Többkritériumú dinamikus útvonaltervezés menetrendkövető autonóm járművek számára

Az autonóm vagy magasan automatizált járművek alkalmazásának lehetőségei elsősorban azok a területek, ahol bizonyos, előre meghatározott pontokat kell érinteni adott sorrendben. Ilyenek például az áruszállítók vagy a közösségi közlekedés járművei. Általában ezek a járművek előre meghatározott útvonalon, valamilyen kritérium – például legrövidebb út – alapján érik el az állomásokat, gyakran függetlenül az úthálózat aktuális forgalmi állapotától. Szemléletes példa erre egy olyan útvonaltervező módszer bemutatása – elsősorban a magasan automatizált járművek számára, – ami az előre meghatározott állomások között az aktuálisan legkedvezőbb útvonalon juttatja el céljához a járművet.

DOI 10.24228/KTSZ.2020.3.4

Horváth Márton Tamás
e-mail: horvath.marton@mail.bme.hu

1 BEVEZETÉS

Szakértők állítása alapján a jövő közlekedése autonóm, elektromos és megosztott [3]. A cikk ezek közül az elsőre reflektál, mivel napjainkban az autonóm járműveket leginkább egyéni járművek vagy kis flották szintjén tesztelik és fejlesztik, elsősorban a közvetlen fizikai környezet felmérésére koncentrálnak szenzoros mérések alapján [19]. Ahogy az autonóm járművek egyre nagyobb mértékben válnak a forgalom részévé, mindinkább fontos lesz, hogy a szomszédos jármű vagy épp az útpadka pozíciójának figyelése mellett a forgalmi állapotot is vegyék figyelembe az útvonal meghatározása során.

Az autonóm járművek tervezésének kezdetekor a járművek előre meghatározott útvonalakon haladtak, függetlenül attól, hogy az úthálózaton hol volt éppen torlódás. Természetesen a valóságban elvárás, hogy reagáljunk a hálózatban bekövetkező változásokra. A [15]-ben egy, a Dijkstra-algoritmuson [5] alapuló útvonaltervezési metódust mutattak be autonóm személygépjárművek részére. A hálózatot szegmensekre bontották fel, amelyek mindegyikéhez egy-egy ellenállásértéket rendeltek. Az ellenállást minden út megkezdése előtt egyszer számították ki, figyelembe véve az útszakasz hosszát, az időjárási körülményeket, valamint a hálózaton tapasztalható zavarokat. Az útvonal bejárása közben az eredeti

tervet nem vizsgálták felül, így az a hálózaton menet közben bekövetkező változásokra már nem reagált. A közösségi közlekedés, illetve az áruszállítás esetében az útvonalat érdemben befolyásoló tényező a megállók helyzete és a következő megálló elérésének időpontja.

A [11] egy olyan információmegosztási rendszert dolgozott ki, amely alapján a járművek közvetlenül egymást értesítették a forgalmi helyzetről, ezért nem volt szükség mobilitásmenedzsment központ alkalmazására. A járművek az általuk bejárt útvonal információit eltárolták, majd a szemben jövőknek átadták, akik ezt szintén eltárolták. Amikor a szembe irányba közlekedő járművek ismét az eredeti – tehát az elsőként említett járművekkel azonos – irányban haladó járművel találkoztak, akkor átadták nekik az első járműtől származó információkat, valamint a saját információikat is. Így a forgalomban részt vevők tudomást szerezhettek arról, hogy mire számíthatnak maguk előtt, és szükség esetén módosíthatták útvonalukat.

Az autonóm közösségi közlekedési járművek – hasonlóan a jelenlegi fejlett rendszerekhez – várhatóan közvetlen kapcsolatban fognak állni a központtal, így a forgalmi állapotról az értesítést majd innen kaphatják. Ugyanez igaz a gyárakban vagy raktárakban mozgó járművek esetében is. A folyamatos kommunikáció lehetőséget biztosít arra, hogy a járművek a megállók között – ellentétben a jelenlegi gyakorlattal – ne feltétlenül azonos útvonalon haladjanak. A bemutatott megközelítés elvben nem kizárólag autonóm járművekre alkalmazható, hanem humán sofőrök is használhatják, hasonlóan az egyéni közlekedésben elterjedt navigációs alkalmazásokhoz. Ilyen esetekben azonban célszerűbb lehet a „legegyszerűbb", legkevesebb forgalmi kihívást tartalmazó útvonal ajánlása, hogy elkerüljük a vezető figyelmének túlzott lekötését [7].

Az autonóm járművek útvonaltervezése [10] szerint egyszerű Google Maps hálózati adatok és légi felvételek alapján is elvégezhető, nincs szükség arra, hogy az úthálózatot előzetesen gráfként leképezzék. Ipari létesítményekben az úthálózat lényegesen egyszerűbb felépítésű, mint egy városban, így ebben az esetben a jellemző gráf előállítása sem bonyolult. A [14]-ben egy olyan beltéri útvonaltervező algoritmust mutatnak be autonóm járművek részére, amelyben az épület területét virtuális csempékre osztották fel, és a jármű mozgását a csempék foglaltsága befolyásolta. A városi úthálózatokon ez a fajta megközelítés nem lehetséges, ezért helyette heurisztikus legrövidebbút-kereső algoritmusokat használnak. A [9] alapján négy fő kategória készíthető az alkalmazott stratégia szerint: (1) a keresési terület korlátozása, (2) a keresési probléma részekre bontása, (3) a vizsgált útszakaszok körének korlátozása, (4) az előbbiek kombinációja.

Ha az útvonaltervezést több kritérium mentén végezzük el, akkor első lépésben valamilyen Pareto-optimalizálási technikát is lehet alkalmazni. Ezáltal viszonylag alacsony számítási igénnyel megtalálhatók azok a lehetséges megoldások, amelyek az összehasonlítási szempontok alapján egyértelműen jobbak a többi megoldásnál. Minél hosszabb idő áll rendelkezésre a számításokhoz, annál több lehetséges megoldás található, amelyek az összes lehetséges megoldást reprezentálják [4]. Azonban, még ha a Pareto-optimumot sikerül is viszonylag gyorsan megtalálni, a keresőalgoritmusoknak további időre lehet szükségük ahhoz, hogy bebizonyítsák, nincs is több lehetséges legjobb megoldás [6]. Mint azt később látni fogjuk, a cikkben az egyik kritérium (az eljutási idő) domináns a többi kritérium felett, így másik típusú algoritmus kerül felhasználásra.

