



M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem
Közlekedésmérnöki és Járműmérnöki Kar
Közlekedésüzemi és Közlekedésgazdasági Tanszék

Multimodális útvonaltervezés hangyakolónia algoritmussal

TDK dolgozat

.....

Katona Géza
L0I8ZH

2014

Absztrakt: Gyakori kérdések egy utazás során a mikor?, mennyiért?, milyen módon? A társadalom, a technológia fejlődésével a mobilitás iránti igény nő, a mobilitás kultúra pedig változik. Előtérbe kerülnek a zöld technológiák, kiegészítő szolgáltatások, esélyegyenlőségi kérdések. Az interneten elérhető információ mennyiségének növekedésével, illetve a közösségi oldalak fejlődésével együtt a közlekedési rendszerek használói is magasabb igényeket támasztanak a szolgáltatókkal, szolgáltatásokkal szemben, továbbá az információ igényük is nő.

A TDK keretében a multimodális útvonaltervezést fogom megvalósítani a hangya algoritmussal. A hangyakolónia algoritmus különböző paraméterektől függ, ezért e paraméterek vizsgálatával a feladatra legjobb értékek megkeresése lesz a céltól dolgozat keretén belül.

Kulcsszavak: Multimodális útvonaltervező, Hangyakolónia algoritmus

Konzulens: Lénárt Balázs, tanársegéd

Tartalomjegyzék

1	BEVEZETÉS	4
2	HANGYAKOLÓNIA ALGORITMUS	6
3	A HANGYA ALGORITMUS IMPLEMENTÁLÁSA	8
4	FUTÁSI EREDMÉNYEK	15
4.1	α, β, Q VIZSGÁLATA (10 ADATPONT, 5 HANGYA, 50 ITERÁCIÓ)	15
4.2	VÁTOZÓ HANGYA, ITERÁCIÓ ÉS ADATPONTSZÁM	21
5	ÉRZÉKENYSÉGVIZSGÁLAT	25
5.1	AZ ÉRZÉKENYSÉGVIZSGÁLAT ELMÉLETE	25
5.2	AZ ÉRZÉKENYSÉG VIZSGÁLAT ALKALMAZÁSA.....	27
6	TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK.....	31
7	ÖSSZEFOGLALÁS	33
	IRODALOMJEGYZÉK.....	34
	ÁBRAJEGYZÉK.....	35

1 Bevezetés

Mindennapi probléma, hogy el szeretnénk jutni egyik pontból a másikba, és ehhez különböző eszközök segítségével tervezünk útvonalat. Napjainkban leggyakrabban ilyenkor a Google Maps-hoz folyamadunk, vagy az Elvirához esetleg az újonnan indult BKK Futárhoz, azonban ezek a rendszerek sajnos nem átjárhatóak, csak az adott területre, vagy vállalat járatait fedik le. A közúti tervezők tekintetében az iGO vagy a Sygic nevű szoftverek a legelterjedtebbek. Ezen szoftverek mögött általában a Dijkstra vagy az A* útvonalkereső algoritmusok vagy ezek valamilyen továbbfejlesztett változatai. Azonban ezen algoritmusok használatának megvannak a maguk hátrányai is, ezért a programozók különféle megkötésekre szorulnak, ahhoz, hogy megfelelően gyorsan működő algoritmusokat tudjunk használni. Azonban a felhasználói igények egyre inkább arra tartanak, hogy egy egységes, komplex rendszerben szemléljük a közlekedést, ahol az utazás során különféle szolgáltatók szolgáltatásait vesszük igénybe (például közbringa, közösségi autó, közösségi közlekedés, egyéni közlekedés, stb.), ennek eredményeképpen pedig egyre bonyolultabban kell kezelni a hálózatokat, az azokat leképező hálózatok mérete exponenciálisan megnövekszik, mely magával hozza az új megoldások megjelenését.

A közlekedési igények pedig várhatóan egyre inkább növekedni fognak, mivel a lakosság egyre inkább mobilizálódik, erre mutatnak a különböző európai programok, mint például az Erasmus, melyek a tagállamok közötti egyre nagyobb mozgások indikátorai, illetve a közlekedési eszközök sebessége is egyre növekszik, így olyan területek is egyre közelebb kerülnek egymáshoz, melyeket korábban csak távolsági utazások kapcsoltak össze, ma már akár ingázó forgalom is megjelenik ezeken. A különböző kiegészítő rendszerek használatára az egyik legjobb példa Németország, ahol egy vidéki lakos a vonatállomásig valamilyen helyi közlekedési eszközzel eljut, majd onnan gyorsan és pontosan bejut a nagyvárosba, ahol a tömegközlekedést, a közbringákat illetve a különböző autó megosztó vagy autóbérlő rendszerek segítségével jut el a céljához. Ha belegondolunk abba az összefüggésbe, hogy a gazdaság fejlődésével, növekedésével a közlekedési igények másfél kétszer olyan mértékben növekednek, akkor látható, hogy a jövő afelé hat, hogy egyre nagyobb mértékű utazási igényekre kell felkészülnie a különböző szolgáltatóknak, köztük az útvonaltervező cégeknek is. A modern nagyvárosi életformával pedig egyre inkább együtt jár a különböző közösségi illetve megosztó rendszerek használata, melyek a bejáratott közlekedési módokhoz képest teljesen új szemléletet kö-

vetel meg az útvonaltervezés tekintetében, melyek a hagyományos módszerekkel nehezen kezelhetőek le, emellett a felhasználók egyre több információt és egyre jobban megtervezett útvonalakat várnak, melyeknek mindig a legfrissebb közlekedési eseményeknek megfelelően aktuálisnak kell lenniük, illetve módosulniuk kell az aktuális forgalmi állapotoknak megfelelően.

A folyamatosan változó közlekedési rendszerhez pedig új megoldások szükségesek, az egyik lehetséges irányhoz a természet adta az ihletet, mégpedig a hangyák táplálékkeresési módja, mely egy igen hatékony és gyors módja annak, hogy a lehető leggyorsabban megtalálják az élelemhez vezető leggyorsabb útvonalat. A dolgozatom keretén belül ezzel a módszerrel fogok foglalkozni. Mivel a módszer különböző paraméterek segítségével hangolható, ezért a dolgozatban ezek finomhangolásával fogok foglalkozni az útvonalkeresés tekintetében.

2 Hangyakolónia algoritmus

A modell alapjait Marco Dorigo rakta le. [1] Kutatásai során megfigyelte a hangyák élelemszerzési módszerét. Ennek lényege, hogy a hangyák teljesen véletlenszerű úton elindulnak a bolyból élelmet keresni. Ha élelmet találtak, akkor a visszafele úton feromont bocsájtanak ki. A feromon a többi hangya számára vonzó tulajdonságú, ezért amikor egy újabb hangya indul élelemért, akkor az irány választásakor nagyobb valószínűséggel fog ebbe az irányba indulni. Minél több hangya halad el az adott útvonalon, annál erősebb a feromon nyom az útvonalon, így egyre több hangya választja majd ezt az útirányt. Emellett minél közelebb van az adott élelemforrás, annál többször tudnak megfordulni a hangyák, ami szintén a feromon nyom erősödését vonja maga után. [1][4]

Az útvonalkeresésre alkalmazott hangyamodellben a hangyák egy-egy járművet vagy utast reprezentálnak. Az útvonal addig növekszik, amíg az összes meglátogatandó hely bele nem kerül az útvonalba. Az algoritmus működése során minden hangya a kiindulási pontból indul. A következő meglátogatandó helyszín választása során az elérhető helyszínek közül történik a választás a jármű kapacitásának figyelembe vételével. A hangya akkor tér vissza a kiinduló ponthoz, ha az összes meglátogatandó helyszínt bejárta vagy a jármű kapacitását elérte. Az L távolság a virtuális hangya által megtett útból számított. A második hangya az első hangya visszaérkezése után indul, és mindez addig folytatódik, amíg az előre megadott hangyaszámot el nem éri a program. Az algoritmus működése során minden hangyának minden pontot érintenie kell. A kiválasztáshoz a következő formula alkalmazható [1]

$$j = \arg \max \{ (\tau_{iu}) (\eta_{iu})^\beta \} \text{ amíg } u \notin M_k, \text{ ha } q \leq q_0, \text{ különben } S, \quad (1)$$

ahol τ_{iu} az i jelenlegi pozíció és az u lehetséges pozíció közötti feromon mennyiség, η_{iu} pedig i és az u közötti távolság inverze, β a távolság fontosságát a feromon mennyiségével szemben kifejező tényező, M_k pedig a hangya által már bejárt utat tartalmazza. q egy véletlen változó a $[0,1]$ zárt intervallumon, q_0 pedig egy paraméter. Ha minden választás megtörtént, a hangya a legnagyobb értékű élt választja az (1) egyenletből, ha csak q nem nagyobb, mint q_0 . Ekkor a hangya egy véletlenszerű S -t választ, mely a p_{ij} eloszlás valószínűségén alapszik, ami a magas feromon tartalmú útvonalaknak kedvez [1]

$$p_{ij} = \begin{cases} \frac{(\tau_{ij})(\eta_{ij})^\beta}{\sum_{u \notin M_k} (\tau_{iu})(\eta_{iu})^\beta} & \text{ha } j \notin M_k \\ 0 & \text{ha } j \in M_k \end{cases} \quad (2)$$

Az (1)-es és (2)-es kifejezéseket alkalmazva a hangyák vagy a legkedvezőbb utat követik vagy véletlenszerűen választanak a feromonok alapján. Az algoritmus addig folytatódik, amíg minden helyszín meg nincs látogatva és a bejárás kész nincs teljesen.

Az algoritmus legfontosabb része a feromon mennyiséget frissítő algoritmus. Az útvonalak aktualizálása magában foglalja a lokális és a globális frissítést a legjobb útvonalaknak egy előre meghatározott m -ig. A lokális frissítés módszere a következő [1]

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + (\alpha)\tau_0 \quad (3)$$

ahol α a feromon kibocsátását szabályozó paraméter, τ_0 pedig az él kezdeti feromon értéke.

Miután az előre meghatározott m darab hangya létrehozta a lehetséges útvonalakat, a globális aktualizálás során a legjobb útvonal által tartalmazott élekhez feromon kerül hozzáadásra egy darab hangya által az m -ből. [1]

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + \alpha(L)^{-1} \quad (4)$$

Ez a megoldás a legjobb út használatára ösztönöz. Az eljárás egy előre meghatározott alkalomszor fut le. [1]

A hangyamodellt továbbfejlesztve lehetőség nyílik többféle befolyásoló tényező figyelembe vételére, mint például a jármű kihasználtság, pénzbeli haszon, időszükséglet... Ez a következőképpen írható fel [4]

$$\tau_{ij} = \tau_{ij} + \mathcal{B} \cdot \tau_{ij} \cdot \mathbb{H}_{\ell_{ij}} \quad (5)$$

ahol $\mathcal{B}$ a konzervatív és a felfedező keresés egyensúlyát biztosítandó szám, $\mathbb{H}_{\ell_{ij}}$ a hasznossági függvény. [4]

Az egyenletekben szereplő α és β értékeket a szakirodalom alapján célszerű kiválasztani. [1][3][4][5][6]

