogy két pick lista megrendeléseit összevonjuk-e egy pick listává. Ha elfogytak a megengedett összerendelések, akkor lokális kereséssel még megpróbálunk javítani a megoldáson. Ezt az eljárást ismételjük meg minden hangyára. Amikor egy iterációban az utolsó hangya eljárása is befejeződött, akkor a megoldások feromonjainak számát kicsit módosítjuk: a kevésbé jó megoldásokét csökkentjük, míg a "jó" megoldásokét növeljük.

A *genetikus algoritmus* szintén populáció alapú megközelítés, amely nagyszámú lehetsé-

ges megoldást generál. Ezek közül a fittségük alapján választja ki a legjobb megoldásokat, melyeket keresztezve illetve mutálva kap egy újabb generációt. Ezt a megközelítést Hsu (2005) alkalmazta először a megrendelések csoportosítására. Minden megoldást egészek sorozatával jellemez, amely megmondja, hogy melyik megrendelés melyik pick listán szerepel. Egy megoldás fittségét úgy számolhatjuk ki, hogy vesszük a megoldás által adott túra hosszának és a populációban szereplő leghosszabb túra hosszának különbségét. Az írók számos keresztezési és mutációs módszert írnak a cikkükben, melyekre most nem térünk ki.

3.4. Kiszedés hullámokban

Ez a megközelítés a megrendelések csoportosításánál a kiszállítási folyamatot veszi alapul, így olyan megrendelések kerülnek egy hullámba, melyek megrendelési helye azonos. A gyakorlatban például egy ilyen hullámot alkothat az egy teherautóra kerülő megrendelések halmaza. Mivel általában egy ilyen hullámra időbeli megkötés vonatkozik, a kiszedés alatt megtett összes távolság nem feltétlenül optimális. Amíg az egyik hullámba tartozó összes megrendelés nincs kiszedve, addig nem lehet elkezdeni a következő hullámba tartozó megrendeléseket kiszedni. (Gademann 2001)

Ekkor a kiszedés és kiszállítás folyamata a következő optimalizálási problémaként fogalmazható meg. Hogy lehet adott raktár bejárási útvonalakkal, fix kapacitás mellett a megrendeléseket úgy pick listákra bontani, hogy a pick listák maximális kiszedési ideje minimális legyen, ha minden termék lokációja ismert. Általában föltesszük, hogy a dolgozók nem akadályozzák egymást, azaz nem kell azért megállniuk, mert egy másik kiszedő feltartja őket.

A megengedett megoldásokat leíró egészértékű programhoz használjuk a következő jelöléseket.

- J: a megrendelések halmaza, $J = \{1, \ldots, n\}$;
- I: a megengedett csoportosítások halmaza;
- $a_i = (a_{i1}, \ldots, a_{in})$: bináris változó - a j. megrendelés rajta van-e az i. pick listán;
- x_i: bináris változó - az i. pick listát kiválasztjuk-e;
- pt_i: az i. pick lista keszedési ideje, $i \in I$;

- m: a dolgozók száma, azaz ennyi pick listát lehet párhuzamosan kiszedni.

Ekkor a feladat a következőképp írható fel.

$$\min \max_{\mathrm{i} \in \mathrm{I}} \sum_{i \in I} pt_i x_i \tag{3.5}$$

$$\sum_{i \in I} a_{ij} x_i = 1 \qquad \forall j \in J \tag{3.6}$$

$$\sum_{i \in I} x_i \leq m \tag{3.7}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{3.8}$$

Ahol a (3.6) és (3.8) korlátok biztosítják, hogy egy megrendelés csak egy pick listára kerüljön, a (3.7) korlát pedig kizárja, hogy m-nél több pick listát válasszunk ki. Ennek a modellnek egy megoldásában előfordulhat olyan eset, hogy az egyik dolgozó nem kap pick listát, ami a valóságban nem biztos, hogy optimális megoldás emberi szempontból. (Bozer és Kile 2008) Továbbá ehhez az optimalizálási feladathoz kapcsolódó döntési feladat $\mathcal{NP}$-teljes. (Gademann 2001)

3.5. Pick listák ütemezése

Az eddigi modellekben a megrendelések határideje nem volt fontos szempont, a gyakorlatban viszont ez sokszor meghatározó. Tehát ebben a részben azt vizsgáljuk, hogy hogyan lehet egy adott megrendelésállományt csoportosítani adott bejárási útvonalak, fix költség, ismert lokációk, és határidők mellett úgy, hogy a megrendelések határidőtől való eltérése minimális legyen. Ez azt jelenti, hogy azt is büntetjük, hogyha korábban, és azt is, ha később készül el egy megrendelés, mint a határideje.

