Multidiszciplináris tudományok, 10. kötet. (2020) 2 sz. pp. 292-302 **https://doi.org/10.35925/j.multi.2020.2.34**

MILKRUN ÚTVONALAK OPTIMALIZÁLÁSA EGY AUTÓIPARI BESZÁLLÍTÓNÁL

Francuz Ádám
hallgató, Miskolci Egyetem, Gépészmérnöki- és Informatikai Kar
3515 Miskolc, Miskolc-Egyetemváros, e-mail: francuzadam13@gmail.com

Bányai Tamás
egyetemi docens, Miskolci Egyetem, Gép- és Terméktervezési Intézet
3515 Miskolc, Miskolc-Egyetemváros, e-mail: alttamas@uni-miskolc.hu

Absztrakt

A cikk témája az autóipari beszállítóknál alkalmazott Milkrun útvonalak optimalizálása az idő függvényében. A létrehozott metódus első esetben egy Milkrun útvonalra hajtja végre az optimalizálást, majd ezt kiterjeszti több útvonalra annak érdekében, hogy az adott időkorlátnak megfeleljen az optimalizálás. Az optimalizálási folyamat és annak megvalósíthatósága egy konkrét számpéldán keresztül lesz bemutatva.

Kulcsszavak: *Milkrun, optimalizálás, logisztikai folyamat, autóipar*

Abstract

The topic of this article is the optimization of Milkrun routes used at the suppliers of the automobile industry. The following method first optimizes the route for one Milkrun, than expands the number of routes in order to fit in the given time limit. The optimization process and its practicability will be demonstrated through a concrete example.

Keywords: *Milkrun, optimization, logistic process, automobile industry*

1. Bevezetés

A folyamatosan növekvő társadalmi igények lényegesen megváltoztatják az autóipar szerepét és működését. Már a XX. század második felétől kezdve egyre nagyobb jelentőséget tulajdonítanak az autóipari folyamatok rugalmas működésének, valamint ezen folyamatok idő szerinti minimalizálásának. Az autóipari beszállítók logisztikai rendszerében nagy átfutási idő kapcsolódik a termelési logisztikai folyamatokhoz, ebből adódóan minden vállalat próbál kedvező megoldást találni, és minél nagyobb mennyiségben csökkenteni a termelési logisztika átfutási idejét. Ilyen kedvező megoldásoknak tekinthetők a Milkrun megoldások. A Milkrun konkrét időpontokhoz kötött, változó mennyiségű, de célszerűen hasonló méretű alapanyagok utánpótlását biztosító rendszer. Valójában ez a termelésben felhasznált alapanyagok és/vagy félkész termékek aktuális fogyásának megfelelő menetrendszerűen történő kiszolgálását, utántöltését jelenti. A Milkrun rendszerek előnye, hogy a végrehajtását szolgáló kocsik rugalmasan bővíthetők, valamint az alapanyagellátás mellett a termelésben keletkező hulladékot és göngyöleget is el tudja szállítani, azaz egy alapanyag körforgás alakítható ki [1-3]. Jelen cikk keretében a szerzők egy olyan optimalizálási algoritmust ismertetnek, amely alkalmas komplex, ipari környezetben működő milkrun alapú anyagellátási rendszerek tervezésére. A cikkben egy olyan megoldási módszer kerül egy autóipari beszállító gyakorlati példáján keresztül bemutatásra, mely az Excel Solver adottságait kihasználta kötegelt futtatással alkalmas a nagyméretű milkrun rendszerű üzemen belüli ellátási lánc optimális kialakításának meghatározására.

