Problém obchodného cestujúceho

Martin Macej

Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 2, Bratislava
martinmacej8@gmail.com

Abstrakt. Keďže efektívne plánovanie a vytváranie dopravnej trasy môže znížiť prepravné náklady rôznych spoločností, je dobré poznať a chápať algoritmy, ktoré sa na túto problematiku zameriavajú. V súčasnej dobe sa viacero štúdií zameriava na problém obchodného cestujúceho, ktorý je vlastne paralelou k rýchlemu hľadaniu najkratšej cesty medzi viacerými bodmi.
V tejto práci sa zameriame práve na tento problém a na možnosti jeho riešenia. Porovnáme a vyskúšame viaceré algoritmy a pokúsime sa nájsť všeobecne čo možno najoptimálnejšie riešenie či už jedným algoritmom alebo kombináciou viacerých.

Kľúčové slová: problém obchodného cestujúceho, najkratšia cesta, prírodou inšpirované algoritmy, optimalizácia

1 Úvod

Problém obchodného cestujúceho je jedným z najznámejších a najpopulárnejších problémov, ktorý bol predstavený v roku 1800 W.R. Hamilton-om a T. Kirkman-om [1] .

Tento problém je nasledujúci: máme niekoľko miest a poznáme náklady spojené s presunom medzi jednotlivými mestami, resp. poznáme dĺžky ciest medzi nimi. Cieľom je nájsť najkratšiu cestu (cestu s minimálnymi nákladmi) tak, aby sme každé mesto navštívili práve raz.
Ak by sme chceli nájsť všetky cesty a nájsť tak určite najkratšiu cestu, museli by sme nájsť $\frac{1}{2} * (n-1)!$ riešení, kde n je počet miest a platí podmienka n > 2. Ako je vidno zo výrazu $\frac{1}{2} * (n-1)!$, zložitosť tohto problému je pomerne vysoká (n!) a preto nie je časovo možné skúšať všetky možnosti pri veľkom množstve vrcholov. Kvôli tejto výpočtovej zložitosti patrí problém obchodného cestujúceho medzi NP ťažké problémy. Na jeho riešenie sa používajú rôzne techniky a algoritmy, ktoré budú bližšie opísané a vysvetlené nižšie.

Problém obchodného cestujúceho môže byť riešený viacerými metódami, akými sú napríklad neurónové siete, memetické výpočty, simulované žíhanie, genetické algoritmy alebo rôzne prírodou inšpirované algoritmy, akými sú napríklad optimalizácia kolóniou mravcov, optimalizácia rojom častíc a mnohé ďalšie [2, 3, 4, 5] .

V súčasnosti sa najviac na problémy nájdenia najkratšej cesty, do ktorej patrí aj problém obchodného cestujúceho, používa metóda optimalizácie kolóniou mravcov a jej rôzne kombinácie s inými algoritmami [1].

V tejto práci sa pokúsime pozrieť na pár algoritmov, ktoré riešia tento problém a porovnať ich pri rôznych typoch vstupov. Pokúsime na zistiť, ktorý algoritmus nám dáva najlepšie výsledky, čo sa týka dĺžky trasy a najlepšie výsledky z hľadiska dĺžky vykonávania algoritmu. Zameriame sa primárne na rojové algoritmy (algoritmus kolónie mravcov a včelie algoritmy), keďže práve tieto algoritmy sú pomerne dobre známe a sú spomínané vo väčšine existujúcich riešení.
Implementované algoritmy sme sa rozhodli overiť na 4 rôznych scenároch, a to na hľadaní najkratšej cesty pre 10, 20, 50 a 100 vrcholov. Vyhodnocovať úspešnosť algoritmu budeme na základe dĺžky najkratšej cesty, ktorú nájde.

2 Podobné práce

V práci [5] sa autori zamerali na porovnanie viacerých algoritmov, ktoré riešia problém obchodného cestujúceho.
Porovnávali genetický algoritmus (GA), algoritmus roju častí (PSO), algoritmus kolónie mravcov (ACO) a algoritmus umelých včiel (ABC).

