EGY HEURISZTIKUS ÚTVONALTERVEZŐ ALGORITMUS TÖBBNAPOS TÚRÁK TERVEZÉSÉRE[1]

APÁTHY M. SÁNDOR
Budapesti Corvinus Egyetem

Az útvonaltervező algoritmusok megalkotói az utazó ügynök probléma óta hagyományosan a csúcsokban gyűjthető profitok összegét tekintik az optimalizálandó célfüggvénynek, ezzel azonban figyelmen kívül hagynak jó pár gyakorlati megfontolást, éppen ezért ritkán vezetnek jó eredményre. Ennek fényében olyan hasznossági függvény és célfüggvény kerül jelen dolgozatban bemutatásra, mely a korábbi pontösszeg-maximalizálás egy kiterjesztéseként értelmezhető, hiszen a paraméterek bizonyos értékei mellett visszakapjuk azt, mégis képesek figyelembe venni a felhasználók igényeit is. Többnapos túraútvonalak tervezéséhez olyan heurisztikus algoritmust alkottunk, melynek célja, hogy egyszerűségével, és ebből adódóan rövid futási idejével lehetőséget adjon annak későbbi gyakorlati alkalmazhatóságára. A 3-napos útvonalak esetén is átlagosan 4 másodperc alatti eredmény, valamint a célfüggvénynek köszönhető attraktív útvonaltervek megfelelő alapját képezik egy személyre szabott túrautakat tervező alkalmazás megalkotásának, mely a felhasználói elégedettség optimalizálását tartja legfőbb céljának.

Kulcsszavak: Team Orienteering Problem, Route Planning, Heuristic Algorithm, Tourism. *JEL kód:* C60, C61, Z32

1 Bevezetés

Az utazó ügynök probléma óta az útvonaltervező algoritmusok megalkotói hagyományosan a csúcsokban gyűjthető profitok összegét tekintik az optimalizálandó célfüggvénynek, ezzel azonban figyelmen kívül hagynak jó pár gyakorlati megfontolást, éppen ezért ritkán vezetnek jó eredményre. Ilyen gyakorlati megfontolás például, hogy a felhasználó által alacsony értékelést kapott pontokat akkor se vegyük be az útvonaltervbe, ha azok igen kis költséggel megtehetőek, vagy éppen az, hogy igyekezzünk fajlagosan a lehető legtöbb időt a helyszínek meglátogatásával tölteni (a gráf élein történő séták helyett). Ennek fényében olyan hasznossági függvényt és célfüggvényt javasolunk, mely a korábbi pontösszeg-maximalizálás egy kiterjesztéseként értelmezhető, hiszen a paraméterek bizonyos értékei mellett visszakapjuk azt, mégis figyelembe veszik a felhasználók igényeit is.

A kutatásainkhoz kapcsolódó korábbi eredményeket a 2. szakaszban foglaljuk össze, melyet követően a 3. szakaszban megfogalmazásra kerül a Team Orienteering Problem (TOP) egy módosított változata, mely mind a korlátok

[1]Beérkezett: 2016. szeptember 16. E-mail: sandor.m.apathy@gmail.com.

kezelését tekintve, mind a célfüggvény megalkotását illetően a gyakorlati probléma minél életszerűbb leképezésére koncentrál. A feladatra adott heurisztikus algoritmus célja az volt, hogy egyszerűségével, és ebből adódóan rövid futási idejével lehetőséget adjon annak későbbi gyakorlati alkalmazhatóságára. Eredményeinket a 4. szakaszban ismertetjük: a 3-napos útvonalak esetén is átlagosan 4 másodperc alatti eredmény, valamint a célfüggvénynek köszönhető attraktív útvonaltervek megfelelő alapját képezik egy személyre szabott túrautakat tervező alkalmazás megalkotásának, mely a felhasználói elégedettség optimalizálását tartja legfőbb céljának. A tanulmány záró szakaszában megállapítjuk következtetéseinket, valamint kijelöljük a kutatása lehetséges jövőbeli irányait.

2 Kapcsolódó kutatások

Az egyik első útvonaltervező alkalmazás az utazó ügynök probléma (*Traveling Salesman Problem*, röviden *TSP*), melyet először az 1930-as években Karl Menger formalizált, és adott rá megoldást [1], de az elnevezés Hassler Whitney-től származik [2]. Lényege, hogy az ügynöknek adott telephelyeket kell felkeresnie, és dönteni csak arról tud (az élköltségek ismeretében), milyen sorrendben teszi ezt, hogy a lehető legkisebb költséggel járja körbe a telephelyeket. Tehát minimális összköltségű Hamilton-kört keresünk a gráfon. Birkhoff [3] munkájának köszönhetően lehetővé vált a hozzárendelési feladatok megoldása lineáris programozási feladatként, melyet Dantzig, Fulkerson és Johnson alkalmazott elsőként a TSP megoldására [4].

Az utazó ügynök problémából fejlődött ki az *Orienteering Problem (OP)*, vagy más néven a *Selective Traveling Salesman Problem (STSP)*, ahol az egyes ügyfelekhez már profitot rendelnek, és az ügynököt szorító időkorláton belül a legnagyobb összprofitot kell begyűjtenie az útja során az ügyfelek meglátogatásával. Az elnevezés 1996-ban Chao et al. [5] cikkében szerepel, de már 1984- ben megjelent Tsiligirides-nél [6], ahol a TSP-ben az ügynöknek nincs elég ideje, hogy az összes várost meglátogassa egyedül. Cikkében olyan sztochasztikus algoritmust alkalmaz az optimális útvonal közelítő megoldására, melyben minden iterációban *Monte-Carlo-módszerrel* keresi a következő csúcsot, a távolság és a begyűjthető profit függvényében. A problémát már formalizálta Kataoka és Morito 1988-ban [7], ám ők még *Maximum Collection Problem* néven hivatkoztak rá. A témáról bővebben Feillet et al. összefoglaló cikkében olvashatunk [10].

Már a kezdetektől ismert volt ennek a technikának a természetjárásban és általában a turizmusban való alkalmazhatósága, hiszen az OP elnevezés is a tájfutásból ered, ahol a versenyzőknek egy térkép és egy iránytű segítségével kell felkeresniük az előre kijelölt pontokat a lehető legrövidebb időn belül. Innen datálható a tudományág sport és turizmus területén történő hasznosítása, és terjedt ki nem csak a természetjárásra, de a városnézésre is. Ennek jó példája Wang et al. [8], ahol a turista a legérdekesebb látványosságokat látogatja végig a szállodából indulva, és a nap végén oda tér vissza. Golden,

Levy és Vohra megmutatták, hogy az OP NP-nehéz [11], így az erre adott egzakt megoldás csak viszonylag kis számú csúcs esetén lehetséges. Ramesh et al. [13] branch-and-bound algoritmust használ, mellyel egzakt megoldást ad akár 150 csúcsot tartalmazó gráfra is, míg Fischetti et al. [14] cikkükben *brach-and-bound* eljárással akár 500 csúcsra is egzakt megoldást tudnak adni. Ramesh és Brown [15] 4 fázisból álló heurisztikus megoldást adnak az OP-re, melyben az *2-opt* és *3-opt* eljárásokat alkalmazzák, melyek a *local search* algoritmusok családjába tartoznak, melyekről Lin cikkében olvashatunk bővebben [33]. Ennél jobb eredményeket ad Chao et al. [22] 5 lépésből álló megoldása, mely mohó algoritmust, sztochasztikus eljárást és *2-opt* algoritmust ötvözve építi fel az útvonalat. A fenti heurisztikus megoldások egy komoly hátránya, hogy könnyen be tudnak ragadni egy lokális optimumba, melyet Gandreau et al. [16] *tabu search* megoldása hatékonyan hidal át. Mivel az eredmények turisztikában történő felhasználása igen nagy figyelmet kap, így cikkek sora foglalkozik azok térinformatikai beágyazásával is (mobil applikációk formájában), erre jó példát találunk az OP esetére Souffriau et al. 2008-as cikkében [9].

Az OP egy természetes kiterjesztése a *Team Orienteering Problem (TOP)*, ahol a turista „feladata", hogy P nap alatt a rendelkezésére álló időben a lehető legtöbb (számára érdekes) látványosságot meglátogasson, és minden nap végén visszatérjen a szállodájába, ez igen hasonlít a *Vehicle Routing Problem with Time Windows-ban (VRPTW)* megfogalmazott feladathoz. Ezt először Butt és Cavalier formalizálta 1994-ben [12], ahol egy toborzási feladat megoldására alkalmazták. A TOP megfogalmazását a *3. mellékletben* találjuk. Az egzakt megoldások közül igen hatékonyan működnek az oszlop generáló algoritmuson [19] alapuló eljárások. Ekkor LP feladatként oldjuk meg a problémát, de redukáljuk a dimenziók számát a gyorsabb futási idő érdekében, melyre jó példa Butt és Ryan 1999-es cikke [20], ahol akár 100 csúcsra is egzakt megoldást kaphatunk viszonylag rövid idő alatt. Később Boussier et al. [21] alkalmazta az oszlopgeneráló algoritmust, de már kombinálva a *branch-and-bound* eljárással, hogy javítsanak az algoritmus teljesítményén. A heurisztikus megoldások közül a legkorábbi a már az OP kapcsán ismertetett Chao et al. [22] cikkében szereplő 5 lépcsős eljárás kis átalakítással: itt az első P legjobb utat listázzuk ki eredményül. Tang és Miller-Hooks [23], valamint Archetti et al. [24] is *tabu search* eljárást alkalmaz az TOP megoldására, míg Ke et al. [25] *hangya kolóniák* módszerét javasolja cikkében. Az első lépésben 4 eljárást is teszteltek, amivel egy megvalósítható eljáráshoz lehet jutni. Közülük az utakat szekvenciálisan felépítő algoritmus bizonyult a leghatékonyabbnak. Az egyes iterációkban elkészült megoldást *2-opt* algoritmussal javítják, majd kiegészítik annyi csúccsal, amennyi az időkorlátba belefér. Vansteenwegen et al. két heurisztikus eljárást is kifejlesztettek. Mind a *Guided Local Search (GLS)* [26], mind a *Skewed Variable Neighborhood Search (SVNS)* [27] eljárások ugyanazokon a lépéseken alapulnak: egy kezdeti eljárásbók kiindulva „gyengébb" útszakaszokat törlünk, illetve kisebb útszakaszokat illesztünk össze, majd az így kapott út összprofitját igyekszik javítani cserékkel, illetve a menetidőket csökkenteni, és új pon-

tokat beilleszteni a megtakarított idő terhére. Az SVNS más sorrendben variálja ezeket a lépéseket, és így jóval megelőzi a GLS-t. Az útvonaltervező algoritmusokról bővebb összefoglalót Vansteenwegen et al. [17] cikkében olvashatunk, ahol külön kitérnek az egyes eljárások számítási igényére is.