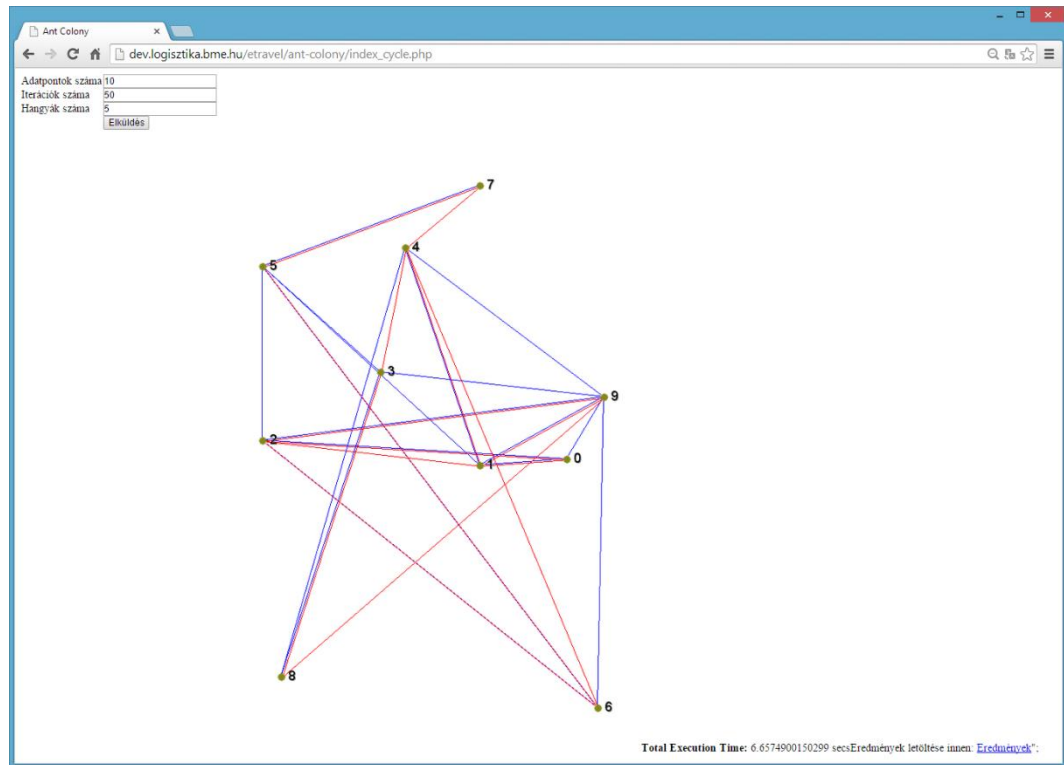
3 A hangya algoritmus implementálása

Az algoritmust az eddigiekben arra használták elsősorban, hogy a szállítási problémára adjanak megoldást, azaz egy adott feladóponttól hogyan lehet a leghatékonyabban eljuttatni az feladóponttól a célpontig. Ezért az útvonalkereséshez kis mértékben módosítani kellett az algoritmust.

Az algoritmus jelenleg egy véletlenszerűen generált hálózaton működik, így minden futáskor egy újabb hálózaton fut le a keresés, ezzel hatékonyan vizsgálható a működés. A hálózatban lévő pontok távolsága szimmetrikus, azonban a kapcsolati mátrix nem szimmetrikus. Az első futásnál a feromon mátrix csak 1-es elemeket tartalmaz, így minden pont azonos mértékben vonzó.

A tesztrendszer implementálása „php” nyelven történt, hogy a későbbiekben könnyen és jól bővíthető rendszert kapjak. Így lehetőség van a későbbiekben egy valós hálózatra is ráilleszteni a működést. A korábbi években a Dijkstra és az A* algoritmusokkal foglalkoztam, ahol már felépítettem egy teszt hálózatot [7]. A későbbiekben az ehhez való illeszkedést tartottam figyelembe. Ez a hálózat egy multi modális hálózat, ahol különböző tömegközlekedési szolgáltatók járatai kerültek felvitelre. Az algoritmus megtervezésénél szem előtt tartottam, hogy a későbbiekben ezen illetve ehhez hasonló hálózatokon kell egy útvonaltervező algoritmusnak működnie, ahol is több tömegközlekedési ág és szolgáltató illetve az egyéni közlekedési lehetőségek hálózatán kell tervezni. Jelenleg a tesztelés során a megfelelő paraméterek megtalálása volt a legfontosabb, így maradtam a véletlenszerű hálózatnál, azonban ezt az irányt végig figyelembe kellett vennem.

A tesztrendszer a következőképpen néz ki (1. ábra). 3 bemenő paraméter adható meg, az adatpontok száma, az iterációk száma és a hangyák száma. A leggenerált hálózat látható középen, az élek színei azt jelentik, hogy melyik irányba járható az összeköttetés. A „kék” szín jelenti az egyik irányt, a „piros” pedig a másikat, ahol mind a két színű él látható, ott két irányú az összeköttetés, ahol pedig csak az egyik, ott vagy az egyik, vagy a másik irányba. Jelenleg a kiinduló pont mindig a nullás csomópont, a célpont pedig a 9-es. A program számolja a futáshoz szükséges időt, ez látszik alul, illetve a paramétervizsgálathoz generál egy Excel-t, mely letölthető a további elemzésekhez.



1. ábra Tesztrendszer

A program első lépésben a különböző mátrixok legenerálását végzi el. Legelőször a távolságmátrix készül el, úgy, hogy minden pont x és y koordinátájához 0 és 100 közötti random számok generálódnak, majd ezek kerülnek elosztásra százzal, így 0 és 1 közötti számok várhatóak 2 tizedesjegy pontossággal. +

$$cords_{x_i} = \frac{rand(0,100)}{100} \quad (6)$$

Így rendelkezésre állnak a pontok koordinátái, majd kiszámolásra kerülnek a távolságok, melynek eredménye bekerül a távolságmátrixba

$$d_{i,j} = \sqrt{(cords_{x_i} - cords_{x_j})^2 + (cords_{y_i} - cords_{y_j})^2} \quad (7)$$

Példát pedig a 2. ábra mutat 10 csomópont esetére.

d_{ij}	0	1	2	3	4	5	6	7	8	9
0	0	0,6801	0,3956	0,5660	0,7430	0,4280	0,3996	0,4518	0,5948	0,3105
1	0,6801	0	0,3231	0,3523	0,7155	0,5341	0,5743	0,8438	0,2110	0,3864
2	0,3956	0,3231	0	0,4220	0,7603	0,4657	0,4810	0,7000	0,1992	0,2062
3	0,5660	0,3523	0,4220	0	0,3700	0,2236	0,2729	0,5412	0,4785	0,2843
4	0,7430	0,7155	0,7603	0,3700	0	0,3257	0,3444	0,4472	0,8465	0,5731
5	0,4280	0,5341	0,4657	0,2236	0,3257	0	0,0500	0,3195	0,5968	0,2631
6	0,3996	0,5743	0,4810	0,2729	0,3444	0,0500	0	0,2731	0,6243	0,2751
7	0,4518	0,8438	0,7000	0,5412	0,4472	0,3195	0,2731	0	0,8698	0,5025
8	0,5948	0,2110	0,1992	0,4785	0,8465	0,5968	0,6243	0,8698	0	0,3680
9	0,3105	0,3864	0,2062	0,2843	0,5731	0,2631	0,2751	0,5025	0,3680	0

2. ábra Távolságmátrix példa

Hasonlóképpen generálódik a kapcsolati mátrix, itt is 0 és 75 közötti értékek között generál véletlenszerű értékeket, így a tapasztalatok alapján megfelelő a hálózat sűrűsége. Ezután következik egy százzal osztás, és végül egy kerekítés, így 0 és egy közötti számok kerülnek be a mátrixba.

$$connection_{i,j} = round\left(\frac{rand(0,75)}{100}\right) \quad (8)$$

Példát pedig a mutat 10 csomópont esetére.

$connection_{i,j}$	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	0	0	0	1	0
2	1	0	1	1	1	0	1	1	1	0
3	1	0	1	0	1	1	0	1	1	1
4	0	1	1	1	0	1	1	0	1	1
5	1	1	1	0	0	1	1	1	0	1
6	0	1	1	1	1	1	0	0	0	0
7	0	0	1	1	0	1	1	1	0	1
8	1	1	0	0	0	1	0	1	1	0
9	1	1	1	0	0	1	0	1	1	0
10	1	0	1	1	0	1	1	1	1	0

3. ábra Kapcsolati mátrixra példa

A feromon mátrix pedig az első lépésben 1-el van végig feltöltve.

$$\tau_{i,j} = 1 \quad (9)$$

Az iteráció pedig a következőképpen működik (4. ábra):

1.	For $i := 0$ to iteration	<i>Iteráció kezdete</i>
2.	$\Delta\tau$ törlése	
3.	For $ant := 0$ to $NrAnt$	
4.	$Route_{[ant][0]} := Start$	
5.	$Length_{[ant]} := 0$	
6.	$Length_{min[ant]} := Végtelen$	
7.	For $r := 0$ to $NrOfDatapoints$	
8.	$p_{denominator_{[ant][r]}} := 0$	
9.	For $t := 0$ to $NrOfDatapoints$	<i>Valószínűségi mátrix nevezőjének számítása</i>
10.	$r \neq t$	
11.	$p_{denominator_{[ant][r]}} := p_{denominator_{[ant][r]}} + \tau_{[r][t]}^\alpha \cdot \left(\frac{1}{distance_{[r][t]}}\right)^\beta \cdot connection_{[r][t]}$	<i>Szumma</i>
12.	For $r := 0$ to $NrOfDatapoints$	
13.	$p_{max[r]} := 0$	
14.	For $s := 0$ to $NrOfDatapoints$	
15.	$r \neq s$	
16.	$p_{[ant][r][s]} := \frac{\tau_{[r][s]}^\alpha \cdot \left(\frac{1}{distance_{[r][s]}}\right)^\beta \cdot connection_{[r][s]}}{p_{denominator_{[ant][r]}}}$	$p_{[ant][r][s]} := 0$
17.	$p_{max[r]} := p_{max[r]} + p_{[ant][r][s]}$	
18.	unset <i>route</i>	
19.	$route_{[ant][0]} := start$	
20.	$length_{[ant]} := 0$	
21.	For $step := 0$ to $NrOfDatapoints$	
22.	$roulette := 0$	
23.	$selection := \frac{rand(0, round(p_{max[rown]} \cdot 99))}{100}$	
24.	$s := 0$	
25.	$roulette \leq selection$ vagy $in_{array}(s - 1, route_{[ant]})$	
26.	$row := route_{[ant][step]}$	
27.	$roulett := roulett + p_{[ant][row][s]}$	
28.	$s := s + 1$	
29.	$s > NrOfDatapoints$	
30.	$length_{[ant]} := length_{[ant]} + (length_{[ant]} + 1)^2$	
31.	Break	
32.	$route_{[ant][step+1]} := s - 1$	
33.	$startPoint := route_{[ant][step]}$	
34.	$endPoint := route_{[ant][step+1]}$	
35.	$length_{[ant]} := length_{[ant]} + distance_{[startPoint][endPoint]}$	

36.		$\Delta\tau_{[ant][startPoint][endPoint]} := \frac{Q}{length_{[ant]}}$	
37.		$length_{min[ant]} > length_{[ant]}$	
38.		$length_{min[ant]} := length_{[ant]}$	
39.		$route_{best[ant]} := route$	
40.	For $r := 0$ to $NrOfDatapoints$		
41.	For $s := 0$ to $NrOfDatapoints$		
42.	$r \neq s$		
43.	$sumOf\Delta\tau_{[r][s]} := 0$		
44.	For $ant := 0$ to $NrOfAnt$		
45.	$isset \Delta\tau_{[ant][r][s]}$	$sumOf\Delta\tau_{[r][s]} := 0$	
46.	$sumOf\Delta\tau_{[r][s]} := sumOf\Delta\tau_{[r][s]} + \Delta\tau_{[ant][r][s]}$		
47.	For $r := 0$ to $NrOfDatapoints$		
48.	For $s := 0$ to $NrOfDatapoints$		
49.	$r \neq s$		
50.	$\tau_{[r][s]} := \tau_{[r][s]} \cdot (1 - \varrho) + sumOf\Delta\tau_{[r][s]}$		

4. ábra Az Hangyakolónia algoritmus működése

Mint ahogy látható, az algoritmus egy nagy iteráció, mely az egyes lépések során kiszámolja az épp aktuálisan kapott legjobb utakat, majd ezt a folyamatot addig folytatja, amíg egy előre meghatározott szám szerint (*iteration*) le nem fut a számítás.

Első lépésben (2) a $\Delta\tau$ törlésre kerül, melyben majd a feromon mennyisége fog tárolódni.

A (3)-as lépésben létrehozásra kerülnek a hangyák, amik az útvonalakat fogják bejárni. Mindig az $NrAnt$ nevű változó értékében tárolt számú hangya jön létre. (4)-től (6)-ig a *Route*, a *Length* és *Length_{min}* tömbök kapják meg a megfelelő kezdő értéküket.