Az autonóm közösségi közlekedés – egyelőre folyamatos humán helyszíni felügyelet mellett – már több város alacsony forgalmú területein üzemel, például Bécsben [22] vagy a svájci Sionban [16]. A következő fejlődési lépcső, hogy ilyen közösségi közlekedési járműveket nagy forgalmú területeken is üzembe állítsanak. Az [1]-ben igényvezérelt közlekedésre dolgoztak ki egy hasonló metódust autonóm megosztott taxik alkalmazásával. Az útvonaltervezés során kombinálták az aktuális és az előrebecsült utazási igényeket a New York-i taxik historikus adataival [21]. A szimulációk eredményei szerint a jelenleginél kevesebb jár-

mű is elegendő az utasok kiszolgálására anélkül, hogy a várakozási idő túlzottan növekedne. Ugyanezt a taxis adatbázist felhasználva a [24]-ben autonóm igényvezérelt járművek részére fejlesztettek ki egy útvonaltervező algoritmust, makroszkopikus szinten, torlódott úthálózatokon.

A továbbiakban egy többkritériumú dinamikus útvonaltervező algoritmus kerül bemutatásra elsősorban olyan autonóm járművek részére, amelyeknek meghatározott állomásokat kell elérnie meghatározott időben. Ilyenek például a közösségi közlekedés járművei vagy egyes ipari termelő létesítményekben alkalmazott áruszállító járművek. Az útvonalválasztási logika alapja, hogy egy ellenállásfüggvényt minimalizálunk, miközben figyelembe veszszük azt is, hogy az egyes állomásokra mikor kell megérkeznie a járműnek.

2. AZ ÚTVONALTERVEZŐ ALGORITMUS

A modellben alkalmazott algoritmus az úthálózat egyes szakaszaihoz különféle típusú ellenállásokat rendel [18], majd ezeket összegzi. Alapesetben a legkisebb ellenállású útvonal lesz a tervezés eredménye, azonban, ha a jármű késésben van, akkor az algoritmus olyan útvonal választására törekszik, amelyen a késés behozható. Ebben az esetben az útvonalválasztás az eljutási idő alapján történik.

2.1. A logika bemutatása

A kidolgozott útvonaltervező algoritmus négy fő lépésből áll: (1) általános előkészítés, amely előre, bármely időpontban elvégezhető, (2) a következő állomáshoz vezető útvonal megtervezése valós időben, amikor a jármű elérte a megelőző állomást, (3) a következő állomás megközelítése és közben folyamatos újratervezés a hálózat állapotától (kialakuló torlódás, egyéb akadály) függően, valamint (4) az utolsó állomás elérése, ami egyben a menet vége is.

(1) Az általános előkészítés szerepe, hogy a lehető legtöbb számítást offline, még a valós idejű futás megkezdése előtt elvégezzük, amivel valós idejű számítási kapacitás és adatforgalom spórolható meg. Az egymást követő állomások között egy-egy alhálózatot határozunk meg, amelyen az algoritmus a forgalmi állapotot monitorozza, és szükség esetén alternatív útvonalat választ. Az alhálózatot a jármű és az útszakaszok fizikai attribútumai határozzák meg, különösen a szélesség (szűk kanyarok) és a forgalmi szabályok (kanyarodás tiltás, egyirányú utcák). Szintén ebben a lépésben adjuk meg az egyes állomásokról a továbbindulás elvárt időpontját.

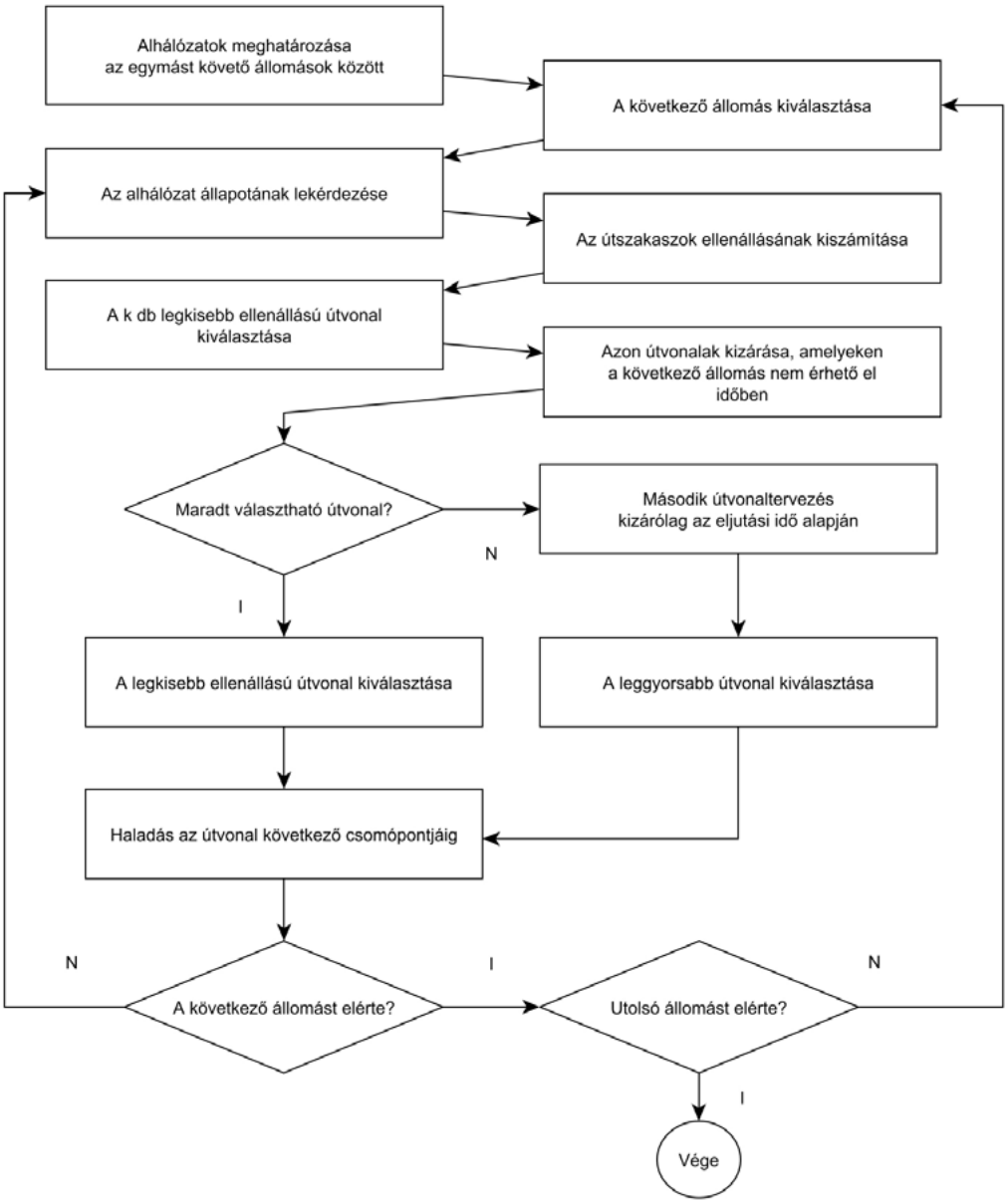
(2) Előre meghatározott az, hogy az egyes állomásokat milyen sorrendben kell érinteni, és mikor kell azokról továbbindulni. Ahogy a jármű két állomás között halad, a kettő közötti alhálózat forgalmi állapotát folyamatosan lekérdezzük, és minden lekérdezés után kiszámítjuk a „*k*" db legkisebb ellenállású útvonalat a Yen-algoritmus alapján [23]. Az útszakaszok ellenállását olyan, a felhasználó számára releváns tényezők határozzák meg, mint például az eljutási idő, a távolság stb. Ezek szabadon megválaszthatók. Ha kevesebb, mint „*k*" db útvonal létezik, akkor ezzel az útvonalkészlettel dolgozunk tovább.