3 Az útvonaltervező algoritmus

Az alábbiakban bemutatásra kerül a tanulmány magját képező útvonaltervező algoritmus. Elsőként bemutatjuk a tervezéshez felhasznált adatokat, majd egy saját, merőben új célfüggvényt, végül a probléma megfogalmazása után egy heurisztikus eljárást adunk annak megoldására.

3.1 A felhasznált adatok

Az útvonaltervező algoritmus teszteléséhez létrehoztunk egy adatbázist, mely 150 budapesti turisztikai látványosságot tartalmaz az alábbi adatokkal:

- Helyzeti adatok: a látványosságok szélességi és hosszúsági koordinátái, 3 méter pontossággal.
- Az adott hely látogatásához szükséges idő percben
- Az egyes helyszínek költségei (belépő díjak), euróban
- A felhasználó által az egyes helyszínekre adott értékeléseket az útvonaltervezés során adottságnak tekintjük, és feltételezzük, hogy a 3. fejezetben adott eljárás alapján kalkuláltuk, így leírják az adott turista preferenciáit.
- A szálloda (pontosabban annak koordinátái), melyből a turista a nap elején elindul, és a nap végén oda érkezik vissza.

A fentieken túl az OpenStreetMap alkalmazás segítségével, mely tartalmazza a város teljes útvonalhálózatát, kiszámoltuk az összes pont többitől vett távolságát, melyet egy távolság mátrixban foglaltunk össze. Ennek a_{ij} eleme az i pontból a j pontba való leggyorsabb eljutásához szükséges időt jelenti (másodpercben). A két pont közötti legrövidebb utat Dijkstra-algoritmussal számoltuk. Így tehát a várost egy olyan gráffal modellezzük, melynek csúcsai a meglátogatható látványosságok (ide értve a fix szállodát is), valamint az azokat összekötő, időben legrövidebb utak, mint a gráf élei. Az élek költségei az él kezdő- és végpontja közötti menetidők, a csúcsokban pedig a látogatások során gyűjthető profitok (a turista adott csúcsra vonatkozó értékelései), valamint a csúcsnál eltöltendő idők és belépődíjak jelentik a költségeket.

Fontos tisztázni e ponton, hogy az élköltségek számításakor mindvégig gyalogos menetidőkkel kalkuláltunk, így maguk a tervezett útvonalak is végig gyalogos túrákat hivatottak modellezni. Autós vagy tömegközlekedési eszközt is figyelembe vevő algoritmus esetén a távolságmátrixot kalkulálhatjuk az

OpenStreetMap vagy a GoogleMaps által becsült menetidőkkel, ám ezek nagyban függnek az aktuális forgalomtól, illetve tömegközlekedés esetén a menetrendtől is. Gavalas et al. [34] athéni helyszíneket és tömegközlekedést modellező kutatásukban klasztereken alapuló heurisztikus eljárásukat tovább fejlesztve 3 algoritmust is adnak a *Time dependent Team Orienteering Problem with Time Windows (TDTOPTW)* közelítő megoldására, melyek az időablakok mellett kezelni tudják az időben változó útiköltségeket és a tömegközlekedési menetrendet is. Az eljárásaik hátránya, hogy nem veszik figyelembe az újabb csúcsok útvonalba történő beillesztésénél a következő helyszín várakozási idejében okozott változást, mikor a beillesztésről döntenek.

3.2 A turista célfüggvénye

A turisták maximalizálandó célfüggvényéről azért érdemes szót ejteni, mert még a legfrissebb és igen haladó megközelítésekben is, lásd Gavalas et al. [19], vagy Vansteenwegen et al. [27], a feladat nem más, mint a TSP-ben is meghatározott csúcsoknál gyűjthető profitok összegének maximalizálása. Ennek megértése érdekében egy pillanatra tegyük fel, hogy a csúcsoknál begyűjthető profitok lehetséges értékei legyenek az $[1, 10]$ intervallumba eső egész számok. Ekkor, ha egy meglátogatott ponthoz igen közel eső, de kis profitú pont meglátogatása mégis jó ötletnek tűnik, hiszen annak az útvonalba történő beillesztése nagy fajlagos profittal kecsegtet. Azonban a gyakorlati problémára koncentrálva ez mégsem jó ötlet, hiszen egy 10-es skálán 2-esre értékelt látnivaló általában nem nevezhető a turista ízlésvilágával összeegyeztethetőnek. Ez a megközelítés még a TSP megfogalmazása óta része az útvonaltervező algoritmusoknak, ahol pénzben mérhető profitról lévén szó, összeadható volt, és reális elvárás, hogy az összprofitot maximalizáljuk. Helyszínekre adott értékelések esetén azonban ez már nem igaz. Javasoljuk tehát, hogy ne „pontgyűjtő akcióként" kezeljük a feladatot, és ennek érdekében egy olyan célfüggvényt alakítsunk ki, mely igyekszik garantálni a felhasználót leginkább kielégítő útvonal megtervezését. Az alábbi megfontolásokat tesszük a célfüggvény megalkotása során:

- A turistát a kialakult pontok alapján kevésbé érdeklő nevezetességeket töröljük a listából. Ez egyrészt csökkenti a feladat számítási igényét, másrészt garantálja, hogy csak valóban személyre szabottan kurrens helyszíneket veszünk számításba. A továbbiakban jelöljük s^*-gal azt a minimális értékelést, amit el kell érnie egy csúcsnak a bent maradáshoz, ellenkező esetben töröljük a gráfból.

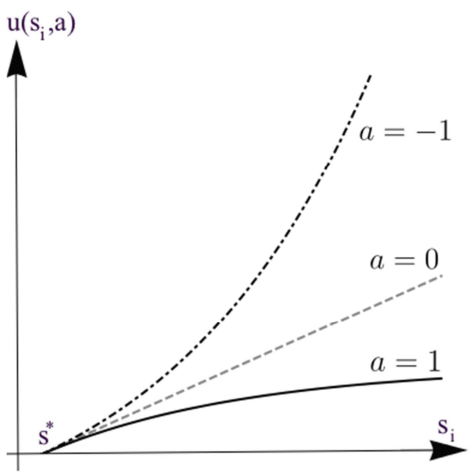
- A lehető legtöbb időt töltse a turista a helyszíneken, tehát igyekezzünk minimalizálni a csúcsok közötti közlekedésre fordított időt. Ezt egy α paraméterrel építjük be a célfüggvénybe: minél érzékenyebb erre a turista, annál nagyobb büntetést számol fel a látványosságok közötti távolságok megtételéért.

- A turisták különbözhetnek egymástól sétára vonatkozó hajlandóságukban is. Céltalanul hosszú utakat (két pont között) az egyéni prefe-

renciáktól függően sújtsuk külön büntetéssel. Úgy vélem, kevés turista örülne egy két órás sétának két helyszín között. Amennyiben lehetséges, vegyünk fel egy látogatható pontot a hosszabb utakat megtörve. Ennek érdekében a célfüggvényben ne az utazással töltött idők összegét szerepeltessük, hanem azoknak egy 1-nél nagyobb hatványát (β „lustasági" paraméter), és azokat adjuk össze. Ezzel büntetjük az útvonaltervezésben indokolatlanul hosszú utak felvételét.

- Ne hagyatkozzunk pusztán az egységnyi összköltségre (menetidő + látogatási idő) eső profitra a döntésnél, hiszen így sok időintenzív látnivalót hagyunk ki az útvonaltervezésből: például Párizsban nem javasolnánk meglátogatni az Eiffel-tornyot, mert annak látogatási ideje hozzávetőlegesen 2 óra, míg profitja bár igen magas lehet, de egy kicsivel alacsonyabb profitú pont meglátogatása fél óra alatt nagyobb fajlagos haszonnal kecsegtet. Itt javasoljuk olyan kategória létrehozását, amely az úgynevezett kötelező látnivalókat tartalmazza, melyeket fel kell venni a meglátogatandó helyszínek listájába függetlenül attól, mennyire időintenzívek. Ezek személyre szabottan kerülnek meghatározásra, például az egyén értékelése alapján maximális pontszámot kapott látnivalók lehetnek ezek.

- Legyen az értékelések figyelembevételekor személyre szabható, mennyivel értékel többre az adott felhasználó például egy 9-es értékelésű helyszínt egy 8-ashoz képest. Az általunk javasolt $u(s_i, a)$ hasznosság függvényt úgy alkottuk meg, hogy $a = 0$ mellett az adott látványosság eredeti s_i értékeléseit adja vissza (illetve azok s^* küszöbértékkel csökkentett értékét), míg $a < 0$ esetén progresszíven nő az értékelések hasznossága. Az $a > 0$ esetben csökkenő határhasznossággal bír az értékelés egységnyi növekedése (*1. ábra*).



1. ábra. Hasznosság függvény

$$u(s_i, a) = \begin{cases} \frac{1-e^{-a(s_i-s^*)}}{a} & , \text{ha } a \neq 0 \\ s_i - s^* & , \text{ha } a = 0\,. \end{cases}$$

Az a elméletben $(-\infty; \infty)$ intervallumon bármilyen értéket felvehet, gyakorlati megfontolások alapján $[-2, 2]$ intervallumban vizsgáljuk majd az útvonaltervre gyakorolt hatását. Mivel $u(s^*, a) = 0$, így tehát azok a helyszínek, melyek értékelése küszöbértéken van, 0 profitot hoznak, és csak az ennél jobb értékelés helyszínek jelentenek pozitív profitot, melyek növelik a célfüggvény értékét. Ez függvényforma természetesen csak javaslat, ám a kutatás jelen fázisának eredményei alapján reményt keltő annak alkalmazása.

