Multidiszciplináris tudományok, 8. kötet. (2018) 1 sz. pp. 3-8

A TÖBBÜGYNÖKÖS UTAZÓ ÜGYNÖK PROBLÉMA MEGOLDÁSA LOKÁLIS OPTIMALIZÁLÁSSAL

Agárdi Anita
hallgató, Miskolci Egyetem, Informatika Intézet
3515 Miskolc, Miskolc-Egyetemváros, e-mail: agardianita@iit.uni-miskolc.hu

Absztrakt

Az utazó ügynök problémának számos változata létezik a szakirodalomban. Léteznek együgynökös és többügynökös feladatok, melyek segítségével számos gyakorlati járattervezési feladat megoldható, különösen logisztikai feladatok esetében. Jelen cikk keretében annak a lehetőségét vizsgálom, hogy az előklaszterezés milyen hatással van a többügynökös utazó ügynök probléma megoldására.

Kulcsszavak: *utazó ügynök probléma, optimalizálás, előklaszterezés, lerakatok, heurisztika*

Abstract

The are different types of models of the travelling salesmen problems. By the aid of these various models including single-depot and multi-depot models it is possible to represent a wide range of routing problems in the field of logistics. Within the frame of this article the author analysis the impact of pre-clustering on the performance of local optimisation methods of multiple travelling salesman problems.

Keywords: *traveling salesman problem, optimization, pre-clustering, depots, heuristics*

1. Bevezetés

Az utazó ügynök optimalizálási feladat célja az ügynök által megtett út hosszának minimalizálása, ahol az útnak minden várost pontosan egyszer érintenie kell. Az utazó ügynök problémának számos változata létezik, melyek rendszerint a logisztikai problémákhoz kapcsolódnak. A probléma változatokat csoportosíthatjuk aszerint, hogy egy, vagy több ügynök látogatja-e meg a városokat, és hogy ezek az ügynökök egy vagy több lerakatból indulnak-e ki. Eszerint megkülönböztetünk együgynökös utazó ügynök problémát (TSP - Traveling Salesman Problem), többügynökös egy lerakatos problémát (MTSP - Multiple Traveling Salesman Problem) és többügynökös több lerakatos problémát (MDMTSP – Multi-Depot Multiple Traveling Salesman Problem). Tapasztalat szerint a szomszédos városok egymáshoz közel helyezkednek el, így érdemesnek találtam előklaszterezést bevezetni az egyes utazó ügynök problémák megoldására. Dolgozatom célja tehát az előklaszterezés hatástanulmánya az egyes utazó ügynök problémákra.

2. Az egy- és többügynökös egy lerakatos utazó ügynök probléma

Az együgynökös utazó ügynök probléma (TSP - Traveling Salesman Problem) során adott n darab város pozíciója, amit egyetlen ügynöknek kell meglátogatni úgy, hogy minden várost csak egyszer látogat meg, és abba a városba érkezik vissza, ahonnan elindult [1].

A többügynökös egy lerakatos utazó ügynök probléma (MTSP - Multiple Traveling Salesman Problem) során adott n darab város pozíciója, melyet m darab ügynöknek kell meglátogatnia. Minden várost csak egy ügynök látogathat meg. Ezenkívül adott egy központi lerakat (depó) pozíciója. A depóból indul ki minden ügynök, és ide is érkezik vissza a városok meglátogatása után. Ez a probléma bonyolultabb mint egy szimpla TSP, mert azt is meg kell határozni, hogy egy ügynökhöz mely városok tartozzanak [2].

3. A többügynökös több lerakatos utazó ügynök probléma

A többügynökös több lerakatos utazó ügynök probléma (MDMTSP – Multi - Depot Multiple Traveling Salesman Problem) során adott n darab város pozíciója, melyet m darab ügynöknek kell meglátogatnia. Minden várost csak egy ügynök látogathat meg. Ezenkívül adott m darab lerakat pozíciója is. A feladat során az ügynökök egy-egy lerakatból indulnak ki, meglátogatnak bizonyos városokat, majd ugyanabba a lerakatba térnek vissza. A problémánál nem csak városokat kell rendelni az ügynökökhöz, hanem egy lerakatot is. Egy lerakathoz csakis egy ügynök tartozhat [3].

4. Heurisztikus módszerek az utazó ügynök probléma megoldására

Az utazó ügynök problémákat számos heurisztikus módszerrel megoldották már. Ezek közül a legismertebbek: genetikus algoritmus, hangya kolónia algoritmus, hegymászó algoritmus, PSO, Lin-Kernighan algoritmus, tabu keresés, memetikus algoritmus.