A következő modellt Elsayed és Lee állította fel (1993), és a megrendelések összes késését minimalizálja. Ehhez használjuk a következő jelöléseket.

- J: a megrendelések halmaza, $J = \{1, \ldots, n\}$;
- I: a megengedett csoportosítások halmaza;
- $a_i = (a_{i1}, \ldots, a_{in})$: bináris változó - a j. megrendelés rajta van-e az i. pick listán;
- pt_i: az i. pick lista keszedési ideje, $i \in I$;
- dd_j: a j. megrendelés határideje, $j \in J$;

- J_i: az i. pick listán szereplő megrendelések halmaza, $i \in I, J_i = \{j \in J | a_{ij} = 1\}$;
- M: elég nagy pozitív konstans;
- α: az összes korábban elkészülő megrendelés súlya;
- β: az összes késő megrendelés súlya;
- x_i: bináris változó - az i. pick listát kiválasztjuk-e;
- ct_i: változó - az i. pick lista kiszedésének befejezési ideje, tehát az az időpont, amikor a dolgozó visszatér a depóhoz a kiszedett pick listával;
- v_{ik}: bináris változó - az i. megrendelés a k. megrendelés után következik-e közvetlenül a sorban, $i, k \in I$;
- ta_{ij}: változó - a j. megrendelés késése az i. pick listán, $i \in I, j \in J$;
- ea_{ij}: változó - a j. megrendelés az i. pick listán mennyivel hamarabb készül el, mint a határideje, $i \in I, j \in J$.

Ekkor a következő MIP feladatot írhatjuk fel.

$$\min \alpha \sum_{i \in I} \sum_{j \in J} a_{ij} ea_{ij} + \beta \sum_{i \in I} \sum_{j \in J} a_{ij} ta_{ij} \tag{3.9}$$

$$\sum_{i \in I} a_{ij} x_i = 1 \qquad \forall j \in J \tag{3.10}$$

$$pt_i x_i \leq ct_i \qquad \forall i \in I \tag{3.11}$$

$$ct_i - ct_k + M v_{ik} \geq pt_i x_i \qquad \forall i, k \in I, i < k \tag{3.12}$$

$$ct_k - ct_i + M(1 - v_{ik}) \geq pt_k x_k \qquad \forall i, k \in I, i < k \tag{3.13}$$

$$ct_i - ta_{ij} \leq dd_j x_i \qquad \forall j \in J_i, i \in I \tag{3.14}$$

$$ea_{ij} - ct_i \leq dd_j x_i \qquad \forall j \in J_i, i \in I \tag{3.15}$$

$$ct_i \geq 0 \qquad \forall i \in I \tag{3.16}$$

$$ta_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{3.17}$$

$$ea_{ij} \geq 0 \qquad \forall i \in I, j \in J \tag{3.18}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{3.19}$$

$$v_{ik} \in \{0, 1\} \qquad \forall i, k \in I, i < k \tag{3.20}$$

A korábbi modellekhez hasonlóan a (3.10) és (3.19) korlát biztosítja, hogy egy megrendelés csak egy pick listára kerüljön. A (3.11) korlát azt mondja, hogy egy pick lista befejezési

ideje nem lehet korábban, mint a kiszedési ideje. A (3.12) és (3.13) korlátok azt fejezik ki, hogy a közvetlen rákövetkező pick lista befejezési ideje legalább annyival később legyen, mint az azt megelőző kiszedési ideje. A (3.14) és (3.15) korlát határozza meg a késések, illetve a korábbi befejezés nagyságát. A (3.16), (3.17) és (3.18) korlát a befejezési idő, a késés és a koraiság nemnegativitását fejezi ki.

