Využitie algoritmu včelej kolónie pri riešení optimalizačných úloh

Application of bee colony algorithm in solving optimization problems

Marián Šedivý

Abstract:

This paper aims on solving optimization problems, specially solving the vehicle routing problem for Bratislava Self-Governing Region, by use of bee colony algorithm. Bees are highly organized social insects. The survival of the entire colony depends on every individual bee. Bees use systematic task segregation among them to ensure a continues existence of their colony. By simulating behavior of bees during foraging we can efficiently find optimal solutions to optimization problems.

Keywords:

BCO, VRP, Bee Colony Optimization, Vehicle Routing Problem, Swarm Intelligence

ACM classification: G1.6, I.2.8, J.4

Úvod

Rozvozný problém (Vehicle Routing Problem – VRP), alebo aj problém okružných jázd radíme medzi najviac študované problémy kombinatorickej optimalizácie. Riešením tohto problému je nájdenie optimálneho diagramu ciest, po ktorom pôjdu jednotlivé vozidlá tak, aby uspokojili požiadavky všetkých zákazníkov s minimálnymi nákladmi na prepravu. Taktiež nesmieme zabudnúť na dodržanie všetkých obmedzujúcich podmienok, ktoré súvisia s povahou zadania tohto problému. Je potrebné uvažovať o počte centrálnych skladov, kapacite vozidiel, dopyte jednotlivých zákazníkov, ale aj o tom, či riešením úlohy je minimalizácia počtu vozidiel alebo celkovej vzdialenosti prejdenej všetkými vozidlami. Od roku 1959, kedy tento problém ako prví popísali vo svojej práci Dantzig a Ramser (Dantzig, et al., 1959), bolo venovaných veľa štúdií exaktným a aproximatívnym riešeniam tohto problému, ako aj jeho variácií.

Algoritmus včelej kolónie je inšpirovaný správaním včiel pri obstarávaní potravy. Toto správanie bolo aplikované ako výpočtový algoritmus na riešenie komplexných problémov v rôznych oblastiach. V literatúre sa spomína aplikácia algoritmu pri dynamickej alokácii serverov (Nakrani, et al., 2004), optimalizácii numerických funkcií (Karaboga, et al., 2007), (Karaboga, et al., 2008), smerovaní paketov[1)] v telekomunikačných sieťach (Wedde, et al., 2004), stochastickom

1) Paket je ucelený blok dát, ktorý sa prenáša v sieti a spracováva ako celok. Je nedeliteľný na menšie časti.

VRP (Lučić, et al., 2003), (Teodorović, 2008) a rozvrhovacom probléme – Job Shop Scheduling Problem(Chong, et al., 2006), (Chong, et al., 2007).

Súčasný stav problematiky riešenia okružných úloh

VRP pokrýva distribúciu tovarov alebo služieb v danom časovom horizonte pre skupinu zákazníkov, pomocou súboru vozidiel, ktoré sú umiestnené v jednom alebo viacerých centrálnych skladoch, obsluhované posádkou a jazdiace po vhodnej cestnej sieti. Konkrétne riešenie VRP spočíva v nájdení množiny trás, z ktorej každá prislúcha jednému vozidlu tak, aby začalo a skončilo svoju cestu v prislúchajúcej stanici, boli uspokojené všetky požiadavky zákazníkov a splnené všetky obmedzujúce podmienky pri minimalizácii globálnych cestovných nákladov.

Zložitosť VRP problému vo veľkej miere vyplýva z faktu, že VRP problém sa nachádza na rozhraní dvoch dobre známych a študovaných problémov: problému batohu (Bin Packing Problem – BPP[2)]) a problému obchodného cestujúceho (Travelling Salesman Problem – TSP[3)]). Pre vyriešenie VRP potrebujeme vyriešiť obidva tieto problémy. Treba podotknúť, že tieto problémy patria do triedy NP – ťažkých problémov a nájdenie riešenie pre každý z nich osobitne vyžaduje veľkú námahu a je časovo náročné.

Sieť ciest využívaná na transport tovarov a služieb je vo všeobecnosti opísaná ako graf, kde jeho uzly predstavujú stanice, zákazníkov alebo križovatky a hrany (spojnice týchto uzlov) sú úseky trás. Tieto úseky, ako aj celé grafy môžu byť orientované alebo neorientované. Orientované grafy sú typické pre mestské trasy, kde je potrebné rešpektovať jednosmerné ulice alebo obchádzky. Každá hrana má svoje ohodnotenie vyjadrené napríklad ako dĺžku v km, resp. ako čas potrebný na jej prejdenie (ten závisí od použitého vozidla, ale aj od času, kedy daným úsekom prechádzame).