2. Szakirodalmi áttekintés

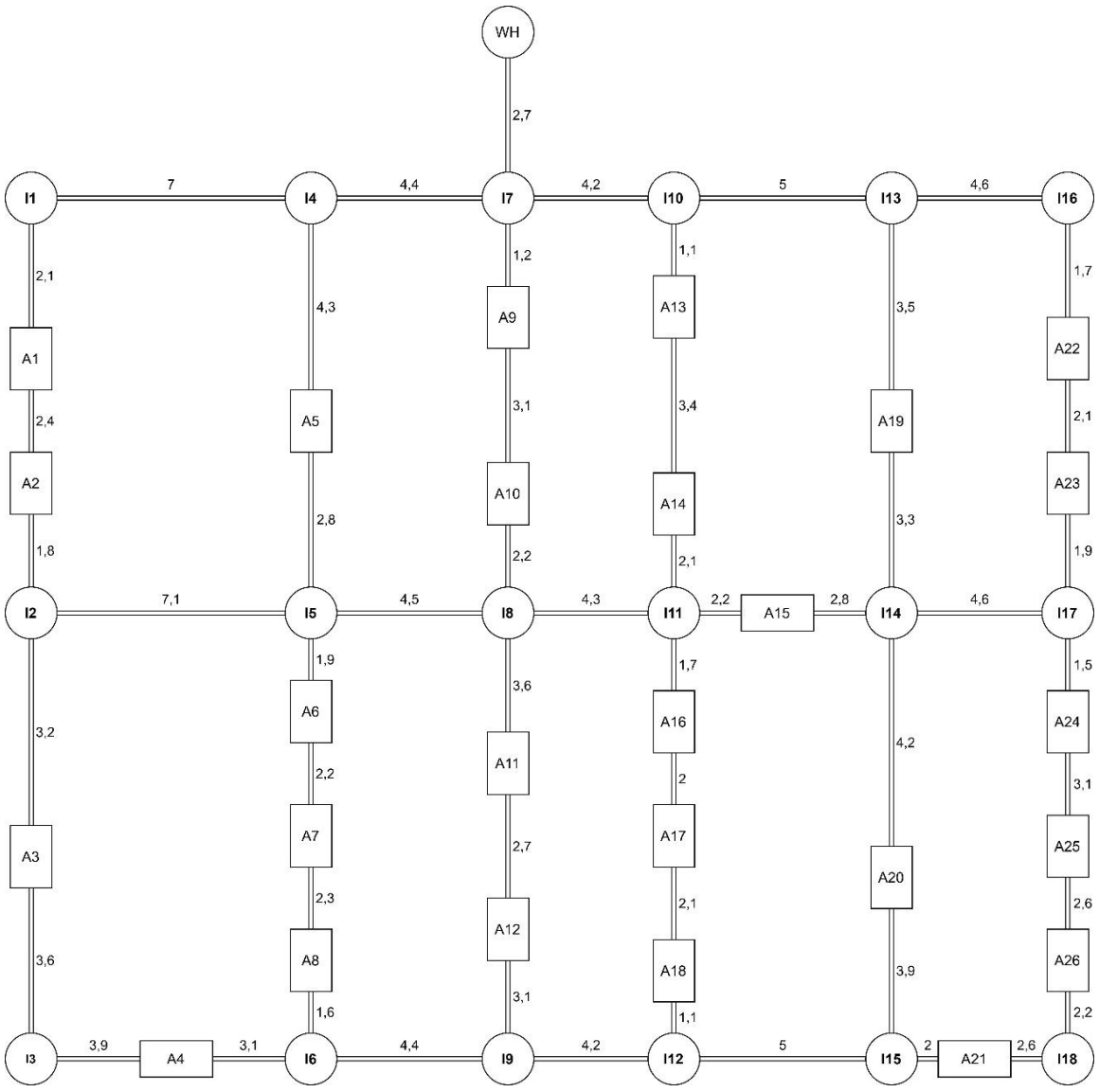
A milkrun alapú anyagellátási rendszerek tervezésének a szakirodalma az elmúlt egy évtizedben igen kiszélesedett, ami egyrészt annak volt köszönhető, hogy ezen megoldások egyre szélesebb körben elterjedtek az iparban, így különösen a mechatronikai összeszerelés és az autóipari beszállítói környezetben [4]. Mivel jelen tanulmány célja egy a milkrun anyagellátási rendszerek tervezésére alkalmas módszer bemutatása, ezért jelen szakirodalmi áttekintésben azon főbb források rövid bemutatását végezzük el, melyek a tervezési módszereket, a tervezés során alkalmazott főbb algoritmusokat tekintjük át. Egyértelműen megállapítható, hogy a milkrun anyagellátási rendszerek tervezése azok komplexitásának okán alapvetően heurisztikus algoritmusokkal valósítható meg. Egy autóipari beszállító példáján keresztül kerül bemutatása egy olyan genetikus algoritmuson alapuló optimalizálási módszer, mely egy NP-nehéz vegyes egészértékű programozási feladat megoldását illusztrálja [5]. Egy harmony search és szimulált lehűtés alapján készült hibrid metaheurisztikus megoldás képezi az alapját a [6] szakirodalmi forrásban tárgyalt algoritmusnak, mely segítségével a szerzők egy üzemen kívüli milkrun anyagellátási rendszert és egy crossdocking létesítményt tartalmazó ellátási lánc optimalizálására mutatnak példát. A C-W saving algoritmus alkalmazására találhatunk példát az [7] forrásban, ahol egy a járatok úthosszát és az üzemeltetési költséget minimalizáló tervezési módszer kerül ismertetésre. Egy dinamikus optimalizálási feladat példáján keresztül szemlélteti a [8] szakirodalmi forrás a hangyafarm algoritmus alkalmazhatóságát. A bemutatott példában az anyagellátási feladatok végrehajtási idejének a minimalizálása szerepel fő célfüggvény komponensként. A greedy és a tabu keresés alapján készített kétfázisú metaheurisztikus algoritmus alapján készült a [9] cikkben bemutatott tervezési módszer, melynek célja a nagy járatsűrűségű milkrun anyagellátási rendszerekben az optimális járművek kiválasztása. A [10] szakirodalmi forrásban az evolúciós stratégiák alkalmazására láthatunk példát. A bemutatott példa egyedisége abban rejlik, hogy a milkrun anyagellátási rendszer tervezésekor a göngyöleg inverz folyamatait is integrálja a rendszerbe, mely mind az üzemen belüli, mind az üzemen kívüli milkrun folyamatoknak szerves része [12]. Természetesen a heurisztikus megoldások mellett számos szimulációs alkalmazás is nagymértékben segítheti a milkrun rendszerek tervezését [13-16]. A cikk ezen rövid szakirodalmi áttekintést követően bemutatja a konkrét optimalizálási feladatot. Ezután leírásra kerül a milkrun útvonal optimalizálása egy, illetve több milkrun járat esetében.