Elsayed (1993) egy három lépésből álló módszert javasol a fent leírt problémára. Első lépésben minden megrendelés, mint önálló pick lista, kap egy prioritási értétek a határidejének és a kiszedési idejének súlyozott összege alapján. Ezután a pick listákat a prioritási értékük szerinti növekvő sorrendbe állítja, ami megadja a célfüggvény értékét. Ha van két szomszédos megrendelés, melyek cseréjével a célfüggvény értéke csökkenthető, akkor azokat felcseréli. A második lépésben a megrendeléseket csoportosítja az első lépéseben felállított sorrend alapján, azaz minden megrendelésről sorban eldönti, hogy egyedül érdemesebb-e kiszedni, vagy hozzávenni egy már létező pick listához. Az utóbbi esetben a megrendelést az első olyan pick listához adja hozzá, amelyikkel a célfüggvény értéke csökkenthető. A harmadik lépésben megállapítja minden pick listára a legkorábbi időpontot, amikor a kiszedést el lehet kezdeni. Mivel a célfüggvényben a korábban elkészülő megrendelések is büntetve vannak, ha az egyik pick lista legkorábbi kezdési ideje későbbre esik, mint az őt megelőző befejezési ideje, akkor ott állásidő keletkezik.

Elsayed és Lee (1996) adta a következő algoritmust arra az esetre, ha csak az összes késést kell minimalizálni. Hasonlóan az előző esethez a megrendeléseket a határidejük és kiszedési idejük alapján sorba rendezzük. Az így kapott sorrend felső korlátot ad a célfüggvény értékére. Annak érdekében, hogy a korlátot javítsuk, a következő három lépés valamelyikével csoportosítjuk a megrendeléseket. (1) A *legközelebbi ütemezési szabály* a sorrendben első megrendelést választja mag megrendelésnek, majd ehhez keres további megrendeléseket úgy, hogy az összes késést ne növelje, és betartsa a kapacitás korlátot. Ha ehhez a csoporthoz már nem lehet több megrendelést hozzávenni, akkor a sorrendben következő megrendelést választja az új pick lista magjának, és így tovább. (2) A *legrövidebb kiszolgálási idő szabály* úgy ad a mag megrendeléshez további megrendeléseket, hogy a kiszedési időt minimalizálja. (3) A *legtöbb közös lokáció szabály* a mag megrendeléshez azokat a megrendeléseket adja, amelyekkel a legtöbb közös lokációt tartalmazzák. Numerikus kísérletekkel megmutatták, hogy a legközelebbi ütemezési szabály jobb, mint a másik kettő, és az optimálishoz közeli megoldást ad.

Tsai (2008) vizsgálta azt az esetet, amikor a késések és korai befejezések mellett az összes utazási időt is minimalizálni szeretnénk. A korábbiakkal ellentétben megengedte,

hogy egy megrendelés több részre legyen bontva. Az így kapott problémát genetikus algoritmussal oldotta meg.

Won és Olaffson (2005) arra az esetre adott megoldást, ahol ismert, hogy a megrendelések mikor érkeznek be a raktár rendszerébe. Erre írtak fel egy optimalizálási feladatot, amiben a célfüggvény súlyozott összege az összes utazási időnek, és az összes olyan várakozási időnek, amikor a megrendelés már a rendszerben van, de még nem kezdték el kiszedni. Az általuk adott heurisztika a megrendeléseket az FCFS szabállyal rendezi csoportokba, majd az útvonalat egy 2-közelítő algoritmussal adják meg.

3.6. Dinamikus csoportosítás

Az eddigi modelleknél mindig feltettük, hogy a megrendelésállomány adott, azonban a valóságban a megrendelések az időben folyamatosan érkeznek, és a pontos tételek csak akkor válnak ismertté, ha már bekerültek a rendszerbe.

Ilyen feltételek mellett úgynevezett időablakok szerint szokás csoportosítani a megrendeléseket. Ennek egyik fajtája a *változó időablakos csoportosítás*, ahol megvárják, míg a rendszerbe kellő számú megrendelés kerül (de nem több, mint egy kiszedő kocsi kapacitása), és azt egy csoportban kiszedik. A másik a *fix időablakos csoportosítás*, ahol egy bizonyos intervallumba eső megrendeléseket bontják pick listákra.

