Multidiszciplináris tudományok, 11. kötet. (2021) 4 sz. pp. 120-126 **https://doi.org/10.35925/j.multi.2021.4.14**

KLASZTEREZÉS HANGYA KOLÓNIA OPTIMALIZÁCIÓVAL ÉS TABU KERESÉSSEL

Agárdi Anita
tanársegéd, Miskolci Egyetem, Informatika Intézet, Általános Informatikai Intézeti Tanszék
3515 Miskolc, Miskolc-Egyetemváros, e-mail: agardianita@iit.uni-miskolc.hu

Absztrakt

A cikk az egyik legfontosabb adatbányászati feladattal, a klaszterezéssel foglalkozik. A klaszterezés elemek csoportosítását jelenti. A cikkben olyan klaszterezés kerül bemutatásra, mely során metaheurisztikákat használtam csoportosításra. Jelen cikkben két, népszerű metaheurisztika, a hangya kolónia optimalizáció egyes algoritmusai és a tabu keresés algoritmus lett használva a csoportosítási probléma megoldására. A teszt eredmények azt mutatják, hogy hatékony klaszterezést nyújtanak az algoritmusok.

Kulcsszavak: *klaszterezés, hangya kolónia optimalizáció, tabu keresés*

Abstract

The article presents one of the most important data mining tasks, clustering. Clustering is the grouping of elements. The paper presents a clustering in which I used metaheuristics for grouping. In this paper, two popular metaheuristics, some algorithms for ant colony optimization and the tabu search algorithm, were used to solve the grouping problem. Test results show that the algorithms provide efficient clustering.

Keywords: *clustering, ant colony optimization, tabu search*

1. Bevezetés

A cikkben egy népszerű adatbányászati problémát, a klaszterezést mutatom be. A klaszterezés az adatsor elemeinek csoportosítását jeleni, ahol azok az elemek amelyek egymástól különböznek külön csoportba, míg azok az elemek amelyek egymáshoz hasonlítanak azonos csoportba kerülnek. Számos módon csoportosíthatunk egy adatsort, így számos klaszterező algoritmus látott már napvilágot. A partíciós módszerek [1] egy kezdeti csoportosítás után iteratívan javítják a klaszterezést, míg nincs javulás. A hierarchikus módszerek [1] egy fa struktúrát alakítanak ki. A sűrűség alapú klaszterezés [1] során sűrű pontokat keresünk. Egy pont bizonyos környezetébe kell esnie bizonyos számú pontnak. Ezen klaszterezés a zajos adatok kiszűrésére is alkalmas. Az utóbbi években metaheurisztikákkal is történtek klaszterezések, például genetikus algoritmus [2-3], tabu keresés [4-5], részecske-raj alapú optimalizálás [6-7], hangya kolónia optimalizáció [8-9].

A hangya kolónia optimalizációt [8-9] a hangyák viselkedése ihlette. A hangyák útjuk keresése során feromont helyeznek az útvonalra. Azon útszakaszokat választják nagyobb valószínűséggel a hangyák, amelyek a rövidebbek, és nagyobb feromon tartalmuk van. Az idő múlásával (iterációszámmal) a feromontartalom csökken, de mikor egy-egy hangya átkel egy útszakaszon, akkor feromont tesz le. A hangya kolónia algoritmus egy gyűjtőfogalom, amely számos algoritmust tartalmaz. A cikkben az

Ant System [10], Ant Colony System [11], MAX-MIN Ant System [12], Elitist Strategy of Ant System [13], Rank Based Version of Ant System [14].

A tabu keresés [4-5] egy tabu listát tart fent, ahol a keresés eredményei találhatóak. Amikor a tabu lista betelik, akkor a legrégebbi elem törlődik. A tabu lista szomszédsági alapú keresést végez, az aktuális megoldás szomszédait vizsgálja. Mindig a jobb szomszédot próbálja meg elfogadni.