3. A feladat leírása

Az optimalizálási algoritmus a *Microsoft Office* termékcsalád *Excel* szoftverében lett létrehozva az *Excel Solver* bővítmény és a VBA (*Visual Basic Application*) fejlesztőkörnyezet segítségével. Az általános algoritmus bemutatással párhuzamosan a cikk egy számpélda segítségével mutatja be az algoritmus működését. A példa egy kitalált gyártócsarnok elrendezés, amely tartalmaz:

- 26 db gyártósort (a térképen „A” karakterrel jelölve),
- 18 db elágazást (a térképen „I” karakterrel jelölve),
- a raktári kiinduló és érkezési pontot (a térképen „WH” karakterrel jelölve),
- ezeket a gyártósorokat, elágazásokat és raktári pontot összekötő egyenes szakaszokat.

A feladat jobb megoldhatósága érdekében az algoritmus a gyártósorok méretét és kapacitását azonosnak tekinti. A szakaszok hossza egy mértékegység nélküli, egymáshoz viszonyított arányszám. Az egységnyi szakasz megtételéhez szükséges idő 3 s, minden gyártósoron az állásidő 10 s, ezek az adatok szükségesek a végső időeredmény meghatározásához.



1. ábra. *A gyártócsarnok elrendezése*

Az optimalizálás fő célja olyan idő függvényben minimalizált útvonal/útvonalak létrehozása, amely/amelyek megfelelnek a feladat végrehajtásához szükséges feltételeknek:

- A létrejött útvonalak megtételéhez szükséges idő nem haladhatja meg a Milkrun rendszer időkapacitását. Az időkapacitás eredhet humán forrásból (pl. ebédszünet, munkaidő vége), vagy eredhet technikai forrásból (pl. lemerült akkumulátor). Jelen számpéldában a Milkrun rendszer időkapacitása 500 s. Abban az esetben, ha az útvonalhoz szükséges idő meghaladja az időkorlátot, akkor növelni kell a Milkrun kocsik számát, és újra végrehajtani az optimalizálást.
- A Milkrun kocsik számát csak addig lehet növelni, ameddig az igényelt mennyiség nem lépi át a rendelkezésre álló Milkrun kocsik számát. Abban az esetben, ha az összes rendelkezésre álló kocsit felhasználásra került, de az időkorlát feltételünk nem teljesül, akkor a feladat nem megvalósítható.

Idő függvényében minimalizált útvonal meghatározásához az egyváltozós, idő szerint változó függvény szélsőértékét kell meghatározni, vagyis deriválás után a derivált függvény értéke 0.

$$f(t) - idő\ szerint\ változó\ függvény$$

$$f'(t) = 0 \tag{1}$$

Az idő értéke és a távolság értéke között egyenes arányosság van (1 útvonalegység → 3 s), ezért az idő szerint változó függvény és a távolság szerint változó függvény csak konstansban különbözik. Mivel konstans deriváltja 0, ezért az idő szerinti minimalizáláshoz elegendő távolság szerint optimalizálni az útvonalat.

$$g(s) - távolság\ szerint\ változó\ függvény$$

$$f(t) = g(s) * c \tag{2}$$

$$f'(t) = g'(s) \tag{3}$$

Ebből eredően az optimalizálási metódus távolság értékekkel dolgozik, csak a végén kerül átváltásra idő mértékegységre.

4. Milkrun útvonal optimalizálása egy Milkrun járatra

Az egy útvonalra történő optimalizálásnál a fő célunk, hogy a raktári pont legyen a kezdő és érkezési pont és minden gyártósor pontosan egyszer legyen érintve.