Ez után a hangyák választási valószínűségének számítása következik, mely a (2)-es egyenleten alapul. A számítás praktikus okokból számlálóra és nevezőre lett bontva, így elsőként (7)-től (11)-ig a nevező számítása történik meg, mely a következőképpen írható fel:

$$p_{nevező_{r,t}} = \sum_{t=1}^n (\tau_{r,t})^\alpha \cdot (\eta_{r,t})^\beta \quad (10)$$

ahol $\eta_{r,s} = \frac{1}{\text{távolság } r. \text{ és } t. \text{ pont között}}$, r és t pedig a csomópontok mátrix azonosítói.

A számítás eredménye például 10 csomópont esetén a következő vektor:

$$p_{nevező_{r,t}} = \begin{bmatrix} 4,0991319574649 \\ 10,4243351300520 \\ 9,8400572193381 \\ 10,5742047365380 \\ 9,7615805339839 \\ 6,7004995669906 \\ 11,6479280690660 \\ 5,4177293877860 \\ 8,0808305145619 \\ 12,7881196984000 \end{bmatrix} \quad (11)$$

A hányados számítása pedig a következőképpen néz ki:

$$p_{r,t,s} = \frac{(\tau_{r,t,s})^\alpha \cdot (\eta_{r,t,s})^\beta}{p_{nevező_{r,t}}} \quad (12)$$

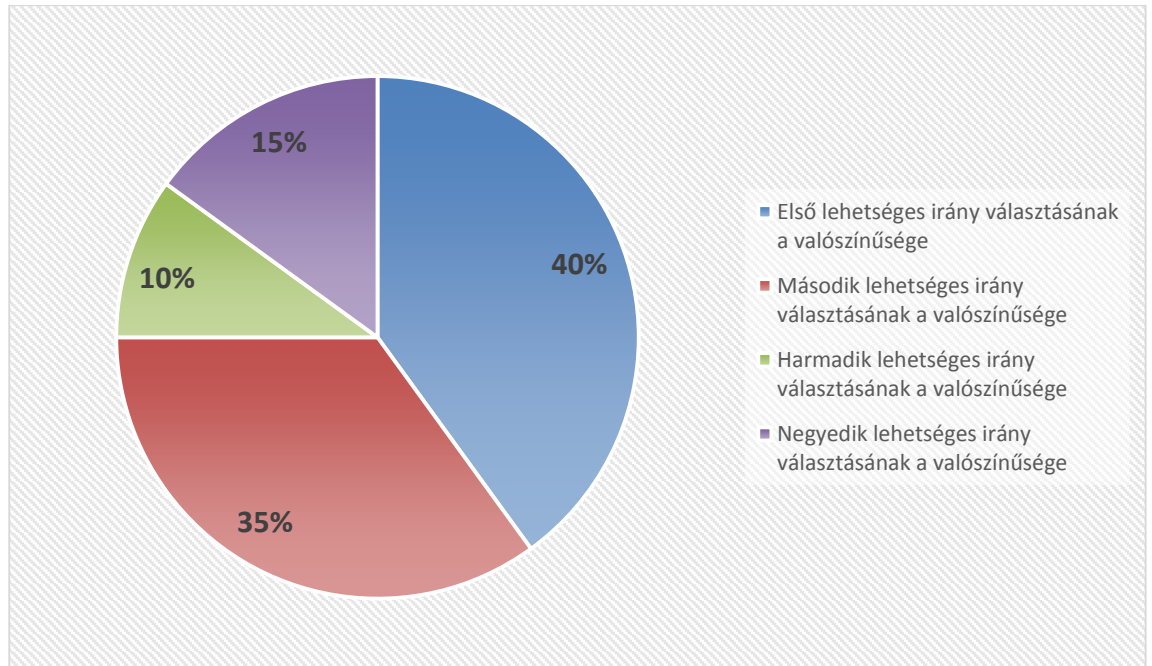
Melynek eredménye a valószínűségi mátrix, amelyre egy példa látható a következő ábrán (5. ábra)

$p_{r,t,s}$	1	2	3	4	5	6	7	8	9	10
1	0	0,2958	0,3879	0	0	0	0	0	0,3163	0
2	0,1163	0	0,1688	0,1616	0,1134	0	0,1266	0,1044	0,2089	0
3	0,1616	0	0	0	0,1166	0,1489	0	0,1215	0,2277	0,2238
4	0	0,1593	0,1456	0	0	0,2000	0,1810	0	0,1367	0,1774
5	0,1188	0,1211	0,1175	0	0	0,1795	0,1746	0,1532	0	0,1353
6	0	0,2042	0,2187	0,3156	0,2615	0	0	0	0	0
7	0	0	0,1238	0,1643	0	0,3839	0	0,1643	0	0,1637
8	0,2746	0,2009	0	0	0	0,3265	0	0	0,1979	0
9	0,1605	0,2694	0,2772	0	0	0,1602	0	0,1327	0	0
10	0,1403	0	0,1722	0,1467	0	0,1525	0,1491	0,1103	0,1289	0

5. ábra Valószínűségi mátrixra példa

A (18)-tól a (20)-as lépésig az útvonalhoz szükséges változók kezdő értékei vannak beállítva. Ezt követően a „következő lépést” számító programrészlet következik (21)-(39). Itt egy úgynevezett Rulett eljárás került alkalmazásra, melynek lényege, hogy a rulett kerékhez hasonlóan generál százalékos értékeket, és megnézi, hogy a megadott érték benne található-e ebben a területben. Jelen esetben a rulett kerék mezői az adott pontból elérhető pontok valószínűség értékeivel arányosak, így mindig van annak valószínűsége, hogy az adott pontból a legkisebb valószínűségű irányt válassza a hangya, de mégis a legnagyobb valószínűségű választása az esélyesebb, ahogy a valóságban is. A példa kedvéért legyen a gráfnak egy olyan pontja, ahonnan 4 pont érhető el, melyeknek a választási

valószínűsége legyen: **40%, 35%, 10%** és végül 15%, ekkor a „rulett kerék” a következőképpen néz ki (6. ábra).



6. ábra „Rulett kerék” példa

A futások során a *row* változóba belekerül a hangya adott lépéséhez, hogy ott melyik gráf ponton haladt keresztül. Amennyiben az *s* elérné a maximális pontok számát (*NrOfDatapoints*) (29), akkor leállítja a ciklust, így nem tud a végtelenig menni, illetve ezen éleknek a távolságához egy nagy érték állítódik be (30), így gyakorlatilag nulla valószínűséggel kerülhetnek kiválasztásra. Ez után a *route* változó megfelelő értékébe belekerül, hogy hány lépésben érte el a hangya a végcél, illetve a kezdő és végpontok letárolódnak a *startPoint* és *endPoint*-ba, kiszámolásra kerül a *length* változóba hangyánként a megtett út, és frissítésre kerül hangyánként, kezdő és végpontonként a feromon értéke (32)-(36). Majd pedig ha az épp aktuális hossz kisebb, mint az eddigi minimum, akkor ennek az útvonalnak a tartalma és hossza kerül bele a minimális útvonalhosszba (38) és a legjobb útba (39).

A következőkben pedig a τ mátrix kerül frissítésre (40)-(50).

4 Futási eredmények

A dolgozat keretein belül azt vizsgáltam, hogy a különböző paraméterek, melyek meghatározzák az algoritmus működését, hogyan is változtatják meg a futási eredményeket.

Paraméternek felvettem a következő értékeket és a következő határok között változtattam őket:

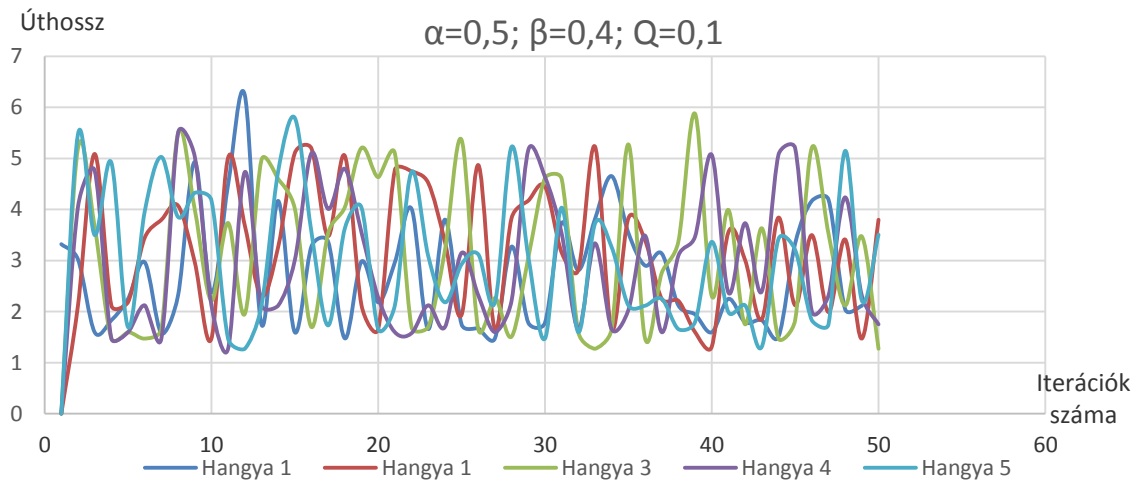
Paraméter	Érték tartomány
<i>NrOfDataPoints</i>	10 ... 100
<i>iterations</i>	50 ... 100
<i>NrOfAnts</i>	5 ... 20
α	0,5 ... 3,5
β	0,2 ... 2
Q	0,1 ... 0,5
ϱ	0,5

1. táblázat A vizsgálandó paraméterek tartományai

4.1 α , β , Q vizsgálata (10 adatpont, 5 hangya, 50 iteráció)

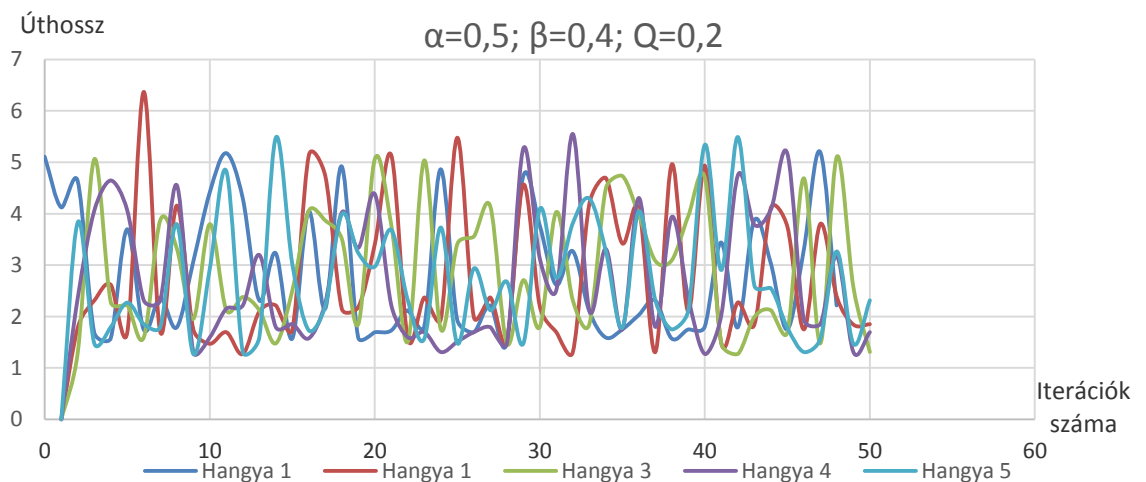
A futtatások során az α , β , Q és a ϱ voltak a futó változók, melyeket az egyes futások során paraméterként változtat a program. Első lépésben a Q változó értékét fogom változtatni a többit pedig változatlanul hagyom.