Azt az időpontot, amikor egy megrendelés elérhetővé válik (azaz ki lehet szedni), a megrendelés *beérkezésének* hívjuk. A *várakozási idő* az az időtartam, amennyi a megrendelés beérkezésétől a kiszedés megkezdéséig eltelik. A megrendelés *átfutási ideje* az az idő, amennyit a rendszerben tölt a beérkezésétől a kiszállításig.

A dinamikus csoportosítás mérőszáma a megrendelések átlagos átfutási ideje. Ez egyrészt a raktár kiszolgálási minőségét, másrészt a kapacitását is jellemzi. Az átfutási időt minimalizálva nő a kiszolgálás színvonala és a kapacitása is.

Időablakos csoportosítás az átfutási idő minimalizálására

A változó időablakos csoportosítás esetén a megrendelések átfutási ideje konvex függvénye a pick listákon lévő megrendelések számának, amit szokás a pick lista méretének is hívni. A nagy méretű pick listák alacsony átlagos átfutási időt adnak, viszont minél nagyobb egy pick lista, annál több a várakozási idő. A kis méretű pick listák viszont las-

sabb átfutási időt eredményeznek. A fix időablakos csoportosítás esetén az átfutási idő konvex az időablak hosszának méretében. Tehát mindkét esetben meg lehet adni a pick lista méretét vagy az időablak hosszát úgy, hogy azzal az átfutási időt minimalizáljuk. Van Nieuwenhuysen és De Koster (2009) numerikus kísérletekkel igazolta, hogy a fix időablakos csoportosítás egy kevéssel jobb átlagos átfutási időt ad, mint a változó időablakos csoportosítás.

Chew és Tang (1999) adott egy modellt, arra az esetre, amikor egy blokkban vannak elrendezve a termékek, és az időablak hossza változtatható, továbbá a raktárat S-alakban járják be. A modellben a megrendeléseket két sorba rendezik. Az első sorba a megrendelések Poisson-folyamat szerint érkeznek, majd ez alapján kerülnek pick listákra az FCFS-szabály alapján. Azok a megrendelések, amelyek már szerepelnek valamelyik pick listán, a pick listával együtt átkerülnek a második sorba. Innen veszik le őket egymás után a dolgozók. A két blokkos esetre Le-Duc és De Koster (2007) adott hasonló megoldást, míg a fix időablakos esetre Van Nieuwenhuysen és De Koster (2009).

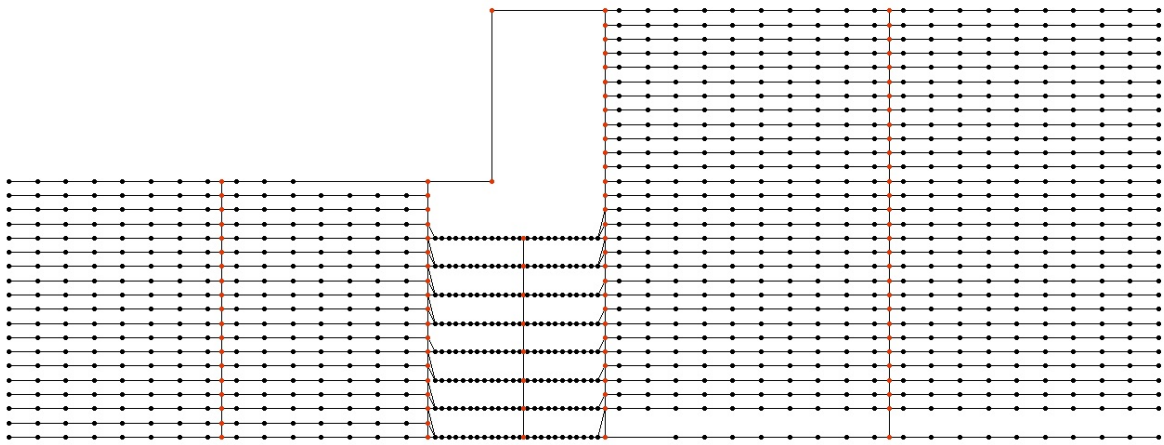
4. fejezet

Eredmények a gyakorlatban

4.1. Útvonal minimalizálása a pick listák bontásával

Ahogy az első fejezetben említettük, az első projekt célja annak megállapítása volt, hogy mennyi idő takarítható meg a pick listák bontásával.