(3) Az előző lépésben meghatározott útvonalakon megnézzük az eljutási időt. Kizárjuk azokat az útvonalakat, amelyeken a következő állomás elérése a tervezett időre nem valósítható meg. A fennmaradó alternatívák közül a legkisebb ellenállású útvonalat választjuk ki, és a jármű ezen halad a következő számításig. Ha a „*k*" db legrövidebb út közül egy sem garantálja, hogy időben el lehessen érni a következő állomást, akkor egy újabb legrövidebbút-keresési algoritmust futtatunk le, ezúttal kizárólag az eljutási időt figyelembe véve. Ezáltal a késés minimalizálható.

(4) A (2) és (3) lépést addig ismételjük, amíg a jármű el nem éri az utolsó állomást. Ekkor véget ér a teljes folyamat.

Az útvonaltervezési folyamatot az 1. ábra mutatja.

1. ábra: Az útvonaltervezési folyamat



2.2 A MODELL LEÍRÁSA

Az útvonaltervező algoritmus a következőképpen írható le. Az *s* és *s*+1 állomások közötti *G* alhálózatot egy súlyozott, irányított gráfnak tekintjük.

$$G = (N,L,w), \tag{1}$$

N csomópontokkal, *L* élekkel és w: $L \rightarrow \mathbb{R}^+$ ellenállásfüggvénnyel, amely minden $(i,j) \in L$ élhez egy ellenállást rendel. Formálisan az ellenállásfüggvény kiterjeszthető a következők szerint:

$$w: N \times N \rightarrow \mathbb{R}^+ \cup 0 \cup \infty, \tag{2}$$

ahol $w(i,i)=0$ minden $i \in N$-re és $w(i,j)=\infty$ minden $(i,j) \notin L$-re. Az $i=x_0 \in N$-től $j=x_n \in N$-ig tartó útvonalat a *G* alhálózatban található $x_0 l_1^p x_1^p l_2^p \ldots l_n^p x_{n,}^p$ lehetséges útvonalak összehasonlításával választjuk ki, *p*=1-től *P*-ig, ahol *P* az *i* és *j* közötti, a *G* alhálózatban szereplő útvonalalak száma. A *p*-edik útvonalváltozat ellenállása:

$$C^p = w\left(x_0, x_1^p\right) + \sum_{m=2}^{n} w\left(x_{m-1}^p, x_m^p\right). \tag{3}$$

Alapesetben az algoritmus egy általános ellenállással (C_g^p) számol, amelyet a felhasználó szabadon határozhat meg, különböző, számára releváns tényezőket kombinálva. Később a bemutatott példában ez a hossz és az idő kombinációja lesz, de számos egyéb mutatót is tartalmazhat, mint például a forintosított költséget vagy az utaselégedettséget [2], vagy akár externális költségeket is, mint például a légszennyezettség hatása az egészségre [12]. Speciális esetben, ha a jármű késésben van, az ellenállás azt jellemzi, hogy a következő állomásig mennyi időbe telik eljutni (C_t^p). A későbbiekben ellenállás alatt az általános ellenállás értendő. Ahol az eljutási idővel kell számolni, az külön kiemelésre kerül.

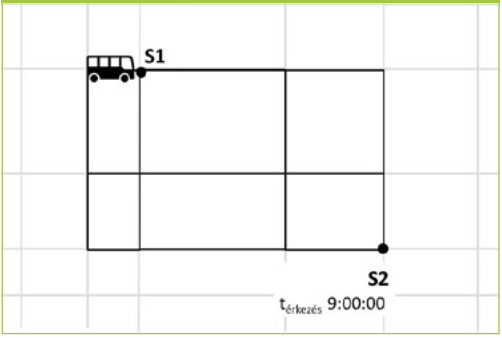
Az útvonaltervezés első lépéseként meghatározzuk a „*k*" db legkedvezőbb útvonalat (p=1,2…$k \in K$) az általános ellenállások $(C_g^1, C_g^2 \ldots C_g^k)$ alapján. Ezt követően meghatározzuk, hogy ezeken az útvonalakon mennyi idő alatt érhető el a következő állomás. Kizárjuk azokat az útvonalakat, amelyeken a következő állomás nem érhető el a megadott limitidőre (t_{limit}). A megmaradó útvonalak közül a legkisebb ellenállásút választjuk: $\min(C_g^p)$, $p \in (P \cap K)$; $C_t^p \leq t_{limit}$). Ha a „*k*" db legkisebb ellenállású útvonal mindegyikén $C_t^p > t_{limit}$, vagyis egyiken sem lehet elérni a következő állomást a megadott időre, akkor egy újabb útvonalkeresést hajtunk végre, ezúttal kizárólag az eljutási idő alapján. Ebben az esetben a leggyorsabb útvonalat választjuk ki: $\min(C_t^p)$, $p \in P$.