Ezeket figyelembe véve a célfüggvényünk, mely alapján az újabb pontokat vesszük fel az útvonalba:

$$C(\alpha, \beta, a, R) = \Big(\frac{\sum_{p=1}^{P}\sum_{i=1}^{N}\theta_{ip}v_i}{\sum_{i=1}^{N-1}\sum_{j=2}^{N}\tau_{ijp}t_{ij}^{\beta}}\Big)^{\alpha} \times \Big(\sum_{p=1}^{P}\sum_{i=1}^{N}\theta_{ip}u(s_i, a)\Big)^{1-\alpha}\,,$$

ahol P a rendelkezésre álló napok száma, N a gráf csúcsainak száma, s_i, v_i, t_i rendre a következő pont értékelése, látogatási ideje és az odaút menetideje, $s*$ a felhasználó értékeléseinek azon küszöbértéke, amely alatt nem került be látványosság a potenciálisan látogatható helyszínek közé, α a célfüggvényben szereplő két szempont (a hasznosság és az egységnyi menetidőre eső látogatási idő) súlyozására szolgál, β a „lustasági" paraméter. A τ_{ijp} értéke legyen 1, ha a p-edik útnál az i-edik csúcs után a j-edik következik az úton, és 0 különben. Legyen θ_{ip} értéke 1, ha a p-edik úton az i-edik csúcsot meglátogatják, és 0 különben. A P napra tervezett útvonalak összességét R jelöli. Az a paraméter hivatott tükrözni, mennyivel értékel többre a felhasználó egy s-re értékelt látványosságot egy $(s-1)$-re értékelthez képest.

A választott célfüggvény forma tehát alapvetően két törekvést szolgál: egyrészt az egységnyi megtett útra jutó fajlagos látogatási időt igyekszik növelni, másrészt a legnagyobb hasznossággal bíró csúcsok meglátogatását szorgalmazza. Az α paraméterrel ezek súlyát szabályozhatjuk. Fontos látni, hogy a célfüggvény a mások által széles körben használt profitösszeg-maximalizálás egy kiterjesztése, hiszen $\alpha = a = 0$ választással éppen ezt kapjuk vissza.

3.3 A feladat formalizálása

Legyen adott egy $G(V, E)$ gráf, amelynek minden c_i csúcsához egy s_i nemnegatív értékelés van rendelve, mely a turista számára $u(s_i, a)$ hasznossággal bír, ha meglátogatja a c_i csúcsot. A c_i és c_j csúcsok közötti e_{ij} élhez t_{ij} élköltséget rendelünk, ami a turista számára a távolság megtételéhez szükséges idő gyalogosan. Az c_i csúcs meglátogatása v_i időt vesz igénybe (látogatási idő). Jelölje továbbá h_{ip}, hogy a p-edik útnál az i-edik csúcs hányadik lépésben kerül sorra az úton, valamint τ_{ijp} értéke legyen 1, ha a p-edik útnál az i-edik csúcs után a j-edik következik az úton, és 0 különben. Legyen θ_{ip} értéke 1, ha a p-edik úton az i-edik csúcsot meglátogatják, és 0 különben. A turistának P napja van a látványosságok megtekintésére, és naponta $T_{\max}$ perc ideje.

Az i-edik látványosság megtekintése b_i költséggel jár (belépődíj), melyet a napi B költségkeretéből fedezhet. Fontos, hogy a költségvetési korlát tetszés szerint átcsoportosítható a napok között, így összességében a P napra BP költségkerettel rendelkezik. Ez nem vonatkozik az időkorlátra, mely minden napon betartandó. Minden nap elején a szállodából indul, és a nap végén oda érkezik vissza. A modellben ezt az általánosság jegyében külön kezeljük (az 1-es és N-nel jelölt csúcs), de ezek megegyezhetnek egymással. A feladat: P nap alatt olyan P darab utat bejárni a $G(V,E)$ gráfon, hogy maximalizáljuk a célfüggvény értékét, miközben betartjuk az idő- és költségkorlátokat, és minden csúcs legfeljebb egyszer látogatható meg. Ekkor a feladat megfogalmazható a következőképpen:

$$\max_{\tau_{ijp}} \Big(\frac{\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} v_i}{\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij}^{\beta}} \Big)^{\alpha} \times \Big(\sum_{p=1}^{P} \sum_{i=1}^{N} \theta_{ip} u(s_i, a) \Big)^{1-\alpha} \tag{1}$$

$$\sum_{p=1}^{P} \sum_{j=2}^{N} \tau_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} \tau_{iNp} = P \tag{2}$$

$$\sum_{p=1}^{P} \theta_{kp} \le 1\,, \quad k = 2, \dots, N-1 \tag{3}$$

$$\sum_{j=2}^{N} \tau_{kjp} = \sum_{i=1}^{N-1} \tau_{ikp} = \theta_{kp}\,, \qquad k = 2, \dots, N-1, \quad p = 1, \dots, P \tag{4}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij} + \sum_{i=1}^{N} \theta_{ip} v_i \le T_{\max}\,, \quad p = 1, \dots, P \tag{5}$$

$$\sum_{p=1}^{N} \sum_{i=1}^{N} \theta_{ip} b_i \le PB \tag{6}$$

$$h_{ip} - h_{jp} + 1 \le (N-1)(1 - \tau_{ijp})\,, \quad i, j = 2, \dots, N,\ p = 1, \dots, P \tag{7}$$

$$2 \le h_{ip} \le N\,, \quad i = 2, \dots, N,\ p = 1, \dots, P \tag{8}$$

$$\tau_{ijp}, \theta_{ip} \in \{0, 1\}\,, \quad i, j = 1, \dots, N,\ p = 1, \dots, P \tag{9}$$

Az egyes kifejezések jelentése a következő: (1) a maximalizálandó célfüggvény; (2) minden út az 1-es csúcsnál kezdődik, és az N-ediknél ér véget, (ezek a korábbiak alapján megegyezhetnek); (3) minden csúcsot csak legfeljebb egyszer látogatunk meg; (4) minden út egyenként összefüggő; (5) betartjuk az időkorlátot: a napi látogatási és menetidők összege nem lehet több, mint $T_{\max}$; (6) betartjuk a költségkorlátot: a belépődíjak összege a P napra együttesen nem lehet több BP-nél; (7) és (8) együtt garantálja, hogy ne legyenek körök az útban, Miller–Tucker–Zemlin javaslata alapján [18]; (9) a τ_{ijp} és θ_{ip} értékkészlete 0 vagy 1.

A kitűzött feladatra adott heurisztikus megoldásunkat a következőkben ismertetjük.

3.4 Az útvonaltervezés

Mivel a megoldandó feladatunk megfeleltethető a TOP egy speciális esetének, így az NP-nehéz feladat, vagyis csak igen kis méretű gráf esetén van reményünk egzakt eljárással optimális megoldásra jutni, éppen ezért egy heurisztikus eljárást javasolunk. Célunk gyakorlati megfontolásokon alapszik: egyrészt szeretnénk egy valóságos problémákon alkalmazható eljárást adni, így az eljárás futási idejét szeretnénk alacsonyan tartani, másrészt olyan eljárást keresünk, mely a felhasználók számára kielégítő megoldással szolgál. Ennek köszönhető a rendhagyónak számító hasznosságfüggvényünk is.

Először két olyan eljárást ismertetünk, melyet több ponton használunk majd az algoritmus során:

Lexikografikus rendezés: ennek során mindig egy csúcsokból álló halmazt rendezünk egy másik csúcsokból álló halmaz és az erőforrás keretek (pénz és idő) szűkössége alapján (mely meghatározásának menetét később pontosan ismertetjük). Szükségünk van továbbá arra az s^* küszöbértékre, melynél alacsonyabb értékelésű csúcsokat törölni fogunk a releváns pontok halmazából (lásd az algoritmus első lépése). Formálisan tehát $L(C_1, C_2, sc, s^*)$, aholC_1 a csúcsok azon halmaza, melyet rendezni szeretnénk, C_2 amely halmazhoz rendezzük, sc az erőforrások szűkösségének mértékét állítja sorrendbe (idő vagy pénz), és s^* az értékelések küszöbértéke. Képezzük az alábbi értékeket:

$$\frac{\frac{u(s_i,a)}{u(s^*+1,a)}}{d^*(c_i,C_2)+v_i}\ , \qquad \frac{\frac{u(s_i,a)}{u(s^*+1,a)}}{b_i}$$

ahol $d^*(c_i, C_2)$ a c_i csúcs és a C_2 halmaz közötti átlagos távolságot jelöli. A számláló tehát azt fejezi ki, hogy az adott s_i értékelésű pont hány $(s^* + 1)$ értékelésű pont hasznosságával egyenértékű (a releváns pontok között ugyanis $(s^* + 1)$ értékelésű a minimális, hiszen az s^* értékelésűeket és az annál kisebbeket töröljük a gráfból). Ezt osztjuk a nevezőben a pont felvételének várható költségével, ami az odaút menetideje és a látogatási idő összege. A pénzben kifejezett költségek rendezésekor ugyanezen elv alapján a nevezőben a belépődíj szerepel. A következő lépésben rendezzük a C_1 halmaz csúcsait, először aszerint, amelyik korlát szűkösebb. Ha ez például az idő, akkor a fenti hasznosság/időköltség mutató alapján rendezzük csökkenő sorrendbe, majd az így kapott listát nagyjából 6 egyenlő részre osztjuk kvantilisek segítségével (csak az utolsó csoport elemszáma különbözhet a többitől). A második korláthoz tartozó mutató alapján is sorba rendezzük a csúcsokat a 6 csoporton belül. A csoportosításra azért van szükség, mert az első mutatószám értékei alapján már egyértelműen sorba rendezhetjük általában a csúcsokat, így azok kis eltérése esetén sem lenne lehetőségünk a lexikografikus rendezésnél a második mutató alapján felülvizsgálni a sorrendet.