Schéma klasifikácie variantov problému okružných jázd podľa (Genderau, et al., 1998)

Tabuľka 1: *Varianty problému okružných jázd*

Väzba / Kapacita	M:M[4)]	1:M[5)]
Obmedzená kapacita	Odvoz na telefón	Systém doplňovania
Neobmedzená kapacita	Expresné doručenie pošty	Služby kuriéra a opráv

Problémy typu M:M s kapacitnými obmedzeniami patria medzi najviac komplikované z toho dôvodu, že miesta vyzdvihnutia a doručenia musia ležať na jednej trase a vyzdvihnutie musí predchádzať doručeniu.

Pri probléme odvozu na telefón (Dial-a-Ride) taktiež známeho aj ako Stacker Crane Problem uvažujeme o dvoch variáciách. Pri preemptívnom type problému je možné nechať prepravované objekty na medzizastávke a neskôr ich opäť naložiť a doručiť, čím sa zefektívni celý proces. Pri nepreemptívnom type túto možnosť nemáme. Odvoz na telefón je často využívaný pri prevoze starších ľudí, prípadne ľudí s obmedzenou pohyblivosťou či zdieľaných taxíkoch.

2) Bin Packing Problem je problém, pri ktorom treba uložiť čo najviac objektov do čo najmenšieho počtu kontajnerov s konštantnou veľkosťou.

3) Travelling Salesman Problem – riešením problému obchodného cestujúceho je nájsť takú okružnú cestu medzi mestami, ktorá bude najlacnejšia. Okružná cesta značí poradie, v akom je potrebné prejsť všetkými miestami práve raz a na záver sa vrátiť do začiatočného miesta.

4) Každá požiadavka zahŕňa obe lokality (miesto vyzdvihnutia aj doručenia)

5) Každá požiadavka zahŕňa iba jednu lokalitu (doručenie alebo vyzdvihnutie)

Systém doplňovania môžeme charakterizovať ako lokálny odvoz na telefón, ktorý „doplňuje“ iný väčší transportný systém v určitej lokalite. Napríklad môže ísť o zozbieranie zákazníkov menším autobusom, ktorý ich následne odvezie k vlaku alebo na letisko.

Pri expresnom doručení pošty je dôležitým faktorom nutnosť aby vyzdvihnutie a doručenie pošty v danej oblasti boli vykonané tým istým vozidlom.

Služby kuriéra a opráv môžeme prirovnať k problému okružných jázd s časovými oknami, kde vykonanie služby nemusí vždy nasledovať za požiadavkou, ale je centrálne naplánované tak, aby sa minimalizovali náklady na prepravu. Taktiež veľmi dôležitú úlohu zohráva čas potrebný na vykonanie služby.

Algoritmus včelej kolónie

Tento algoritmus, často označovaný aj ako optimalizácia prostredníctvom včelej kolónie (Bee Colony Optimization – BCO), alebo umelá včelia kolónia (Artificial Bee Colony – ABC), simuluje správanie včiel pri zbere potravy. Keďže lietanie je pre včely veľmi náročné a vyčerpávajúce, snažia sa optimalizovať zber potravy, aby získali čo najväčšie množstvo peľu preletením najkratšej vzdialenosti. Základnou ideou BCO je vytvorenie multi – agentového systému (kolónie umelých včiel), ktorý je schopný riešiť zložité problémy kombinatorickej optimalizácie.

Správanie včiel pri obstarávaní potravy bolo dlhé roky záhadou, až kým Karl von Frisch[6)] nedešifroval význam pohybov, ktoré robia včely pri svojom kývavom tanci (waggle dance). Tento tanec včely používajú na vzájomnú komunikáciu. Predpokladajme, že včela našla bohatý zdroj potravy. Po jej návrate do úľa začne tancovať v tvare osmičky. Prostredníctvom tohto informatívneho tanca včela dokáže informovať ostané zberačky potravy o smere a vzdialenosti k novoobjavenému zdroju a tým k nemu môže pritiahnuť viac včiel.

