FÖLDESI PÉTER[1] – KÓCZY T. LÁSZLÓ[2] – TÜŰ-SZABÓ BOLDIZSÁR[3]

A Diszkrét Bakteriális Memetikus Evolúciós Algoritmus gyorsításának lehetőségei a szimmetrikus Utazó Ügynök Probléma megoldására

A cikkben a 2016-ban általunk publikált Diszkrét Bakteriális Memetikus Evolúciós Algoritmus gyorsítási lehetőségeit fogjuk bemutatni. Az algoritmus a bakteriális evolúciós algoritmust lokális kereséssel (2-opt, 3-opt) kombinálja, ezáltal hatékonyan alkalmazható az Utazó Ügynök Probléma megoldására. A lokális keresés tartományának szűkítésével, valamint párhuzamosítással jelentős gyorsulást értünk el. A továbbfejlesztett algoritmus szimmetrikus VLSI referenciapéldákon teszteltük 5000 pontos feladatnagyságig. A tesztelés során elért útvonalhosszak minden referenciafeladat esetén 0,12%-on belül vannak az optimális útvonaltól, valamint a futási idő is jól becsülhető a feladatméret ismeretében.

1. BEVEZETÉS

1.1. Az Utazó Ügynök Probléma

Az Utazó Ügynök Problémát Karl Menger fogalmazta meg először az 1930-as években, és azóta az egyik intenzíven kutatott területe a kombinatorikus optimalizálásnak.

Az Utazó Ügynök Probléma a következőképpen fogalmazható meg: egy ügynök feladata, hogy a cég központjából elindulva látogassa meg az összes meglátogatandó helyet, majd térjen vissza a kiinduló helyre úgy, hogy közben a legrövidebb útvonalat tette meg.[4]

Ezen optimalizálási probléma jelentősége abban rejlik, hogy az eredeti probléma és különböző változatai számos területen alkalmazhatók, például a logisztika, tervezés és mikrochip-gyártás területén.

Az Utazó Ügynök Probléma megfogalmazható gráfkeresési problémaként:

$$G_{TSP} = (V_{cities}, E_{conn})$$
$$V_{cities} = \{v_1, v_2, \dots, v_n\}, E_{conn} \subseteq \{(v_i, v_j) | i \neq j\} \quad (1)$$
$$C : V_{cities} \times V_{cities} \rightarrow R, C = (c_{ij})_{nxn}$$

[1] Egyetemi tanár, Széchenyi István Egyetem, Logisztika Tanszék.
[2] Egyetemi tanár, Széchenyi István Egyetem, Informatika Tanszék.
[3] Széchenyi István Egyetem, Informatika Tanszék.
[4] Applegate, D. L. – Bixby, R. E. – Chvátal, V. – Cook, W. J.: *The Traveling Salesman Problem: A Computational Study*, 2006, Princeton University Press, Princeton; Gutin, G. – Punnen, A. P.: *The Traveling Salesman. Problem and Its Variations*, 2007, Springer-Verlag US., 1–28 o.

ahol C a költségmátrix, c_{ij} pedig a i. és a j. csúcs közötti csúcs távolságát jelenti.

A cél megtalálni a legkisebb költségű irányított Hamilton-kört. Másképpen fogalmazva, a csúcsok azon permutációját, amelyik a legkisebb költséget (úthosszat) eredményezi.

$$C(i) = \left(\sum_{i=1}^{n-1} c_{p_i p_{i+1}}\right) + c_{p_n p_1} \tag{2}$$

A költségmátrix tulajdonságától függően a probléma két csoportba sorolható: ha a költségmátrix szimmetrikus, akkor szimmetrikus, ha nem, akkor aszimmetrikus Utazó Ügynök Problémáról beszélhetünk.

Az Utazó Ügynök Probléma az NP-nehéz osztályba tartozik. Minden NP-beli probléma visszavezethető rá, de feltétlenül van NP-ben.[5]

A jelenlegi tudásunk szerint az NP-nehéz problémákhoz nem tudunk olyan algoritmust megadni, amely garantáltan megoldja a feladatot szubexponenciális idő alatt. Az NP-nehéz problémák heurisztikus módszerekkel kezelhetők, mert elfogadható időn belül tudnak optimális, illetve közel optimális megoldásokat eredményezni.