3. ALKALMAZÁSI PÉLDA

A fentiekben egy olyan útvonaltervezési metódus került bemutatásra, amelyben a járműveknek egy menetrendet kell teljesíteniük, meghatározott állomásokkal, érkezési, illetve indulási időpontokkal, figyelembe véve az aktuális eljutást befolyásoló tényezőket. Ebből következően a teljes utat fel lehet bontani az egymást követő megállók közötti egyenértékű részutakra. A legfontosabb kitétel, hogy egyetlen állomást sem lehet korábban elhagyni, mint az a menetrendben szerepel, a feladási pontra időben érkező áru vagy utas nem késheti le a járművet. Ha a jármű esetleg korábban érkezik az állomásra, az nem jelent akkora gondot, mivel ott ki tudja várni a menetrend szerinti indulás idejét.

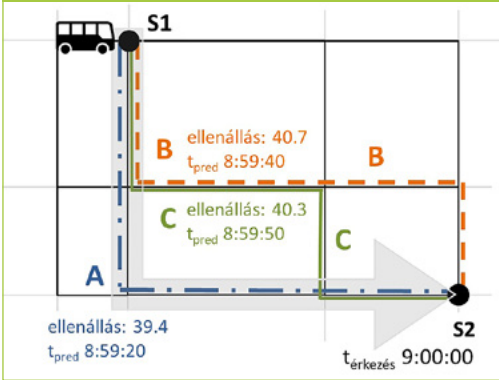
Minden megállópár között előre definiálunk egy alhálózatot, a példában S1 megállótól S2 megállóig. A jármű az S1 megállótól indul és lekérdezi az ellenállásokat az alhálózat útszakaszain, lásd a fekete szakaszokat a 2. ábrán.

2. ábra: A monitorozott alhálózat



A „*k*” db (esetünkben három) legkisebb ellenállású utat kiszámítjuk, majd meghatározzuk a hozzájuk tartozó érkezési időket. A három változatot jelöli a 3. ábrán az A (pontvonal), a B (szaggatott vonal) és a C (folytonos vonal) változat.

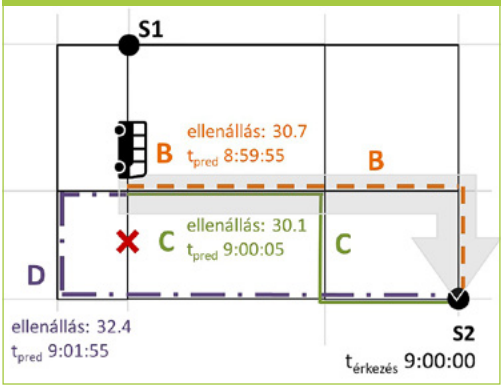
3. ábra: Első útvonalválasztás



Ahogy a jármű megközelíti a következő csomópontot, újra kiszámítjuk a három legkedvezőbb útvonalat az alhálózatban. A példában az előzőleg választott útvonalon történik egy incidens, amelyet a 4. ábrán egy piros „x” jelöl. Ilyen módon az A jelű útvonal már nincs a legjobb három választás között. Az újonnan választott három útvonal B (szaggatott vonal), C (folytonos vonal) és D (pont-vonal), amelyek közül egyedül a B garantálja, hogy időben elér a jármű a megállóhoz, így ezt választjuk, még akkor is, ha nem ennek a legalacsonyabb az ellenállása. Érdemes megfigyelni, ahogy a jármű egyre jobban megközelíti az S2 megállót, az ellenállás értéke egyre csökken, mivel mind a hátralévő távolság, mind a hátralévő idő csökken (3. és 4. ábra).

Ezt követően a jármű a B útvonalon kezd el haladni, és amikor eléri a következő csomópontot, újra kiszámítjuk a három legkisebb ellenállású útvonalat, megvizsgáljuk, hogy ezek közül melyik garantálja a következő megálló időben történő elérését, majd kiválasztjuk ezek közül a minimális ellenállásút. Ha az első útvonalkeresés után nincs a kritériumnak megfelelő választás, akkor egy második útvonalkeresést végzünk el, kizárólag az eljutási idő alapján. A részfolyamat akkor ér véget, amikor a jármű eléri az S2 megállót. A teljes folyamat akkor zárul, amikor a jármű az utolsó megállót is elérte.

4. ábra: Második útvonalválasztás



Az algoritmus használhatóságának mérőfoka, hogy valóban képes-e csökkenteni a járművek által bejárt útvonal ellenállását, miközben időben elérik a megállókat. Referenciaesetnek a hagyományos megközelítést vesszük, amikor a járművek a megállók között fixen ugyanazon az útvonalon haladnak. Az összehasonlítás Matlab szoftveres környezetben történt. Az útvonalkeresés a fenti példában szereplő S1 és S2 megállók között történt a 2. ábra szerinti alhálózaton. A hagyományos megközelítés szerinti, fix útvonalat követő referenciaesetnek a 3. ábra szerinti A jelű útvonalat tekintjük. Üres hálózaton ez eredményezi a legkisebb ellenállást. A szimulációs futások kiindulópontja az S1 megálló volt, és egészen az S2 megálló eléréséig tartottak. Minden futtatás végén összehasonlításra került az algoritmus szerinti ellenállás és eljutási idő a hagyományos útvonalon mért mennyiségekkel. Összesen 1100 db eset vizsgálata történt meg, amelyek közül 990 db rendes körülmények között, míg 110 db incidenses körülmények között zajlott. Az incidens helyszíne a 4. ábrán piros x-szel jelzett szakasz volt minden esetben. A szimulációs forgatókönyvek futtatása során minden egyes útszakaszon a sebesség véletlenszerűen, 3 és 4 m/s között változott. A sebesség megválasz-

tásának alapját a [13] adta, amely a közösségi közlekedési buszok átlagos, torlódásos körülmények közötti sebességét vizsgálta. Incidens esetén a jelzett szakaszon az eljutási idő a megszokott ötszörösére emelkedett. Ahogy a jármű elérte a következő csomópontot, az egyes útszakaszokon a sebesség mindig újragenerált értéket kapott (3 és 4 m/s között). Ezáltal a hálózat állapota dinamikusan változott.