2. Hangya kolónia algoritmusok

A hangya kolónia optimalizációnak számos változata alakult ki az utóbbi években. A cikkben az Ant System [10], Ant Colony System [11], MAX-MIN Ant System [12], Elitist Strategy of Ant System [13], Rank Based Version of Ant System [14] kerülnek bemutatásra, tesztelésre. Ezen algoritmusok csupán néhány lépésben térnek el egymástól. Mindegyik algoritmus során a hangyák iteratívan keresik az utat, és bizonyos hangyák tesznek le feromont az útjukra. Ezenkívül minden algoritmus figyelembe veszi a feromon párolgását is.

Ant System

Az út konstruálása az alábbi képlettel történik, ahol annak a valószínűsége, hogy a k. hangya az i. pontból a j. pontba megy az algoritmus t. iterációjában az alábbi [10]:

$$p_{ij}^{k} = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}(t)]^{\alpha} * [\eta_{il}]^{\beta}} \quad ha\ j \in N_i^k \tag{1}$$

$\eta_{ij} = \frac{1}{d_{ij}}$, két csomópont közötti távolság reciproka, $\tau_{ij}(t)$ pedig az (i,j) él feromontartalma

A feromon frissítésre pedig az alábbi képletet alkalmazzák:

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{2}$$

ahol

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L^k(t)} & ha\ az\ (i,j)\ élen\ átment\ a\ k.hangya \\ 0 & különben \end{cases} \tag{3}$$

Ant Colony System

Itt a csak a globálisan legjobb hangya tehet le feromont az egyes iterációk során, és egy lokális feromon frissítés is történik. A globális feromon frissítés képlete megegyezik az Ant System feromon frissítésének képletével, csak a globálisan legjobb hangya feromonját kell figyelembe vennünk.

A lokális frissítés során ha egy hangya áthaladt egy útszakaszon akkor az alábbi képletet használják az utak frissítésére [11]:

$$\tau_{ij} = (1-\xi) * \tau_{ij} + \xi * {\tau_{ij}}^{0} \tag{4}$$

MAX-MIN Ant System

Annyiban különbözik az előző algoritmusoktól, hogy egyetlen hangya tehet le feromont, és az utak feromonja korlátozva van, csak egy intervallumbeli értéket vehet fel. Az utak feromontartalma kezdetben a legmagasabb értékű lesz [12].

Elitist Strategy of Ant Sytem

Az Ant System algoritmus javítására fejlesztették ki. Annyiban különbözik tőle, hogy az eddigi legjobb útvonalra plus hangsúlyt fektet, amit az alábbi képlet jelent [13]:

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) + w * \Delta\tau_{ij}^{gb}(t) \tag{5}$$

Rank Based Version of Ant Sytem

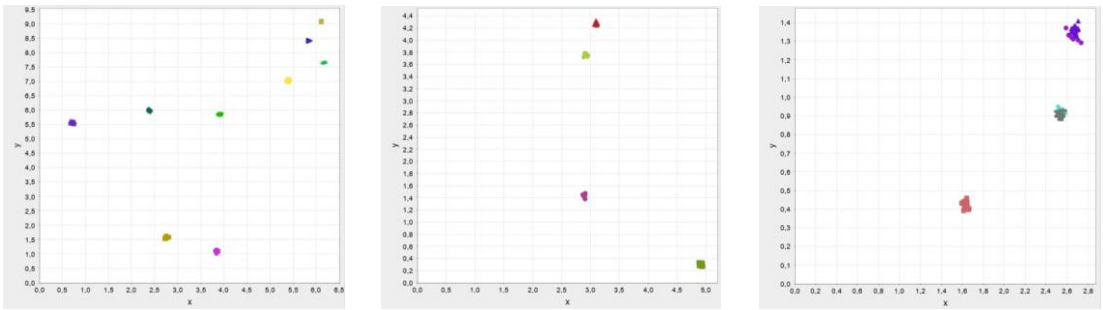
Ez az algoritmus is az Ant Sytem egy módosítása. Itt a hangyákat útvonal hosszuk alapján rangsorolják, és minden hangya a rangsorának megfelelő súllyal tehet le a bejárt útvonalára feromont. Ez mutatja az alábbi képlet [14]:

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \sum_{r=1}^{w-1} (w-r) * \Delta\tau_{ij}^{r}(t) + w * \Delta\tau_{ij}^{gb}(t) \tag{6}$$

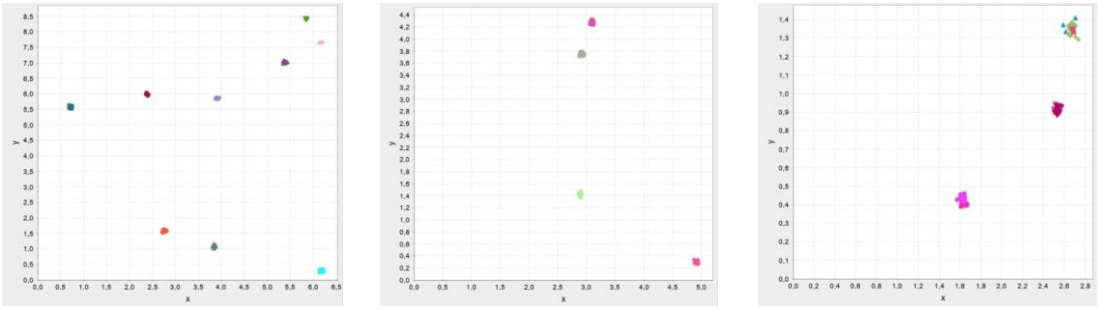
3. Teszt eredmények

Ezen fejezetben a teszteredményeket fogom bemutatni. Három, jól klaszterezhető adatsort generáltam. Az első adatsor 100 kétdimenziós pontot, a második 50 kétdimenziós pontot, a harmadik pedig 90 kétdimenziós pontot tartalmaz. Az első adatsort optimálisan 10 klaszterre, a másodikat 5 klaszterre, a harmadikat pedig 3 klaszterre érdemes bontani. A klaszterezést úgy valósítottam meg, hogy az adatsor elemeit mint permutáció tekintettem. A permutáció elemein sorban végigmentem, és azok a permutációban szomszédos elemek, amelyek közel helyezkedtek el egymáshoz azonos klaszterbe kerültek. Az optimalizációs algoritmusoknak szükségük van egy fitnesz értékre is, ugyanis iteratívan egy vagy több permutációt készítenek. Az optimalizáció célfüggvénye az azonos klaszterbeli elemek közötti távolságok összege. Az adatsor és a klaszterezés kialakítási stratégia a [16] cikkben bemutatottak alapján készült.

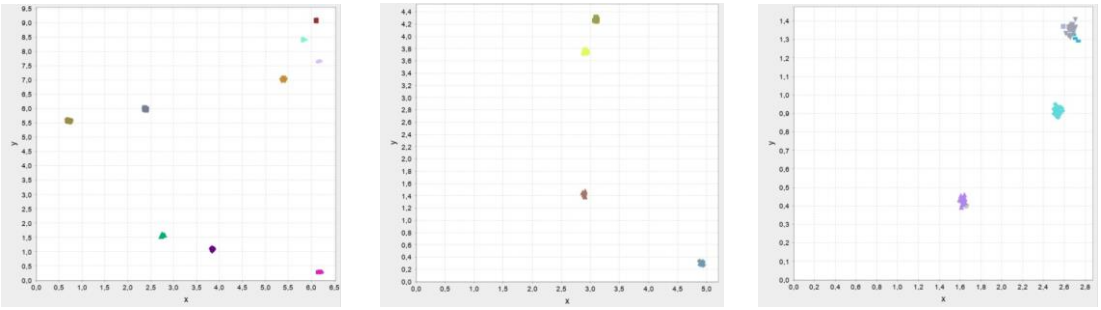
Elsőként a kialakult klaszterezést mutatom be, majd a futási időket és fitnesz értékeket. Az *1.-6. ábra* szerint az első két adatsorra szépen megoldották az algoritmusok a klaszterezést, a kialakult klaszterek olyanok, amelyeket a felhasználó vár. A harmadik adatsorra az Elitist Strategy of Ant System és a Rank Based Version of Ant Sytem algoritmusok azok, amelyek majdnem tökéletesen megoldották a klaszterezést. A többi algoritmus a vártnál több csoportra bontotta az adatsort.