1.2. Korábbi munkánk

Az utóbbi évek során számos populáció alapú evolúciós algoritmust (genetikus algoritmus,[6] bakteriális evolúciós algoritmus,[7] részecske raj optimalizálás[8] és memetikus változataik[9]) vizsgáltunk, és hasonlítottunk össze tulajdonságaik szerint különböző függvényeken tesztelve őket.

2005-ben Moscato[10] ötletét alapul véve bemutatásra került egy memetikus bakteriális evolúciós algoritmus, amely a bakteriális evolúciós algoritmust a Levenberg-Marquardt módszerrel kombinálja.[11]

2016-ben publikáltuk az Utazó Ügynök Probléma megoldására kifejlesztett

[5] Karp, R. M.: *Reducibility among Combinatorial Problems. Complexity of Computer Computations*, 1972, New York, 85–103. o.
[6] Holland, J. H.: *Adaption in Natural and Artificial Systems*, 1992, The MIT Press, Cambridge.
[7] Nawa, N. E. – Furuhashi, T.: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, in *IEEE Transactions on Fuzzy Systems*, 7. évf., 1999, 608–616 o.
[8] Kennedy, J. – Eberhart, R.: Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks ICNN 1995, Perth, WA, Australia*, vol. 4, 1942–1948. o., 1995.
[9] Balázs, K. – Botzheim, J. – Kóczy, T. L.: Comparison of Various Evolutionary and Memetic Algorithms, in *Integrated Uncertainty Management and Applications, AISC*, nr. 68, 2010, Springer-Verlag, Berlin–Heidelberg, 431–442. o.; Botzheim, J. – Cabrita, C. – Kóczy, L. T. – Ruano, A. E.: Fuzzy rule extraction by bacterial memetic algorithms, in *Proceedings of the 11th World Congress of International Fuzzy Systems Association, IFSA 2005*, Beijing, China, 1563–1568. o.
[10] Moscato, P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts. Towards Memetic Algorithms*, Technical Report Caltech Concurrent Computation Program, Report 826, California Institute of Technology, Pasadena, USA, 1989.
[11] Botzheim – Cabrita – Kóczy – Ruano: i. m.

Diszkrét Bakteriális Memetikus Algoritmust az Utazó Ügynök Probléma megoldására.[12] Az algoritmust 1500 pont alatt nagyságú problémákon teszteltük, amelyeknél optimális, illetve közel optimális útvonalakat kaptunk. Más a szakirodalomban szereplő módszerekkel ellentétben a futási idő előre megbecsülhető, a pontok elhelyezése nincs nagy hatással a futási időre.

A 2. fejezetben a DBMEA algoritmus gyorsítási lehetőségeit (párhuzamosítás, lokális keresés gyorsítása) fogjuk bemutatni. A 3. fejezet a teszteredményeket, a 4. fejezet pedig a következtetéseket, illetve a későbbi terveket tartalmazza.

2. DISZKRÉT BAKTERIÁLIS MEMETIKUS EVOLÚCIÓS ALGORITMUS

A Diszkrét Bakteriális Memetikus Evolúciós Algoritmus (DBMEA) algoritmus egy memetikus algoritmusnak tekinthető, hiszen a globális keresést (bakteriális evolúciós algoritmus) lokális kereséssel (2-opt, 3-opt) kombinálja. Minden iterációban a populáció egyedeire lokális keresést is alkalmazunk.

A klasszikus evolúciós algoritmusok a globális térben végzik a keresést, így lassan konvergálnak az optimális megoldáshoz. A lokális keresési módszerek a megoldás egy szűk környezetében próbálnak meg javítani, ennek köszönhetően a legközelebbi lokális optimumhoz konvergálnak, de a konvergálás sebessége nagyobb lesz. A kettő kombinálása gyakran jó tulajdonságokat, közel-optimális megoldásokat és gyors konvergenciát eredményez.

A bakteriális evolúciós algoritmus a baktériumok evolúciós fejlődésén alapszik. Két operátort használ a populáció egyedeinek javítására, a bakteriális mutációt és a génátadást.[13] A bakteriális evolúciós algoritmus folyamata a következő:

- Kezdeti populáció felállítása
- Minden iterációban:
  - Bakteriális mutáció
  - Lokális keresés
  - Génátadás

2.1. Kezdeti populáció előállítása

A populáció egyedekből áll, amelyek mindegyike egy megoldást jelent az adott feladatra. A DBMEA algoritmus esetén a populáció egyedei egy-egy lehetséges útvonalat reprezentálnak. Minden csúcs egy indexszel van összepárosítva (0..*n-1*), ahol *n* a csúcsok száma. A 0. sorszámú csúcs jelöli a kiindulási helyet.