Outlier számítás: Az outlier kereső eljárásunk $O(H, cr)$ egy adott H Hamilton-kör outlier értékeit adja meg egy cr kritikus időérték mellett. Meghatározzuk H minden csúcsára a ki- és bemenő élek összidejét, majd azok átlagát és szórását. Azon i elemeket tartjuk outliernek, melyek esetén

$$t_{i,be} + t_{i,ki} > t^*_H + cr\sigma_H\ ,$$

ahol t_H^* az átlagos be- és kimenő élköltség és σ_H a szórás. A cr értéke az algoritmus egyes lépéseinél változhat. Ezt külön jelezzük majd.

Heurisztikus algoritmusunk az alábbi lépésekből áll:

1. *A probléma egyszerűsítése:* Töröljük a gráf minden csúcsát (az azokba bemenő és azokból kimenő élekkel együtt), melynek értékelése kisebb vagy egyenlő s^* küszöbértéknél, melyet a gyakorlatban választhatunk úgy, hogy a megmaradó csúcsok összes látogatási ideje ne haladja meg a $PT_{\max}$ rendelkezésre álló összidőkeret kétszeresét. Gyakorlati tapasztalatunk alapján az ennél több pont szerepeltetése nem javít az optimumon, ellenben az algoritmus számítási igényét fölöslegesen növeli. Ennél általában konzervatívabb megoldás, ha s^* értékét úgy választjuk, hogy megegyezzen az értékelések átlagával, hiszen ha $s_i < s^*$, akkor $u(s_i, a) < 0$, tehát nem javíthat a célfüggvény értéken. (Az általunk vizsgált esetekben mindig volt annyi átlagon felüli értékelésű csúcs, hogy azok összes látogatási ideje meghaladja a $PT_{\max}$ rendelkezésre álló összidőkeret kétszeresét.) Az így kapott halmazt a továbbiakban a releváns csúcsok halmazának nevezzük.

2. *Fix csúcsok:* Rögzítsük a kötelezően meglátogatandó csúcsokat. Ezek az eljárás során soha nem kerülhetnek az útvonalból törlésre. A korábbi megegyezés alapján azokat tekintjük kötelezőnek, melyekre vonatkozóan a turista értékelése maximális volt.

3. *Csoportosítás:* A releváns csúcsok alapján megbecsüljük, mely erőforrás korlátunk szűkösebb. Ennek érdekében összevetjük az alábbi két hányadost:

- a releváns csúcsok látogatási idejéhez hozzáadjuk a releváns csúcsok közötti átlagos távolságot, és ezt az összeget szorozzuk a releváns csúcsok számával, majd elosztjuk a rendelkezésre álló $PT_{\max}$ időkerettel,

- a releváns csúcsok látogatási költségének összegét osztjuk a BP költségvetési kerettel,

Amelyik érték nagyobb, azt tekintjük szűkösebb korlátnak, és a lexikografikus rendezések alkalmával a pontszámok után rögtön azt a korlátot vesszük előre. Ha tehát a szűkös korlát az idő, akkor a rendezésnél az alábbi sorrendet vesszük figyelembe a változók körében: értékelés, idő, pénz. A maximális pontszámú (tehát kötelező) csúcsok mellé a maradék releváns csúcsra lexikografikus rendezés után választjuk az első $5P$ darab csúcsot. Ez az egyetlen olyan lépés, ahol a fent ismertetett lexikografikus rendezéstől eltértünk annyiban, hogy első rendező elvként a pontszámot használtuk. A releváns pontok halmazának outlier csúcsait rendhagyó módon egy a teljes halmazra meghatározott legrövidebb Hamilton-kör meghatározásával kezdjük (melynek kezdő- és végpontja a szálloda). Ebből $cr = 1$ értékválasztás mellett használjuk az $O(H, cr)$ függvényt az outlierek kiszűrésére (természetesen csak a nem maximális értékelésű csúcsok lehetnek outlierek). Azért ezt az eljárást választottuk, mert bár eshet távol néhány csúcs a „központtól", ám ha oda egy viszonylag rövid élköltségekből felépíthető út vezet, akkor tapasztalataink alapján nem érdemes rögtön eldobni. Jó példa erre a Városliget, míg tipikus outliernek nevezhető a Nagytétényi Kastély.

4. *Napi utak építése:* A megmaradó csúcsokat P darab (napok száma) klaszterre bontjuk *Hartigan-Wong klaszterező eljárással* [28], melyet relatív hatékonysága miatt választottunk. Minden klaszterre kiszámoljuk a szállodával alkotott legrövidebb Hamilton-kört a TSP megoldására adott algoritmussal. Ennek hátterében az a megfontolás áll, hogy amennyiben csak egy fix csúcsokból álló halmazon szeretnénk a célfüggvényünket maximalizálni, az ekvivalens a csúcsokat összekötő élek élköltségeinek β-adik hatványaival vett gráfon történő legrövidebb Hamilton-kör meghatározásával, hiszen a csúcsok változatlansága miatt mind a hasznosságok, mind a látogatási idők értéke állandó az adott halmazra. Ezt kihasználva az R szoftverben beépített TSP optimalizáló *Repetitive Nearest Neighbor Algorithm* eljárást alkalmaztuk (bővebben lást [29]). Ezt a klaszterező eljárást 10-szer ismételjük meg, hiszen a klaszterezés gyakran vezet különböző eredményre. Az 10 eredmény közül azt választjuk, ahol P darab Hamilton-körre számolt célfüggvény értékünk maximális.

5. *Feltöltés:* Az előző lépésben P darab utat kaptunk, mely a szállodánál kezdődik és ott ér véget. Amennyiben még nem értük el a napi idő- és pénzkeret 1,2-szeresét, akkor az $L(C_r, C_i, sc, s^*)$ eljárással rendezzük az i-edik napra a releváns csúcsok halmazát, melyeket még egy útba sem illesztettünk be (jelölje ezt C_r). Itt fontos megemlíteni két elvet:

- Azokat a csúcsokat, melyeket egy adott napra beillesztünk, automatikusan kivesszük a még megmaradt releváns csúcsok C_r halmazából.

- Ha egy csúcsot kiveszünk egy napból, azt automatikusan visszarakjuk a C_r halmazba. Így minden napra rendeztük C_r elemeit, és a lista elejéről kezdve elkezdjük feltölteni a csúcsokkal a napokat, amíg el nem érjük az idő- és pénzkorlát 1,2-szeresét. Amennyiben egy csúcs két nap szerinti rendezésben is bekerülne az útba, oda helyezzük, ahol magasabb marginális célfüggvény javulást eredményez. Az i-edik napra való felvétel kritériuma, hogy az így keletkező Hamilton-körben az adott pont ne legyen outlier, ahol az $O(H, cr)$ függvényt $cr = 1{,}5$ mellett értékeljük ki.

6. *Csere:* Minden nap csúcsaira meghatározzuk a többi nap pontjaitól vett 3 legkisebb érték átlagát, ezt a csúcs saját napjára is kiszámítjuk (ahol a másodiktól a negyedik legkisebb értékig vesszük az átlagot, hiszen a legkisebb érték, az önmagával vett távolság, ami 0). Ezután minden csúcsot arra a napra helyezünk át, hol ez az érték minimális. Ezt az iterációt 10-szer ismételjük meg egymás után.

7. *Levágás:* Ha van olyan nap, ahol meghaladtuk a napi időkeretet több mint 5%-kal (ennyit engedélyezünk legfeljebb), akkor az $L(C_i, C_i, sc, s^*)$ alapján (vagyis saját magával) rendezve a naphoz tartozó csúcsok halmazát az utolsó elemeket addig távolítjuk el a napi útból, míg az időkihasználása a keret 105%-ánál nem lesz kevesebb. A pénzkorlát túllépése esetén az(oka)t a ponto(ka)t távolítjuk el, ahol az egységnyi pénzköltségre eső marginális célfüggvény növekedés minimális.

8. *Feltöltés:* Amennyiben van olyan nap, ahol még van szabad időkapacitás, a C_r elemeit $L(C_r, C_i, sc, s^*)$ eljárással rendezzük az i-edik napra, és az első elemtől kezdve elkezdjük a napot feltölteni, míg az időkorlátot és a P napra szánt költségvetési korlátot át nem lépjük. Itt ismét a felvétel kritériumaként az $O(H, 1, 5)$ függvényt alkalmazzuk, mint korábban.

3.5 Az eredmények kiértékelése

A fenti algoritmust a 150 budapesti turisztikai látványosságot tartalmazó adathalmazon tesztelve az alábbi megállapításokat tehetjük:

Pozitív a értékek (konkáv hasznossági görbe) jellemzően nagy kitérőket eredményeznek. Amennyiben ezek alacsony α és β értékekkel párosulnak, úgy az utak „szétesőek", tehát viszonylag kevés pontot, és nagy élköltségeket tartalmaznak. Viszonylag kis α érték (0,5 alatt) csak magas β ($> 1{,}5$) és alacsony a ($< -0{,}5$) értékek mellett ad „jó" megoldást.