Celý algoritmus, založený na procese obstarávania a zberu potravy, môžeme opísať pomocou nasledovných krokov[7)]:

- Počas prvej iterácie, keď ešte včely nemôžu sledovať žiaden tanec, vyletia z úľa náhodne do miest s potravou. Vo svojej práci (Wong, et al., 2009) využívajú pre túto fázu dva heuristické prístupy a to heuristiku prechodu stavov[8)] a heuristiku najbližšieho suseda[9)],
- Následne sa včely vrátia do úľa, kde odovzdajú potravu a majú na výber z viacerých možností:
  - zavrhnúť daný zdroj potravy a stať sa nerozhodnutým nasledovníkom,
  - pokračovať v zbere potravy na danom zdroji bez toho, aby presvedčili ďalšie včely,
  - začať tancovať a tým presvedčiť ostané včely nasledovníčky, aby išli s nimi zbierať potravu na danom zdroji,

Včela si vyberá z uvedených možností na základe určitej pravdepodobnosti. Mechanizmus, kedy si včela vyberá nasledovanie určitej tancujúcej včely, nie je veľmi známy, ale existujú domnienky, že regrutovanie včiel je funkciou kvality daného zdroja (Camazine, et al., 1991).

---

6) Podrobný opis experimentov, ktoré vykonal Karl Von Frisch pre dešifrovanie správania včiel je dostupný na http://www.nobelprize.org/nobel_prizes/medicine/laureates/1973/frisch-lecture.pdf

7) Daný postup bol odvodený od formulácií použitých v prácach autorov (Wong, et al., 2009) a (Karaboga, et al., 2008).

8) Včela začína v náhodnom mieste a na základe pravidla prechodu stavov si vyberie ďalšie miesto kam poletí.

9) Včela začína v ľubovoľnom mieste a vyberá si najbližšie susedné miesto kam poletí. Ak tejto podmienke vyhovuje viac miest, aplikuje sa náhodný výber.

Konštrukcia cesty umelými včelami

Počas fázy zberu potravy musí včela prejsť z jedného miesta do druhého miesta pokiaľ nevykoná celý okruh (vráti sa na začiatok svojej cesty) na základe tranzitívneho pravidla. Toto pravidlo definuje pravdepodobnosť prechodu z miesta i do miesta j po n prechodoch, p_{ij},n ako je zapísané vo vzťahu (1.1). Keď je konštrukcia cesty hotová, získame permutáciu miest, ktorá je jedným z riešení nášho VRP. Označme d_{ij} vzdialenosť dvoch miest a ρ_{ij} , n fitness hodnotu hrany spájajúcu tieto dve mestá. Nech $A_{i,n}$ je množina zatiaľ nenavštívených miest, ktoré sú dosiahnuteľné z miesta i na prechode n. Parametre α a β určujú relatívnu významnosť fitness hodnoty hrany verzus heuristickú vzdialenosť.

$$P_{ij,n} = \frac{\left[\rho_{ij,n}\right]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta}}{\sum_{j \in A_{i,n}} \left(\left[\rho_{ij,n}\right]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta} \right)} \qquad (1.1)$$

Vzdialenosť d_{ij} nadobúda hodnoty nepriamo úmerné k $P_{ij,n}$. Čím je vzdialenosť medzi i a j kratšia, tým väčšia je pravdepodobnosť, že j bude vybrané ako nasledujúce miesto návštevy. Fitness hodnota hrany $P_{ij,n}$ meria pravdepodobnosť, že v preferovanej ceste bude existovať spojenie medzi i a j. V prírode ak včela nájde nový zdroj potravy, začne tancovať. Ak týmto tancom osloví inú včelu, tá bude zbierať potravu na novom zdroji. Pri BCO je táto situácia modelovaná implementáciou preferovanej cesty, ktorú označíme θ. θ je permutáciou prechodov jednotlivými miestami, ktoré včela získala od inej včely (navádzanie pri obstarávaní potravy). Ďalej uvedieme vzťah pre $P_{ij,n}$:

$$\rho_{ij,n} = \begin{cases} \lambda & , \; j \in F_{i,n} \,, \left|A_{i,n}\right| > 1 \\ \dfrac{1-\lambda\left|A_{i,n} \cap F_{i,n}\right|}{\left|A_{i,n} \cap F_{i,n}\right|} & , \; j \notin F_{i,n} \,, \left|A_{i,n}\right| > 1 \\ 1 & , \; \left|A_{i,n}\right| = 1 \end{cases} \qquad (1.2)$$