[12] Kóczy, L. T. – Földesi, P. – Tüű-Szabó, B.: A Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem, in *IEEE World Congress on Computational Intelligence (WCCI 2016), Vancouver, Canada*, 2016, 3261–3267. o.; Kóczy, L. T. – Földesi, P. – Tüű-Szabó, B.: An effective Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem, in *International Journal of Intelligent Systems*, 2017, http://onlinelibrary.wiley.com/doi/10.1002/int.21893/full.

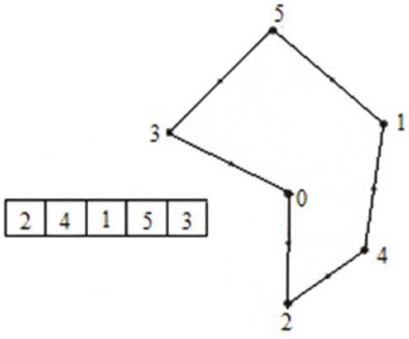
[13] Nawa, N. E. – Furuhashi, T.: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, in *IEEE Transactions on Fuzzy Systems*, 7. évf., 1999, 608–616 o.

Az egyed kódolása nem fogja tartalmazni a kiinduló helyet, tehát a kód *n-1* hosszúságú lesz, mivel metrikus problémákat vizsgálunk, ezért minden csúcs egyszer fog szerepelni a kódban. A kód azt a sorrendet követi, ahogy a kiinduló csúcsból az egyes csúcsok látogatásra kerülnek (1. ábra).

A DBMEA algoritmus esetén a kezdeti populáció tartalmaz random és determinisztikus egyedeket is, mivel a determinisztikus egyedek javítják az algoritmus hatékonyságát.[14] A kezdeti populáció determinisztikus egyedei a következők lesznek, amelyek mindegyike könnyen implementálható és gyors lefutású:

- Legközelebbi Szomszéd Heurisztika: Olyan determinisztikus módszert jelöl, amelynél mindig a legközelebbi még nem meglátogatott csúcs kerül meglátogatásra.
- 2. Legközelebbi Szomszéd Heurisztika: Minden lépésben a 2. legközelebbi még meglátogatatlan csúcs kerül meglátogatásra.
- Váltakozó legközelebbi és 2. Legközelebbi Szomszéd Heurisztika: Ez az előző kettő heurisztika kombinációja, ahol váltakozva hol a legközelebbi, hol a 2. legközelebbi még meglátogatatlan csúcs kerül meglátogatásra.

1. ábra: Egyedek kódolása



2.2. Bakteriális mutáció

A bakteriális mutáció művelete egyedenként kerül végrehajtásra. A bakteriális mutáció folyamata a következő (2. ábra):

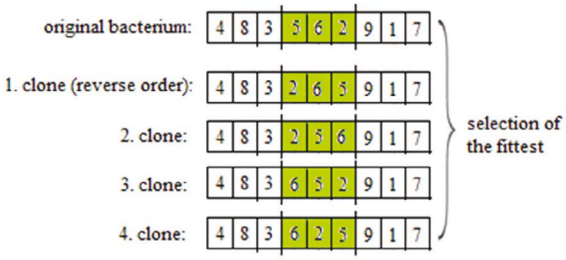
- A kezdeti lépésben N_{clones} darab klónt képezünk az eredeti egyedet másolva. Az egyed kódját megadott hosszúságú szegmensekre osztjuk. Szegmens elemeinek nem feltétlenül kell egymás mellettinek lenniük.
- Véletlenszerűen kiválasztunk egy szegmenst, és a klónoknál a szegmens elemeinek sorrendjét véletlenszerűen megváltoztatjuk. Az eredeti egyedben

[14] Földesi, P. – Botzheim, J.: Modeling of loss aversion in solving fuzzy road transport traveling salesman problem using eugenic bacterial memetic algorithm, in *Memetic Computing*, 2. évf., 2010/4. szám, 259–271. o.

a kiválasztott szegmens elemeinek sorrendje változatlan marad. Az egyik klón a szegmens elemeit fordított sorrendben tartalmazza.