Az első futás eredménye (7. ábra):



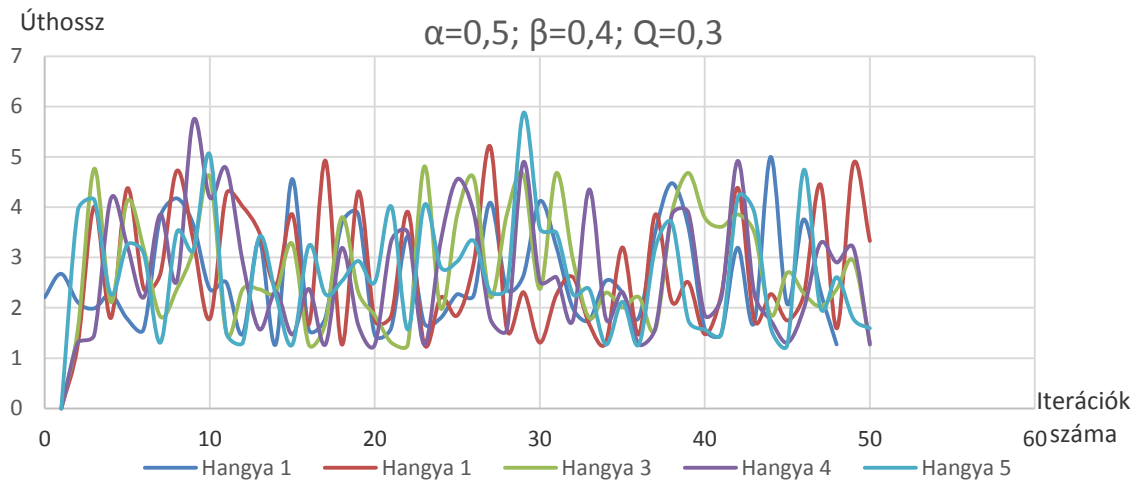
7. ábra $\alpha = 0,5; \beta = 0,4; Q = 0,1$ paraméterek diagramja

A diagramon az öt hangya által az iterációk során megtett út látható. Ideális esetben a hangyáknak az 50. iterációnál a legkisebb értékhez kellene tartaniuk, ami azt jelentené, hogy megtalálták a legrövidebb utat a kiinduló és célpont között. Az első futásnál az látható (7. ábra), hogy az úthosszak teljesen hektikusan változnak az iterációk során, így nem találják meg a hangyák a legjobb utat, teljesen esetlenek a választásaik.



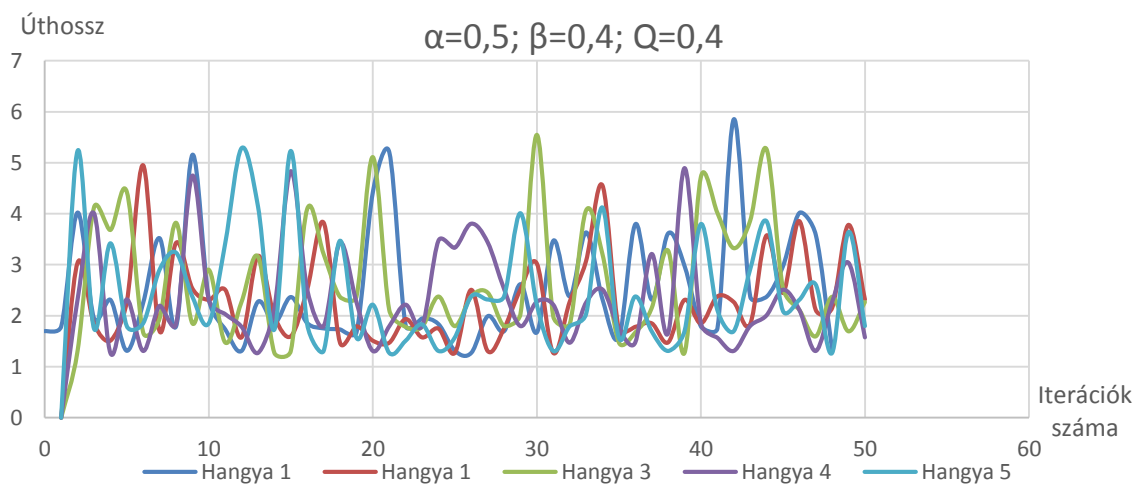
8. ábra $\alpha = 0,5; \beta = 0,4; Q = 0,2$ paraméterek diagramja

A következő esetben (8. ábra) kis mértékben javulnak az eredmények, de továbbra is széttartóak a görbék.



9. ábra $\alpha = 0,5; \beta = 0,4; Q = 0,3$ paraméterek diagramja

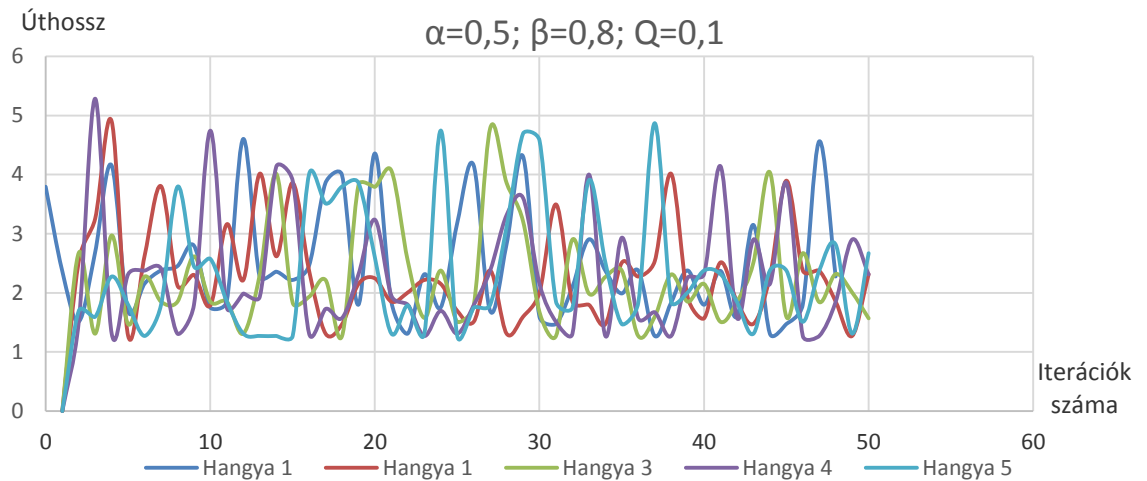
Az 9. ábra alapján elmondható, hogy a Q további növelésével kis mértékben tovább simíthatóak a görbék.



10. ábra $\alpha = 0,5; \beta = 0,4; Q = 0,4$ paraméterek diagramja

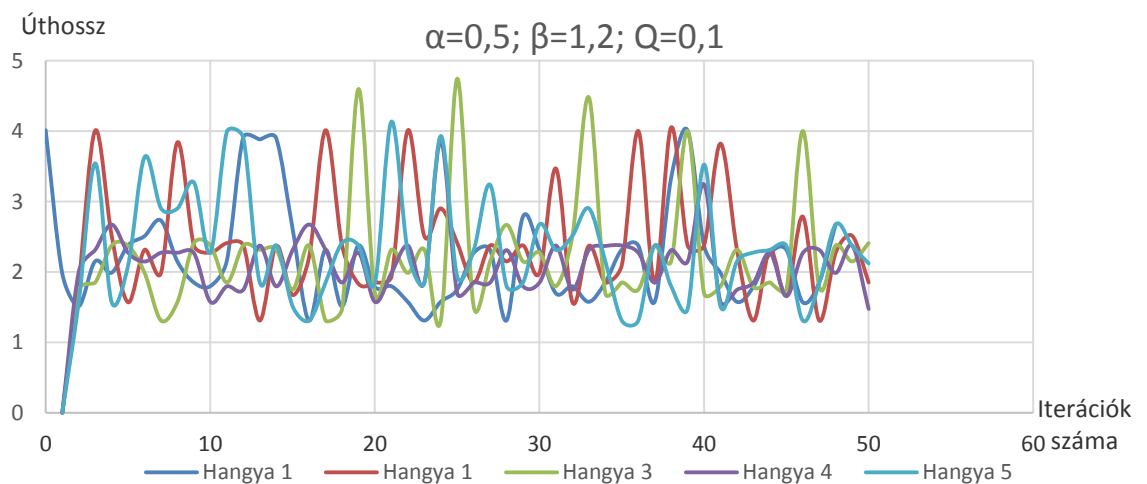
A $Q = 0,4$ értéknél további kisebb mértékű javulás tapasztalható (10. ábra).

A következőkben a β értékének változását fogom vizsgálni. Az könnyebb összehasonlíthatóság okán a $Q = 0,1$ értéknél fogom nézni a β változásának hatását.



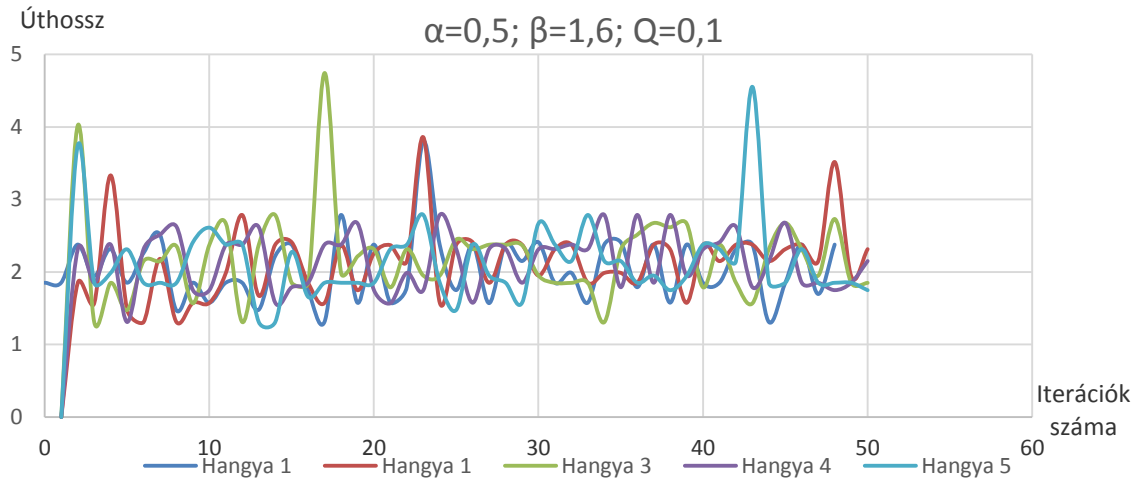
11. ábra $\alpha = 0,5; \beta = 0,8; Q = 0,1$ paraméterek diagramja

A diagramon (11. ábra) látható, hogy a β növelésével a diagram esetén nagyobb javulás érhető el. Tovább növelve az értéket a következő eredmények születtek (12. ábra)



12. ábra $\alpha = 0,5; \beta = 1,2; Q = 0,1$ paraméterek diagramja

Megfigyelhető, hogy az úthosszak lényegesen, egy egész értékkel csökkentek, azonban az eredmények továbbra is hektikusak, tehát továbbra sem stabil az adott út.

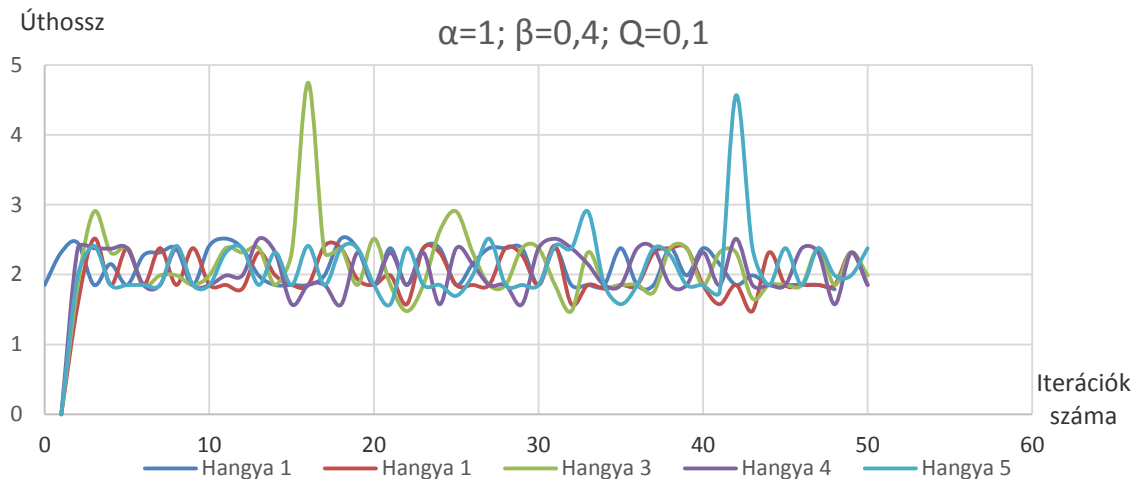


13. ábra $\alpha = 0,5; \beta = 1,6; Q = 0,1$ paraméterek diagramja

Az előbbi ábra (13. ábra) is abba a tendenciába illik, hogy a β növelésével a minimum értékeket egyre jobban megközelítő értékeket kaphatunk.