Ehhez először megkonstruáltam a raktár alaprajzát gráfként. A gráf csúcsai a lokációk, illetve olyan fiktív csúcsok, melyek az útvonalakat reprezentálják. A gráf élei azon lokációk és/vagy fiktív csúcsok között vannak, melyek egymásból közvetlenül elérhetők. Ezzel a módszerrel kaptam egy 1179 csúcsból álló gráfot, amin a Floyd-Warshall algoritmussal kiszámoltam a távolságot minden pontból minden pontba. Az algoritmus nagyjából 50 perc alatt futott le, de mivel a távolságokat csak egyszer kellett kiszámítani, az algoritmus futási ideje nem volt lényeges szempont. A számításokat AIMMS-ben végeztem. Az alábbi ábrán a megkonstruált gráf látható:



Miután megkaptuk az összes lokáció egymástól vett távolságát, azt elemeztük, hogy a jelenlegi pick listák bontásával átlagosan az összes megtett távolság hány százaléka takarítható meg. Az elemzéshez többféle mintát néztünk. Az első mintában olyan pick listákat néztünk, melyek az ipari és ruházati termékek bontása után a részlisták száma pontosan a duplája volt az eredetinek, azaz a bontás után rész listák nem lettek sem összevonva, sem szétszedve. Ebben az esetben a megtakarítás 14% volt, tehát összesen 14%-kal rövidebb útvonalon lehetett kiszedni a termékeket. A második típusú mintában olyan pick listák szerepeltek, melyek közül néhányat a szétválasztás után össze lehetett vonni. Ekkor a megtakarítás sokkal nagyobb, 32% volt. A harmadik mintában viszont olyan pick listák szerepeltek, melyek száma a szétválasztás után megnőtt egyéb szempontok miatt (például kiszállítás iránya). Ez az eset nagyrészt veszteséges volt, ha a listák száma több, mint 30%-kal nőtt meg. Mindennek ellenére a várakozás az, hogy a harmadik eset csak elhanyagolható mennyiségben jelenik meg.

Az új kiszedési rendszert a következő napokban tesztelik, és az előzetes felmérések alapján nagyjából 15%-os megtakarítást várnak tőle.

4.2. Útvonal minimalizálása a termékek optimális elhelyezésével

Az idő előre haladtával felmerült az igény, hogy vizsgáljuk meg, hogy lehet-e a jelenlegi lokáció kiosztásnál jobbat találni. Azaz elemezzük azt, hogy a termékeket lehet-e jobban elhelyezni a raktárban, hogy azok kiszedését rövidebb útvonalon meg lehessen tenni.

Jelenleg a raktárban a vállalatirányítási rendszer a legközelebbi nyitott lokáció elvét használja azzal a kiegészítéssel, hogy a termékek az SUT-juk (Storage Unit Type) és az ipari jellegük alapján a raktár más-más részében kapnak helyet.

Ezen szeretnének módosítani, és áttérni az osztályozás alapú raktározásra. Ezért a cél annak meghatározása volt, hogy melyik binbe milyen típusú termékek kerüljenek, A, B vagy C. Ahogy a második fejezetben említettük, az osztályok optimális elhelyezése nagyban függ a raktár bejárási útvonalától. Jelen esetben a raktárat kétszer két blokkosnak lehet tekinteni. Ezt a két részt úgy lehet csak bejárni, hogy a középső úton lehet végig tolni a kiszedő kocsit, a sorokba azzal nem lehet bemenni, mivel ha fizikailag be is fér a sorba, a termékek kiszedéshez már nincs elég hely. A számítások során feltettük, hogy kocsit tolva lassabban lehet haladni, mint gyalog, és a magasan, illetve alacsonyan lévő

polcokról több idő levenni a termékeket, mint a kézmagasságban lévőkről.

Ahhoz, hogy megállapítsuk egy adott bin típusát, vagyis hogy melyik osztályba tartozó termékeket lehet benne tárolni, a következő lépéseket tettük. Először minden binre kiszámoltuk, hogy mennyi időbe telik odaérni a kezdőponttól (ez az a pont, ahol a dolgozók felveszik a pick listákat). Ezt az időt úgy állapítottuk meg, hogy először a kocsival el kell jutni a bin soráig, majd a sorban gyalog be kell menni a bin oszlopáig. A gyaloglási sebességet 5 km/h-nak vettük, a kocsival való haladást pedig az alapsebesség 80%-ának, azaz 4 km/h-nak. Mivel a magasság csak kisebb mértékben játszik szerepet a bin elérési idejében, ezért azt kisebb súllyal számítottuk bele az időbe. Ezután az elérési időket normáltuk, tehát elosztottuk a legtöbb időre lévő bin idejével. Ezzel $0,3$ és 1 közötti értékeket kaptunk, ami után az osztályok volumenének százalékos megoszlása alapján határoztuk meg a binek típusát.