Na testovanie pustili každý z algoritmov 10 krát a porovnávali výsledky t.j. najkratšie cesty. Algoritmus ABC v implementácii, ktorú použili mal najlepší výkon v porovnaní s ostatnými algoritmami a to najmä pri problémoch, kde bolo viacero miest. Ďalším zistením bolo, že algoritmus ACO znižuje svoj výkon pri vyššom počte miest kvôli komplexnosti algoritmu. A teda je vhodnejší pre problémy, kde je počet miest menší ako 80. Algoritmus ABC má dobré schopnosti globálneho vyhľadávania a dokáže nájsť optimálnu cestu. Najrýchlejším algoritmov spomedzi vyššie spomenutých je práve ABC. Výkon ostatných sa pri vysokom počte uzlov značne redukuje.
V ďalšom riešení [6] sa opäť zamerali na porovnanie viacerých algoritmov, ktoré riešia problém obchodného cestujúceho. Porovnali algoritmus najbližších susedov (NN), genetický algoritmus (GA), simulované žíhanie (SA), tabu search (TS), optimalizáciu kolóniou mravcov (ACO) a optimalizáciu fyziológie stromov (TPO). Pri porovnaniach zistili, že najdlhší čas výpočtu trasy je u TS a ACO. Oba z týchto algoritmov taktiež vykazovali exponenciálny rast času potrebného pre výpočet najlepšej cesty. Naopak, najrýchlejší čas výpočtu bol pri algoritmoch NN, TPO a GA.

Na základe týchto zistení sme sa rozhodli implementovať algoritmus kolónie mravcov, ktorý by mal dávať dobré výsledky pri výpočte trasy avšak dĺžka jeho trvania je relatívne vysoká.

Ďalej sme sa rozhodli implementovať algoritmus kolónie včiel a algoritmus inteligentných včiel, ktoré môžu mať horšie výsledky ale doba výpočtu cesty týmito algoritmami by mala byť omnoho rýchlejšia ako pri kolónii mravcov.

3 Opis použitých algoritmov

Algoritmy optimalizácie kolóniou včiel a kolóniou umelých včiel sú implementované na základe článkov. Algoritmus optimalizácie pomocou kolónie mravcov je inšpirovaný existujúcim kódom [1], keďže má veľmi dobré výsledky, a je teda zbytočné ho znova implementovať. Kód je mierne upravený a má zmenené parametre na také, ktoré nám dávajú lepšie výsledky ako pôvodné.

3.1 BCO – optimalizácia kolóniou včiel

Ako prvé sa určí, koľko miest má navštíviť každá včela v jednom kole. Následne každá z včiel náhodne navštevuje definovaný počet miest. Po skončení kola sa určí najúspešnejšia včela, t.j. tá, ktorá zatiaľ prešla najmenšiu vzdialenosť a zistí sa doposiaľ najkratšia a najdlhšia vzdialenosť. Následne sa pre každú zo včiel vypočíta, či jej doposiaľ navštívené miesta predstavujú jednu z lepších alebo z horších trás a teda či prežije (trasa patrí medzi kratšie) alebo zahynie a vznikne nová včela, ktorá preberie miesta inej včely, ktorá má lepší predpoklad nájsť kratšiu cestu. Takýmto spôsobom sa populácia včiel rozdelí na 2 skupiny. Prvá skupina, tzv. recruiterov, ostáva nezmenená, druhá skupina sa nahrádza novými včelami, ktoré preberajú navštívené miesta niektorej zo včiel patriacich medzi recruiterov.

Tento postup sa opakuje, až pokým včely nenavštívia všetky mestá [5] .

Výhody riešenia:

- Jednoduchosť, flexibilita, robustnosť
- Používa málo parametrov oproti ostatným algoritmom
- Jednoduchá implementácia

```
BCO pseudokód:
  bees = createBees()
  while best_bee.not_all_cities_visited():
    move_all_bees(move_count)
    best_bee = bees.selecte_best_bee()
    max_distance, min_distance = get_ distance()
    for bee in bees:
      bee.is_recruiter = false
    for bee in bees:
      if bee.is_recruiter == false:
        random_number = random(0, 1)
        if random_number < 0.5:
          bee = get_random_recruiter_bee()
        else:
          bee = best_bee
  return best_bee.best_cost, best_bee.best_path
```

[1] https://github.com/ppoffice/ant-colony-tsp

3.2 ACO – optimalizácia kolóniou mravcov

Algoritmus kolónie mravcov funguje nasledovne. Každý mravec začína z náhodne vybraného bodu/mesta/vrcholu grafu. V každom kroku sa pohybuje po hrane grafu. Každý mravec si pamätá cestu, ktorú absolvuje a v každom svojom nasledujúcom kroku si vyberá hranu vedúcu k vrcholu, ktorý ešte nenavštívil.