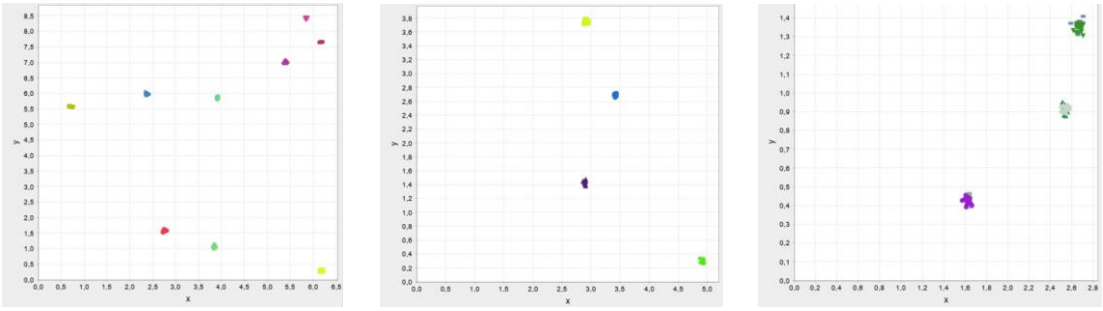
1. ábra. *Ant Colony System algoritmus eredménye az első, második és harmadik adatsorra*



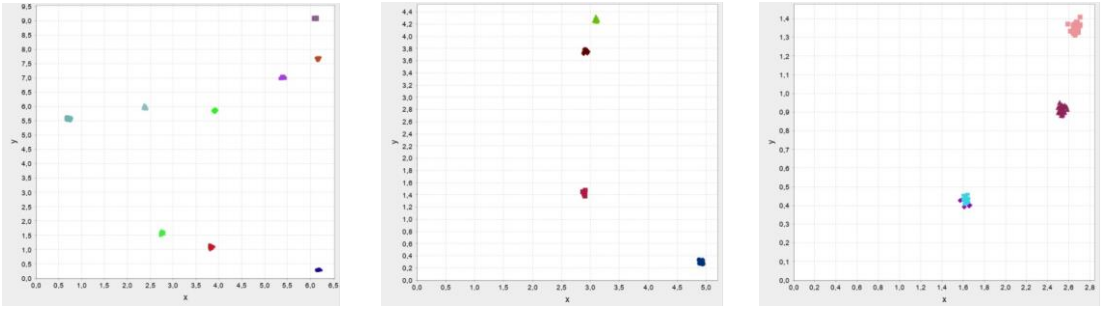
2. ábra. *Ant System algoritmus eredménye az első, második és harmadik adatsorra*