$$\forall j \in A_{i,n} \,, \qquad 0 \le \lambda \le 1$$

kde λ reprezentuje pravdepodobnosť navštívenia miesta vo θ. $F_{i,n}$ je množina, ktorá obsahuje jedno preferované miesto odporučené v θ, do ktorého sa včela presunie z miesta i na prechode n. Ak včela v nultom kroku začala prehľadávanie z úľa H, potom položíme $F_{H,0} = \{\theta(1)\}$. Ak práve navštívené miesto i je na m-tej pozícii v preferovanej ceste po n prechodoch, potom $F_{\theta(m),n} = \{\theta(m+1)\}$. $F_{i,n}$ obsahuje iba jedno miesto, pretože uvažujeme iba o $\theta(m+1)$ (nasledujúce priľahlé miesto k miestu i z θ). Prvé dve podmienky z (1.2) zabezpečujú, že hrany odporúčané v θ sú priradené s pravdepodobnosťou λ, zatiaľ čo zvyšné hrany sú priradené s rovnakou pravdepodobnosťou. Ak uvažujeme o l hranách, jedna hrana (ktorá existuje v θ) sa bude rovnať λ a ostatné $\frac{(1-\lambda)}{l-1}$. Ak žiadna hrana nie je podobná v porovnaní s θ, všetky uvažované hrany z $A_{i,n}$ sa budú rovnať $\frac{\cdot}{l}$. Tretia podmienka priradí hodnotu 1 pre $\rho_{ij,n}$ ak už ostáva iba jedno miesto z $A_{i,n}$. Ide teda o posledný prechod predtým, než sa včela vráti do začiatočného miesta a tým ukončí svoju trasu.

Opis práce algoritmu

Včely v úle sú rozdelené do troch skupín a to aktívne včely, neaktívne včely a prieskumníci. Aktívne včely sú všetky, ktoré práve využívajú zdroj potravy. Neaktívne včely sú v úli a čakajú na

informácie, od aktívnych včiel. Na základe kvality nájdeného zdroja jedla, aktívne včely tancujú pre neaktívne, ktoré si potom vďaka týmto informáciám vyberajú zdroj jedla kam poletia. Lepšie zdroje preto pritiahnu viac včiel. Prieskumníci sú včely, ktoré hľadajú nové zdroje potravy v okolí úľu.

- **Aktívna včela (Foraging)**: generuje susedné riešenia k svojmu aktuálnemu riešeniu, ktoré má práve v pamäti,
- **Neaktívna včela (Observing)**: čaká v úli a na základe tancovania aktívnych včiel a prieskumníkov, môže s určitou pravdepodobnosťou prijať niektoré z ich riešení. Aktívna včela, ktorá pre zadaný počet iterácií nedokázala zlepšiť svoje riešenie sa stáva neaktívnou a zároveň sa náhodná neaktívna včela aktivuje.
- **Prieskumník (Scouting)**: generuje náhodné riešenia, a tým je zabezpečené vyviaznutie z lokálneho extrému.

Každá včela má vo svojej pamäti riešenie, ktoré reprezentuje jednotlivé trasy pre všetky vozidlá. Ide teda vždy o kompletné riešenie. Aktívne včely používajú na nájdenie susedného riešenia metódu GetNextSolution(), ktorá sa skladá z dvoch krokov SwapCitiesInOneRoute() a SwapCitiesAmongRoutes(). Prieskumníci generujú náhodné riešenia úplne nezávislé na ich aktuálnom. Ak nájde aktívna včela alebo prieskumník lepšie riešenie ako mala doteraz, aktualizuje svoje riešenie, zavolá metódu DoWaggleDance() a zároveň ho porovná s globálnym najlepším riešením. Ak je nájdené riešenie lepšie ako globálne, tak ho aktualizuje. Prácu jednotlivých včiel a pomocné metódy môžeme vyjadriť nasledovne:

Aktívna včela

Aktívna včela sa snaží prehľadať okolie najbližšie k svojmu aktuálnemu riešeniu, čo je zabezpečené metódou GetNextSolution(). Taktiež je tu zavedená pravdepodobnosť chyby ProbabilityMistake, keď včela môže odmietnuť lepšie riešenie, alebo aj prijať horšie. Ide o veľmi malé číslo, napríklad 0,01, čiže včela sa pomýli s pravdepodobnosťou 1 percento. Ak aktívna včela našla lepšie riešenie ako mala doteraz, zavolá metódu DoWaggleDance(), pomocou ktorej sa snaží presvedčiť neaktívne včely, aby ju nasledovali. Pokiaľ včela nedokáže zlepšiť svoje riešenie počas zadaného počtu návštev jej zdroja MaxVisits, stane sa z nej neaktívna včela a zároveň sa náhodná neaktívna včela aktivuje. Niektorí autori uvádzajú, že maximálny počet návštev by mal byť rovný približne päť násobku počtu miest.