A dolgozat célja egy speciális optimalizálási modul, az előklaszterezési modul kipróbálása. Az előklaszterezés indokoltságát azzal lehet magyarázni, hogy a tapasztalat szerint az optimális útvonal rendszerint a szomszédos elemeket köti össze ezért a klaszteren belüli elemek nagyobb valószínűséggel lesznek szomszédosak, mint a klaszteren kívüli elemek. A dolgozat célja az előklaszterezés hatékonyságának elemzése.

A szakirodalomban néhány művet sikerült csak találnom, amik az egy- és többügynökös problémát a klaszterezést is bevetve oldják meg. Ezek közül az alábbiakat szeretném kiemelni.

A klasszikus utazó ügynök probléma előklaszterezését vizsgáló [4]-ben K-means klaszterezést használnak a szerzők a keresési tér redukálására. A módszerekben klaszteren belüli és klaszteren kívüli optimalizációt is definiálnak. A klaszteren kívüli optimalizáció célja a klaszterek bejárási sorrendjének a meghatározása. Klaszteren belüli optimalizáció pedig azt, jelenti, hogy a klaszteren belül a városok sorrendjét határozzuk meg. A két szint útvonalának egyesítésével megkapjuk az ügynök útját. A dolgozatomban ezen eljárást használtam a klasszikus TSP előklaszterezésére.

Szintén az előklaszterezett klasszikus utazó ügynök problémára ad megoldást az [5]-ös cikk is. Ezen cikkben úgy oldják meg a problémát, hogy először a városokat közelségük alapján klaszterekbe sorolják. Ezután a genetikus algoritmust vetik be a feladat megoldására. A klaszterezéssel kromoszóma részeket képzenek. Egy-egy kromoszóma részben találhatóak az egyes klaszterek városai. Ezután a szokásos genetikus operátorokat (keresztezés, mutáció) használják a problémára.

A többügynökös egy lerakatos utazó ügynök problémát előklaszterezés segítségével a [6]- os cikk is szerzői is megoldják. Az előklaszterezést arra használják, hogy a többügynökös problémát több, együgynökös problémára redukálják. A klaszterezés dönti el tehát, hogy mely városok fognak egy ügynökhöz tartozni. A dolgozatomban ezen eljárást használtam a több gynökös utazó ügynök problémák (MTSP, MDMTSP) előklaszterezésére.

5. Implementált város meghatározó algoritmusok

A dolgozatomban számos algoritmust implementáltam az utazó ügynök probléma három típusára. Az implementált algoritmusokat futtatási eredmények során kiértékeltem.

Legközelebbi szomszéd algoritmus (TSP): Az algoritmus során egy véletlenszerűen választott városból indulunk ki, és mindig az utolsó városhoz legközelebbi (még ki nem választott) várost választjuk ki. Ha minden várost kiválasztottunk már, akkor az elsőnek kiválasztott városba térünk vissza [7].

Legközelebbi szomszéd algoritmus (MTSP): Az algoritmus során a lerakatból indulunk ki, majd a lerakat utáni legközelebbi várost választjuk ki. Ezután sorban választjuk ki a legutoljára kiválasztott városhoz legközelebbi (még ki nem választott) várost, majd visszatérünk a lerakatba. Ekkor megkaptuk egy ügynök útját. A következő ügynökök első városa a lerakat lesz, a második város pedig az előző ügynök utoljára kiválasztott városához (nem lerakat) legközelebbi (még ki nem választott) város.

Legközelebbi szomszéd algoritmus (MDMTSP): Kiindulunk egy véletlenszerűen választott városból. Kezdetben ez lesz az utoljára kiválasztott város. Ezután sorban választjuk ki az utoljára kiválasztott városhoz legközelebbi (még ki nem választott) városokat, majd visszatérünk az elsőnek kiválasztott városba. A következő ügynök elsőnek kiválasztott városa az előző ügynök utoljára kiválasztott városához legközelebbi (még ki nem választott) város lesz.

Véletlen pont beszúrása algoritmus (TSP): Kiindulunk egy véletlenszerűen választott városból. Ezután sorban választjuk ki (a még ki nem választott) városokat és beszúrjuk azon két város közé, ahol a körút növekedésének költsége minimális lesz. Az eljárás akkor ér véget, ha minden várost beszúrtunk a körútba [8-9].

Véletlen pont beszúrása algoritmus (MTSP): Kiindulunk annyi városból, ahány ügynököt vetünk be a probléma megoldására. Ezután sorban választjuk ki (a még ki nem választott) városokat, és azon két város közé szúrjuk be, ahol a körút növekedése minimális lesz. Ha minden várost kiválasztottunk, akkor a lerakatot is beszúrjuk, ezt viszont minden körútba, és azon két város közé, ahol a körutak növekedésének költsége minimális.