Ennek eredményét egyelőre még nem használták fel a gyakorlatban, azonban egy későbbi projekt alapjául fog szolgálni.

Köszönetnyilvánítás

Szeretnék köszönetet mondani külső témavezetőmnek, Horváth Jánosnak, hogy közbejárásával megteremtette a dolgozat megírásához szükséges feltételeket, illetve belső konzulensemnek, Szabó Csabának, aki észrevételeivel és tanácsaival segítette munkámat, valamint rendelkezésemre bocsátotta a dolgozat elkészüléséhez szükséges szakirodalmat.

Szeretném megköszönni Simona Sava-nak, a Coats Hungary vezérigazgatójának, hogy a cérnagyárban írhattam a szakdolgozatom, illetve hogy hozzájárult a számításokhoz szükséges adatok átadásához. Köszönöm Boros Gergelynek, Bujtár Norbertnek és Józsa Imrének, hogy fáradhatatlanul válaszoltak a kérdéseimre, segítettek összekötni az illetékes emberekkel, és megmutatták a raktározás folyamatát a gyakorlatban. Végül, de nem utolsó sorban szeretnék köszönetet mondani Kántor Tibornak és Konrád Gergelynek, hogy az irodában mindig biztosították a felhőtlen hangulatot és az életmentő kávét. :)

Irodalomjegyzék

[1] Albareda-Sambola, M.; Alonso-Ayuso, A.; Molina E. & de Blas, C.S. (2009) Variable Neighborhood Search for Order Batching in a Warehouse. *Asia-Pacific Journal of Operational Research* 26(5), 655-683

[2] Bozer, Y.A. & Kile, J.W. (2008) Order Batching in Walk-and-Pick Order Picking Systems. *International Journal of Production Research* 46(7), 1887-1909

[3] Brynzér, H., Johansson, M.I. (1996) Storage assignment: Using the product structure to reduce order picking times. *International Journal of Production Economics* $46-47$, 595-603

[4] Bullnheimer, B.; Hartl, R.F. & Strauss, C. (1999) A New Rank Based Version of the Ant System - A Computational Study. *Central European Journal of Operations Research* 7(1), 25-28

[5] Chew, E. & Tang, L. (1999) Travel Time Analysis for General Item Location Assignment in a Rectangular Warehouse. *European Journal of Operational Research* 112(3), 582-597

[6] Choe, K. & Sharp, G. P. (1991) Small parts order picking: design and operation. *Elérhető on-line: http://www.isye.gatech.edu/logisticstutorial/order/article.htm,(2005. május).*

[7] de Koster, R., Le-Duc, T., Roodbergen, K.J. (2007) Design and control of warehouse order picking: a literature review. *European Journal of Operational Research* 182(2), 481-501

[8] de Koster, R.; Roodbergen, K.J. & van Voorden, R. (1999) Reduction of Walking Time in the Center of De Bijenkorf. *New Trends in Distribution Logistics*, Speranza, M.G. & Stähly, P. (eds.), 215-234, Springer: Berlin

[9] de Koster, R.; van der Poort, E. & Wolters, M. (1999)
Efficient Orderbatching Methods in Warehouses. *International Journal of Production Research* 37(7), 1479-1504

[10] Elsayed, E.A. & Lee, M.-K. (1996)
Order Processing in Automated Storage/Retrieval Systems with Due Dates. *IIE Transactions* 28(7), 567-577

[11] Elsayed, E.A. & Stern, R.G. (1983)
Computerized Algorithms for Order Processing in Automated Warehousing Systems. *International Journal of Production Research* 21(4), 579-586

[12] Elsayed, E.A. & Unal, O.I. (1989)
Order Batching Algorithms and Travel-Time Estimation for Automated Storage/Retrieval Systems *International Journal of Production Research* 27(7), 1097-1114