4. SZIMULÁCIÓS EREDMÉNYEK

Az algoritmus teljesítményének vizsgálata három kritérium mentén történt: (1) a szimulációs futtatási időre milyen hatással van, ha a keresett legrövidebb utak számát növeljük, valamint milyen módon változik (2) az ellenállás és (3) az eljutási idő, összehasonlítva a referencia-útvonallal, különböző α és β értékeket alkalmazva (ennek részleteit lásd később).

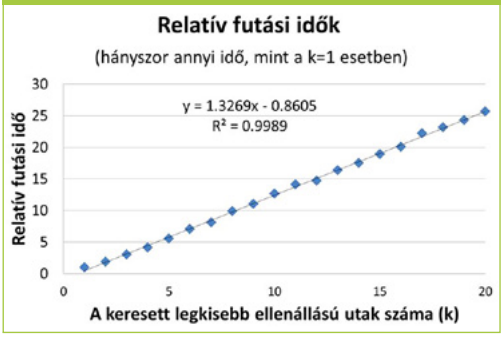
Az egyes forgatókönyveken az útvonaltervezés k=1 és k=20 közötti legrövidebb út kiválasztásával történt, amelyeknek az átlaga adta a későbbi vizsgálatok alapját. Természetesen a szimulációs futási idők függnek a hálózat komplexitásától, így egy adott *"k"* érték eltérő hálózaton eltérő futási időt eredményez. Ilyen módon nem egy abszolút, hanem egy mértékegység nélküli relatív értéket használtunk úgy, hogy a „*k*" legkisebb ellenállású utat figyelembe vevő esetben végzett szimulációs futtatások átlagos futási idejét ($\overline{T_k}$) elosztjuk a k=1 esetben végzett futtatások átlagos futási idejével ($\overline{T_1}$). Utóbbi esetben tehát csak a legkisebb ellenállású útvonalat számítjuk ki. Ezek alapján a *"k"* legkisebb ellenállású útvonalat vizsgáló útvonaltervező algoritmus átlagos relatív futási ideje ($\overline{T_k^r}$) a következőképpen írható fel:

$$\overline{T_k^r} = \overline{T_k} / \overline{T_1} \tag{4}$$

Az átlagos relatív futási idők az 5. ábrán láthatók.

Az átlagos relatív futási idő közel lineárisan függ attól, hogy hány darab legkisebb ellenállású utat keresünk meg. Így a *"k"* érték lényegében önkényesen megválasztható, függően a használt számítógép teljesítményétől. A további vizsgálatokhoz a mintapéldában szereplő k=3 érték került rögzítésre, mivel az alhálózat mérete viszonylag kicsi, emiatt, ha ennél több alternatívával is számolunk, az nem vezetne jobb eredményre. Természetesen, nagyobb hálózatokon van hozzáadott értéke, ha a *"k"* értéket növeljük.

5. ábra: Átlagos relatív futási idők különböző k étékek esetén



Ahogy arról már szó esett, az ellenállásfüggvényt a felhasználó választhatja meg, függően a saját preferenciáitól, hogy ő maga milyen tényezőket tart fontosnak. Az ellenállásfüggvényt a következők szerint határozzuk meg:

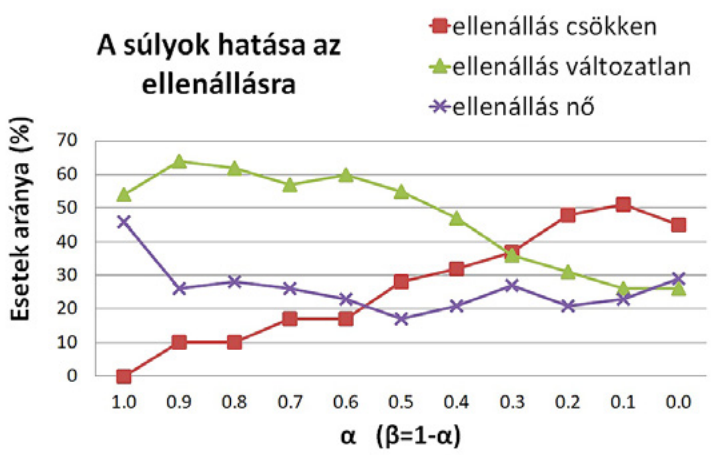
$$C_i = \alpha l_i + \beta t_i, \tag{5}$$

ahol C_i az i-edik útszakasz ellenállása, l_i és t_i az i-edik útszakasz hossza és a rajta aktuális eljutási idő, míg α [1/*m*] és β [1/*s*] a távolság és az eljutási idő súlyozó faktorai. A példában a faktorok között az alábbi összefüggés áll fenn:

$$\alpha + \beta = 1. \tag{6}$$

Ez utóbbi összefüggés nem kötelező, α és β egymástól teljesen függetlenek is lehetnek, azonban érdemes valamilyen megszorítással élni. Ha például szimultán emelnénk mindkét értéket, annak nem lenne semmilyen hatása az útvonalválasztásra. Az alkalmazási példában az útszakaszok hossza (l_i) méterben számított, míg az eljutási idők (t_i) másodpercben. Incidensmentes körülmények között a sebesség nagyobb, mint 1 m/s. Ez azt jelenti, hogy ha α és β értékei egyeznek, akkor a mérőszámokat mértékegység nélkül tekintve

6. ábra: A súlyozó paraméterek hatása az ellenállásra (változó és fix útvonal összehasonlítása)