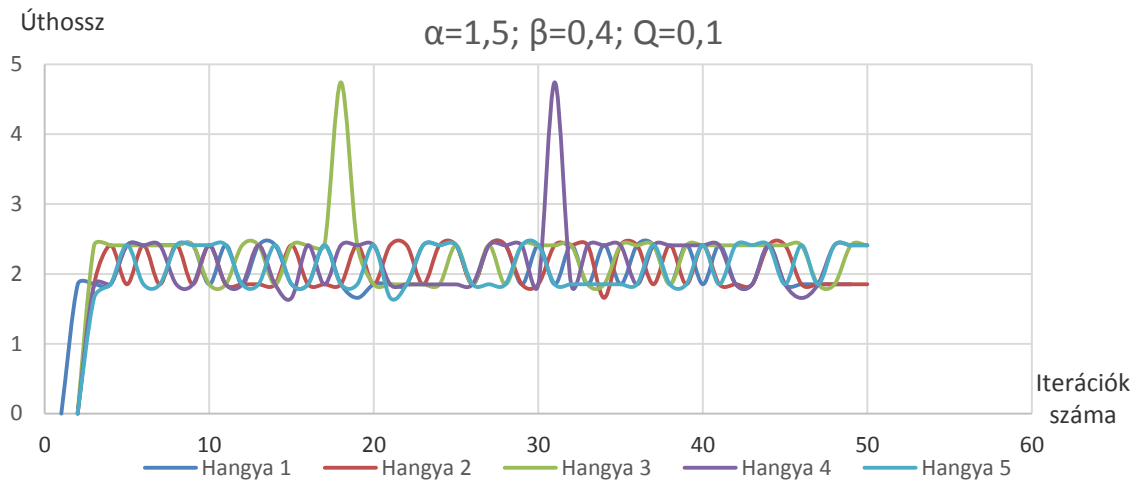
A következő lépésben az α értékét fogom változtatni. Hasonlóan a β értékéhez itt is a többi értéket visszaállítom az alap értékre és így vizsgálom az α hatását.

Az α első értékének az 1-et választottam, az ehhez tartozó diagram pedig a következő ábrán látható (14. ábra)



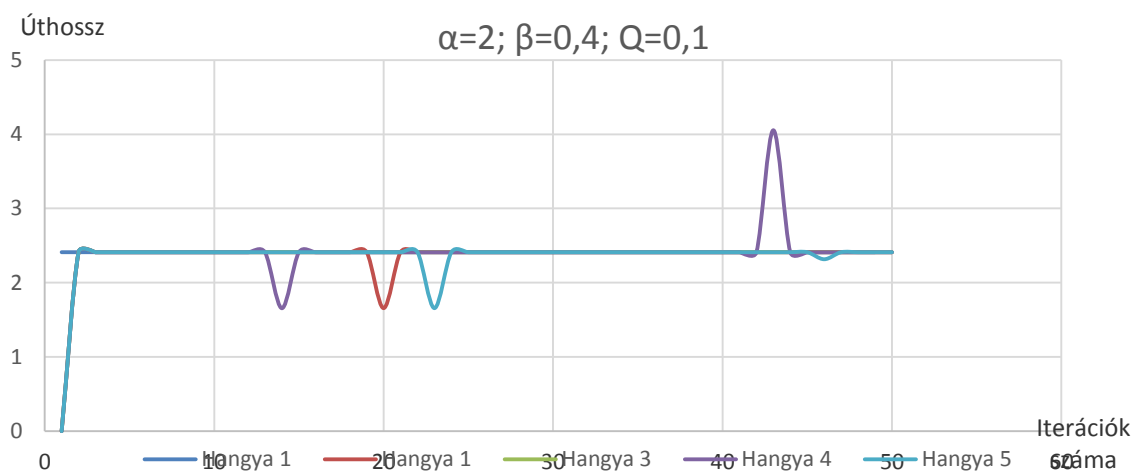
14. ábra $\alpha = 1; \beta = 0,4; Q = 0,1$ paraméterek diagramja

Látható, hogy az α növelése lényegesen nagyobb mértékben hatott az eredményekre. Növelésével néhány kiugró értéket leszámítva lényegesen stabilabb a folyamat. A kiugró értékeket leszámítva a maximum és minimum értékek közötti különbség 1-re csökkent. Így tovább növeltem az α -t (15. ábra)



15. ábra $\alpha = 1,5; \beta = 0,4; Q = 0,1$ paraméterek diagramja

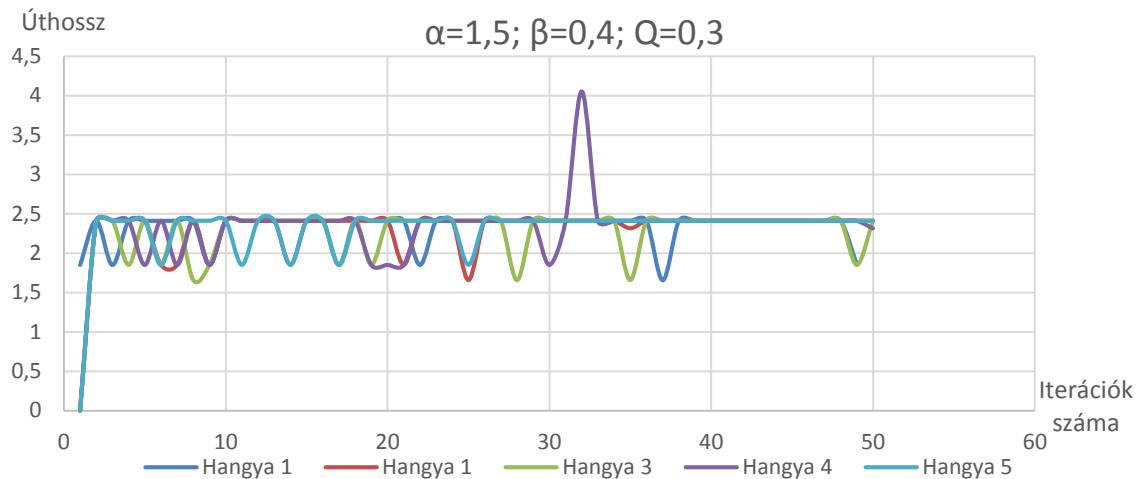
Az értékekből látható, hogy tovább csökkent az értékek szórása, tehát egyre stabilabbá válnak az futási eredmények. Így tovább növeltem az α értékét, amely hatását mutatja a 16. ábra.



16. ábra $\alpha = 2; \beta = 0,4; Q = 0,1$ paraméterek diagramja

Érdekes eredmény, hogy ezzel a növeléssel teljesen stabillá váltak az értékek, azonban a minimumhoz képeset egy magasabb értéken állandósult az útvonal. Néhány iteráció során a hangyák megtalálták a rövidebb utat, azonban a másik út feromon felhője túl erős volt, így az elnyomta a rövidebb út feromonját. Tovább növelve az értéket hasonló eredmények kaphatóak, így az α értékét nem érdemes tovább növelni.

Az eddigiek alapján a legjobb paraméter kombináció a mutatja a 17. ábra



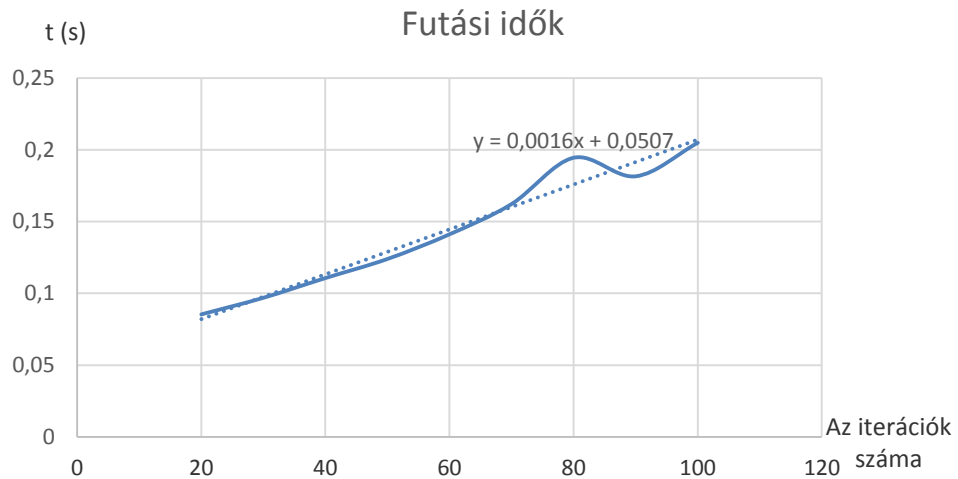
17. ábra $\alpha = 1,5; \beta = 0,4; Q = 0,3$ paraméterek diagramja

Ezen paramétereknél még az algoritmus megfelelő biztonsággal megtalálja a minimum értékeket, azonban elegendően stabil is az eredmény, kis mértékű az eredmények szórása.

4.2 Vátozó hangya, iteráció és adatpontszám

A 4.1-es fejezetben az α , β , Q változását vizsgáltam, mely paraméterek a legjobb útvonal megtalálásának valószínűségét szabályozzák. Azonban az algoritmus másik fontos tulajdonsága még a futási idő, illetve ezen belül az, hogy legalább hány iteráció szükséges a legjobb útvonal megtalálásához. Ebben a fejezetben ezen futási eredményeket fogom bemutatni. Változtatni fogom az adatpontok számát, az iterációk számát és a hangyák számát. Le fogom jegyezni a futási időket, majd ezeket fogom diagramban ábrázolni.

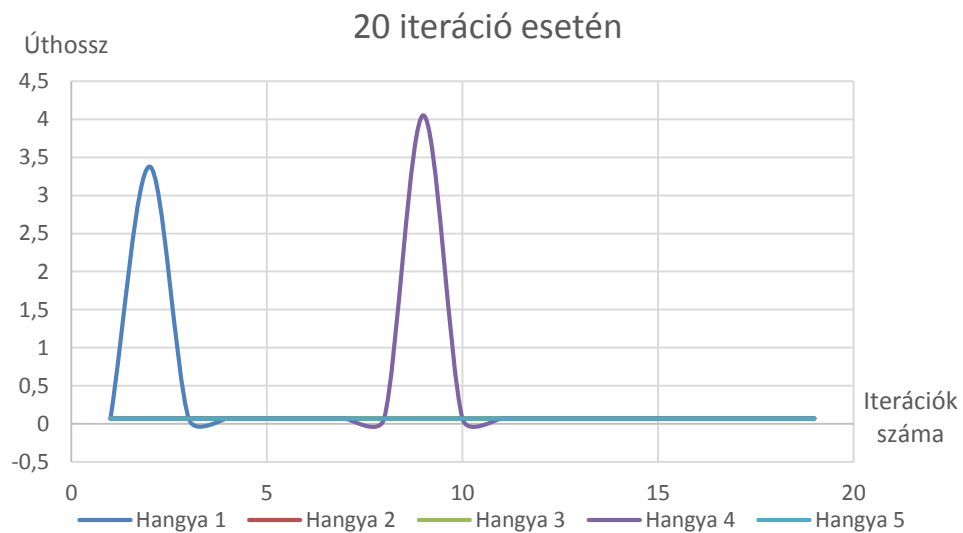
Első lépésben az iterációk számának változását fogom vizsgálni, itt figyelni fogom azt is, hogy hány iteráció szükséges ahhoz, hogy a megtalálható legyen a legjobb eredmény. A futások során most az adat pontok számát 10-re állítottam, a hangyák számát pedig 5-re. Ekkor a következő értékeket kaptam (18. ábra)



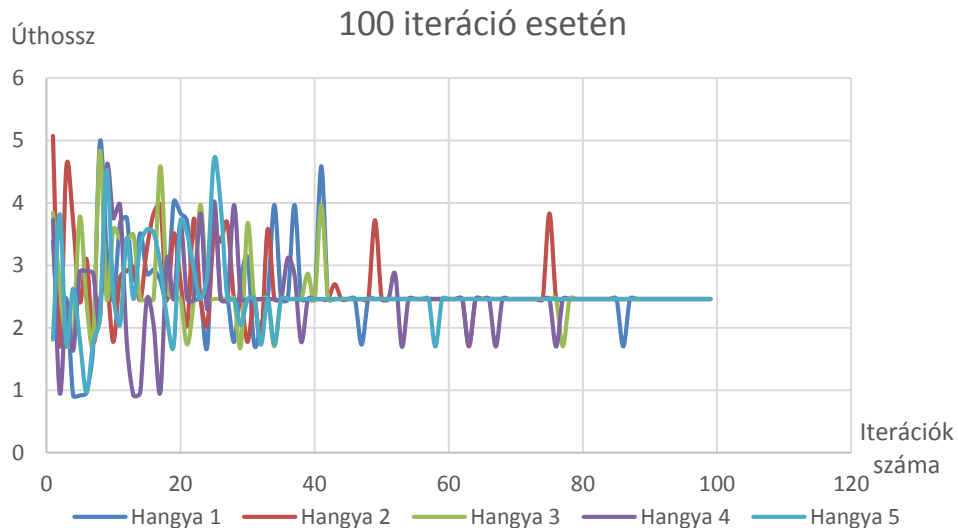
18. ábra Futási idők az iterációk függvényében

Látható, hogy az eredmények alapján, hogy az iteráció növelése lineárisan hat a futási időre, mely igen kedvező, mert nagy hálózatok esetén az erőforrások korlátai végesek, és könnyen elérhetőek.