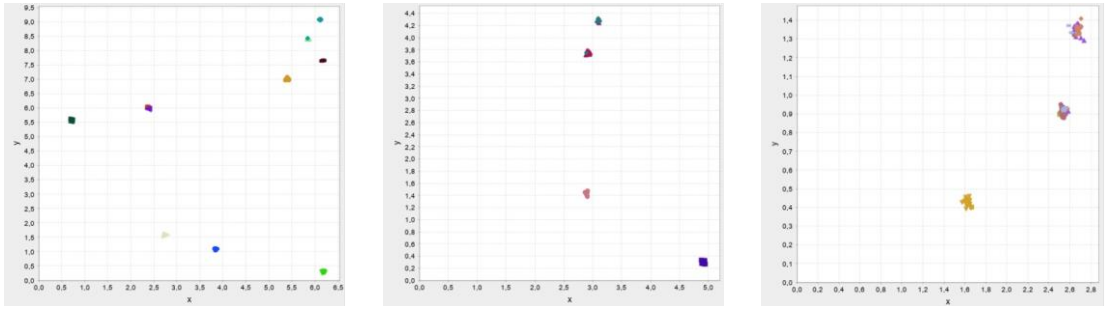
3. ábra. *Elitist Strategy of Ant System algoritmus eredménye az első, második és harmadik adatsorra*



4. ábra. *MAX-MIN Ant System algoritmus eredménye az első, második és harmadik adatsorra*



5. ábra. *Rank Based Version of Ant System algoritmus eredménye az első, második és harmadik adatsorra*



6. ábra. *Tabu keresés algoritmus eredménye az első, második és harmadik adatsorra*

A következőkben az egyes adatsorokra a futások fitnesz értékeit, a futási idő és a klaszterek számát is bemutatom.

1. táblázat. *Az első adatsor eredményei*

Algoritmus	Klaszterek száma	Fitnesz érték	Futási idő (perc)
Ant Colony System	9	0.3531	0.4497
Ant System	9	0.3644	0.4279
Elitist Strategy of Ant System	9	0.3535	0.4206
MAX-MIN Ant System	9	0.3644	0.4813
Rank Based Version of Ant System	9	0.3664	0.4497
Tabu Search	10	0.6370	0.0211

Az első adatsort az összes hangya kolónia algoritmus verzió 9 csoportra bontotta. Közel azonos klaszterezést alakítottak ki, mert a fitnesz értékek is közel azonosak. A tabu keresés megtalálta mind a 10 csoportot. A futási idők minden esetben alacsonyak.

2. táblázat. *A második adatsor eredményei*

Algoritmus	Klaszterek száma	Fitnesz érték	Futási idő (perc)
Ant Colony System	4	0.3662	0.0158
Ant System	4	0.3662	0.0153
Elitist Strategy of Ant System	4	0.3662	0.0175
MAX-MIN Ant System	4	0.3574	0.0130
Rank Based Version of Ant System	4	0.3662	0.0142
Tabu Search	4	0.8252	0.0035

A második adatsorra 5 klaszter lett volna az optimális, az algoritmusok négyet találtak meg. A fitnesz értékek alapján látható, hogy ezek bizonyos algoritmusoknál különböző klasztereket jelentenek.

A harmadik adatsornál 3 klaszter lett volna az optimális. Az Ant Colony System és a Rank Based Version of Ant System 5 csoportra, a többi algoritmus pedig 6 klaszterre bontotta az adatsort. A fitnesz értékek is különbözőek az egyes algoritmusoknál, ez azt jelenti, hogy különböző csoportosításokat végeztek még akkor is, ha a klaszterszám azonos.

3. táblázat. *A harmadik adatsor eredményei*

Algoritmus	Klaszterek száma	Fitnesz érték	Futási idő (perc)
Ant Colony System	5	0.6297	0.2786
Ant System	6	0.7539	0.2160
Elitist Strategy of Ant System	6	0.8241	0.2270
MAX-MIN Ant System	6	0.6938	0.2225
Rank Based Version of Ant System	5	0.7609	0.2560
Tabu Search	6	1.1504	0.0188

4. Összefoglalás

Jelen cikkben a klaszterezést két metaheurisztikus algoritmussal mutattam be. A tabu keresést, és a hangya kolónia optimalizáció egyes algoritmusait, mint az Ant Colony System, Ant System, Elitist Strategy of Ant Sytem, MAX-MIN Ant System, Rank Based Version of Ant System. Három különböző adatsoron mutattam be az algoritmusok hatékonyságát. A teszt eredmények alapján az első két adatsorra mindegyik algoritmus szinte tökéletesen megoldotta a csoportosítást, a harmadik adatsorra viszont csak az Elitist Strategy of Ant System és a Rank Based Version of Ant Sytem algoritmusok adtak hatékony klaszterezést.