- Kiválasztjuk a legkisebb költségű egyedet a klónok és az eredeti egyed közül, és ezen klón szegmensét átmásoljuk a többi klónba és az eredeti egyedbe.
- A bakteriális mutáció végén, miután az összes szegmenst megvizsgáltuk, a legkisebb költségű klón lesz az új egyed, a populációban a többi klón törlésre kerül.

2. ábra: Bakteriális mutáció



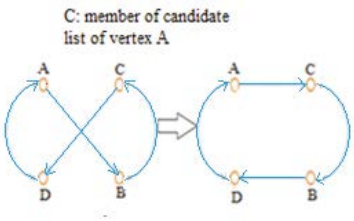
A bakteriális mutáció időkomplexitása generációként $O(N_{ind}N_{clones}n^2)$, tárkomplexitása $O(N_{ind}N_{clones}n)$.[15]

2.3. Lokális keresés és gyorsítása

A lokális keresés során a feladat egy lehetséges megoldásából indulunk ki, és ez iteratívan kerül javításra a megoldás környezetében keresve.[16]

A DBMEA algoritmus esetén 2-opt és 3-opt lokális keresést alkalmazunk. 2-opt lokális keresés esetén úgy javítjuk iteratívan az útvonalat, hogy két élt (AB, CD) másik két éllel (AC, BD) cseréljük, ha a csere csökkenti az útvonal hosszát (|AB|+|CD|>|AC|+|BD|) (3. ábra).

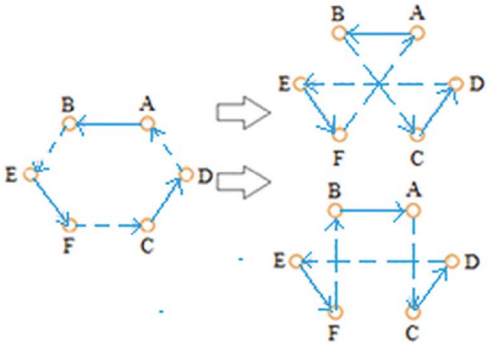
3. ábra: 2-opt lokális keresés



[15] Földesi – Botzheim: i. m.
[16] Hoos, H. H. – Stutzle, T.: *Stochastic Local Search: Foundations and Applications*, 2005, Morgan Kaufmann, San Francisco, 373–376. o.

3-opt lokális keresés esetén 3 élt cserélünk ki másik 3 éllel, ha a csere az útvonal hosszának csökkenését eredményezi. 3-opt lokális keresés esetén kétféleképpen lehet összekapcsolni a részútvonalakat, hogy körutat eredményezzen. Az eredeti körút és a 3 él cseréjével létrejövő két körút közül a legrövidebb lesz az új megoldás (4. ábra).

4. ábra: 3-opt lokális keresés



A lokális keresés gyorsítására az alábbi módszereket használtuk:

- megvizsgálandó csúcsok listája (candidate list): Minden csúcshoz létrehozunk egy listát, amely az adott csúcshoz legközelebbi csúcsok sorszámát tartalmazza. A listák hossza a probléma méretének négyzetgyöke. A lokális keresés során nem vizsgáljuk meg az összes élt útvonalcsökkenést keresve, hanem csak azokat, amelyek a listában szerepelnek.
- don't look back bitek alkalmazása: Minden egyes csúcshoz egy bitet rendelünk hozzá, amelyek kezdetben 0 értékűek. Ha a keresés során egy adott csúcshoz nincs olyan élcsere, amely az úthossz javulását eredményezné, akkor a csúcshoz tartozó bit értékét 1-re állítjuk, és a következő keresési lépésben már nem vizsgáljuk meg ezt a csúcsot javulást keresve. Azoknak a bitjét, amelyek részt vettek az élcserében, az iteráció végén 1-re állítjuk.[17]
- fix sugarú keresés: Minden u_i csúcs esetén egy fix sugarú (az útvonalban a csúcs és a vele összeköttetésben levő egyik u_j csúcs távolsága c_{u_i,u_j}) körön belül keressük az u_k csúcsot az útvonal javításához. 3-opt lokális keresés esetében 2 fix sugarú keresésre van szükség: egyik c_{u_i,u_j} sugárral, a másik $c_{u_i,u_j} + c_{u_k,u_l} - c_{u_i,u_k}$ sugárral, ahol u_l az u_k csúcs szomszédja az útvonalban.[18]

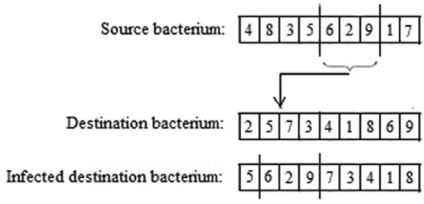
[17] Hoos – Stutzle: i. m.
[18] Uo.