Mravec našiel riešenie, ak navštívil všetky vrcholy grafu. Pri každom kroku sa rozhoduje, ktorú z hrán vedúcich k nenavštívenému vrcholu si vyberie. To, akou cestou sa vyberie, je pravdepodobnostne ovplyvnené hodnotou feromónu, ktorá je na hranách grafu tak, že hranu s vyššou hodnotou feromónu si vyberie s vyššou pravdepodobnosťou ako hranu, na ktorej je hodnota feromónu nižšia.

Keď všetky mravce dokončia svoju cestu, hodnoty feromónov na hranách grafu sa aktualizujú. Každá hrana stratí isté percento z jej prislúchajúcej hodnoty feromónu a následne je každej hrane táto hodnota feromónu zvýšená priamo úmerne tomu, ako často bola daná hrana navštívená. Tento postup sa opakuje, až po kým nie je splnené kritérium ukončenia algoritmu, ktorým môže byť napríklad počet iterácii [1, 2, 6] .

Výhody riešenia:

- Podporuje paralelizmus
- Efektívne pre problém obchodného cestujúceho a podobné druhy problémov
- Dá sa aplikovať aj do dynamických aplikácií, keďže sa prispôsobuje zmenám, akými môžu byť napríklad pridanie nových vzdialeností

Nevýhody riešenia

- Teoretická analýza je zložitá
- Obsahuje postupnosť náhodných rozhodnutí
- Čas konvergencie je neistý (aj keď konvergencia je istá)

```
ACO pseudokód:
  for i in generations:
    ants = createAnts()
    for i in ants:
      for i in range(0, all_cities):
        ant.select_next_town()
      ant.update_path_cost()
      if ant.cost < best_cost:
        best_cost = ant.cost
      # increase pheronome
      ant.update_ant_pheronome()
    # decrease pheronome after all ants
    update_pheronome()
  return best_cost, best_path
```

3.3 ABC – kolónia umelých včiel

ABC algoritmus je založený na inteligencii roja. V tomto modeli vystupujú 2 hlavné elementy, a to samotné včely a zdroje potravy (v prípade problému obchodného cestujúceho mestá). Roj včiel sa delí na 3 základné skupiny: zamestnané včely, nezamestnané včely (divákov) a skautov. Percentuálne rozdelenie je také, že polovicu populácie tvoria zamestnané včely a druhú polovicu nezamestnané včely. Následne v priebehu algoritmu zamestnané včely, ktoré majú dobré výsledky a sú teda nasledované veľkým množstvom nezamestnaných včiel, stávajú skautami a náhodne skúmajú ďalšie cesty.

Najkratšia cesta sa hľadá pomocou 3 pravidiel. Ako prvé sa pošlú zamestnané včely pre zdroje potravy a zistí sa „kvalita“ zdroju potravy, čo reprezentuje prejdenú vzdialenosť. Následne sa pošlú nezamestnané včely pre potravu a pri svojej ceste berú do úvahy informácie od zamestnaných včiel o dĺžke cesty a podobne. Posledným krokom je stanovenie skautských včiel, ktoré sa pošlú hľadať možné zdroje potravy.

Zdroje potravy si včely vyberajú v počiatočnej fáze náhodne a počíta sa kvalita nektáru. Tieto informácie potom zamestnané včely zdieľajú aj medzi čakajúcimi (nezamestnanými) včelami. Po zdieľaní informácií sa každá zamestnaná včela vráti k zdroju potravy navštívenému v cykle predtým (keďže ho má zapamätaný) a potom v blízkosti aktuálneho zdroja potravy vyberá iný zdroj. Nakoniec nezamestnané včely použijú získané informácie od zamestnaných včiel na výber potravy. Pravdepodobnosť výberu sa zvyšuje tým, ako vysoká bola kvalita nektáru. Práve kvôli tomu včela s najvyššou kvalitou nektáru prijíma nezamestnané včely na jej zdroj potravy. Následne si vyberie iný zdroj potravy v susedstve aktuálneho a stáva sa skautom. Ako skaut náhodne vygeneruje nový zdroj potravy, ktorý nahradí ten, ktorý opustili nezamestnané včely [5], [7], [8] .

```
ABC pseudokód:
  create_distance_table()
  assign_roles(onlookers, foragers)
  while cycle < max_cycle_limit:
    distance, path = bees_waggle()
    if distance < best_distance:
      best_distance = distance
      best_path = path
    recuit_distance, recruit_path = recruit()
    if recruit_distance < best_distance:
      best_distance = recruit_distance
      best_path = recruit_path
  return best_cost, best_path
```