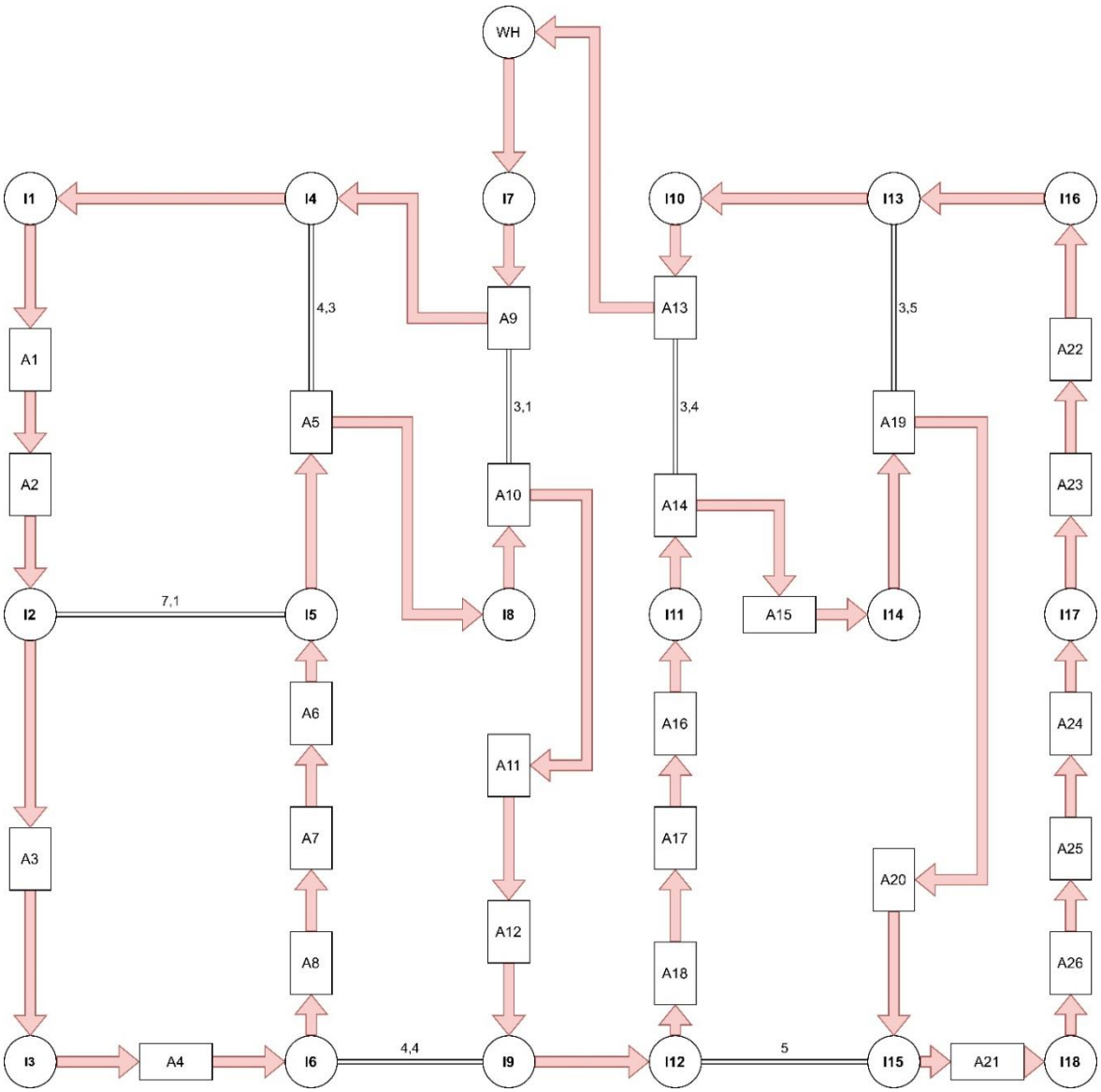
Első lépésben a gyártósorok és a raktári pont azonosítóit számazonosítókká kell alakítani. Erre azért van szükség, mert a Solver bővítmény csak számfeltételekkel tud optimalizálni, egyéb azonosító feltételekkel nem. Az átalakítás az 1. táblázatban vázoltak szerint folyik.

1. táblázat. *Példák a gyártósor számazonosítókhoz*

Gyártósor azonosítója	Gyártósor számazonosítója
WH	0
A1	1
A2	2
A3	3
A4	4
A5	5

Az azonosítók átalakítása után az Excel táblába fel kell írni az összes olyan lehetséges útszakaszt, ami két gyártósort (vagy egy gyártósort és a raktári pontot) köt össze. Ezekhez a szakaszokhoz is azonosítót kell hozzárendelni, ugyanis ezeknek a szakaszoknak az alakításával lehet az optimalizálást végrehajtani. A szakaszok azonosítója a kiindulási gyártósor számazonosítója és az érkezési gyártósor számazonosítója kötőjellel összekötve. Fontos megjegyezni, hogy irányított szakaszról van szó, tehát egy szakasznak mindkét végpontja lehet kiindulási és érkezési pont is, így minden szakasszal duplán kell számolni.

Az optimáláshoz szükséges adatok és feltételek meghatározása után következik a tényleges optimalizálás. A Solver bővítmény a Microsoft Excel „Adatok” fülön belül az „Elemzés” csoportban található.