2.4. Génátadás

A génátadás a populáción belül az egyedek közötti információcserét teszi lehetővé. A génátadás folyamata a következő (5. ábra):

- Az egyedeket útvonalhossz alapján növekvő sorrendbe állítjuk, majd a populációt két részre osztjuk. Az egyik rész tartalmazza a jó (rövid útvonalhosszú) egyedeket, a másik a rossz (hosszú útvonalú) egyedeket.
- N_{inf} alkalommal véletlenszerűen kiválasztunk egy-egy egyedet a populáció mindkét feléből (forrás- és célbaktérium). Ezután a jó egyed a megoldásának egy meghatározott hosszú részét átadja a rossz egyednek. Majd a módosult célbaktérium beillesztésre kerül a sorba rendezett populációba.

5. ábra: Génátadás



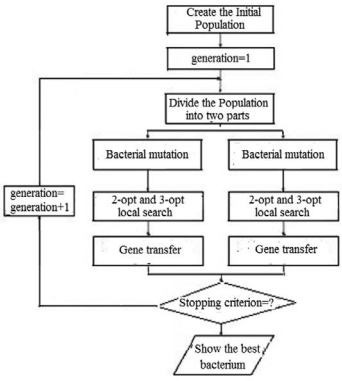
A génátadás műveletének időkomplexitása , mivel az útvonalhossz kiszámítása $O(N_{ind}n)$, a sorba rendezés $O(N_{ind}logN_{ind})$ a módosult célbaktérium visszaillesztése a sorba rendezett populációba $O\left(N_{\inf(n+N_{ind})}\right)$ nagyságrendű.[19] $C_{GT} = O\left(N_{ind}(n + logN_{ind}) + N_{\inf(n+N_{ind})}\right)$

2.5. A DBMEA algoritmus párhuzamosítása

Elkészítettük a DBMEA algoritmus párhuzamosított változatát OpenMP[20] segítségével. A párhuzamos algoritmusban a bakteriális mutáció, a lokális keresés és a génátadás kerül párhuzamosan végrehajtásra. A párhuzamos algoritmus folyamatábrája két szálon történő futtatása esetén a 6. ábrán látható.

[19] Földesi – Botzheim: i. m.
[20] Chandra, R. – Menon, R. – Dagum, L. – Kohr, D. – Maydan, D. – McDonald, J.: *Parallel Programming in OpenMp*, 2000, Academic Press, USA.

6. ábra: Párhuzamos DBMEA algoritmus folyamatábrája 2 szál esetén



3. TESZTELÉSI EREDMÉNYEK

A javított (párhuzamosított és szűkített lokális keresésű) algoritmust VLSI referenciaproblémákon teszteltük.[21] A csúcsok közötti távolságok valós értékként kerültek kiszámításra. A Concorde algoritmust úgy módosítottuk, hogy valós értékű úthosszokat számoljon. Többféle paraméterbeállítást kipróbáltunk, az alábbi mutatta a leggyorsabb konvergenciát:

- populáció egyedeinek száma (N_{ind}=100)
- klónok száma a bakteriális mutáció során (N_{clones}= n_{cities}/25)
- infekciók száma a génátadás során (N_{inf}= n_{cities}/25)
- bakteriális mutáció szegmenshossza (I_{seg}=10)
- átadott szegmens hossza a génátadás során (I_{trans}=n_{cities}/10)
- a lokális keresés során megvizsgálandó csúcsok listájának (candidate list) hossza (problémaméret négyzetgyöke)
- megállási kritérium: az egyedek legjobb 50%-a megegyezik.