[13] Elsayed, E.A.; Lee, M.-K.; Kim, S. & Scherer, E. (1993)
Sequencing and Batching Procedures for Minimizing Earliness and Tardiness Penalty of Order Retrievals. *International Journal of Production Research* 31(3), 727-738

[14] Elsayed, E.A. (1981)
Algorithms for Optimal Material Handling in Automatic Warehousing Systems. *International Journal of Production Research* 19(5), 525-535

[15] Gademann, N. & van de Velde, S. (2005)
Order Batching to Minimize Total Travel Time in a Parallel-Aisle Warehouse. *IIE Transactions* 37(1), 63-75

[16] Gademann, N.; van den Berg, J.P. & van der Hoff, H.H. (2001)
An Order Batching Algorithm for Wave Picking in a Parallel-Aisle Warehouse. *IIE Transactions* 33(5), 385-398

[17] Gibson, D.R. & Sharp, G.P. (1992)
Order Batching Procedures. *European Journal of Operational Research* 58(1), 57-67

[18] Glover, F. (1986)
Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research* 13(5), 533-549

[19] Hall, R.W. (1993)
Distance approximation for routing manual pickers in a warehouse. *IIE Transactions* 25(4), 76-87

[20] Henn, S. & Wächser, G. *Tabu Search Heuristics for the Order Batching Problem in Manual Order Picking Systems*, 2010.

[21] Henn, S., Köch, S., Wächser, G. *Order Batching in Order Picking Warehouses: A Survey of Solution Approaches*, 2011.

[22] Henn, S. *Algorithms for On-line Order Batching in an Order-Picking Warehouse*, 2010.

[23] Ho, Y.-C. & Tseng, Y.-Y. *A Study on Order-Batching Methods of Order-Picking in a Distribution Center with two Cross Aisles*, 2006.

[24] Ho, Y.-C.; Su, T.-S. & Shi, Z.-B. *Order-Batching Methods for an Order-Picking Warehouse with two Cross Aisles*, 2008.

[25] Hsu, C.-M.; Chen, K.-Y. & Chen, M.-C. *Batching Orders in Warehouses by Minimizing Travel Distance with Genetic Algorithms*, 2005.

[26] Kovács A. *Optimizing the storage assignment in a warehouse served by milkrun*, 2009.

[27] Le-Duc, T. & de Koster, R. (2007)
Travel Time Estimation and Order Batching in a 2-Block Warehouse. *European Journal of Operational Research*

[28] Pan, J.C.-H. & Liu, S. (1995)
A Comperative Study of Order Batching Algorithm. *International Journal of Management Science* 23(6), 691-700

[29] Petersen, C.G. and Aase, G. (2004)
A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics* 92, 11-19

[30] Petersen, C.G., Aase, G. and Heiser, D.R. (2004)
Improving order-picking performance through the implementation of class-based storage. *International Journal of PhysicalDistribution & Logistics Management* 34(7), 534-544

[31] Petersen, C.G. (1997)
An evaluation of order picking routing policies. *Journal of Operations & Production Management* 17(11), 1098-1111

[32] Ratliff, H.D. & Rosenthal, A.S. (1983)
Orderpicking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research* 31(3), 507-521

[33] Rosenwein, M.B. (1996)
A Comparison of Heuristics for the Problem of Batching Orders for Warehouses Selection. *International Journal of Production Research* 34(3), 657-664

[34] Ruben, R.A. & Jacobs, F.R. (1999)
Batch Construction Heuristics and Storage Addignment Strategies for Walk/Ride and Pick Systems. *Management Science* 45(4), 575-596

[35] Tsai, C.-Y.; Liou, J.J.H. & Huang, T.-M. (2008)
Using a Multiple-GA Method to Solve Batch Picking Problem: Considerint Travel Distance and Order Due Time. *International Journal of Production Research* 46(22), 6533-6555

[36] van Nieuwnhuyse, I. & de Koster, R. (2009)
Evaluating Order Throughput Time in 2-Block Warehouses with Time Window Batching. *International Journal of Production Economics* 121, 654-664

[37] Won, J. & Olafsson, S. (2005) Joint Order Batching and Order Picking in Warehouse Operation. *International Journal of Production Research* 43(7), 1427-1442