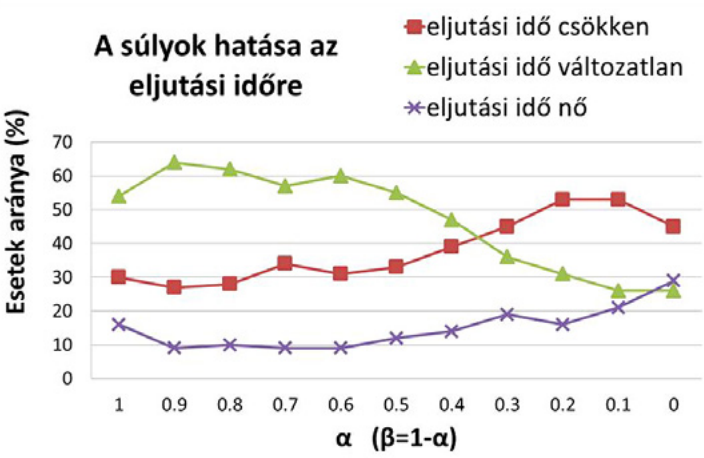
$\alpha l_i > \beta t_i$, vagyis az útvonal hosszának erősebb hatása lesz a teljes ellenállásra. Ahogy α értéke csökken és β értéke nő, úgy lesz az eljutási idő egyre dominánsabb az útvonaltervezés során. Érdemes megjegyezni, hogy az eljutási idő kapcsolatban áll az útszakasz hosszával, azonban ez a legtöbb figyelembe vehető mutatóra igaz, például az emisszióra, a megtett úttal arányos útdíjra stb.

Megjegyzendő továbbá, hogy a számítási időt csak minimálisan befolyásolja az alkalmazott költségfüggvény. A futási idők analízise szerint a költségfüggvény számítása a teljes folyamatnak megközelítőleg 5%-át teszi ki. Ez az időigény akkor sem emelkedik számottevően, ha bonyolultabb költségfüggvényt alkalmazunk. A futási idő legnagyobb részét, átlagosan 81,5%-át a Yen-algoritmus alkalmazása igényli. A maradék 13,5%-ot a szoftver belső folyamatai teszik ki.

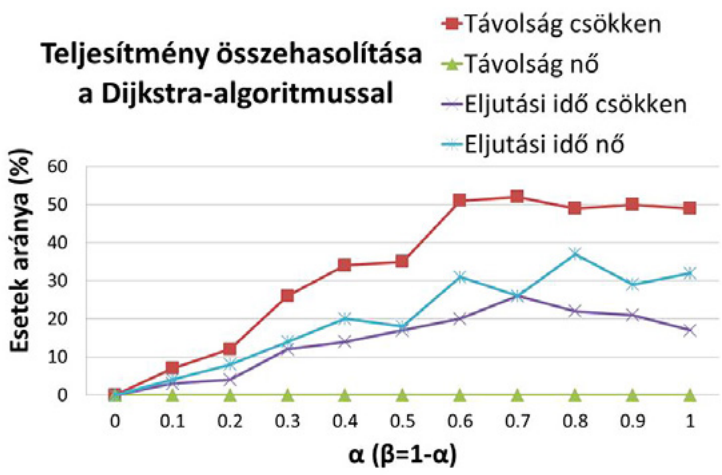
Az α és β súlyok ellenállásra és eljutási időre gyakorolt hatását úgy állapítjuk meg, hogy a jármű által ténylegesen befutott út ellenállását és eljutási idejét összehasonlítjuk a fix útvonal hasonló menynyiségeivel. A számítások során minden esetben az éppen aktuális periódusban érvényes eljutási időkkel számolunk. Az előre definiált fix útvonal és az algoritmus által javasolt dinamikusan változó útvonal ellenállását összehasonlítva kapunk képet arról, hogy az algoritmus jobb, rosszabb vagy azonos eredményt ad a fix útvonalhoz képest.

Összesen 1100 szimuláció került lefuttatásra, az α és β súlyok lehetséges kombinációival egyenként 100, α=1 és β=0-tól α=0 és β=1-ig. (Lásd a 6. és a 7. ábrát.) Mind az ellenállás, mind az eljutási idő az α=0,9 és β=0,1 esetekben csökkent a legnagyobb mértékben, a

7. ábra: Súlyozó paraméterek hatása az eljutási időre (fix és változó útvonal összehasonlítása)



8. ábra: Teljesítmény összehasonlítása a Dijkstra-algoritmussal



szimulációk valamivel több, mint 50%-ában az eredmény javult, igaz, az esetek majdnem 20%-ában romlott a fix útvonalas referenciaesethez képet. (Lásd ismét a 6. és 7. ábrát.)

Konklúzióként kijelenthető, hogy α=0,5 és β=0,5 értékek választása a megfelelő kompromisszum, mivel az esetek közel 30%-ában mind az ellenállás, mind az eljutási idő csökken, miközben a teljesítmény csak 10%-ban romlik. Ugyanakkor a paraméterek és értékeik megválasztása mindig függ az aktuális hálózat jellegétől (pl. topológia, környezetvédelmi normák).

Az algoritmus teljesítménye összevetésre került a metódus alapját adó Dijkstra-algoritmussal is, referenciaesetnek az eljutási időre optimalizált változatot tekintve. Az 1100 szimulációs futás során megvizsgáltuk, hogy az eljutási idő és a befutott út hogyan alakult a két algoritmusnál. Az eredmények a 8. ábrán láthatók.

Ahogy α értéke nő (α=0,6-ig), egyre nő azon futások aránya, ahol a jármű rövidebb útvonalon haladt, mint a referencia Dijkstra-algoritmus esetében. Mivel a referencia Dijkstra-algoritmus kizárólag az eljutási idő alapján optimalizál, míg a cikkben bemutatott algoritmus a távolságot is figyelembe veszi, a jármű sosem halad hosszabb útvonalon, mint a referenciaesetben. Megjegyzendő, hogy az α=0 és β=1 eset pontosan egyezik a referencia Dijkstra-esettel.

Az eljutási időkben csak akkor tapasztalható különbség, ha a két algoritmus eltérő útvonalat javasol, ami az optimalizáció jellege miatt egyben azt is jelenti, hogy a cikkben bemutatott algoritmus rövidebb útvonalat javasol. Az eljutási idő ezen esetek valamivel kevesebb, mint felében csökken, a többiben emelkedik. Megjegyzendő, hogy az útvonal választásakor a jövő nem látható előre pontosan, így előfordulhatnak a hálózaton olyan véletlenszerű incidensek, amelyek miatt a bemutatott algoritmus – bár nem tisztán eljutási időre optimalizált – végül mégis gyorsabb útvonalon juttatja el a járművet a célba, mint a kifejezetten az eljutási időre optimalizált Dijkstra-algoritmus. Természetesen, amikor a jármű a tervek szerint nem éri el időben a következő állomást, akkor a bemutatott algoritmus tisztán idő alapon optimalizál, vagyis a referencia Dijkstra-algoritmust alkalmazza.