4 Experimenty

Vo vyššie opísaných algoritmoch sme sa pri experimentoch zamerali na dĺžku cesty, akú našiel algoritmus a na čas, ako dlho daný algoritmus trval.

Vytvorili sme si 4 súbory so zoznamom miest, ktoré chceme navštíviť. Každý súbor obsahoval iný počet vrcholov (10, 20, 50 a 100) a tieto vrcholy boli rozmiestnené náhodne. Ide o úplný graf, kde každé miesto je prepojené s každým, je možné prechádzať jednu hranu oboma smermi a hrany nie sú individuálne ohodnotené ale ich hodnota je rovná vzdialenosti medzi miestami, ktoré spája.

Intervaly boli volené pre x aj y súradnice náhodne. Pre 10 to bolo z intervalu <0, 10>, pre 20 miest <0, 20>, pre 50 miest <0, 50> a pre 100 miest <0, 100>.

Keďže máme viacero algoritmov a chceli sme docieliť čo možno najväčšiu objektivitu, rozhodli sme sa pre rovnaký počet jedincov a to konkrétne pre 100 (100 jedincov nie je málo, dokážu dať dobré výsledky a zároveň to nie je ani veľa, aby algoritmus bežal priveľmi dlho).

Algoritmus kolónie umelých včiel obsahuje parameter, ktorý nám hovorí o percentuálnom obsadení včiel, ktoré sa pozerajú a ktoré pracujú. Tento parameter sme nastavili na 0,5 (polovica pracujúcich, polovica čakajúcich), kde tento pomer bol odporúčaný aj napríklad v jednom z článkov [7]. Percento včiel, ktoré sú prieskumné (scout) sme nastavili na 20%, počet cyklov, koľko maximálne môže včela na prieskume vykonať na 5. Toto číslo nám vyšlo ako jedno z najlepších, po pár experimentoch dávalo o trochu lepšie výsledky ako ostatné čísla (vykonanie po 10 experimentov s počtom miest 2, 5, 8, 12 a 20 na 1 cyklus, všeobecne najlepšie výsledky dávalo číslo 5).

Pre algoritmus kolónie mravcov sa nastavuje počet cyklov, koľko krát majú mravce vyraziť na cestu a hľadať cestu. Toto číslo sme sa rozhodli nastaviť na hodnotu 100.

Algoritmus kolónie včiel má tiež iba jeden špeciálny parameter a to počet pohybov, po koľkých sa vyberá najlepšia včela. Tento parameter sme nastavili na 6.

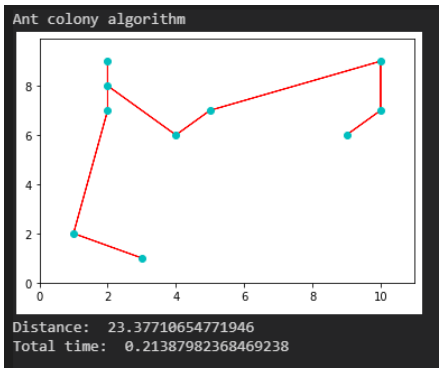
V nasledujúcich experimentoch nižšie je stále uvedený opis experimentu, tabuľka s výsledkami a grafické znázornenie nájdených ciest.

4.1 Hľadanie cesty pre 10 vrcholov

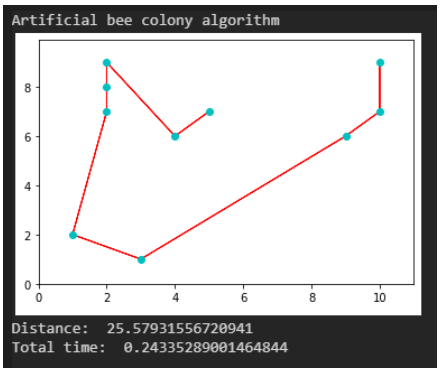
Najprv sme experimentovali s malým počtom miest (10). Výsledky je možné si pozrieť v tabuľke nižšie. Algoritmy boli spustené viac krát, výsledky sa samozrejme trochu líšili ale rozdiely neboli zásadné, keďže počet vrcholov je iba 10.