3.1. Útvonalhosszak összehasonlítása

Az 1. táblázat mutatja a tesztelés során kapott úthossz-eredményeket. Megállapítható, hogy a javított DBMEA algoritmus mindegyik referenciafeladat esetén optimális, illetve közel optimális útvonalhosszokat eredményezett, az optimális útvonaltól való eltérés a legrosszabb esetben 0,12% volt.

[21] VLSI TSP dataset, http://www.math.uwaterloo.ca/tsp/vlsi/index.html, 2015. április.

1. táblázat: A tesztelés során kapott útvonalhosszak

Név	Legrövidebb útvonal hossza a javított DBMEA algoritmus esetén	Átlagos útvonal hossza javított DBMEA algoritmus esetén (5 futtatás)	Optimális útvonal hossza	Eltérés az optimális útvonaltól [%]
xql662	2 550,84	2 551,63	2 550,84	0
dkg813	3 243,41	3 243,41	3 243,91	0
dka1376	4 738,61	4 742,29	4 736,87	0,04
dca1389	5 156,56	5 157,34	5 152,69	0,07
dja1436	5 333,84	5 335,56	5 328,21	0,1
icw1483	4 470,57	4 473,83	4 465,88	0,1
rbv1583	5 446,09	5 449,01	5 444,67	0,03
rby1599	5 591,06	5 595,8	5 584,16	0,12
pds2566	7 774,91	7 781,57	7 766,708	0,1

3.2. Futási idők összehasonlítása

A 2. táblázatban a javított DBMEA algoritmus futási időit hasonlítottam össze a módosított (egészekre kerekítés helyett valós értékű élhosszokat számoló) Concorde algoritmus futási idővel, amely az egyik leghatékonyabb pontos megoldást biztosító algoritmus.[22] Látható, hogy az esetek többségében a javított DBMEA algoritmus gyorsabb, valamint prediktabilitás szempontjából is jobban teljesít a Concorde algoritmusnál. A Concorde esetén a futási idő nem becsülhető előre, mert a pontok elhelyezkedése nagymértékben befolyásolja a futási időt. A javított DBMEA algoritmusnál viszont a pontok elhelyezkedése nincs nagy hatással a futási időre, a feladat méretének ismeretében előre megbecsülhető.

Az eredeti DBMEA algoritmus futási eredményei Kóczy – Földesi – Tüű-Szabó An effective Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem című cikkében találhatók. Például az xql662 esetén az ezen cikkben javasolt gyorsításokkal 23,14-szeres gyorsulást sikerült elérni.[23]

A szakirodalomban található leggyorsabb heurisztikus módszerhez (Helsgaun-féle Lin-Kernighan) képest a javított DBMEA algoritmus ugyan kb. 1 nagyságrenddel lassabb, viszont az a módszer csak a szimmetrikus Utazó

[22] Applegate, D. L. – Bixby, R. E. – Chvátal, V. – Cook, W. J. – Espinoza, D. – Goycoolea, M. – Helsgaun, K.: Certification of an Optimal Tour through 85900 Cities, in *Operations Research Letters*, 37. évf., 2009/1, 11–15. o.

[23] Kóczy, L. T. – Földesi, P. – Tüű-Szabó, B.: An effective Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem, in *International Journal of Intelligent Systems*, 2017, http://onlinelibrary.wiley.com/doi/10.1002/int.21893/full.

Ügynök Probléma esetén alkalmazható, a mi módszerünk viszont kis módosításokkal alkalmas lesz más Utazó Ügynök Probléma változatok (időablakos, időfüggő Utazó Ügynök Probléma stb.) kezelésére is.

2. táblázat: A futási idők összehasonlítása

Név	A javított DBMEA algoritmus átlagos futási ideje	A javított DBMEA algoritmus futási idejének tartománya	Módosított Concorde algoritmus futási ideje	Párhuzamosítással nyert gyorsulás
xql662	287,35 s	256,77 – 321,59 s	157,27 s	1,54
dkg813	513,49 s	474,11 – 534,53 s	2 031,11 s	1,69
dka1376	4 640,81 s	3 934,12 – 5 494,63 s	807,11 s	1,75
dca1389	5 018,35 s	4 729,46 – 5 491,89 s	380 155,1 s	1,78
dja1436	4 951,08 s	4 528,02 – 5 305,59 s	69 811,21 s	1,85
icw1483	6 118,88 s	5 686,68 – 6 589,64 s	1 690,34 s	1,85
rbv1583	6 553,3 s	5 993,29 – 6 956,67 s	1 134,22 s	1,75
rby1599	7 810,28 s	7 177,69 – 8 541,25 s	211 312,32 s	1,89
pds2566	13 456,76 s	12 177,71 – 15 946,74 s	102 797,26 s	1,82