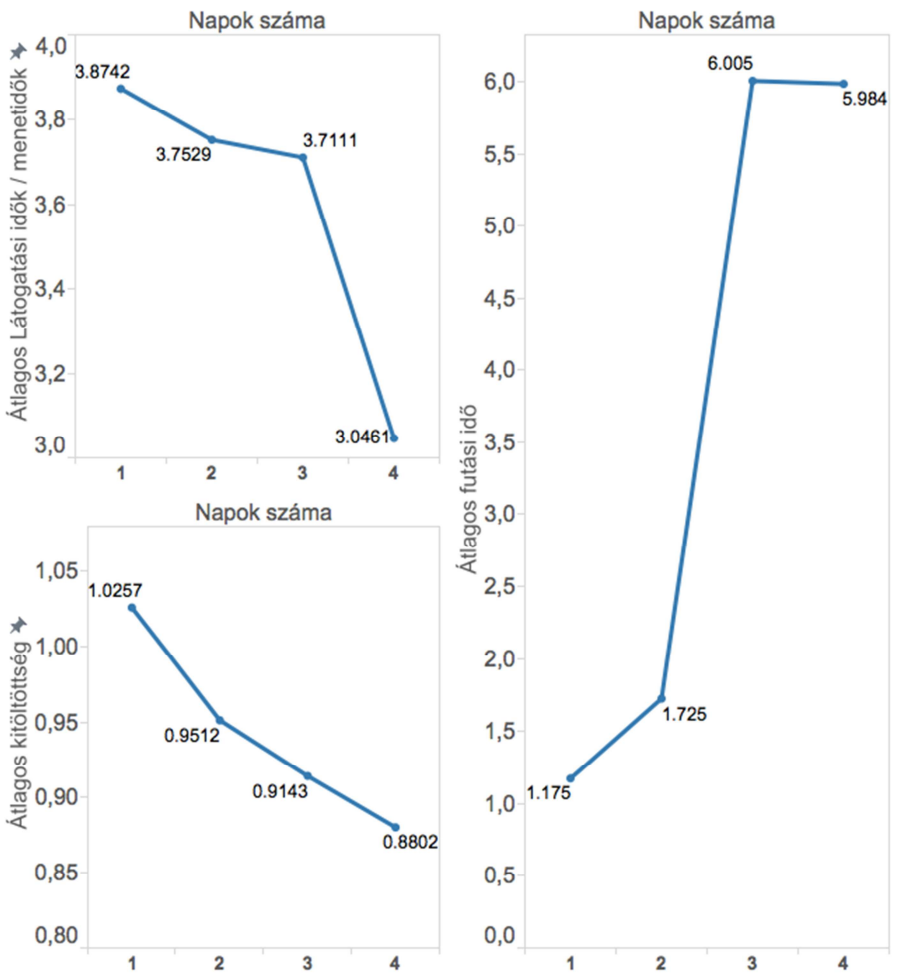
Általában elmondható, hogy $a < -0{,}5$; $\alpha > 0{,}5$ és $\beta > 1{,}5$ esetén kaptunk jó megoldásokat.

Az $\alpha = 0{,}75$; $a = -1$ és $\beta = 2$ választása mellett adja összességében 1-2-3 és 4 napos túrákra a legjobb megoldást (hosszabb túrákat nem vizsgáltunk), ahol a napok feltöltöttsége is igen jó, és a teljes túrákra kalkulált látogatási idő – menetidő hányados is kiemelkedően magas. Ezzel a paraméter kombinációval tervezett 4 napos túrát mutatunk be az 1. mellékletben, ahol összehasonlításként szerepeltetjük $\alpha = a = 0$ esetet, mely a szakirodalomban széles körben elterjedt profitösszeg-maximalizálást jelenti. Ennek tanulsága alapján a profitösszeg maximalizálás jóval alacsonyabb látogatási idő – menetidő arányt eredményez (átlagosan 1,43, szemben az általunk kiemelt eset átlagosan 3,8-es eredményével), a napok kitöltöttsége mindkét esetben átlagosan 95% körüli, és a futási idő érthető módon átlagosan nagyjából 1 másodperccel hosszabb az általunk választott paraméterek esetében.

Az algoritmus 3 napos útvonal megtervezését 100-szor futtatva átlagosan 3,73 másodperc alatt végezte el. Sajnos ezt nem tudjuk összevetni Vansteenwegen et al. [31] vagy Gavalas et al. [30] eredményeivel, hiszen merőben más feladatot oldottunk meg, sőt azok optimális megoldása számolható LP feladatként, míg célfüggvényünk miatt a miénk egy NLP feladat. Az összevetés kedvéért egy másik mobil applikációhoz tervezett heurisztikus eljárást említve Sylejmani és Dika 2011-es cikkében [32] 40 látványosságra tervezett 3 napos túrájuk számítási ideje átlagosan 81,7 másodperc volt *tabu search* algoritmussal.

A futási időről általában elmondhatjuk, hogy egy 4 napos túra megtervezése is átlagosan 6 másodpercen belül tartható. Mivel már az első lépésben csökkentjük a gráf méretét, kijelölve a releváns csúcsok részhalmazát, ezért a napok számától (P) és a napi időkeretektől ($T_{\max}$) függ a probléma mérete. Amennyiben egy olyan furcsa városban végeznénk a kísérletet, ahol minden csúcs meglátogatása költséggel jár, ez esetben a pénzügyi korlátunk is ugyanúgy lehet effektív, mint esetünkben az időkorlát. A túrák megtervezésének átlagos számítási idejét a *2. ábrán* foglaljuk össze. A teszteket

az alábbi paraméterekkel rendelkező laptopon futtattuk: 3,8 GB RAM, Intel Core i3-3217U CPU, 1.80GHz 4 processzor. Minden paraméter kombinációra 20 alkalommal végeztük el a tesztet, és az eredményeket (outlier értékektől való tisztítás nélkül) átlagoltuk. A futási idők érdekessége a 2 és 3 napos útvonalak tervezése közötti nagy eltérés az átlagos futási időben, míg ez nem növekszik 4 nap megtervezése esetén. A *2. mellékletben* összefoglaltuk az a, α és β paraméterek függvényében is a futási időket a napok száma szerinti bontásban. Az a értékének változása látszólag nincs hatással a futási időre és a β paraméter is csak 3 és 4 napos túrák esetén növeli kis mértékben azt. Az α növekedése azonban 3 és 4 napos túrák esetén drasztikusan növeli a futási időt, mintegy 3-szorosára. Itt vélhetően az áll a háttérben, hogy ilyenkor egyre nehezebb olyan pontokat találnia az algoritmusnak, mely növelni tudná a célfüggvény értékét.



2. ábra. Az útvonaltervező algoritmus eredményei

A napok kitöltöttsége csökkenő tendenciát mutat a napok számának növelésével (2. ábra). Az összes általunk vizsgált esetre átlagosan 94,3%-os értéket mértünk. A paraméterek közül csak az α paraméter növekedése van negatív hatással a napok kitöltöttségére (2. melléklet), hiszen itt a célfüggvény hasznosság tényezője egyre kevésbé számít, így nehezebb olyan csúcsokat találni, melynek hasznossága ellensúlyozni tudja a látogatási idő – menetidő hányadosban bekövetkező romlást.

A látogatási idő − menetidő hányados (mely a P napra együtt értendő) a napok számának növekedésével csökkenő tendenciát mutat, hiszen egyre nagyobb távolságra találjuk a következő csúcsokat, amelyeket még felvehetünk az utakba. Az a és β paraméter változása csekély hatással van a hányadosra, az α paraméter növelésével azonban drasztikusan növelhető a hányados értéke.

4 Következtetések és lehetséges kutatási irányok kijelölése

Az előző alfejezetben bemutatásra került a TOP megoldására adott heurisztikus algoritmus, mely bár az alkalmazott módszerekben is sok ponton eltér a szakirodalomban található megoldásoktól, mégis legnagyobb vívmánya az a hasznossági- és célfüggvény, mely gyakorlatias megközelítésben a felhasználó preferenciáit tartja szem előtt a „pontgyűjtéssel" szemben. Az eredmények értékelése igen nehéz, hiszen célunk olyan útvonaltervező algoritmus megalkotása volt, mely a felhasználók személyre szabott igényei (helyszínek értékelése) alapján képes másodpercek alatt, számukra megfelelő útvonalat tervezni. Bár esetünkben is értelmezhető az optimális megoldástól való eltérés mértéke, ennek kiszámítása mégis nehezen vethető össze más kutatások eredményeivel, hiszen eltérő célfüggvénnyel dolgoztunk. Az útvonaltervezés értékelésének egy lehetséges módja, ha az algoritmusunk által készített útvonalakat és egy a szakirodalomban szereplő másik eljárás eredményeit értékeltetjük tesztalanyokkal, hiszen az ő értékeléseikre tekintünk a útvonalterv jóságának végső fokmérőjeként. Az algoritmus által generált útvonalak jóságának mérésén túl néhány további tervünk is van a kutatás folytatását illetően.

A kezdeti útvonalak klaszterezésen alapuló kialakítása helyett jó megoldás lehet P darab egymástól kellő távolságra található csúcs kijelölése: ezek egyrészt a kötelező pontok lehetnek, másrészt egyéb, magas értékelésű csúcsok. Az egyes napokra ezekből kiindulva építhetünk fákat, melyeket úttá rendezhetünk át Lin-Kernighan-algoritmussal. A klaszterező eljárás ugyanis nem mindig vezet ugyanarra az eredményre, ezért is van szükségünk az algoritmus kezdeti lépésében 10 ilyen iterációra. Az iterációk számát növelve ugyan biztosíthatjuk a jó klaszterezést, ám jelentősen növeljük vele a futási időt (20 klaszterezéssel már átlagosan további 2,5 másodperccel).

A jelenlegi kutatási szakaszban egyetlen α, β, a kombinációt emeltünk ki, mint optimálisnak tűnő megoldást. Ezzel magasan tudjuk tartani a na-

pok kitöltöttségét, jellemzően a legmagasabb értékelésű csúcsokon rendre áthalad az útvonal, és alacsony élköltségeivel a célfüggvény értékét magasan tartja. Amennyiben a napok kitöltöttségét és az egységnyi megtett útra eső látogatási időt (vagy hasznosságot) elfogadjuk az útvonaltervezés jóságának fokmérőjeként, akkor lehetőség nyílna rögzített a érték mellett az optimális α és β kombinációk meghatározására, sőt akár megadható lenne β az α függvényében, és ezzel csökkenthetjük a paraméterek számát.