Tabuľka 1. Porovnanie algoritmov pre 10 vrcholov.

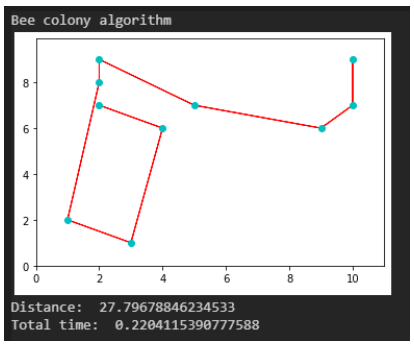
Algoritmus	ACO	ABC	BCO
Najkratšia vzdialenosť	**23,377**	25,579	27,797
Dĺžka výpočtu [s]	**0,214**	0,243	0,22



Obr. 1 ACO - 10 vrcholov.



Obr. 2 ABC - 10 vrcholov.



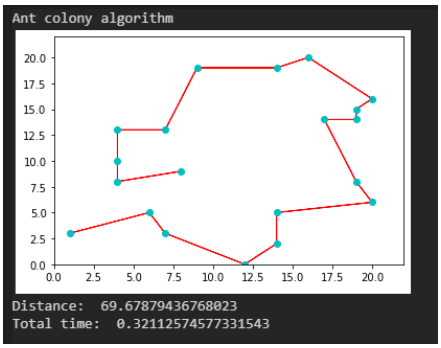
Obr. 3 BCO - 10 vrcholov.

4.2 Hľadanie cesty pre 20 vrcholov

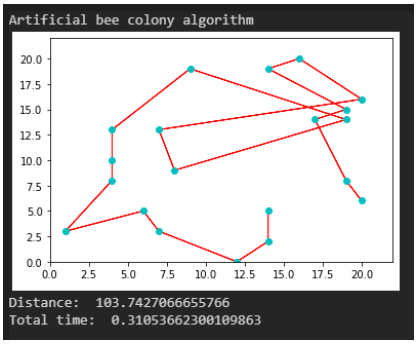
Ďalej sme zvýšili počet miest dvojnásobne (20 miest) a opakovali sme predchádzajúci postup. Tuto sa výsledky líšili už viac. Napríklad rozdiel medzi minimálnou a maximálnou hodnotou bol pri algoritme ACO 2, pri BCO až 30. Uvádzané sú najlepšie hodnoty z 10 pokusov. Vidíme, že algoritmus kolónie mravcov preukazuje omnoho lepšie výsledky ako algoritmy kolónií včiel. Časovo je algoritmus umelých včiel najrýchlejší (najrýchlejšie nám dá výsledok). Najhoršie z tohto experimentu vyšiel algoritmus kolónie včiel, ktorý našiel najkratšiu cestu zhruba 2x dlhšiu ako algoritmus kolónie mravcov a navyše aj trval 1,4-násobne dlhšie.

Tabuľka 2. Porovnanie algoritmov pre 20 vrcholov.

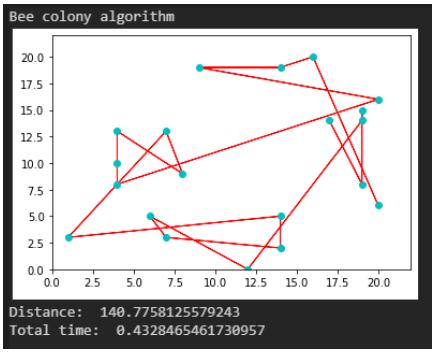
Algoritmus	ACO	ABC	BCO
Najkratšia vzdialenosť	**69,679**	103,743	140,776
Dĺžka výpočtu [s]	0,321	**0,311**	0,433



Obr. 4 ACO - 20 vrcholov.



Obr. 5 ABC - 20 vrcholov.