Véletlen pont beszúrása algoritmus (MDMTSP): Kiindulunk annyi városból, ahány ügynököt szeretnénk a problémára bevetni. Majd sorban választjuk ki (a még ki nem választott) városokat, és azon két város közé szúrjuk be, ahol a körút növekedése minimális lesz.

Hegymászó algoritmus (TSP): Az algoritmus során kiindulunk egy lehetséges megoldásból, kezdetben ez lesz az aktuális megoldás. Képezzük az aktuális megoldás egy szomszédját [10]. A szomszédot úgy képzem, hogy az aktuális megoldás két (véletlenszerűen választott) élét kicserélem egymással (2-opt algoritmus [7]). Ha az aktuális megoldás szomszédja jobb, mint az aktuális megoldás, akkor a szomszédot fogadjuk el aktuális megoldásnak.

Hegymászó algoritmus (MTSP): Az algoritmus hasonló a TSP során bemutatott eljáráshoz, a különbség az aktuális megoldás szomszédjának képzésében van. A szomszédot úgy képezzük, hogy az aktuális megoldáson élcserét és ügynökcserét (kiválasztunk két ügynököt véletlenszerűen, és az egyik ügynökhöz tartozó városok számát eggyel növelük, a másikhoz tartozó városok számát pedig egyel csökkentjük [11]) alkalmazunk.

Hegymászó algoritmus (MDMTSP): Az algoritmus hasonló az MTSP során bemutatott eljáráshoz, a különbség annyi, hogy nem egy központi lerakat van(ezt gyakorlatilag csak a lehetséges megoldások kiértékelésénél kell figyelembe venni).

Klszterezés: Klaszterezés az elemek csoportosítását jelenti. Azok az elemek, amik hasonlóak egymáshoz azonos klaszterbe kerülnek azok, amelyek pedig különböznek egymástól más klaszterbe. Számos

klaszterezési módszer látott napvilágot, például K-közép, hierarchikus klaszterezés, DBSCAN, BIRCH. A dolgozatban a K-közép és a hierarchikus eljárásokat mutattam be.

Előklaszterezés (TSP): A városokat közelségük alapján klaszterekbe soroljuk. Meghatározzuk a klaszterek sorrendjét (erre a véletlen pont beszúrása, legközelebbi szomszéd vagy hegymászó algoritmust használom). Ezután a klaszteren belül a városok sorrendjét határozzuk meg (erre a véletlen pont beszúrása, legközelebbi szomszéd vagy hegymászó algoritmust használom). Majd a klaszter összekötéseket elvégezzük [4,11].

Előklaszterezés (MTSP): A városokat közelségük alapján klaszterekbe soroljuk. Annyi klasztert alakítunk ki, ahány ügynököt szeretnénk bevetni a problémára. Ezután a klasztereken belül a városok sorrendjét határozzuk meg úgy, hogy egy-egy körutat keresünk. A körutaknak a lerakatot is tartalmazniuk kell [6,11].

Előklaszterezés (MDMTSP): Az algoritmus hasonló az MTSP algoritmusához, csak nem egy központi lerakat van.

6. Algoritmusok értékelése

1. táblázat. *Futtatási eredmények az együgynökös utazó ügynök problémára 1000 városra (Forrás: saját forrás)*

	TSP		**Klaszterezett TSP**	
	Megtett táv	**Futási idő (sec)**	**Megtett táv**	**Futási idő (sec)**
Legközelebbi szomszéd	2784.3079	0.125	2612.8273	17.5312
Véletlen pont beszúrása	3062.2465	0.0312	3405.6912	0.1812
Hegymászó	3538.4512	16.1244	2643.6988	31.0455
Legközelebbi szomszéd+hegymászó	*2530.1547*	16.1547	2605.1245	33.0412
Véletlen pont beszúrása+hegymászó	2673.4578	18.1248	3205.1247	32.0312

A klasszikus utazó ügynök probléma során a legközelebbi szomszéd algoritmus hegymászó algoritmussal javítása adta a legjobb megoldást. Az előklaszterezés nélküli eset jobbnak bizonyult, mint az előklaszterezett.

2. táblázat. *Futtatási eredmények a többügynökös egy lerakatos utazó ügynök problémára 1000 városra (Forrás: saját forrás)*

	MTSP		**Klaszterezett MTSP**	
	Megtett táv	**Futási idő (sec)**	**Megtett táv**	**Futási idő (sec)**
Legközelebbi szomszéd	10122.1622	0.0625	9620.2113	0.0125
Véletlen pont beszúrása	13425.4044	0.0312	9994.8935	0.125
Hegymászó	22794.1245	24.4562	9355.1245	33.0356
Legközelebbi szomszéd+hegymászó	*8905.4512*	*24.3145*	9322.2563	32.25
Véletlen pont beszúrása+hegymászó	11075.1245	24.1456	9335.0478	30.1256

Többügynökös egy lerakatos utazó ügynök probléma során szintén a legközelebbi szomszéd algoritmus hegymászó algoritmussal finomítása adta a legjobb eredményt. Szintén az előklaszterezés nélküli eset bizonyult a jobbnak.