4. ÖSSZEFOGLALÁS, JÖVŐBELI TERVEK

A cikkben bemutatott gyorsítási lehetőségekkel jelentős gyorsulást sikerült elérni az eredeti DBMEA algoritmushoz képest. A javított DBMEA algoritmus a vizsgált referenciaproblémák esetén optimális, illetve közel optimális megoldásokat eredményezett.

Szeretnénk későbbiekben az algoritmust az Utazó Ügynök Probléma különböző változatain tesztelni. Első eredményeink az algoritmus időablakos Utazó Ügynök Problémákon való tesztelése során biztatóak, sikerült az optimális megoldásokat megtalálni az algoritmussal.

FELHASZNÁLT IRODALOM

- Applegate, D. L. – Bixby, R. E. – Chvátal, V. – Cook, W. J.: *The Traveling Salesman Problem: A Computational Study*, 2006, Princeton University Press, Princeton.
- Applegate, D. L. – Bixby, R. E. – Chvátal, V. – Cook, W. J. – Espinoza, D. – Goycoolea, M. – Helsgaun, K.: Certification of an Optimal Tour through 85900 Cities, in *Operations Research Letters*, 37. évf., 2009/1, 11–15. o.
- Balázs, K. – Botzheim, J. – Kóczy, T. L.: Comparison of Various Evolutionary and Memetic Algorithms, in *Integrated Uncertainty Management and Applications, AISC*, nr. 68, 2010, Springer-Verlag, Berlin–Heidelberg, 431–442 o.
- Botzheim, J. – Cabrita, C. – Kóczy, L. T. – Ruano, A. E.: Fuzzy rule extraction by bacterial memetic algorithms, in *Proceedings of the 11th World Congress of International Fuzzy Systems Association, IFSA 2005*, Beijing, China, 1563–1568 o.
- Chandra, R. – Menon, R. – Dagum, L. – Kohr, D. – Maydan, D. – McDonald, J.: *Parallel Programming in OpenMp*, 2000, Academic Press, USA.
- Földesi, P. –Botzheim, J.: Modeling of loss aversion in solving fuzzy road transport traveling salesman problem using eugenic bacterial memetic algorithm, in *Memetic Computing*, 2. évf., 2010/4. szám, 259–271. o.
- Gutin, G. – Punnen, A. P.: *The Traveling Salesman. Problem and Its Variations*, 2007, Springer- Verlag US.
- Holland, J. H.: *Adaption in Natural and Artificial Systems*, 1992, The MIT Press, Cambridge.
- Hoos, H. H. – Stutzle, T.: *Stochastic Local Search: Foundations and Applications*, 2005, Morgan Kaufmann, San Francisco.
- Karp, R. M.: *Reducibility among Combinatorial Problems. Complexity of Computer Computations*, 1972, New York.
- Kennedy, J. – Eberhart, R.: Particle swarm optimization, in *Proceedings of the IEEE International Conference on Neural Networks ICNN 1995, Perth, WA, Australia*, vol. 4, 1942–1948. o.
- Kóczy, L. T. – Földesi, P. – Tüű-Szabó, B.: A Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem, in *IEEE World Congress on Computational Intelligence (WCCI 2016), Vancouver, Canada*, 2016, 3261–3267. o.
- Kóczy, L. T. – Földesi, P. – Tüű-Szabó, B.: An effective Discrete Bacterial Memetic Evolutionary Algorithm for the Traveling Salesman Problem, in *International Journal of Intelligent Systems*, 2017, http://onlinelibrary.wiley.com/doi/10.1002/int.21893/full.
- Moscato, P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts.Towards Memetic Algorithms*, Technical Report Caltech Concurrent Computation Program, Report 826, California Institute of Technology, Pasadena, USA, 1989.

▪ Nawa, N. E. – Furuhashi, T.: Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, in *IEEE Transactions on Fuzzy Systems*, 7. évf., 1999, 608–616 o.
▪ VLSI TSP dataset http://www.math.uwaterloo.ca/tsp/vlsi/index.html.