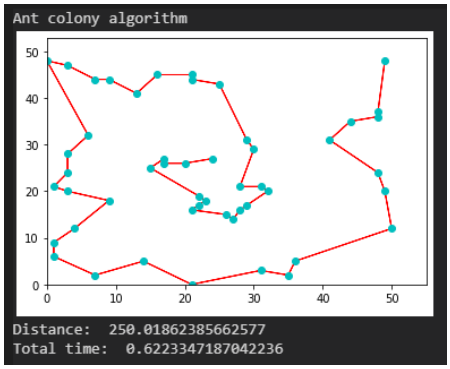
Obr. 6 BCO - 20 vrcholov.

4.3 Hľadanie cesty pre 50 vrcholov

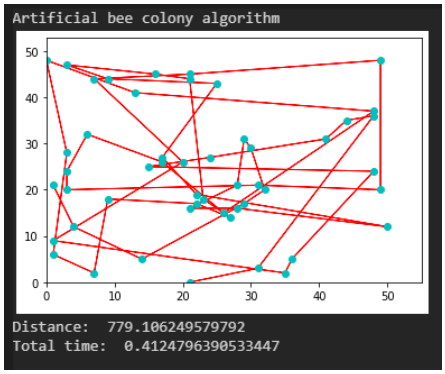
Následne sme pokračovali zvýšením počtu miest, a to na 50. V tomto prípade sme očakávali podobné poradie, čo sa nám po vykonaní experimentu aj potvrdilo. Najkratšiu cestu jednoznačne našiel algoritmus kolónie mravcov, ktorý našiel tretinovú cestu oproti ďalšiemu algoritmu. Z časového hľadiska opäť zvíťazil algoritmus umelých včiel, ktorý skončil viac ako 30% rýchlejšie ako algoritmus ACO. Opäť, ako najhorší sa javí algoritmus BCO, ktorý našiel najdlhšiu cestu a zároveň aj trval najdlhšie, zhruba 1,4-násobne dlhšie ako ABC a 4-krát dlhšie ako ACO.

Tabuľka 3. Porovnanie algoritmov pre 50 vrcholov.

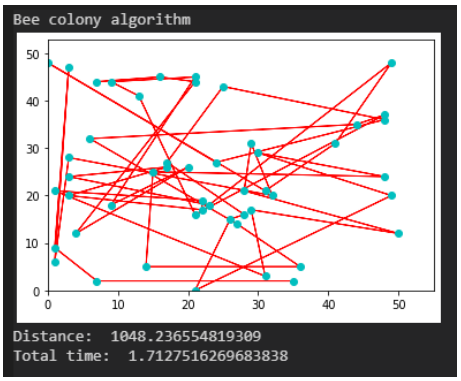
Algoritmus	ACO	ABC	BCO
Najkratšia vzdialenosť	**250,019**	779,106	1 048,237
Dĺžka výpočtu [s]	0,622	**0,412**	1,713



Obr. 7 ACO - 50 vrcholov.



Obr. 8 ABC - 50 vrcholov.



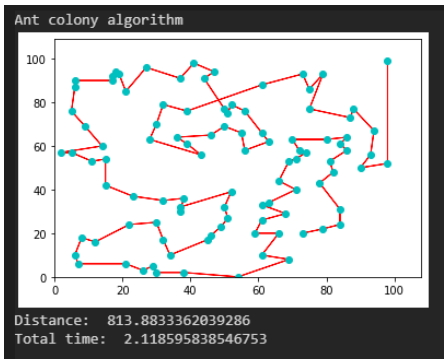
Obr. 9 BCO - 50 vrcholov.

4.4 Hľadanie cesty pre 100 vrcholov

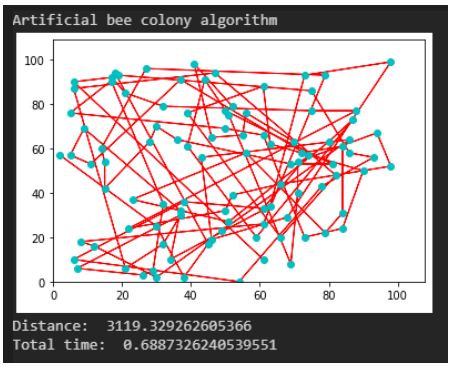
Nakoniec sme počet miest ešte raz zvýšili, a to 2-násobne na 100. V tomto experimente sa nám výsledky z predchádzajúcich experimentov potvrdili, dokonca aj ešte viac prehĺbili. V nasledujúcej tabuľke vidíme priepastný rozdiel v nájdení najkratšej cesty. Algoritmus ACO našiel cestu dĺžky necelých 814, ABC až vyše 3 119 a algoritmus BCO až viac ako 4 175. Rovnako vidíme obrovský rozdiel aj v dĺžke vykonávania programu. Vyhľadať najkratšiu cestu zvládol najrýchlejšie algoritmus umelých včiel, a to za približne 0,69 sekundy. Ďalej nasleduje algoritmus kolónie mravcov, ktorému to trvalo vyše 2 sekúnd. Až na konci je algoritmus BCO, ktorý bežal viac ako 8 sekúnd.