2. ábra. *Egy Milkrun útvonalra történő minimalizálás eredménye*

A Solver felületén az alábbi adatokat szükséges megadni az optimalizáláshoz:
Célérték cella: Az a cella, amelyben összegezve vannak az adott szakaszok távolságai
Cél: Minimalizálás
Változócellák: Azok a cellák, amelyek változtatásával lehet elérni az optimális eredményt. Jelen eseteben ezek a gyártósor számazonosítók, ugyanis ezek változtatásával tudjuk minimalizálni az útvonalat
Vonatkozó korlátozások: A feladat elvégzéséhez szükséges feltételek megadása: Jelen esetben 4 feltétel szükséges az optimalizáláshoz:

- A számazonosítókat tartalmazó cellák értéke legalább 1, legfeljebb 26
- Minden gyártósort csak egyszer érinthet az útvonal, ezért minden cella értéke különböző legyen

- A cellák alapesetben a racionális halmazba tartozó számokat vehetik fel értékül, azonban a számazonosítók egész számok, így feltételként kell megadni, hogy azok a cellák kifejezetten egész számot vehessenek fel értékül.

Megoldási módszer: Evolutív

Az evolutív megoldási módszer során n db fokozaton keresztül választja ki az optimális megoldást a Solver. Első lépésben a bemeneti értékekből képzett véletlenszerű halmazok közül választja ki az optimális megoldást. Ebből a kiválasztott halmazból második lépésben újabb halmazokat generál, és abból is kiválasztja az optimum értéket. Ez az algoritmus addig ismétlődik, ameddig egy megadott időparaméteren belül a Solver nem tudja pontosítani a megoldást.

Az összes paraméter megadása után a Solver a 2. ábrán bemutatott eredményt adta ki minimális útvonalként.

Az optimalizálás eredménye 1 Milkrun járat esetén a 2. táblázatban került összefoglalásra.

2. táblázat. *Egy Milkrun útvonalra történő optimalizálás eredménye*

Útvonal hossza:	141,9
Útvonal megtételéhez szükséges idő:	425,7 s
Állásidő:	260 s
Összes idő:	**685,7 s**

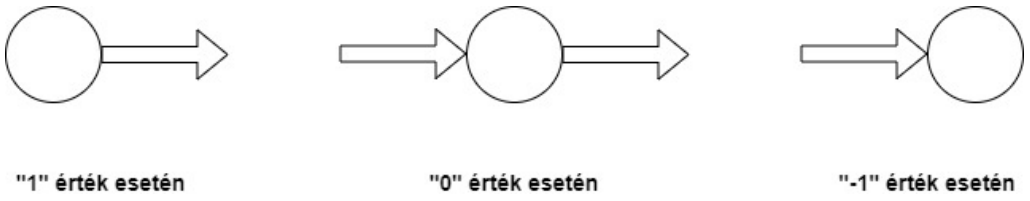
Az optimalizálás eredménye nem felel meg a Milkrun kocsik időkapacitás feltételének, ezért az optimalizálást végre kell hajtani 2 Milkrun járatra is.

5. Milkrun útvonal optimalizálása két járatra

5.1. Legrövidebb útvonal meghatározása

Annak következtében, hogy az egy Milkrun járatra való optimalizálás eredménye nem felel meg a Milkrun időkorlát követelményének, az optimalizálást el kell végezni két Milkrun járatra. Ennek első lépése a legrövidebb útvonal meghatározása a raktári kiindulási ponttól minden gyártósorra. Ehhez szintén az Excel Solver bővítményét kell használni, azonban 26 gyártósorra történő legrövidebb útvonalakat meghatározásra manuális úton sok időt venne igénybe, ezért a VBA fejlesztőkörnyezet segítségével ciklussá alakítjuk az optimalizálást, így egy lépésben el lehet végezni az összes legrövidebb útvonal meghatározását.

Az optimalizáláshoz szükséges táblázatba felírni az összes lehetséges irányított útszakaszt. Az előző minimalizálással ellentétben itt nem a két gyártósor közötti szakaszokat kell figyelembe venni, hanem a gyártósor és elágazás közötti szakaszokat. Minden irányított útszakaszhoz hozzárendeljük annak távolságát és egy bináris számot (0 vagy 1). Ez a szám adja meg, hogy a kívánt legrövidebb útvonalnak része az adott irányított útszakasz, vagy sem. A legrövidebb útvonal hosszát pedig úgy lehet meghatározni a két adatból, hogy minden útszakasz két számértékét összeszorozzuk. Azok a szakaszok, amelyek nem szerepelnek az útvonalban, a 0-val való szorzás miatt kiesnek, így gyakorlatilag az útvonal szakaszi kerülnek összeadásra. Ezt az értéket minimalizálja a Solver. A minimalizálás úgy zajlik, hogy az útszakaszok bináris értékeit változtatja a korlátozó feltételek betartásával. A feltételek megadására szükséges egy újabb táblázatot csinálni, amelyben megtalálható felsorolva a kiindulási pont, az összes elágazás és gyártósorok. A feladat korlátozó feltétele az, hogy a kiindulási pont (mindig a raktári pont) értéke 1, az érkezési pont (gyártósor) értéke -1, a többi gyártósor és elágazás értéke 0.