Fontos még itt megvizsgálni, hogy az iteráció növelése milyen hatással van az eredményekre. Ehhez a következő diagramokat érdemes megvizsgálni (19. ábra és 20. ábra)



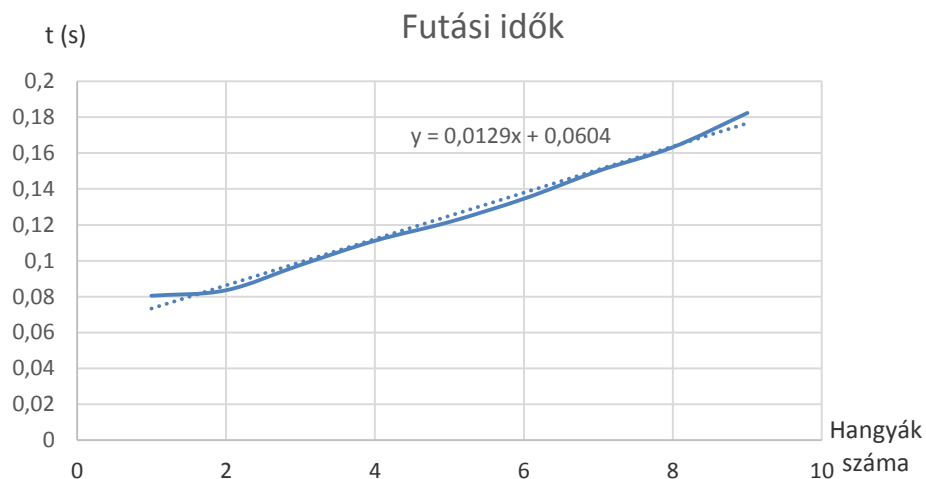
19. ábra Futási eredmények 20 iteráció esetén



20. ábra Futási eredmények 100 iteráció esetén

A diagramokról leolvasható, hogy a nagyobb iterációs szám lényegesen stabilabb eredményeket szolgáltat, és mivel lineáris összefüggést tapasztaltam ezen a téren, ezért érdemes növelni az iterációk számát még nagy hálózatok esetén is.

A következő kérdéskör a hangyák száma. A hangyák számát 1 és 10 között változtattam, és ismét feljegyeztem a futtatási időket. Az iterációk számát a korábbiakhoz hasonlóan 50-re állítottam (21. ábra).

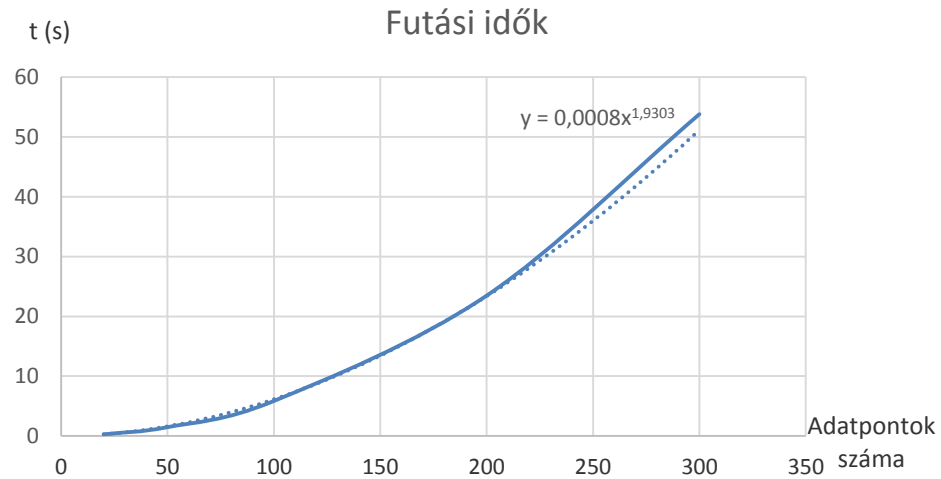


21. ábra Futási idők a hangyák számának függvényében

Itt is lineáris összefüggés mutatkozik a futási idők és a hangyák számának növelésekor. Azonban itt meg kell jegyezni, hogy az algoritmus jelen állapotában 1 programszálként fut, azonban a hangyák esetén érdemes a többszálú futtatást is lemodellezni. A

TDK keretén belül csak az egy szálú futtatással foglalkoztam így a futási idő növekedése az elvártaknak megfelelően alakult, azonban többszálú futtatásnál a hangyák számának növelése várhatóan minimális futási idő növekményt fog okozni.

Végezetül pedig az adatpontok számának a hatását vizsgáltam a futási idők függvényében. Itt az adatpontok számát 10-től 300-ig változtattam (22. ábra).



22. ábra Futási idők az adatpontok számának függvényében

Az eddigiektől eltérően az adatpontok számának változása sokkal nagyobb hatása van a futási időre, mint a többi paraméternek. Ez azért is fontos, mert egy valódi hálózaton való modellezésnél az adatpontok száma hatalmas, a terület növelésével, főleg a nagy városok belefoglalásával exponenciálisan növekszik. Az eredmények alapján látható, hogy a futási idő hatványozottan nő az adatpontok növelésével, így itt van igazán nagy jelentősége a többszálúsítás lehetőségének, melyet a hangyák számának vizsgálata során említettem.

5 Érzékenységvizsgálat

5.1 Az érzékenységvizsgálat elmélete

Egy érzékenységvizsgálat arra keresi a választ, hogy egy paraméterektől függő mennyiség paramétereinek változása milyen hatással van a teljes végeredményre. Általában a gazdasági területen alkalmaznak ilyen vizsgálatot, ahol a különböző előretekintő becsléseknél vizsgálják, hogy az alap esetben alkalmazott legjobb becsléstől való eltérés, illetve az eltérések bekövetkezésének valószínűsége milyen hatással van a teljesítménymutatók alakulására, általában a projekt nettó jelenértékére. Az ilyen vizsgálatok oka, hogy a jövőre vonatkozóan kell például egy beruházás pénzügyeit előre jelezni, ami bármilyen alapos előkészítés mellett is gyakorlatilag csak egy jóslás, így vizsgálni kell más forgatókönyveket is. Mivel a jövő nagy bizonytalansággal becsülhető, ezért az érzékenység vizsgálatával arra keresik a választ, hogy a különböző bemenő paraméterek értékeinek változása milyen változást hoz a projekt megtérülésével kapcsolatosan, vagyis az adott paraméterre mennyire „érzékeny” a megtérülés. Általában a vizsgálat célja, hogy a döntéshozók felmérhessék a kockázatokat, és megalapozott döntést hozhassanak. [7]

Egytényezős érzékenységvizsgálat, melyben az érzékenység rugalmassági mutatókkal van vizsgálva. [7]

$$\text{rugalmassági mutók} = \frac{\text{Magyarázott tényező \% - os változása}}{\text{Magyarázó tényező \% - os változása}} \quad (13)$$

Képlettel leírva

$$\varepsilon = \frac{\frac{\Delta DV}{DV_0}}{\frac{\Delta IV}{IV_0}} = \frac{\frac{DV_1 - DV_0}{DV_0}}{\frac{IV_1 - IV_0}{IV_0}} \quad (14)$$

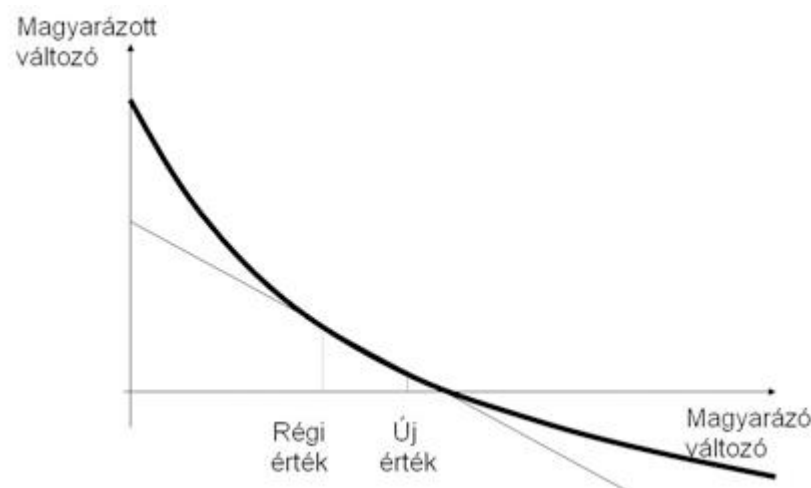
ahol ΔDV a magyarázott tényező változása, ΔIV a magyarázó tényező változása, DV_1 a magyarázott tényező új DV_0 pedig a régi értéke, IV_1 a magyarázó tényező új és IV_0 a régi értéke. [7]

Egy érzékenységvizsgálat első lépése, hogy meg kell vizsgálni a hatástényezőket (kimeneteket), hogy meg lehessen határozni a teljesítménymutatók és a paraméterek közötti kapcsolatot. Ilyenek lehetnek például a piaci részesedés, nagyság, az értékesítési volumen, a növekedési ütem, stb... az ilyen jellemzőket nevezik közvetlen hatótényezők-

nek. Ezt követően a bekövetkezési valószínűséget vizsgálják, majd ez alapján meghatározható a teljesítménymutatók várható értéke. Emellett léteznek a közvetett hatótényezők, melyek hathatnak csak egy paraméterre a többitől függetlenül, vagy akár többre is kihathatnak. Ennek eredményeképpen különböző mértékű hatást fejtenek ki az eredményekre. Az ilyen változóknál sorrendet kell megállapítani, és ez alapján kell az értékelést végezni.

A vizsgálat során először a közvetlen hatótényezők kerülnek tesztelés alá, és megnézzük, hogy a paraméter tekintetében 1%-os változás milyen hatást okoz a végeredmény tekintetében 1%-nál kisebb a változás, akkor nincs szükség további vizsgálatra azon közvetett hatótényezők körében, melyek csak erre hatnak. Azokat a közvetett tényezőket, melyek több közvetlenre is hatnak, kockázatelemzéssel szükséges vizsgálni. [7]

A vizsgálat célja, hogy kiválasztásra kerüljenek azok a kritikus paraméterek, melyek a legnagyobb hatást gyakorolják az alapesetben kiszámítottakhoz képest. A paramétereket az összehasonlíthatóság miatt mindig úgy kell vizsgálni, hogy a többi közben változatlan marad. Azon paramétereket, melyek 1%-os megtérülés esetén legalább ugyanilyen vagy nagyobb mértékű megváltozást okoznak az eredményekben, jellemzően a belső megtérülési rátában vagy a nettó jelenértékben, azokat meghatározó paraméternek tekintjük a vizsgálat szempontjából. Relatív megváltozásnak nevezik a rugalmasságot más néven az elaszticitást. Fontos, hogy a számításokból az egymástól függő értékeket ki kell zárni, mert ez torzítja a végeredményeket. Ezek a lapján a rugalmassági mutatók a következőképp ábrázolhatók (23. ábra) [7]:



23. ábra Rugalmassági mutatók [7]

5.2 Az érzékenység vizsgálat alkalmazása

A jelen problémára alkalmazva meg kell határozni azt a mutatót, amely ebben az esetben az a változó, ami a többi értéktől függ. Erre a legalkalmasabb a $\max$ és $\min$ különbsége, mert ez mutatja azt, hogy az eredmények konvergálnak. Minden futás során felírható, hogy a paraméterek hatására milyen mértékben változik az átlagos különbség, mekkora a szórás és mi volt a két szélsőérték.