Tabuľka 4. Porovnanie algoritmov pre 100 vrcholov.

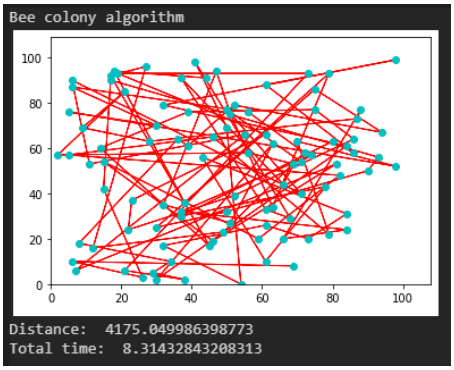
Algoritmus	ACO	ABC	BCO
Najkratšia vzdialenosť	**813,883**	3 119,329	4 175,05
Dĺžka výpočtu [s]	2,119	**0,689**	8,314



Obr. 10 ACO - 100 vrcholov.



Obr. 11 ABC - 100 vrcholov.



Obr. 12 BCO - 100 vrcholov.

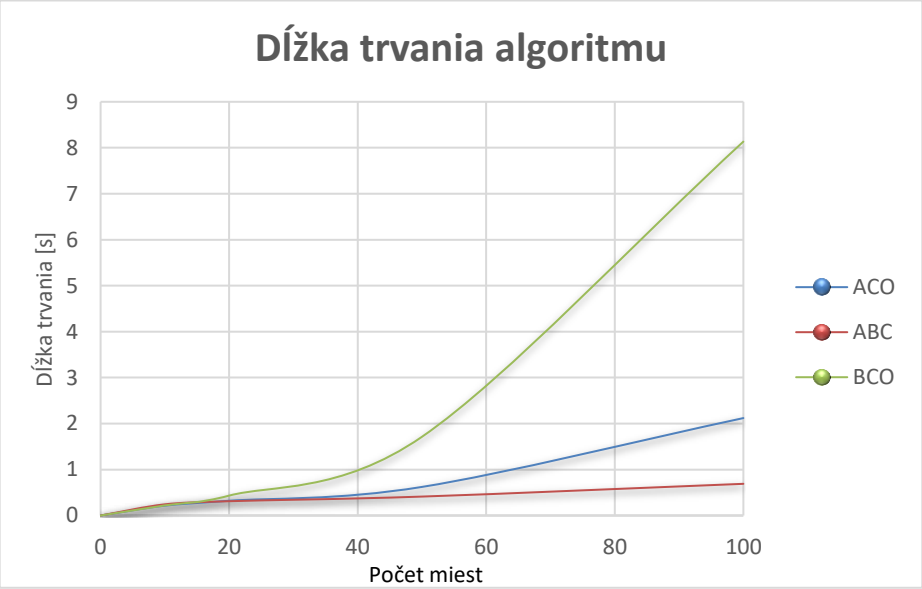
5 Záver

V tejto práci sme sa zamerali na 3 algoritmy, ktoré riešia problém obchodného cestujúceho, konkrétne na algoritmus kolónie mravcov, algoritmus kolónie včiel a algoritmus kolónie umelých včiel. Experimentovali sme, ako sa tieto algoritmy správajú pri malom ale aj vyššom počte vrcholov, ktoré majú navštíviť. Rovnako sme sa zamerali aj na rýchlosť každého z algoritmov.

Po viacerých experimentoch sme dospeli k nasledujúcim záverom. Najkratšiu cestu vie najlepšie nájsť algoritmus kolónie mravcov. Pri 10 navštívených miestach boli výsledky približne rovnaké, pri vyššom počte (od 20 miest) sa tento jav začal prejavovať pomerne jasne. Už pri 50 miestach bol algoritmus ACO 3-3,5x lepší ako zvyšné algoritmy a pri 100 miestach 3,5-4,5násobne.