3. ábra. *A gyártósorok lehetséges értékei kiindulás/érkezés szerint*

A pontok értéke abból adódnak, hogy minden kiindulás 1 értéket ad, minden érkezés -1 értéket. Azok a gyártósorok vagy elágazások, amelyeken áthalad az útvonal, az egyszerre kiindulási és érkezési pont is, így összesen azok 0 értéket adnak, ugyanúgy, mint azok a gyártósorok és elágazások, amelyeken nem halad át az útvonal. A korlátozó feltételt pedig ennek függvényében kell beállítani, és mindig annak a gyártósornak adni feltételnek -1 értéket, ahová szeretnénk, hogy megérkezzen az útvonal. A Solver összes paraméterének beállítása után ciklussá kell alakítani az optimalizálást, a bináris értékek oszlopát, valamint a korlátozó feltételek oszlopát szükséges egymás alá másolni, majd a függvényeket ennek megfelelően beállítani, hogy az optimalizálást egy lépésben el tudjuk végezni.

5.2. Gyártósor-párok képzése

Következő lépésben, a legrövidebb útvonalakat felhasználva gyártósor-párokat képzünk, amiket továbbá egy gyártósornak tekintünk, ezzel lényeges leredukálva a gyártósorok számát.

A gyártósor-párokat az őket alkotó gyártósorok szakaszegyezései alapján képezzük. Azok a gyártósorok kerülnek egy gyártósor-párba, amelyeknek lényeges szakaszegyezései vannak. A szakaszegyezések számát megvizsgáljuk az összes gyártósor-pár lehetőségre. Ez jelen esetben $\frac{26*25}{2} = 325$ lehetőséget jelent, aminek meghatározását egy lépésben, a VBA fejlesztőkörnyezetben végezzük el.

A szakaszegyezéseket a bináris értékekből lehet meghatározni, ha egy adott útszakaszhoz tartozó mindkét gyártósorhoz vezető legrövidebb útvonal értéke 1, akkor szakaszegyezésről beszélünk. Az ilyen egyezéseket megszámoljuk, összeadjuk, és megkapjuk az összes lehetséges gyártósor-párhoz tartozó szakaszegyezések számát. Azonban a gyártósor-pár képzés nem csak a szakaszegyezések számától, hanem a két gyártósorhoz vezető legrövidebb útvonalak távolságától is függ. Erre azért van szükség, mert a kiindulási ponttól távol lévő gyártósor-pároknak lényegesen nagyobb a szakaszegyezések száma, mint a kiindulási pont közelében lévők, így a távolságot is figyelembe véve, abból arányszámot képezve lényegesen reálisabb értéket kapunk.

```
Dim x As Integer, i As Integer, j As Integer
Dim string1 As String, string2 As String


For x = 0 To 25
For i = 4 To 108
For j = 5 To 29

string1 = Cells(2, 4 + x).Value
string2 = Cells(2, j + x).Value


If Cells(i, 4 + x).Value = 1 Then
    If Cells(i, j + x).Value = 1 Then
        Cells(i, j + 30 + 30 * x).Value = 1
    End If
End If

Cells(110, j + 30 + 30 * x).Value = WorksheetFunction.CountIf(Range(Cells(3, j + 30 + 30 * x), Cells(190, j + 30 + 30 * x)), 1)
Cells(111, j + 30 + 30 * x).Value = string1 & "-" & string2

Next j
Next i
Next x
```

4. ábra. *A szakaszegyezéseket vizsgáló algoritmus*

5.3. Gyártósor csoportok meghatározása

Az optimalizálási metódus következő lépése, hogy az előző fejezetben meghatározott arányszámot felhasználva a gyártósor-párokból meghatározzuk azt a két gyártósor csoportot, amelyekre optimalizáljuk majd külön-külön a Milkrun útvonalakat. A gyártósor-párokat a legnagyobb arányszámútól kezdve elkezdjük összekapcsolni csökkenő sorrendben addig, ameddig két nagy csoport alakul a gyártósorokból.

3. táblázat. *A végső Milkrun járatokhoz tartozó gyártósorok*

1. Milkrun járat	2. Milkrun járat
A1	A15
A2	A16
A3	A17
A4	A18
A5	A19
A6	A20
A7	A21
A8	A22
A9	A23
A10	A24
A11	A25
A12	A26
A13	
A14	