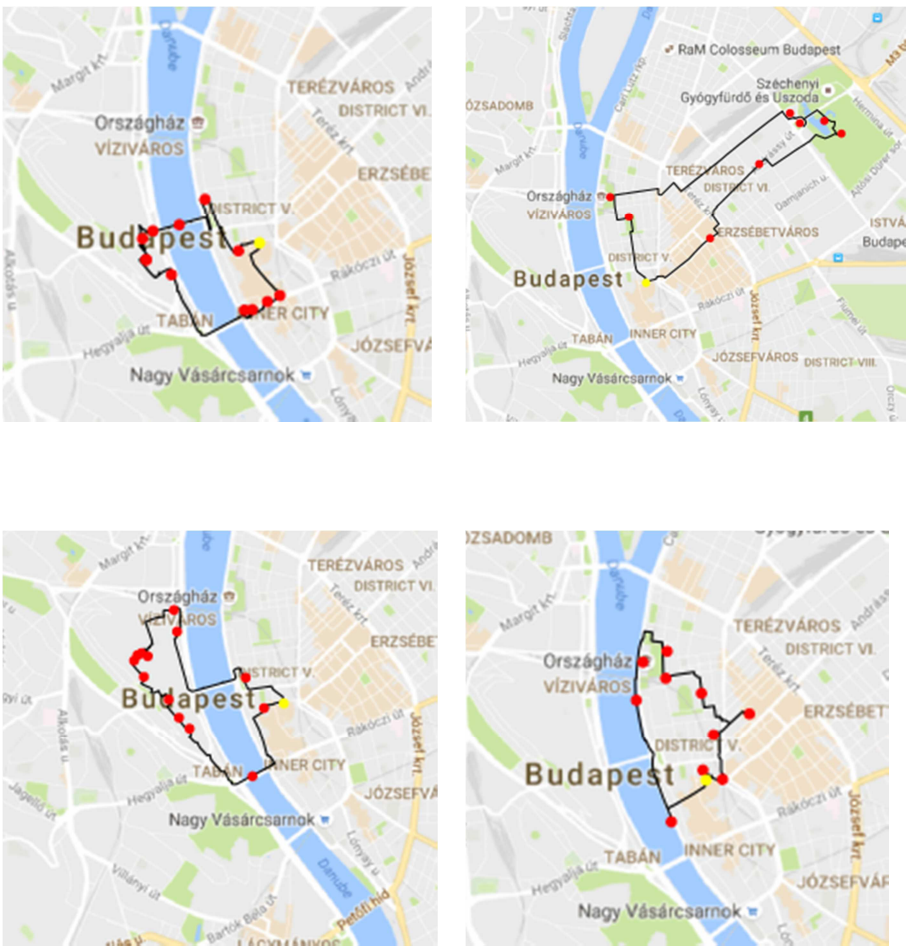
Az előbbi gondolatmeneten tovább haladva, legalább két ilyen függvény definiálása is szükséges lenne, hiszen β eredendően ,,lustasági" paraméter, így annak különböző értékei mellett más-más felhasználói igényeket tudunk kiszolgálni (továbbra is szem előtt tartva az útvonalak optimalitására tett törekvéseket).

Fontos megemlíteni, hogy a célfüggvény konstrukciójából adódóan az utolsó, feltöltő lépésben könnyen lehet, hogy már nem tudunk olyan pontot illeszteni bármelyik nap útvonalába, mely ugyan még a korlátok szerint elférne, de rontana a célfüggvény értékén. Ez akkor következhet be, ha a marginális hasznosságnövelésével nem tudja kompenzálni az addigi átlagos látogatási idő − menetidő arányban okozott romlást. Mivel célfüggvényünk maximalizálása mellett az időkorlát kitöltését is fontos szempontnak tartottuk, így egy valós gyakorlati megoldás esetén akár a célfüggvény rovására is feltölthetjük a napokat további pontokkal. Mi most ettől eltekintettünk.

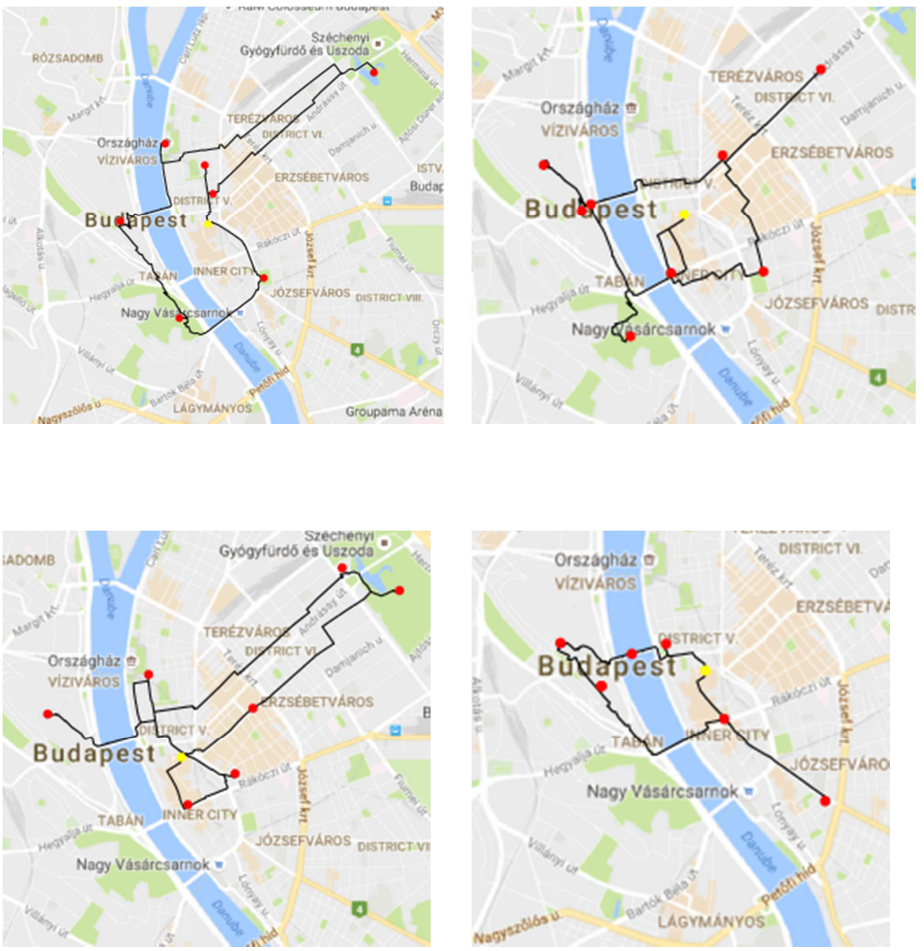
A jelenlegi algoritmus egy kézenfekvő és a gyakorlatban szükséges kiterjesztése lenne az időablakok kezelése, vagyis alkalmassá tenni az eljárást a TOPTW megoldására. A jövőbeli kutatás egyik fő mérföldkövének tekinthető ennek megvalósítása.

További gyakorlati fontossága lenne az algoritmus kiegészítésének, hogy adott árkategóriában választani tudjunk a rendelkezésre álló szállodák közül, és az algoritmus által köréjük szervezett P napos túrák közül a számára legkedvezőbbet választhatja ki, vagy megteheti ezt helyette az algoritmus is.

1. melléklet. Útvonaltervek

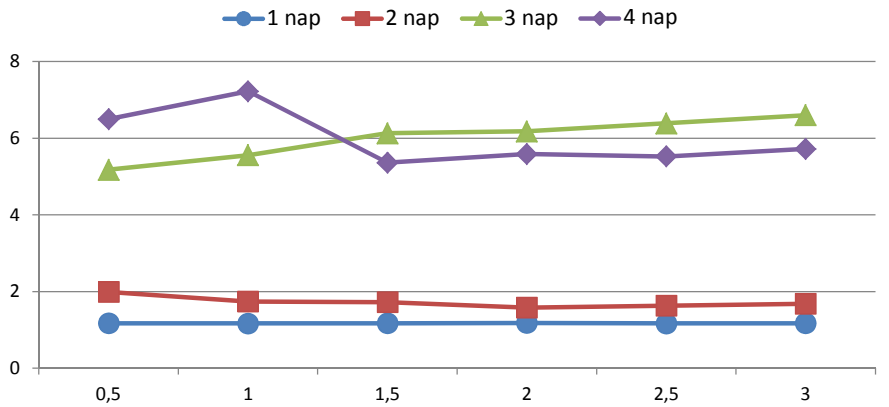


3. ábra. Az $\alpha = 0{,}75$; $a = -1$ és $\beta = 2$ eset útvonalterve

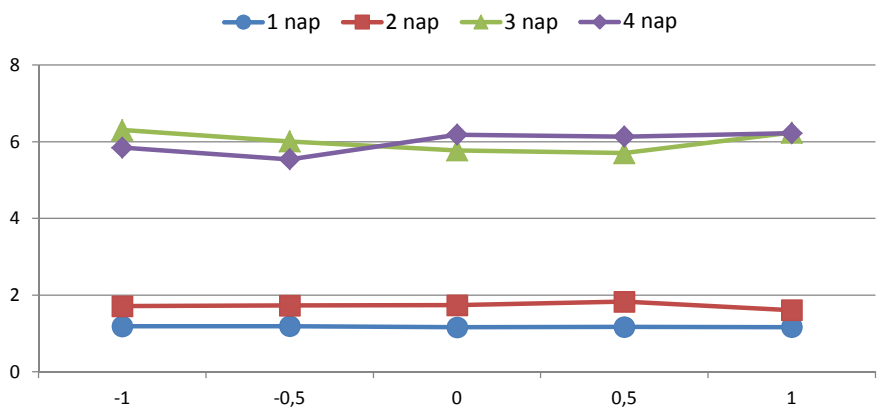


4. ábra. Az $\alpha = 0$ és $a = 0$ eset útvonalterve

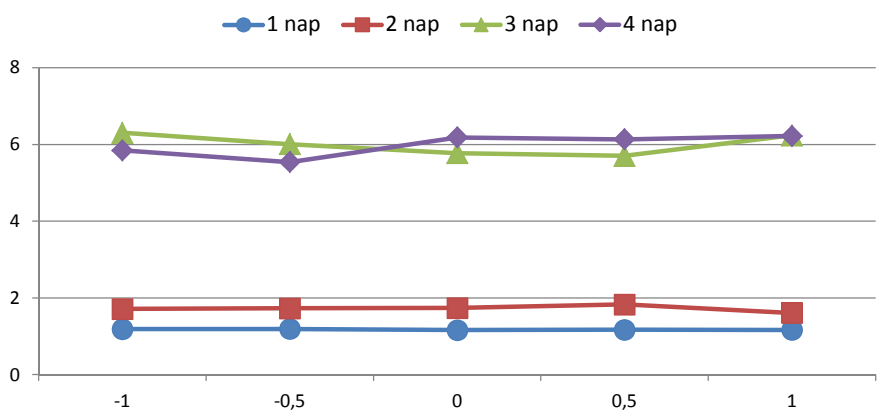
2. melléklet. Az útvonaltervező algoritmus eredményei