Az érzékenységvizsgálatban a különbségre fogom a hangsúlyt fektetni, illetve a minimum fontos még a működés során. Ez alapján felírhatóak a következő egyenletek:

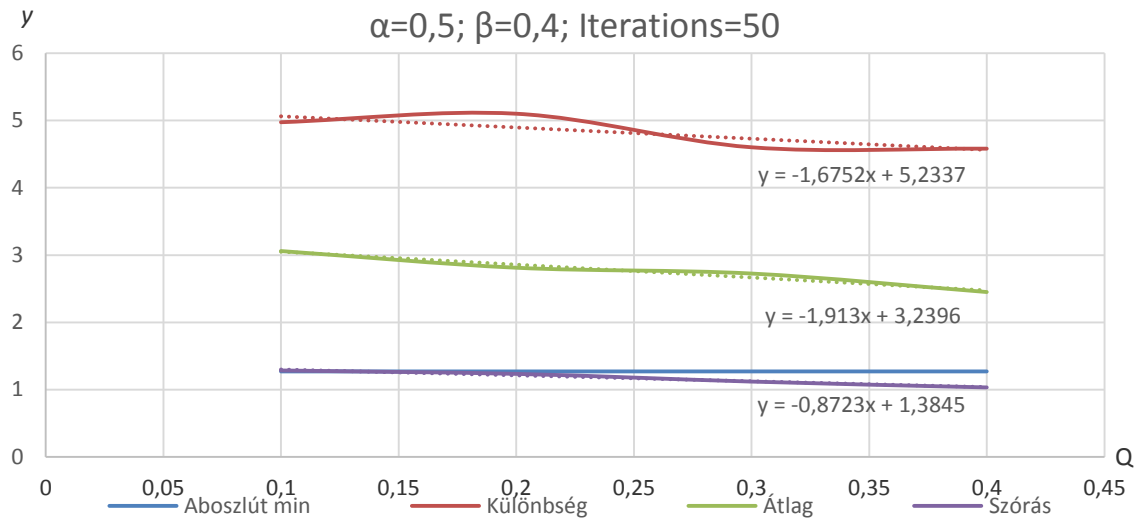
$$\varepsilon = \frac{\frac{\Delta \text{különbség}}{\min}}{\frac{\Delta \alpha}{\alpha_0}} = \frac{\frac{\max - \min}{\min}}{\frac{\alpha_1 - \alpha_0}{\alpha_0}} \quad (15)$$

$$\varepsilon = \frac{\frac{\Delta \text{különbség}}{\min}}{\frac{\Delta \beta}{\beta_0}} = \frac{\frac{\max - \min}{\min}}{\frac{\beta_1 - \beta_0}{\beta_0}} \quad (16)$$

$$\varepsilon = \frac{\frac{\Delta \text{különbség}}{\min}}{\frac{\Delta Q}{Q_0}} = \frac{\frac{\max - \min}{\min}}{\frac{Q_1 - Q_0}{Q_0}} \quad (17)$$

ahol $\max$, $\min$ és a különbség az adott α -hoz, β -hoz, Q -hoz és az iterációk számához tartozó diagramokon lévő megtett utak legnagyobb és legkisebb értékei, illetve ezek különbségei. Az elemzés során a különbség értékén kívül vizsgáltam még az átlagra és a szórásra vett hatást. Így felrajzolhatóak a 4-es bekezdés alapján az érzékenységi függvények.

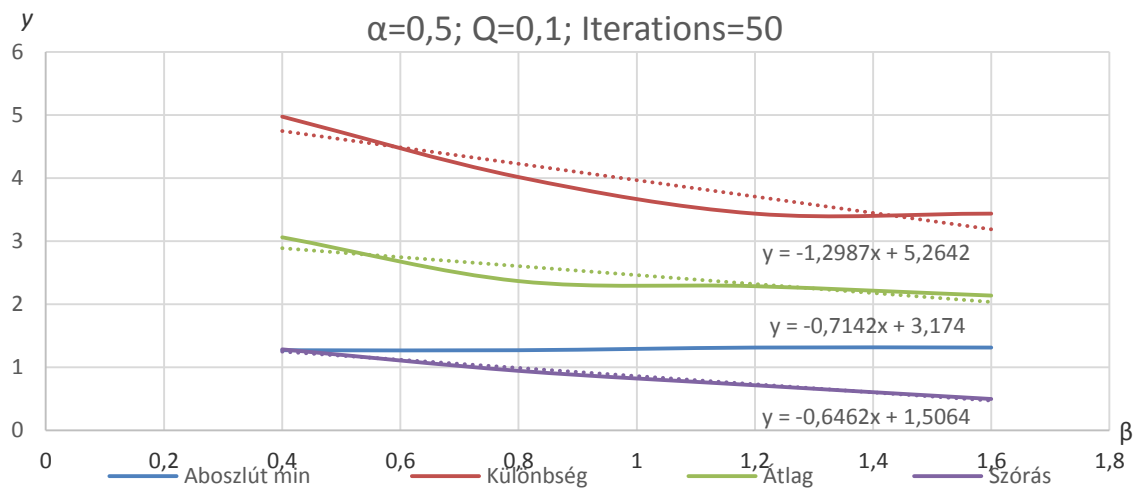
Először a Q hatását vizsgáltam. Az ebből generált diagram látható a következő ábrán (24. ábra).



24. ábra Érzékenységi függvények $\alpha = 0,5, \beta = 0,4$ esetén

Az eredmények alapján látható, hogy a Q változása mind a különbség, az átlag és a szórás tekintetében lineáris összefüggéssel közelíthető. Látható, hogy Q értékének növelése csak kis mértékben javítja az eredményeket. Látható az is, hogy a legkisebb értékhez képest a különbség közel 5-szörös, tehát a Q csak kis mértékben járul hozzá a minimum megtalálásához.

A következő lépésben a β paraméter változását vizsgáltam meg. A kiindulási adatok itt is a 4-es bekezdésben bemutatott eredményeken alapszanak. Az ezeken alapuló diagramot mutatja a 25. ábra.

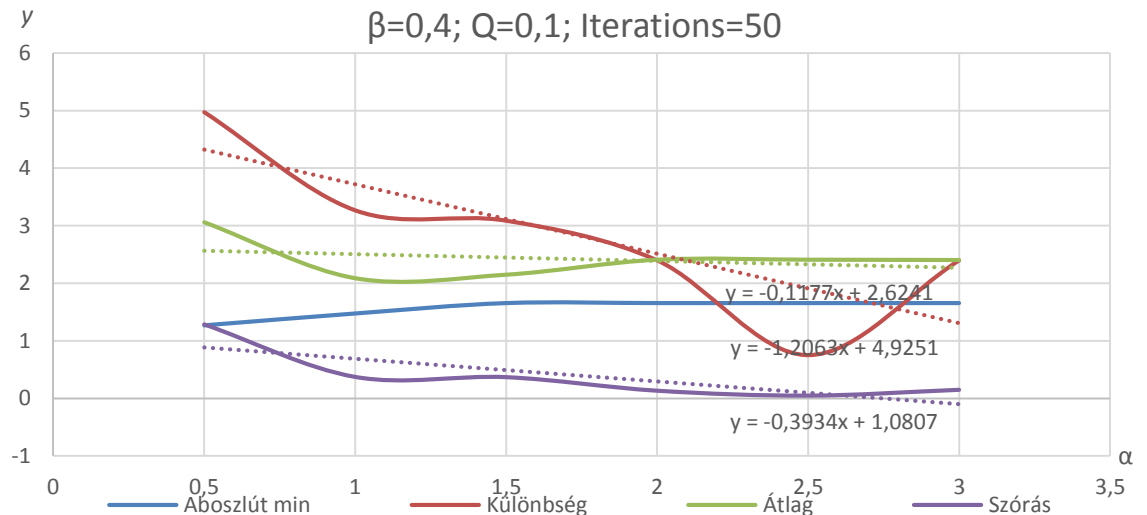


25. ábra Érzékenységi függvények $\alpha = 0,5, Q = 0,1$ esetén

Ahogy a 4-es bekezdésben is arra az eredményre jutottam, hogy a β paraméter lényegesen nagyobb hatással van a végeredményekre, így ez itt is látható. A különbség és

az átlag is lényegesen meredekebb függvényt ad eredményül a Q esetén tapasztalt eredményeknél. Azonban az is látható, hogy a minimum értéket is csak körülbelül 3,5-szere-sére közelíti meg a különbség. Emellett megfigyelhető az is, hogy az 1,2-es érték után a különbség nem javul, de az átlag és a szórás igen, ami azt jelenti, hogy ezen érték felett már nem találja meg a legjobb utat, illetve a hangyák egyre kevésbé indulnak felfedezni.

A következő vizsgált paraméter az α . Ezen értékhez tartozó eredmények láthatóak a következő ábrán (26. ábra)



26. ábra Érzékenységi függvények $\beta = 0,4$, $Q = 0,1$ esetén

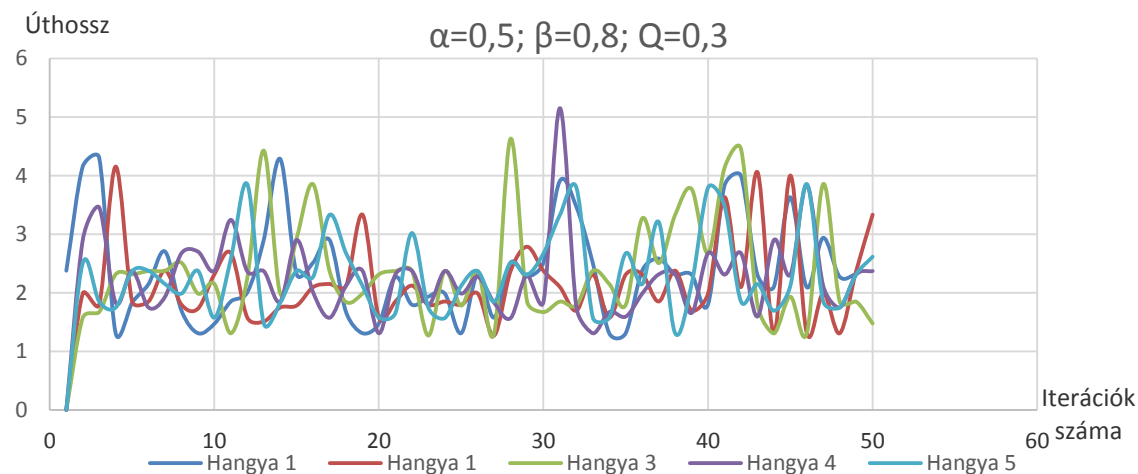
A 4-es bekezdés által előre determinált képet mutatnak az eredmények, tehát az α növelésével csökken a legjobban a különbség, és közelíti meg legjobban a minimumot. Azonban fel kell hívni a figyelmet arra, ahogy növekszik az α , úgy növekszik az abszolút minimum is, ami azt jelenti, hogy a hangyák nem a lokális minimumot találják meg. Így ugyan javul a konvergencia, de csak egy lokális minimum kerül megtalálásra, ezért ez alapján az α értékét nem célszerű növelni.