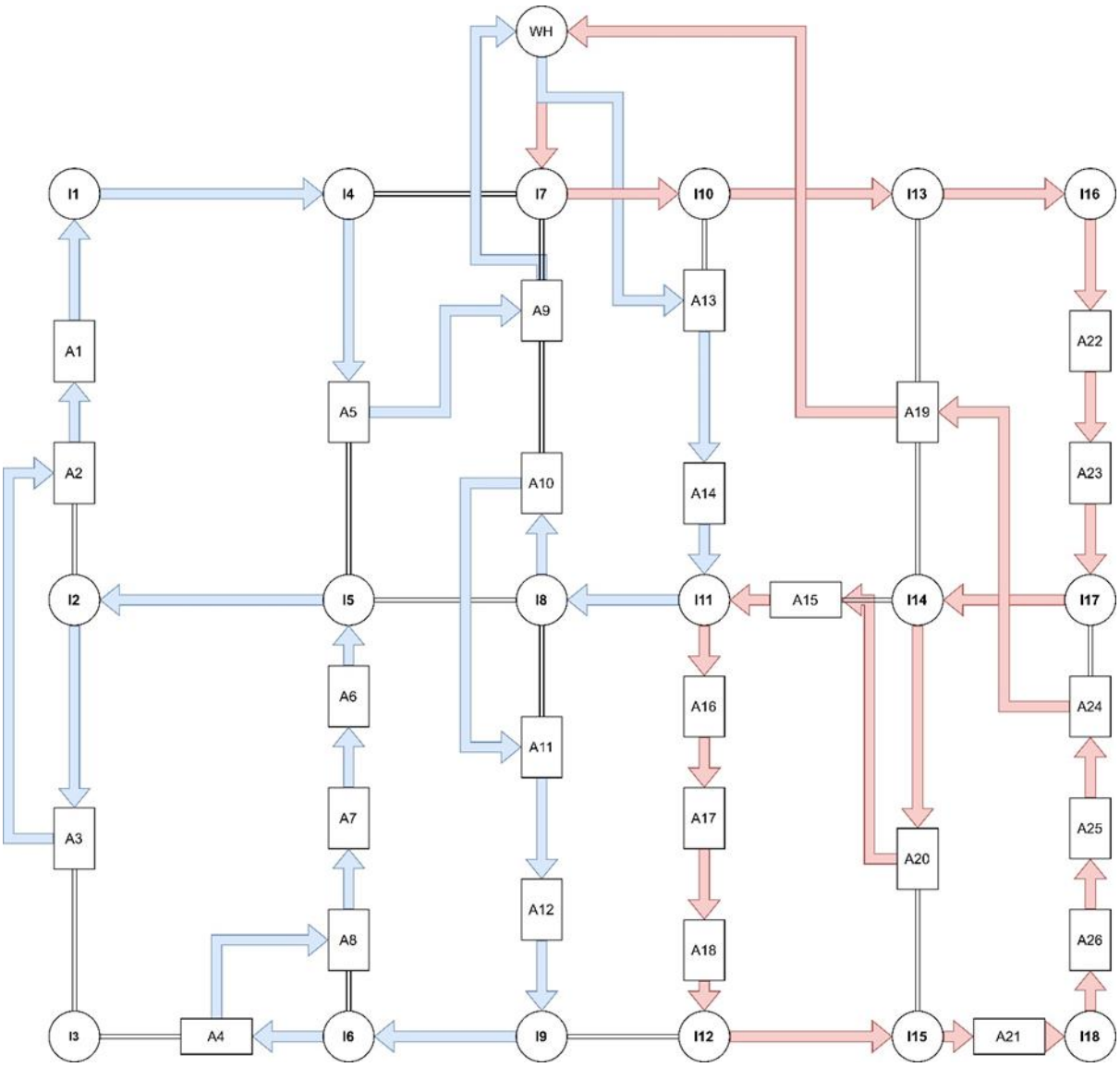
6. Az optimalizálás eredménye

Miután megkaptuk azt a két gyártósor csoportot, amelyre szeretnénk optimalizálni az útvonalakat, elvégezzük a minimalizálást. A Solver beállításai teljesen megegyeznek az egy Milkrun járatra történő optimalizálással, jelen esetben is feltétel, hogy a raktári pont legyen a kiinduló és érkezési pont, és minden gyártósort csak egyszer érinthet az útvonal.

4. táblázat. *Az optimalizálás eredménye*

Útvonal hossza:	95,1	117,4
Útvonal megtételéhez szükséges idő:	285,3 s	352,2 s
Állásidő:	140 s	120 s
Összes idő:	**425,3 s**	**472,2 s**

A kapott eredmények megfelelnek a Milkrun kocsik időkapacitás feltételének, és az útvonal megtételéhez szükséges idők közel megegyeznek, így befejezettnek tekinthető az optimalizálás. Abban az esetben, ha az így kapott időeredmények nem felelnének meg a kapacitás-feltételnek, akkor a Milkrun kocsik növelésével újra el kellene végezni a több Milkrun optimalizálására leírt metódust, figyelve arra, hogy a Milkrun kocsik száma nem lépheti túl a rendelkezésre álló kocsik számát.



5. ábra. *Az optimalizálás eredményeként létrejött Milkrun útvonalak*

7. Összefoglalás

Napjainkban egyre nagyobb vevői igények határozzák meg az autóipar működését. Minden vállalatnak célja az általa gyártott termékeket legyártani, majd eljuttatni a vevőkhöz. Ennek a folyamatnak jelentős részét teszik ki az átfutási idők. Egy autóipari beszállító termelési logisztikai rendszerében számos módszer létezik a rugalmas és jól működő anyagáramlás megvalósítása érdekében. Ilyen rendszernek tekinthetők a Milkrun rendszerek, amelyek egy útvonalat és korfolyamatot létrehozva valósítják meg az anyagáramlást. A cikk témája a Milkrun rendszerek optimalizálása, ahol bemutattunk egy metódust az egy-, illetve több Milkrun útvonalra történő minimalizálásra. A folyamatot egy számpéldán keresztül illusztráltuk, aminek eredményén látható a metódus működése.