Prieskumník

Na rozdiel od aktívnej včely, prieskumník nerobí chyby. To znamená, že prijme riešenie len vtedy, ak je naozaj lepšie ako to, ktoré mal doteraz v pamäti. Následne zavolá metódu DoWaggleDance(). Pomocou včiel prieskumníkov je zabezpečené generovanie náhodných riešení a tým je algoritmu umožnené vyviaznutie z lokálneho extrému.

Neaktívna včela

Neaktívna včela v našom prípade nerobí nič, ako už názov napovedá. Bolo by možné implementovať napríklad metódu GetNextSolution() aj pre neaktívnu včelu a získať tak susedné riešenie k jej aktuálnemu riešeniu.

Tancovanie

Metóda DoWaggleDance() umožňuje s určitou pravdepodobnosťou ProbabilityPersuation prijať neaktívnej včele riešenie od práve tancujúcej včely. Napríklad ak by bola použitá pravdepodobnosť 0,9, znamenalo by to, že s pravdepodobnosťou 90 percent bude neaktívna včela nasledovať práve tancujúcu včelu. Túto metódu využívajú na propagovanie dobrých riešení aktívne včely a prieskumníci a je ňou zabezpečené odovzdávanie informácií medzi včelami v úle.

Generovanie susedných riešení

Metóda GetNextSolution() využíva ďalšie dve metódy pre vygenerovanie susedného riešenia. Zároveň je táto metóda vložená v cykle, ktorý kontroluje či vygenerované riešenie spĺňa obmedzujúce podmienky. Metóda je volaná dovtedy, kým vygenerované riešenie porušuje obmedzujúce podmienky. Tým je zabezpečené, že včely pracujú len s vyhovujúcimi riešeniami.

V prvej metóde SwapCitiesInOneRoute() sa v jednej trase premiestnia navzájom dve susedné miesta[10]. Napríklad pre postupnosť miest A,B,C,D,E,A je susedným riešením A,B,C,E,D,A. Ide vlastne o permutáciu pôvodného riešenia, avšak vzniká tu priestor pre polemizovanie, či susedné riešenie vzniká len výmenou dvoch priľahlých miest, alebo môžu byť vymenené dve ľubovoľné miesta napr. A,D,C,B,E,A.

V druhej metóde SwapCitiesAmongRoutes() vyberieme náhodne dve trasy z riešenia, ktoré má včela v pamäti. Z prvej trasy potom náhodne vyberieme jedno miesto a to vložíme na náhodnú pozíciu v druhej trase.

Generovanie náhodných riešení

Metóda CreateRandomSolution()generuje náhodné riešenie pre prieskumníkov. Jej algoritmus je nasledovný. Pre každé miesto vyber náhodnú trasu, ak sa dá miesto vložiť bez toho, aby porušilo obmedzujúcu podmienku (napríklad presiahlo kapacitu vozidla), tak vlož miesto do vybranej trasy. Ak sa miesto nedá vložiť, zvýš počítadlo neúspechu[11]. Pokiaľ sa počet neúspešných vložení rovná počtu vozidiel, tak poruš obmedzujúcu podmienku a vlož miesto na trasu. Metóda CreateRandomSolution() je volaná v cykle, ktorý kontroluje správnosť vygenerovaného riešenia dovtedy, kým porušuje obmedzujúcu podmienku.

Dosiahnuté výsledky

Opísaný algoritmus bol aplikovaný na riešenie problému okružných úloh pre Bratislavský samosprávny kraj. Problém bol špecifikovaný nasledovne:

- počet vozidiel – 5,
- kapacita vozidla – 40 000 jednotiek,
- dopyt jednotlivých miest – priamoúmerne odvodený od počtu obyvateľov,
- tesnosť – 88,84 %.
- umiestnenie centrálneho skladu v Čunove

V takomto prípade sa celková vzdialenosť prejdená všetkými vozidlami rovnala 652 km. Usporiadanie jednotlivých obcí medzi vozidlá zachytáva nasledujúca tabuľka Tabuľka 1 a graficky znázorňuje obrázok Obrázok 1.