3. táblázat. *Futtatási eredmények a többügynökös több lerakatos utazó ügynök problémára 1000 városra (Forrás: saját forrás)*

	MDMTSP		**Klaszterezett MDMTSP**	
	Megtett táv	**Futási idő (sec)**	**Megtett táv**	**Futási idő (sec)**
Legközelebbi szomszéd	3889.4847	0.0625	2976.4302	0.0938
Véletlen pont beszúrása	7182.0626	0.0625	2784.7974	0.0625
Hegymászó	14769.3042	24.0045	2593.2487	31.0147
Legközelebbi szomszéd+hegymászó	3081.1245	24.8874	2602.4796	30.875
Véletlen pont beszúrása+hegymászó	6529.4156	24.1240	*2590.5623*	20.1978

Többügynökös több lerakatos probléma során a klaszterezett esetek bizonyultak jobbnak. A legjobb megoldást a véletlen pont beszúrása algoritmus hegymászó algoritmussal javított változata adta.

7. Összefoglalás

A kutatás célja a klaszterezés hatástanulmánya az egy- és többügynökös utazó ügynök problémákra. Az implementált város sorrend meghatározó algoritmusok közül a legközelebbi szomszéd algoritmus bizonyult a legjobbnak. Az előklaszterezés sok esetben hosszabb utat eredményezett, mint a nem előklaszterezett esetek. Ennek oka az, hogy a városokra egy kezdeti behatárolást ad, ami azt jelenti, hogy egy klaszterben lévő városokat egymás után kell meglátogatni. Habár néhány publikáció szól az utazó ügynök problémák előklaszterezéséről, én jobban ajánlom a problémát valamilyen generáló algoritmussal (pl. véletlen pont beszúrása vagy legközelebbi szomszéd) megoldani, és ezt javítani egy javító algoritmussal (pl. hegymászó algoritmus).

Felhasznált irodalom

[1] Brezina Jr, I., & Čičková, Z. (2011). Solving the travelling salesman problem using the ant colony optimization. *Management Information Systems, 16(4),* p.010-014.

[2] Hosseinabadi, A. A., Kardgar, M., Shojafar, M., Shamshirband, S., & Abraham, A. (2014, November). GELS-GA: hybrid metaheuristic algorithm for solving multiple travelling salesman problem. In *Intelligent Systems Design and Applications (ISDA), 2014 14th International Conference*. p. 76-81

[3] Benavent, E., & Martínez, A. (2013). Multi-depot Multiple TSP: a polyhedral study and computational results. *Annals of Operations Research, 207(1).* p. 7-25.

[4] Chavan, P. Y., Gundale, A. R., Thorat, M. S. & Jambhulkar , A. H. (2015): Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs. *International Journal of Advance Research in Computer Science and Management Studies*. p. 335-340

[5] Kalyan Bharadwaj, B., Krishna Kishore, G., & Srinivasa Rao, VV. (2011): Solving Traveling Salesman Problem Using Hierarchical Clustering and Genetic Algorithm. *IJCSIT International Journal of Computer Science and Information Technologies Vol. 2 (3).* p. 1096-1098.
[6] Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., & Parthiban, P. (2009): Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science*, *8*(4), p.480-487.
[7] Nilsson, C. (2003): Heuristics for the traveling salesman problem. *Linkoping University.* p.1-6.
[8] Rosenkrantz, Stearns, Lewis (1974): Traveling Salesman Problem Insertion Algorithms. *https://www2.isye.gatech.edu/~mgoetsch/cali/VEHICLE/TSP/TSP009__.HTM* (Letöltve: 2018. 04. 12.)
[9] Hahsler, M., & Hornik, K. (2007). TSP-Infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, *23*(2), p. 1-21.
[10] Zhu, X.: Advanced Search Hill climbing, simulated annealing, genetic algorithm. *http://pages.cs.wisc.edu/~jerryzhu/cs540/handouts/hillclimbing.pdf* (Letöltve: 2018. 04. 12.)
[11] Agárdi A. (2017): Klaszterezési és evolúciós technikák alkalmazása az utazó ügynök probléma megoldásában *Szakdolgozat Miskolci Egyetem*

Jelen cikk a szerzők engedélyével jelent meg másodközlésben. Az első megjelenés bibliográfiai adatai: Agárdi Anita: *A többügynökös utazó ügynök probléma megoldása lokális optimalizálással*. Diáktudomány: A Miskolci Egyetem Tudományos Diákköri Munkáiból 11. pp. 53-58. (2018)