8. Köszönetnyilvánítás

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

Irodalom

[1] *Lean logisztika: Termelés kiszolgáló megoldások – A “Milkrun” és a “Water Spider.* Elérhetőség: http://www.leanlogisztika.hu/termeles-kiszolgalo-megoldasok-a-milkrun-es-a-water-spider/ Letöltve: 2020. július 11.

[2] Mácsay, V., Bányai, T.: *Toyota production system in milkrun based in-plant supply*. Journal of Production Engineering 2017, 9(1):141-146. **http://doi.org/10.24867/JPE-2017-01-141**

[3] Bányai, T., Telek, P., Landschützer, C.: *Milkrun based in-plant supply – An automotive approach*. Lecture Notes in Mechanical Engineering 2018, 170-185. **https://doi.org/10.1007/978-3-319-75677-6_14**

[4] Kilic, H.S., Durmusoglu, M.B., Baskak, M.: *Classification and modeling for in-plant milk-run distribution systems*. International Journal of Advanced Manufacturing Technology 2012, 62(9-12):1135-1146. **https://doi.org/10.1007/s00170-011-3875-4**

[5] Sadjadi, S.J., Jafari, M., Amini, T.: *A new mathematical modeling and a genetic algorithm search for milk run problem (an auto industry supply chain case study).* International Journal of Advanced Manufacturing Technology 2009, 44(1-2):194-200. **https://doi.org/10.1007/s00170-008-1648-5**

[6] Hosseini, S.D., Shirazi, M.A., Karimi, B.: *Cross-docking and milk run logistics in a consolidation network: A hybrid of harmony search and simulated annealing approach*. Journal of Manufacturing Systems 2014, 33(4): 567-577. **https://doi.org/10.1016/j.jmsy.2014.05.004**

[7] You, Z.L., Jiao, Y.: *Development and application of milk-run distribution systems in the express industry based on saving algorithm*. Mathematical Problems in Engineering 2014, 536459. **https://doi.org/10.1155/2014/536459**

[8] Ma, J.H., Sun, G.H.: *Mutation ant colony algorithm of milk-run vehicle routing problem with fastest completion time based on dynamic optimization*. Discrete Dynamics in Nature and Society. 2013, 418436. **https://doi.org/10.1155/2013/418436**

[9] Lin, Y., Xu, T.Y., Bian, Z.Y.: *A two-phase heuristic algorithm for the common frequency routing problem with vehicle type choice in the milk run*. Mathematical Problems in Engineering 2015, 404868. **https://doi.org/10.1155/2015/404868**

[10] Ranjbaran, F. Kashan, A.H., Kazemi, A.: *Mathematical formulation and heuristic algorithms for optimisation of auto-part milk-run logistics network considering forward and reverse flow of pallets*. International Journal of Production Research 2020, 58(6):1741-1775. **https://doi.org/10.1080/00207543.2019.1617449**

[11] Korytkowski, P., Karkoszka, R.: *Simulation-based efficiency analysis of an in-plant milkrun operator under disturbances*. International Journal of Advanced Manufacturing Technology 2016, 82(5-8):827-837. **https://doi.org/10.1007/s00170-015-7442-2**

[12] Fedorko, G., Molnar, V., Honus, S., Neradilova, H., Kampf, R.: *The application of simulation model of a milk run to identify the occurrence of failures*. International Journal of Simulation Modelling 2018, 17(3):444-457. **https://doi.org/10.2507/IJSIMM17(3)440**

[13] Tamás, P.: *Decision support simulation method for process improvement of intermittent production systems*. Applied Sciences-Basel 2017, 7(9):950. **https://doi.org/10.3390/app7090950**

[14] Bohács, G., Kovács, G., Rinkács, A.: *Production logistics simulation supported by process description languages*. Management and Production Engineering Review 2016, 7(1):13-20. **https://doi.org/10.1515/mper-2016-0002**

[15] Fedorko, G., Vasil, M., Bartosova, M.: *Use of simulation model for measurement of milkrun system performance*. Open Engineering 2019, 9(1):600-605. **https://doi.org/10.1515/eng-2019-0067**

[16] Veres, P.; Illés, B.; Landschützer, C. Supply Chain Optimization in Automotive Industry: A Comparative Analysis of Evolutionary and Swarming Heuristics. In Vehicle and Automotive Engineering 2; Jármai, K., Bolló, B., Eds.; Springer: Cham, Switzerland, 2018; pp. 666–676. **https://doi.org/10.1007/978-3-319-75677-6_57**