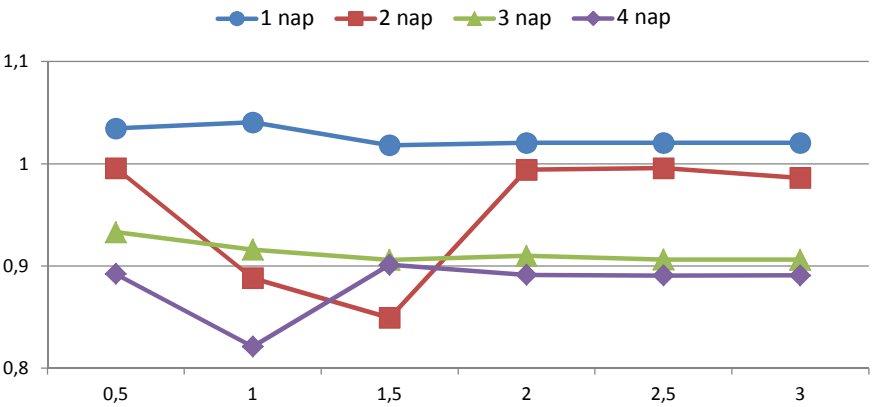
5. ábra. Átlagos futási idő (sec) a béta paraméter függvényében



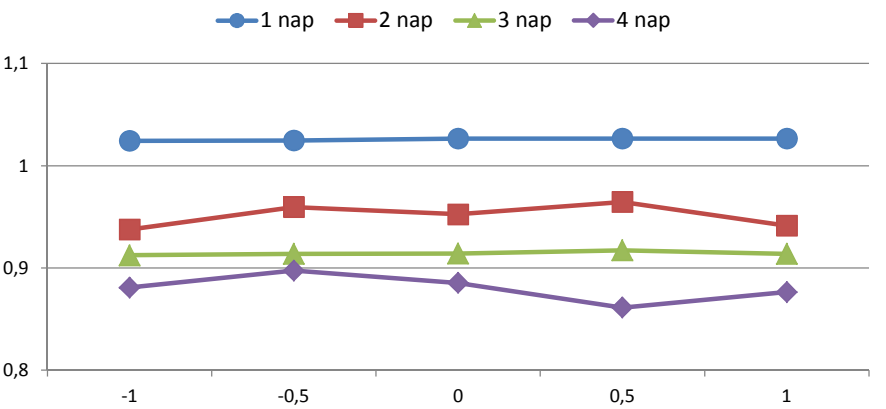
6. ábra. Átlagos futási idő (sec) az *a* paraméter függvényében



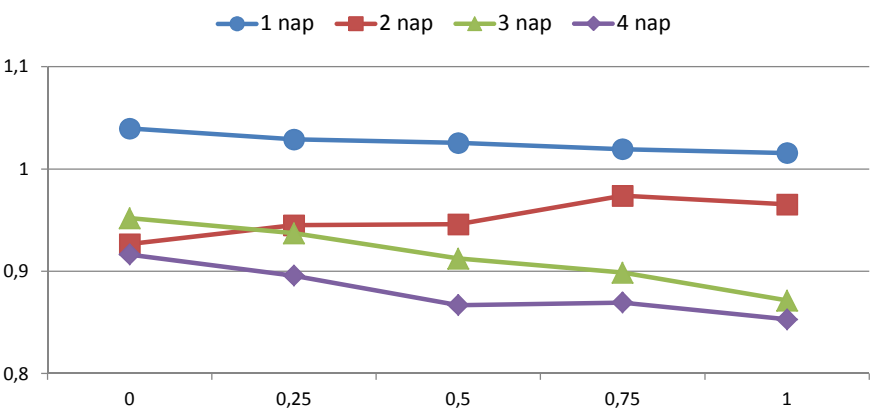
7. ábra. Átlagos futási idő (sec) az alfa paraméter függvényében



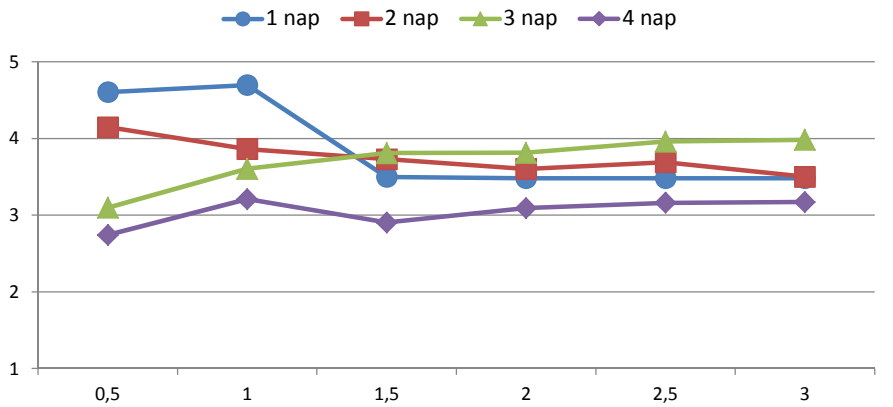
8. ábra. Átlagos kitöltöttség (%) a béta paraméter függvényében



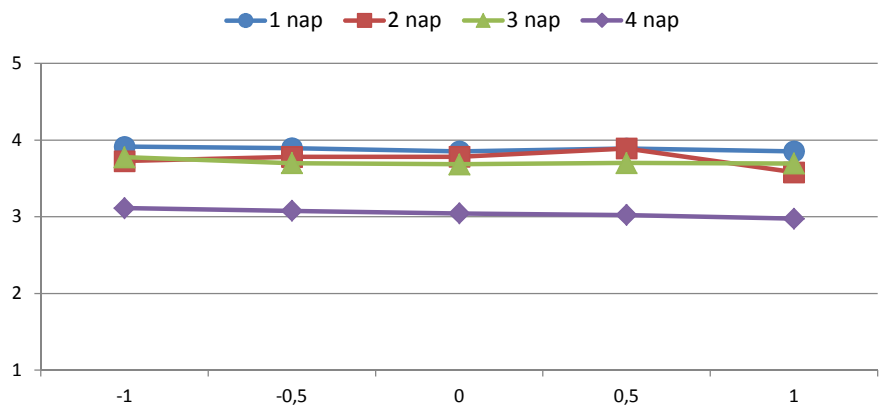
9. ábra. Átlagos kitöltöttség (%) az *a* paraméter függvényében



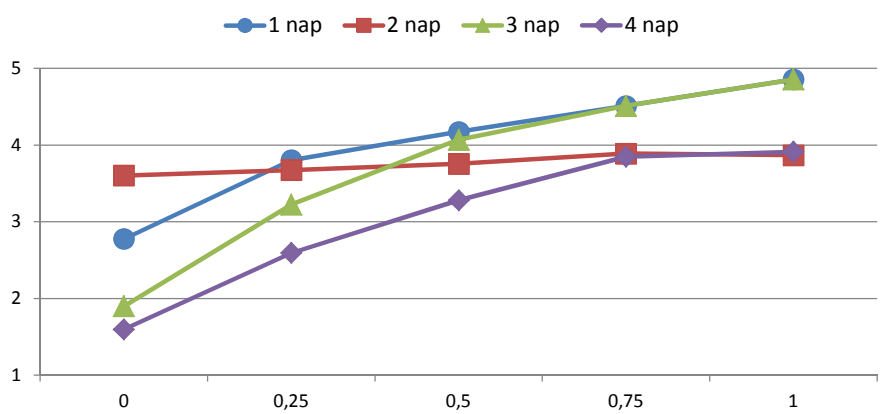
10. ábra. Átlagos kitöltöttség (%) az alfa paraméter függvényében



11. ábra. Átlagos látogatási idők/menetidők a béta paraméter függvényében



12. ábra. Átlagos látogatási idők/menetidők az a paraméter függvényében



13. ábra. Átlagos látogatási idők/menetidők az alfa paraméter függvényében

3. melléklet. Team Orienteering Problem formalizálása

Legyen adott egy $G(V, E)$ gráf, amelynek minden v_i csúcsához egy i nemnegatív profitérték van rendelve, melyet az ügynök megkap, ha meglátogatja a v_i csúcsot, valamint vi és vj csúcsok közötti e_{ij} élhez t_{ij} élköltséget rendelünk, ami a távolság megtételéhez szükséges idő. A feladat $T_{\max}$ idő alatt P darab ügynök számára maximális pontot összegyűjteni úgy, hogy minden csúcs legfeljebb egyszer látogatható meg. A kezdő- és a végpont fix, és gyakran meg is egyeznek egymással. Jelölje továbbá h_{ip}, hogy a p-edik útnál az i-edik csúcs hányadik lépésben kerül sorra az úton, valamint τ_{ijp} értéke legyen 1, ha a p-edik útnál az i-edik csúcs után a j-edik következik az úton, és 0 különben. Legyen θ_{ip} értéke 1, ha a p-edik úton az i-edik csúcsot meglátogatják, és 0 különben. Ekkor a TOP megfogalmazható a következőképpen:

$$\max \sum_{p=1}^{P} \sum_{i=2}^{N-1} \pi_i \theta_{ip} \tag{M1}$$

$$\sum_{p=1}^{P} \sum_{j=2}^{N} \tau_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} \tau_{iNp} = P \tag{M2}$$

$$\sum_{p=1}^{P} \theta_{kp} \leq 1\,, \quad k = 2, \ldots, N-1 \tag{M3}$$

$$\sum_{j=2}^{N} \tau_{kjp} = \sum_{i=1}^{N-1} \tau_{ikp} = \theta_{kp}\,, \qquad k = 2, \ldots, N-1, \quad p = 1, \ldots, P \tag{M4}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} \tau_{ijp} t_{ij} \leq T_{\max}\,, \quad p = 1, \ldots, P \tag{M5}$$

$$h_{ip} - h_{jp} + 1 \leq (N-1)(1 - \tau_{ijp})\,, \quad i, j = 2, \ldots, N, \;\; p = 1, \ldots, P \tag{M6}$$

$$2 \leq h_{ip} \leq N\,, \quad i = 2, \ldots, N, \quad p = 1, \ldots, P \tag{M7}$$

$$\tau_{ijp}, \theta_{ip} \in \{0, 1\}\,, \quad i, j = 1, \ldots, N, \quad p = 1, \ldots, P \tag{M8}$$

Az egyes kifejezések jelentése a következő: (M1) a célfüggvény: a csúcsoknál begyűjtött profitok összege legyen maximális az összes utat figyelembe véve; (M2) minden út az 1-es csúcsnál kezdődik, és az N-ediknél ér véget; (M3) minden csúcsot csak legfeljebb egyszer látogatunk meg; (M4) minden út egyenként összefüggő; (M5) betartjuk az időkorlátot; (M6) és (M7) együtt garantálja, hogy ne legyenek körök az útban, Miller–Tucker–Zemlin javaslata alapján [18]; (M8) a τ_{ijp} és θ_{ip} értékkészlete 0 vagy 1.