5. KONKLÚZIÓ ÉS TOVÁBBI KUTATÁSI IRÁNYOK

A bemutatott kétlépcsős útvonaltervezési metódus a „*k*”–legrövidebbút-keresési eljárást használja. Az algoritmus olyan járművek esetében alkalmazható, amelyeknek meghatározott állomásokat kell elérniük meghatározott időben. Csak a megállási pontok fixek, a köztük befutott útvonal nem előre meghatározott, hanem a hálózat állapota alapján dinamikusan változtatható. A bemutatott útvonaltervező algoritmus az esetek kb. 30%-ában jobb

teljesítményt nyújtott ahhoz képest, mintha a jármű mindig előre meghatározott útvonalon haladna. A konkrét teljesítményt az ellenállásfüggvény határozza meg, amelyet a felhasználó a saját igényei szerint testre szabhat különböző paramétereket alkalmazva, pl. az emisszió vagy a futott kilométerek száma.

A módszer alkalmazható a közösségi közlekedésben és az áruszállításban is, gyárakban vagy raktárakban, de akár igényvezérelt közlekedésben is. Az algoritmus elsősorban az autonóm járművek részére került kifejlesztésre, de azt a járművezetők is alkalmazhatják, ha megfelelő navigációt kapnak egy fedélzeti készülékről, ami egy okostelefon is lehet.

A hagyományos tömeggyártó rendszereket és személyszállítási rendszereket egyre nagyobb arányban helyettesítik olyan rendszerekkel, amelyek az aktuális, valós egyéni igényekre reagálnak [20]. A rugalmasság alapvető követelménnyé vált, amelyhez elengedhetetlen a valós idejű, a körülmények változására azonnal reagáló döntéshozás. Mindez technológiai oldalról nagy mennyiségű, autonóm számítási kapacitást igényel. Egy másik trend, hogy mind a logisztikában, mind a közösségi közlekedésben az útvonaltervezés új értelmet nyer azáltal, hogy az optimális út keresése egy többszintű optimalizálási feladat, amelyben az útvonaltervezés más döntésekkel együtt végzett tevékenység [17]. Ilyenek például a gyártási körülmények: sok esetben, egy gyárban többféle terméket is előállítanak, amelyek termelési folyamatait egymáshoz kell igazítani. Egy következő kérdés a flottamenedzsment, amely az optimalizáció egy másik nézőpontja, és az útvonaltervezésen felül egyéb optimumkritériumokat is teljesíteni kell. Az egyes műveletek elvégzési ideje, valamint a megállókban történő várakozások szintén lehetnek ilyen kritériumok, amelyekkel a bemutatott útvonaltervező algoritmus bővíthető.

Az algoritmus továbbfejlesztési irányaiként a következő kutatási területek tekinthetők.

- Először is, két állomás között nem feltétlenül szükséges minden esetben a teljes alhálózatot monitorozni. Amíg a jármű a kiválasztott útvonalon haladva tudja tartani a szintidőt és az egyéb ellenállásértékek is a limit alatt maradnak, elegendő csak az aktuális útvonalat monitorozni.
- Másodszor, az algoritmus eredeti verziójában a második útvonaltervezés, – ha az első útvonaltervezés „k" alternatívája közül egyik sem garantálja a következő állomás elérését időben – egy egyszerű legrövidebbút-keresés. Azonban ehelyett alkalmazható egy második k–legrövidebbút-keresés, ahol az időbeli elérést garantáló alternatívák közül választanánk útvonalat (a legkisebb ellenállásút), egyéb szempontokat is figyelembe véve, nem csak az eljutási időt.
- Harmadszor, ha egy közösségi közlekedési megállóban nincs le- és felszálló utas sem, az a megálló kihagyható.
- Negyedszer, az útvonaltervezés során az útszakaszok mellett a csomópontok jellemzőit is figyelembe lehet venni, például egy nagy ívű balkanyar bevétele több erőfeszítést igényelhet, mint egy kis ívű jobbkanyaré [8].
- Végül ötödször, az útvonaltervező algoritmus kiterjeszthető egy összetettebb gyártási környezetre egy többszintű gyártási optimalizációs feladat részeként.

KÖSZÖNETNYILVÁNÍTÁS

A cikkben szereplő kutatást a Magyar Kormány és az Európai Szociális Alap támogatta (EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén).

FELHASZNÁLT IRODALOM

[1] Alonso-Mora, J., A. Wallar, and D. Rus. 2017. "Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS): 3583-3590. doi:10.1109/ DOI: http://doi.org/dvgf