Ezt figyelembe véve a legjobb paraméterekkel rendelkező beállítás az érzékenységvizsgálat eredményeit figyelembe véve a következő:

Paraméter	Érték tartomány
α	0,5
β	0,8
Q	0,3

2. táblázat A legjobb eredményeket hozó paraméterek

Ezen paraméterekkel a futási eredmények a következők (27. ábra)



27. ábra A legjobb eredményeket szolgáltató paraméterek futási eredménye

A paraméterekhez tartozó statisztikai értékek pedig a következők:

Paraméter	Érték tartomány
Minimális úthossz	1,27069
Különbség	3,88389
Átlag	2,29822
Szórás	0,74838

3. táblázat A legjobb eredményeket hozó paraméterek statisztikai jellemzői

6 Továbbfejlesztési lehetőségek

Az első és legfontosabb továbbfejlesztési lehetőség a többszálú műveletvégzés leprogramozása az algoritmusban. Ez azért rendkívül fontos, hogy a futási idők kordában tarthatóak maradjanak nagy hálózatok esetén is. Egy ilyen útvonaltervező rendszer jellemzőnek rengeteg maggal rendelkező serveres környezetben fut, ahol az erőforrások optimális kihasználása nagyon fontos szempont, és a számítási feladatok jó elosztása igen fontos. Jelen esetben, első sorban a hangyák száma az, ahol a többszálú futás megoldható. Nagyobb hálózatokon a hangyák számának növelése valószínűsíthetően igen fontos lesz, mivel a hálózat bonyolultságának növekedésével igen fontos, hogy a legkisebb út megtalálásának a valószínűségét minél inkább megnöveljem. Az eredmények alapján látható, hogy az iterációk növelése 50 lépés felett már nem növeli lényegesen a megtalálás pontosságát, ezért annak a növelése nem hoz érdemi javulást, illetve nem is célszerű, mivel soros műveletvégzéssel végezhető csak el, így a párhuzamos műveletvégzés nem jöhet szóba.

Ez után egy valós hálózatra kell megvizsgálni az algoritmust, ahol lehetőleg egy széleskörű közlekedési hálózat minden elemén vizsgálható a működés. Mivel a dolgozat célja volt, hogy egy multi modális hálózaton kell majd az algoritmusnak működni, így itt is majd egy ilyen hálózatot kell alkalmazni.

Ez után továbbfejlesztési irány lehet egy valós idejű dinamikus hálózati modell, melyben a felhasználó mozgását szimuláljuk közlekedés közben, és az épp aktuális helyzetnek megfelelően folyamatosan újratervezzük az útvonalat és alternatívákat nyújtunk számára.

Újabb szintlépés lehet nem csak az útvonal megadása, hanem az utazás költségének kiszámítása, mellyel összehasonlíthatóvá válnak a közösségi közlekedés eljutási lehetőségei és az egyéni közlekedés lehetőségei. Itt a legfontosabb megoldandó probléma az adatok beszerzése mellett, az hogy rengeteg árképzési megoldás létezik. Az árképzési megoldások között két szélső helyzet létezik, az egyik véglet az 1 utazásra jogosító jegy, ahol nem függ sem a távolság, sem az idő, a másik véglet pedig hogy a távolsággal, vagy az idővel vagy mindkettővel arányos a díjszabás. Emellett léteznek a bérletek, amelyek egy teljesen külön kategóriát alkotnak. Ezeknél kezelni kell a gyűjtő jegy típusúakat és a havi vagy éves bérleteket is. Továbbá a különböző kedvezményeket, akciókat is figyelembe kell venni. Ebben a kérdéskörben is hatalmas sokszínűséggel lehet találkozni, és nem csak a különböző szolgáltatók alkalmaznak teljesen eltérő kedvezményeket, hanem

a különböző országok is, melyek teljesen eltérő feltételekkel biztosítják azokat. Ebben a kérdéskörben még a közúti költségeket is meg kell jeleníteni, mert nem hagyható figyelmen kívül a megfelelő összehasonlításhoz, hogy az utazásnak nem csak benzin, hanem egyéb költségei is vannak, mint például az autópálya díjak, komp díjak, behajtási díjak, stb., melyek ritkán jelennek meg a különböző algoritmusokban, programokban.

Végezetül pedig a foglalások megoldása az a kérdéskör, melyet egy ilyen rendszernek tartalmaznia kell, ahhoz, hogy a betölteni kívánt célt teljesítse. Itt az online adatok mellett a jegyek eljuttatásának a kérdésköre az igazán izgalmas, mivel sok szolgáltató még nem rendelkezik elektronikus jeggyel, így azt valamilyen formában el kell juttatni a vevőhöz. Természetesen az online rendszerekhez való kapcsolódás és jegyfoglalásnál felmerülő probléma, hogy azonos, vagy adott időben az azonos helyre történő foglalás elkerülhető legyen.

Az eddigiek mellett még további támogató, illetve utazási élményfokozó bővítés kapcsolódhat egy ilyen rendszerhez, mint például az utas folyamatos végigkísérése az autós GPS-hez hasonlóan, mely tájékoztatja, hogy éppen hol van, mennyi a várható odaérési idő, mikor kell jeleznie a leszállást, hogyan kell eljutnia az átszállásnál a másik peronra vagy megállóba, mely lépcsőt használja, stb.

7 Összefoglalás

Összefoglalva a dolgozat tartalmát, bemutattam a Hangyakolónia algoritmust, mely a hangyák élelemszerzési metódusán alapszik, melynek lényege, hogy ha egy hangya élelmet talál egy útvonalon, akkor oda vonzza a többieket, de bizonyos időközönként felfedező útra is indulnak, ezzel biztosítják az esélyét annak, hogy a minimális útvonalat megtalálják.

Ez után az algoritmust leprogramoztam PHP nyelven, bemutattam a programkódot stuktogram segítségével, bemutattam a főbb lépéseket, illetve a teszt környezetet is. Ezen belül bemutatásra került a rulett eljárás is, amivel az adott pontból lehetséges útirányok körül választ az algoritmus.

Ezután bemutattam a futási eredményeket, és vizsgáltam a paraméterek hatását a hangyák által megtett távolságokra. A paraméterek vizsgálatánál úgy jártam el, hogy egyszerre egy paramétert változtattam, a többit pedig változatlanul hagytam, így jól kimutatható volt, hogy egy paraméter változása milyen hatással van a végeredményekre. Vizsgáltam a hangya algoritmust leíró egyenletben szereplő paraméterek hatását, illetve a hangyák-, a futások- és a csomópontok számának hatását. Itt arra az eredményre jutottam, hogy az α -nak van a legnagyobb hatása, azonban a minimum megtalálásának az esélye csökken a növelésével. Emellett az adatpontok számának a növelése egy kritikus tényező, mivel exponenciálisan nő a számítási idő az adatpontok számának a növelésével.

Végezetül pedig elvégeztem a paraméterek érzékenység vizsgálatát, ahol arra koncentráltam, hogy számszerűsíthetően meghatározzam a paraméterek hatását, ezekre görbékkel illesztettem, így jól kezelhetővé váltak az eredmények, és így jól előre jelezhető

Irodalomjegyzék

- [1] **Marco Dorigo:** *Optimization, learning and natural algorithms (in Italian)*, Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
- [2] **Youfang Huang, Chengji Liang, Yang Yang:** *The optimum route problem by genetic algorithm for loading/unloading of yard crane*, Computers & Industrial Engineering 56, Intelligent Manufacturing and Logistics, 993-1001 oldal, 2009
<http://www.sciencedirect.com/science/article/pii/S036083520800199X> letöltve: 2014.03.29
- [3] **Kovács Gábor:** *Elektronikus fuvar- és raktárbörze rendszermodellje*, Ph.D. értekezés, Budapesti Műszaki és Gazdaságtudományi Egyetem, Közlekedésmérnöki és Járműmérnöki kar, 2011.
- [4] **John E. Bella, Patrick R. McMullen:** *Ant colony optimization techniques for the vehicle routing problem*, Advanced Engineering Informatics, 2004. Január, 41-48 oldal
- [5] **Dorigo, M., Gambardella, L. M.:** *Ant colony system: A cooperative learning approach to the travelling salesman problem*, IEEE Transactions on Evolutionary Computation, 1, 1997, 53-66 oldal
- [6] **Dorigo, M., Stützle, T.:** *Ant colony system: A cooperative learning approach to the travelling salesman problem*, Ant Colony Optimization, MIT Press, 2004, ISBN 0-262-04219-3
- [7] **Katona Géza:** *Közúti kapcsolatok kezelésének integrálása az E-travel multimodális útvonaltervezőbe*, BME TDK Konferencia, 2013
- [8] **Dr. Bozsik Sándor, Dr. Fellegi Miklós, Süveges Gábor, Dr. Szemán Judit:** *Haladó vállalati pénzügyek*, Nemzeti Tankönyvkiadó, 2011, ISBN 978-963-19-7254-2
http://www.tankonyvtar.hu/hu/tartalom/tamop425/0049_10_halado_vallalati_penzugyek/576/index.html

Ábrajegyzék

1. ÁBRA TESZTRENDSZER	9
2. ÁBRA TÁVOLSÁGMÁTRIX PÉLDA	10
3. ÁBRA KAPCSOLATI MÁTRIXRA PÉLDA.....	10
4. ÁBRA AZ HANGYAKOLÓNIA ALGORITMUS MŰKÖDÉSE	12
5. ÁBRA VALÓSZÍNŰSÉGI MÁTRIXRA PÉLDA.....	13
6. ÁBRA „RULETT KERÉK” PÉLDA	14
1. TÁBLÁZAT A VIZSGÁLANDÓ PARAMÉTEREK TARTOMÁNYAI.....	15
7. ÁBRA $\alpha = 0,5; \beta = 0,4; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	16
8. ÁBRA $\alpha = 0,5; \beta = 0,4; Q = 0,2$ PARAMÉTEREK DIAGRAMJA	16
9. ÁBRA $\alpha = 0,5; \beta = 0,4; Q = 0,3$ PARAMÉTEREK DIAGRAMJA	17
10. ÁBRA $\alpha = 0,5; \beta = 0,4; Q = 0,4$ PARAMÉTEREK DIAGRAMJA	17
11. ÁBRA $\alpha = 0,5; \beta = 0,8; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	18
12. ÁBRA $\alpha = 0,5; \beta = 1,2; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	18
13. ÁBRA $\alpha = 0,5; \beta = 1,6; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	19
14. ÁBRA $\alpha = 1; \beta = 0,4; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	19
15. ÁBRA $\alpha = 1,5; \beta = 0,4; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	20
16. ÁBRA $\alpha = 2; \beta = 0,4; Q = 0,1$ PARAMÉTEREK DIAGRAMJA	20
17. ÁBRA $\alpha = 1,5; \beta = 0,4; Q = 0,3$ PARAMÉTEREK DIAGRAMJA	21
18. ÁBRA FUTÁSI IDŐK AZ ITERÁCIÓK FÜGGVÉNYÉBEN	22
19. ÁBRA FUTÁSI EREDMÉNYEK 20 ITERÁCIÓ ESETÉN.....	22
20. ÁBRA FUTÁSI EREDMÉNYEK 100 ITERÁCIÓ ESETÉN.....	23
21. ÁBRA FUTÁSI IDŐK A HANGYÁK SZÁMÁNAK FÜGGVÉNYÉBEN	23
22. ÁBRA FUTÁSI IDŐK AZ ADATPONTOK SZÁMÁNAK FÜGGVÉNYÉBEN	24
23. ÁBRA RUGALMASSÁGI MUTATÓK [7].....	26
24. ÁBRA ÉRZÉKENYSÉGI FÜGGVÉNYEK $\alpha = 0,5, \beta = 0,4$ ESETÉN.....	28
25. ÁBRA ÉRZÉKENYSÉGI FÜGGVÉNYEK $\alpha = 0,5, Q = 0,1$ ESETÉN.....	28
26. ÁBRA ÉRZÉKENYSÉGI FÜGGVÉNYEK $\beta = 0,4, Q = 0,1$ ESETÉN.....	29
2. TÁBLÁZAT A LEGJOBB EREDMÉNYEKET HOZÓ PARAMÉTEREK	29
27. ÁBRA A LEGJOBB EREDMÉNYEKET SZOLGÁLTATÓ PARAMÉTEREK FUTÁSI EREDMÉNYE	30
3. TÁBLÁZAT A LEGJOBB EREDMÉNYEKET HOZÓ PARAMÉTEREK STATISZTIKAI JELLEMZŐI	30