Irodalom

1. K. Menger (1928): Ein Theorem über die Bogenlange, Anzeiger – Akademie der Wissenschaften in Wien – Mathematisch-naturwissenschaftliche, Klasse 65, pp. 264–266.
2. A. Schrijver (2005): On the history of combinatorial optimization (till 1960), *Handbook of Discrete Optimization* (K. Aardal, G. L. Nemhauser, R. Weismantel, eds.), Elsevier, Amsterdam, pp. 1–68. doi: 10.1016/S0927-0507(05)12001-5
3. G. Birkhoff (1946): Tres observaciones sobre el algebra lineal, Revista Facultad de Ciencias Exactas, Puras y Aplicadas Universidad Nacional de Tucuman, Serie A (Matematicas y Fisica Teorica), Vol. 5, pp. 147–151.
4. G. Dantzig, R. Fulkerson, S. Johnson (1954): Solution of a Large Scale Traveling Salesman Problem, *Journal of the Operations Research Society of America*, Vol. 2, pp. 393–410. doi: 10.1287/opre.2.4.393
5. I. Chao – B. Golden – E. Wasil (1996): Theory and methodology – a fast and effective heuristic for the orienteering problem, *European Journal of Operational Research*, Vol. 88, pp. 475–489. doi: 10.1016/0377-2217(95)00035-6
6. T. Tsiligirides (1984): Heuristic methods applied to orienteering, *Journal of the Operational Research Society*, Vol. 35, No. 9, pp. 797–809. doi:10.1057/jors.1984.162
7. S. Kataoka – S. Morito (1988): An algorithm for the single constraint maximum collection problem, *Journal of the Operations Research Society of Japan*, Vol. 31, No. 4, pp. 515–530.
8. X. Wang – B. Golden – E. Wasil (2008): Using a genetic algorithm to solve the generalized orienteering problem, In: B. Golden – S. Raghavan – E. Wasil (Eds.): *The Vehicle Routing Problem: Latest Advances and New Challenges*, pp. 263-274. doi:10.1007/978-0-387-77778-8_12
9. W. Souffriau – P. Vansteenwegen – J. Vertommen – G. Vanden Berghe - D. Van Oudheusden (2008): A personalised tourist trip design algorithm for mobile tourist guides, *Applied Artificial Intelligence*, Vol. 22, No. 10, pp. 964–985. doi: 10.1080/08839510802379626
10. D. Feillet – P. Dejax – M. Gandreau (2004): Traveling Salesman Problems with Profits, *Journal of Transportation Science*, Vol. 39, No. 2, pp. 188–205. doi: 10.1287/trsc.1030.0079
11. B. Golden – L. Levy – R. Vohra (1984): The Team Orienteering Problem, *Naval Research Logistics Quarterly*, Vol. 34, pp. 307–318. doi: 10.1002/1520-6750(198706)34:3<307::AIDNAV3220340302>3.0.CO;2-D
12. S. E. Butt – T. M. Cavalier (1994): A heuristic for the multiple tour maximum collection problem, *Computers and Operations Research*, Vol. 21, pp. 101–111.
13. R. Ramesh – Y. Yoon – M. Karwan (1992): An optimal algorithm for the orienteering tour problem, *ORSA Journal on Computing*, Vol. 4, pp. 155–165. doi:10.1016/0305-0548(91)90086-7
14. M. Fischetti – J. Salazar – P. Toth (1998): Solving the orienteering problem through branch and cut, *INFORMS Journal on Computing*, Vol. 10, pp. 133–148. doi: 10.1007/978-3-319-18161-5_17
15. R. Ramesh – K. Brown (1991): An efficient four-phase heuristic for the generalized orienteering problem, *Computers and Operations Research*, Vol. 18, pp. 151–165. doi: 10.1016/0305-0548(91)90086-7

16. M. Gendreau – G. Laporte – F. Semet (1998): A tabu search heuristic for the undirected selective travelling salesman problem, *European Journal of Operational Research*, Vol. 106, pp. 539–545. doi:10.1016/S0377-2217(97)00289-0

17. P. Vansteenwegen – W. Souffriau – D. Van Oudheusden (2011): The orienteering problem: a survey, *European Journal of Operational Research*, Vol. 209, No. 1, pp. 1–10. doi:10.1016/j.ejor.2010.03.045

18. C. Miller – A. Tucker – R. Zemlin (1960): Integer programming formulations and travelling salesman problems, *Journal of the ACM*, Vol. 7, pp. 326–329. doi:10.1145/321043.321046

19. D. Gavalas – C. Konstantopoulos – K. Mastakas – G. Pantziou – N. Vathis (2015): Heuristics for the Time Dependent Team Orienteering Problem: Application to Tourist Route Planning, *Computers and Operations Research*, Vol. 62, pp. 36–50. doi:10.1016/j.cor.2015.03.016

20. S. Butt – D. Ryan (1999): An optimal solution procedure for the multiple tour maximum collection problem using column generation, *Computers and Operations Research*, Vol. 26, pp. 427–441. doi: 10.1016/S0305-0548(98) 00071-9

21. S. Boussier – D. Feillet – M. Gendreau (2007): An exact algorithm for the team orienteering problem, *4OR*, Vol. 5, pp. 211–230. doi:10.1007/s10288-006-0009-1

22. I. Chao – B. Golden – E. Wasil (1996): Theory and methodology – the team orienteering problem, *European Journal of Operational Research*, Vol. 88, pp. 464–474. doi:10.1016/0377-2217(94)00289-4

23. H. Tang – E. Miller-Hooks (2005): A tabu search heuristic for the team orienteering problem, *Computer and Operations Research*, Vol. 32, pp. 1379–1407. doi:10.1016/j.cor.2003.11.008

24. C. Archetti – A. Hertz – M. Speranza (2007): Metaheuristics for the team orienteering problem, *Journal of Heuristics*, Vol. 13, pp. 49–76. doi: 10.1007/ s10732-006-9004-0

25. L. Ke – C. Archetti – Z. Feng (2008): Ants can solve the team orienteering problem, *Computers and Industrial Engineering*, Vol. 54, pp. 648–665. doi:10.1016/j.cie.2007.10.001

26. P. Vansteenwegen – W. Souffriau – G. Vanden Berghe – D. Van Oudheusden (2009): A guided local search metaheuristic for the team orienteering problem, *European Journal of Operational Research*, Vol. 196, No. 1, pp. 118–127. doi:10.1016/j.ejor.2008.02.037

27. P. Vansteenwegen – W. Souffriau – G. Vanden Berghe – D. Van Oudheusden (2009): Metaheuristics for tourist trip planning, In: M. Geiger – W. Habenicht – M. Sevaux – K. Sörensen (eds.): *Metaheuristics in the Service Industry*, Lecture Notes in Economics and Mathematical Systems, Vol. 624, pp. 15–31. doi:10.1016/j.cor.2015.03.016

28. J. A. Hartigan – M. A. Wong (1979): Algorithm AS 136: A K-Means Clustering Algorithm, *Journal of the Royal Statistical Society*, Series C, Vol. 28, No. 1, pp. 100–108. doi: 10.2307/2346830

29. G. Gutin – A. Yeo – A. Zverovich (2002): Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP, *Discrete Applied Mathematics*, Vol. 117, pp. 81–86. doi: 10.1016/S0166-218X(01) 00195-0

30. D. Gavalas – M. Kenteris (2011): A pervasive web-based recommendation system for mobile tourist guides, *Personal and Ubiquitous Computing*, Vol. 15, No. 7, pp. 759–770. doi:10.1007/s00779-011-0389-x

31. P. Vansteenwegen – W. Souffriau – G. Vanden Berghe – D.D. Van Oudheusden (2011): The city trip planner: an expert system for tourists, *Expert Systems with Applications*, Vol. 38. No. 6, pp. 6540–6546. doi:10.1016/j.eswa.2010.11.085
32. K. Sylejmani – A. Dika (2011): Solving touristic trip planning problem by using taboo search approach, *International Journal of Computer Science Issues*, Vol. 8, Issue 5, No. 3, pp. 139–149. doi:10.1109/HIS.2012.6421351
33. S. Lin (1965): Computer solutions of the traveling salesman problem, *Bell Systems Technology Journal*, Vol. 44, pp. 2245–2269. doi:10.1002/j.1538-7305.1965.tb04146.x
34. D. Gavalas – C. Konstantopoulos – K. Mastakas – G. Pantziou – N. Vathis (2015): Heuristics for the Time Dependent Team Orienteering Problem: Application to Tourist Route Planning, *Computers and Operations Research*, Vol. 62, pp. 36–50. doi:10.1016/j.cor.2015.03.016

A HEURISTIC ROUTING ALGORITHM FOR PLANNING MULTI-DAY TOURS

Routing algorithms are traditionally considered to apply the sum of profits gathered at visited locations as an objective function since the Traveling Salesman Problem. This heritage disregards many practical considerations, hence the result of these models meet with user's needs rarely. Thus a novel objective function will be presented in this paper as an extension of the one inherited from the TSP, that is more aligned with user preferences and aims to maximise the tourist's satisfaction. We also propose a heuristic algorithm to solve the Team Orienteering Problem with relatively low computation time in case of high number of vertices on the graph and multiple tour days. Based on the observations the algorithm is suitable to be implemented in a GIS application considering that even a 3-day tour is designed less than 4 seconds.

Keywords: Team Orienteering Problem, Route Planning, Heuristic Algorithm, Tourism. *JEL code:* C60, C61, Z32