[2] Apáthy M., S. 2017. "Practical Route Planning Algorithm." Periodica Polytechnica Transportation Engineering 45(3): 133-140. DOI: http://doi.org/dvgg
[3] Collie, B., J. Rose, R. Choraria, and A.K. Wegscheider. 2017. "Reimagined Car: Shared, Autonomous, and Electric Vehicle." BCG report, December 18. letöltve: 2020. január 4-én. https://www.bcg.com/publications/2017/reimagined-car-shared-autonomous-electric.aspx
[4] Diakonikolas, I., and M. Yannakakis. 2009. "Small Approximate Pareto Sets for Biobjective Shortest Paths and Other Problems." SIAM Journal on Computing 39 (4): 1340–1371. DOI: http://doi.org/bdn676
[5] Dijkstra, E.W. 1959. "A Note on Two Problems in Connexion with Graphs." Numerische Mathematik 1(1): 269-271. DOI: http://doi.org/dpvk8c
[6] Disser, Y., M. Müller-Hannemann, and M. Schnee. 2007. "Multi-Criteria Shortest Paths in Time-Dependent Train Networks." In Proceedings of the 7th international conference on Experimental algorithms (WEA'08), C. C. McGeoch (Ed.). Springer-Verlag, Berlin, Heidelberg, pp. 347–361. DOI: http://doi.org/fb993d
[7] Duckham, M., and L. Kulik. 2003. "'Simplest' Paths: Automated Route Selection for Navigation." In Lecture Notes in Computer Science (2825):169-185, DOI: http://doi.org/fbm4pd
[8] Eklund, P.W., S. Kirkby, and S. Pollitt. 1996. "A dynamic multi-source Dijkstra's algorithm for vehicle routing." Australian New Zealand Conference on Intelligent Information Systems. Proceedings. ANZIIS 96 pp. 329-333. DOI: http://doi.org/dd45m8
[9] Fu, L., D. Sun, and L. R. Rilett. 2006 "Heuristic shortest path algorithms for transportation applications: State of the art." Computers & Operations Research 33(11): 3324–3343. DOI: http://doi.org/dbws2s
[10] Hoang, V-D., and K-H. Jo. 2015. "Path planning for autonomous vehicle based on heuristic searching using online images." Vietnam Journal of Computer Science 2(2): 109-120. DOI: http://doi.org/dvgn
[11] Hawas, Y.E., and H. El-Shayed. 2015. "Autonomous real time route guidance in inter-vehicular communication urban networks." Vehicular Communications 2(1): 36–46. DOI: http://doi.org/dvgp
[12] Jadaan, K., H. Khreis, and Á Török. 2018. "Exposure to Traffic-related Air Pollution and the Onset of Childhood Asthma: A Review of the Literature and the Assement Methods Used." Periodica Polytechnica Transportation Engineering, 46(1): 21-28. DOI: http://doi.org/dvgq
[13] Oskarbski, J., K. Birr, M. Miszewski, and K. Zarski. 2015. "Estimating the Average Speed of Public Transport Vehicles Based on Traffic Control System Data." Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest. DOI: http://doi.org/dvgr
[14] Pala, M., N. O. Eragi, F. López-Colino, A. Sanchez, A. de Casrto, and J. Garrido. 2013. "HCTNav: A Path Planning Algorithm for Low-Cost Autonomous Robot Navigation in Indoor Environments." ISPRS International Journal of Geo-Information 2(3): 729-748. DOI: http://doi.org/gchs7q
[15] Parulekar, M., V. Padte, T. Shah, K. Shroff, R. Shetty, 2013. "Automatic Vehicle Navigation using Dijkstra's Algorithm." International Conference on Advances in Technology and Engineering (ICATE), Mumbai. pp. 1-5. DOI: http://doi.org/dvgt
[16] PostBus. "Project 'SmartShuttle': Shape the mobility of the future." letöltve: 2020. január 4-én. https://www.postauto.ch/en/project-smartshuttle-0.
[17] Speranza, M. G. 2018. "Trends in Transportation and Logistics." European Journal of Operational Research 264 (3): 830–836. DOI: http://doi.org/gfgkmc
[18] Storandt, S. 2012. "Algorithms for vehicle navigation." PhD dissertation, Universität Stuttgart.
[19] Szalay, Z., T. Tettamanti, D. Esztergár-Kiss, I. Varga, and C. Bartolini. 2018. "Development of a Test Track for Driverless Cars: Vehicle Design, Track Configuration, and Liability Considerations." Periodica Polytechnica Transportation Engineering. 46(1): 29-35. DOI: http://doi.org/dktg
[20] Tavasszy, L., K. Ruijgrok, and I. Davydenko. 2012. "Incorporating Logistics in Freight Transportation Models: State of

the Art and Research Opportunities." Transport Reviews 32 (2): 203–219. DOI: http://doi.org/fxzmms
[21] Taxi trip data, New York. "NYC OpenData: Yellow Taxi Trip Data." 2014. letöltve: 2020. január 4-én. https://data.cityofnewyork.us/view/gn7m-em8n.
[22] Wien. "Selbstfahrender Bus in der Seestadt unterwegs", letöltve: 2020. január 4-én. https://www.wien.gv.at/verkehr/oeffentlich/selbstfahrender-bus.html
[23] Yen, J.Y. 1970. "An algorithm for finding shortest routes from all source nodes to a given destination in general networks." Quarterly of applied mathematics 27(4): 526-530. DOI: http://doi.org/dvgw
[24] Zhang, R., F. Rossi, and M. Pavone. 2016. "Routing Autonomous Vehicles in Congested Transportation Networks: Structural Properties and Coordination Algorithms." In: Proceedings of Robotics: Science and Systems, AnnArbor, Michigan, DOI: http://doi.org/dvgx

Multi-criteria dynamic route planning for scheduled autonomous vehicles

The possibilities for using autonomous or highly automated vehicles are mainly the areas where certain predefined points need to be touched in a given order. Such examples are haulers or public transport vehicles. Typically, these vehicles reach their stations on a predetermined route based on certain criteria, such as the shortest distance, often regardless of the current traffic status on the road network. An illustrative example of this is the presentation of a route planning method – primarily for highly automated vehicles – that navigates the vehicle to its destination on the prevailing most favourable route between predefined stations. The methodology is presented through the example of public transport vehicles, but can also be applied generally, for example in a factory or a warehouse.

Multikriterielle dynamische Routenplanung mit mehreren Kriterien für zeitplanmässig fahrende autonome Fahrzeuge

Die Möglichkeiten zur Verwendung autonomer oder hochautomatisierter Fahrzeuge sind hauptsächlich die Bereiche, in denen bestimmte vordefinierte Punkte in einer bestimmten Reihenfolge berührt werden müssen. Solche Beispiele sind Spediteure oder Fahrzeuge des öffentlichen Verkehrs. In der Regel erreichen diese Fahrzeuge ihre Stationen auf einer vorgegebenen Route, die auf bestimmten Kriterien basiert, z. B. auf der kürzesten Entfernung, häufig unabhängig vom aktuellen Verkehrsstand im Straßennetz. Ein anschauliches Beispiel hierfür ist die Darstellung einer Routenplanungsmethode - hauptsächlich für hochautomatisierte Fahrzeuge -, die das Fahrzeug auf der aktuell günstigsten Route zwischen den vordefinierten Stationen an sein Ziel bringt. Ein anschauliches Beispiel hierfür ist die Darstellung einer Routenplanungsmethode - hauptsächlich für hochautomatisierte Fahrzeuge -, die das Fahrzeug auf der aktuell günstigsten Route zwischen vordefinierten Stationen an sein Ziel bringt. Die Methodik wird am Beispiel von Fahrzeugen des öffentlichen Verkehrs vorgestellt, kann aber auch allgemein, beispielsweise in einer Fabrik oder in einem Lager angewendet werden.