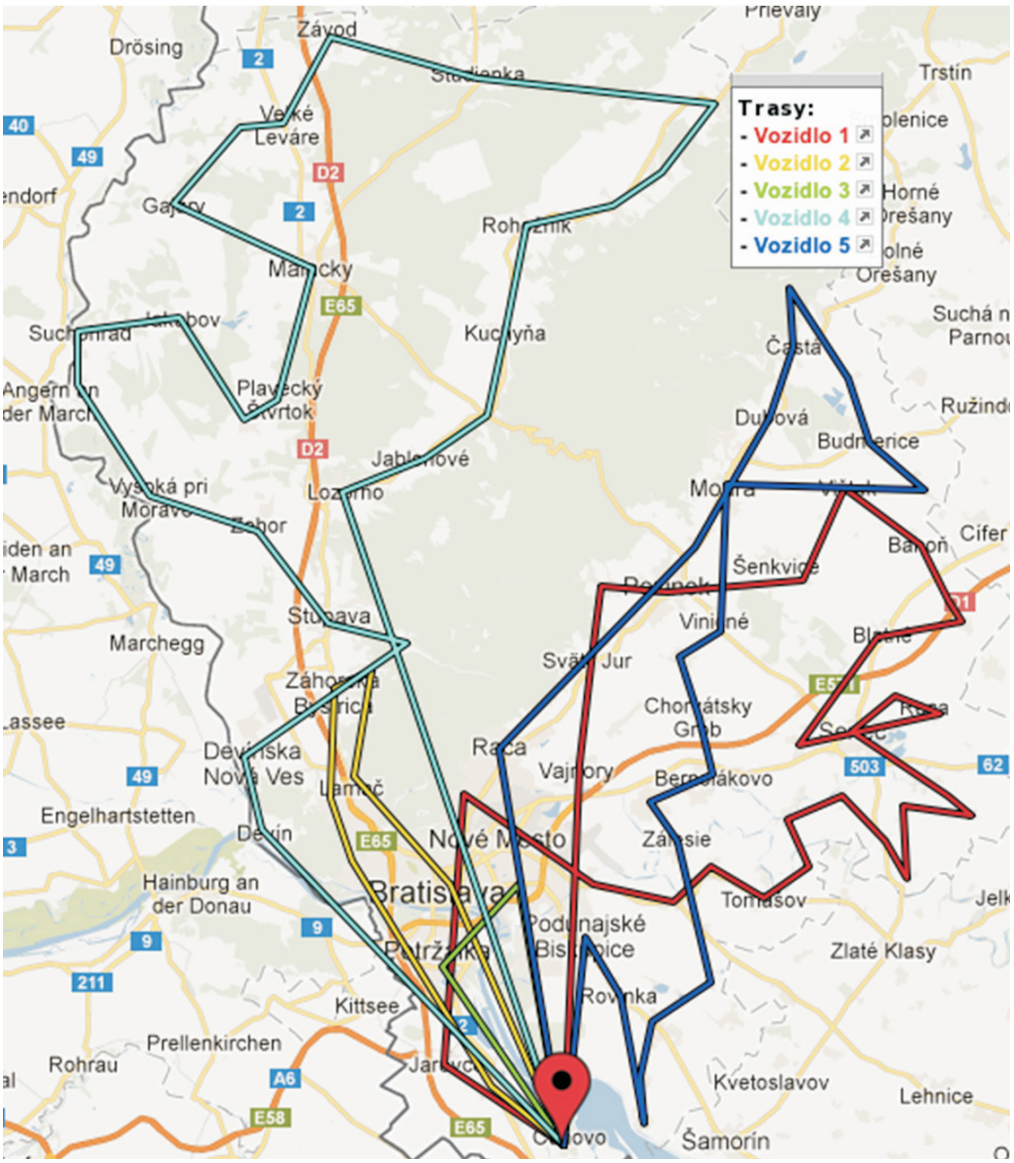
Tabuľka 1 *Rozdelenie obcí medzi vozidlá*