Najrýchlejší algoritmus sa nám ukázal taktiež pomerne jasne. Pri 50 miestach bol algoritmus ABC zhruba o 30% rýchlejší ako ACO ale pri 100 miestach bol rozdiel už omnoho väčší, ABC bol 3,5x rýchlejší.

Najkratšiu cestu vie teda najlepšie z algoritmov ACO, ABC a BCO (v našej implementácii) nájsť algoritmus ACO. Najrýchlejším algoritmom je zase z tejto trojice algoritmus ABC. Ako nevhodné riešenie (čo sa týka výkonu aj správnosti) sa nám javí použitie kolónie včiel BCO.



Obr. 12 Porovnanie trvania algoritmov.

Tabuľka 5. Porovnanie algoritmov.

Algoritmus			ACO	ABC	BCO
Najkratšia vzdialenosť	počet miest	10	**23,377**	25,579	27,797
		20	**69,679**	103,743	140,776
		50	**250,019**	779,106	1 048,237
		100	**813,883**	3 119,329	4 175,05
Dĺžka výpočtu [s]	počet miest	10	**0,214**	0,234	0,22
		20	0,321	**0,311**	0,433
		50	0,622	**0,412**	1,713
		100	2,119	**0,689**	8,314

Výsledky sú pomerne rôzne z nasledujúcich dôvodov. Pri algoritme kolónie mravcov (ACO) každý jeden mravec prechodom cez hranu zanechá istú feromónovú stopu a tým pádom každým prechodom ovplyvní rozhodovanie každého mravca.
Pri algoritme kolónie včiel (BCO) sa za najlepšiu včela včelu prehlasuje tá, ktorá prešla doposiaľ najkratšiu vzdialenosť a preberá na seba následne niekoľké ďalšie (horšie), tento proces sa opakuje až kým najlepšia včela neprešla všetky miesta, takže konvergencia tohto riešenia nie je istá.
Pri algoritme umelých včiel (ABC) máme trochu lepšie výsledky ako pri kolónii včiel (BCO). Na začiatku náhodne navštevujú miesta a následne horšie včely preberajú vrcholy lepších včiel, a pokúšajú sa navštíviť tieto miesta v inom poradí. Rovnako ako pri kolónií včiel, konvergencia riešenia nie je zaručená.

6 Literatúra

[1] X. Wei, L. Han, and L. Hong, “A modified ant colony algorithm for traveling salesman problem,” *Int. J. Comput. Commun. Control*, vol. 9, no. 5, pp. 633–643, 2014.

[2] Z. Yun, S. Shao, and T. Mai, “Solution of TSP Problem of Measurement of Soil Attributes for Precision Agriculture,” *Ann. Adv. Agric. Sci.*, vol. 3, no. 2, pp. 14–20, 2019.

[3] P. Tinarut and K. Leksakul, “Hybrid self-organizing map approach for traveling salesman problem,” *Chiang Mai Univ. J. Nat. Sci.*, vol. 18, no. 1, pp. 27–37, 2019.

[4] A. M. Manasrah *et al.*, “MGA-TSP: Modernized Genetic Algorithm for the Traveling Salesman Problem,” *Int. J. Reason. Intell. Syst.*, vol. 11, no. 1, p. 1, 2019.

[5] S. Sabet, M. Shokouhifar, and F. Farokhi, “a Comparison Between Swarm Intelligence Algorithms for Routing Problems,” *Electr. Comput. Eng. An Int. J.*, vol. 5, no. 1, 2016.

[6] A. H. Halim and I. Ismail, “Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem,” *Arch. Comput. Methods Eng.*, vol. 26, no. 2, pp. 367–380, 2019.

[7] J. C. Bansal, H. Sharma, and S. S. Jadon, “Artificial bee colony algorithm: A survey,” *Int. J. Adv. Intell. Paradig.*, vol. 5, no. 1–2, pp. 123–159, 2013.

[8] W. Qi, H. Shen, and H. Chen, “A discrete artificial bee colony algorithm for RFID network scheduling,” *Int. J. Adv. Comput. Technol.*, vol. 4, no. 14, pp. 324–332, 2012.