Köszönetnyilvánítás

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

Irodalom

[1] Bodon Ferenc, Buza Krisztián (2014): Adatbányászat https://regi.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0064_55_adatbanyaszat/ar01s06.html

[2] Tinós, R., Zhao, L., Chicano, F., & Whitley, D. (2018). NK hybrid genetic algorithm for clustering. IEEE Transactions on Evolutionary Computation, 22(5), 748-761. **https://doi.org/10.1109/TEVC.2018.2828643**

[3] Islam, M. Z., Estivill-Castro, V., Rahman, M. A., & Bossomaier, T. (2018). Combining K-Means and a genetic algorithm through a novel arrangement of genetic operators for high quality clustering. Expert Systems with Applications, 91, 402-417. **https://doi.org/10.1016/j.eswa.2017.09.005**

[4] Lu, Y., Cao, B., Rego, C., & Glover, F. (2018). A Tabu Search based clustering algorithm and its parallel implementation on Spark. Applied Soft Computing, 63, 97-109. **https://doi.org/10.1016/j.asoc.2017.11.038**

[5] Saveetha, V., & Sophia, S. (2018). Optimal tabu k-means clustering using massively parallel architecture. Journal of Circuits, Systems and Computers, 27(13), 1850199. **https://doi.org/10.1142/S0218126618501992**

[6] Alswaitti, M., Albughdadi, M., & Isa, N. A. M. (2018). Density-based particle swarm optimization algorithm for data clustering. Expert Systems with Applications, 91, 170-186. **https://doi.org/10.1016/j.eswa.2017.08.050**

[7] Wang, J., Cao, Y., Li, B., Kim, H. J., & Lee, S. (2017). Particle swarm optimization based clustering algorithm with mobile sink for WSNs. Future Generation Computer Systems, 76, 452-457. **https://doi.org/10.1016/j.future.2016.08.004**

[8] Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering. Analytica Chimica Acta, 509(2), 187-195. **https://doi.org/10.1016/j.aca.2003.12.032**

[9] İnkaya, T., Kayalıgil, S., & Özdemirel, N. E. (2015). Ant colony optimization based clustering methodology. Applied Soft Computing, 28, 301-311. **https://doi.org/10.1016/j.asoc.2014.11.060**

[10] Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1), 29-41. **https://doi.org/10.1109/3477.484436**

[11] Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation, 1(1), 53-66. **https://doi.org/10.1109/4235.585892**

[12] Stützle, T., & Hoos, H. H. (2000). MAX-MIN ant system. Future generation computer systems, 16(8), 889-914. **https://doi.org/10.1016/S0167-739X(00)00043-1**

[13] Jaradat, G. M., Al-Badareen, A., Ayob, M., Al-Smadi, M., Al-Marashdeh, I., Ash-Shuqran, M., & Al-Odat, E. (2019). Hybrid elitist-ant system for nurse-rostering problem. Journal of King Saud University-Computer and Information Sciences, 31(3), 378-384. **https://doi.org/10.1016/j.jksuci.2018.02.009**

[14] Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). A new rank based version of the Ant System. A computational study.

[15] Azar, Y. (1994). Lower Bounds for Insertion Methods for TSP. Combinatorics, Probability & Computing, 3, 285-292. **https://doi.org/10.1017/S096354830000119X**

[16] Agárdi, A. (2021) Klasszikus klaszterező algoritmusok módosítása körút alapon, Multidiszciplináris Tudományok, 11(4), 81-86. **https://doi.org/10.35925/j.multi.2021.4.9**