1. vozidlo	Bratislava – Čunovo, Bratislava – Jarovce, Bratislava – Nové Mesto, Bratislava – Vrakuňa, Most pri Bratislave, Malinovo, Tomášov, Vlky, Nová Dedinka, Tureň, Hrubý Šúr, Hurbanova Ves, Kostolná pri Dunaji, Hrubá Borša, Kráľová pri Senci, Senec, Boldog, Reca, Veľký Biel, Blatné, Čataj, Igram, Kaplna, Báhoň, Vištuk, Šenkvice, Pezinok, Limbach, Svätý Jur, Bratislava – Vajnory, Bratislava – Čunovo

10) Definícia existencie susedných miest je kľúčovou podmienkou pre riešenie úlohy VRP. Ak by totiž neexistoval predpoklad susednosti jednotlivých miest, včelí algoritmus by nebol schopný úlohu riešiť.

11) Počítadlo neúspechu zaznamenáva počet pokusov, kedy sa nepodarilo miesto vložiť na vybranú náhodnú trasu, pretože porušilo kapacitné obmedzenie vozidla.

2. vozidlo	Bratislava – Čunovo, Bratislava – Rusovce, Bratislava – Karlova Ves, Bratislava – Dúbravka, Bratislava – Záhorská Bystrica, Marianka, Bratislava – Lamač, Bratislava – Staré Mesto, Bratislava – Čunovo
3. vozidlo	Bratislava – Čunovo, Bratislava – Petržalka, Bratislava – Ružinov, Bratislava – Čunovo
4. vozidlo	Bratislava – Čunovo, Lozorno, Jablonové, Pernek, Kuchyňa, Rohožník, Sološnica, Plavecké Podhradie, Plavecký Mikuláš, Studienka, Závod, Veľké Leváre, Malé Leváre, Gajary, Kostolište, Malacky, Plavecký Štvrtok, Láb, Jakubov, Suchohrad, Záhorská Ves, Vysoká pri Morave, Zohor, Stupava, Borinka, Bratislava – Devínska Nová Ves, Bratislava – Devín, Bratislava – Čunovo
5. vozidlo	Bratislava – Čunovo, Bratislava – Podunajské Biskupice, Rovinka, Hamuliakovo, Kalinkovo, Dunajská Lužná, Miloslavov, Zálesie, Ivanka pri Dunaji, Bernolákovo, Chorvátsky Grob, Slovenský Grob, Viničné, Modra, Jablonec, Budmerice, Štefanová, Doľany, Píla, Častá, Dubová, Vinosady, Bratislava – Rača, Bratislava – Čunovo



Obrázok 1 *Trasy jednotlivých vozidiel*

Záver

Keďže riešenie problému okružných jázd by aplikovaním exaktných metód trvalo neúmerne dlho, aplikácia včelieho algoritmu sa javí ako veľmi vhodná. Algoritmus včelej kolónie je schopný efektívne riešiť optimalizačné úlohy a v prípustnom čase ponúknuť riešenia, ktoré nie sú veľmi vzdialené od optima.

Použitá literatúra:

- [1] CAMAZINE, S. and SNEYD, J. 1991. *A model of collective nectar source selection by honey bees: Self-organization through simple rules* . U.S.A. : Elsevier, 1991.
- [2] DANTZIG, G. B. and RAMSER, J. H. 1959. *The truck dispatching problem.* Filadelfia : Management Science, 1959.
- [3] GENDERAU, M. and POTVIN, J-Y. 1998. Dynamic Vehicle Routing and Dispatching. [book auth.] T. G. Crainic and G. Laporte. *Fleet management and logistics.* Boston : Kluwer, 1998.
- [4] CHONG, C. S., et al. 2006. *A bee colony optimization algorithm to job shop scheduling.* Arizona : Winter Simulation Conference, 2006 – 2007.*Using a bee colony algorithm for neighborhood search in job shop scheduling problems.* Praha : 21. European Conference on Modeling and Simulation, 2007.
- [5] KARABOGA, D. and BASTURK, B. 2008. *On the performance of artificial bee colony (abc) algorithm.* Turecko : Springer, 2008.
- [6] KARABOGA, D. and BSTURK, B. 2007. *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm.* Turecko : Springer, 2007.
- [7] LUČIĆ, P. and TEODOROVIĆ, D. 2003.*Vehicle routing problem with uncertain demand at nodes: The bee system and fuzzy logic approach.* Berlín : Springer-Verlag, 2003.
- [8] NAKRANI, S. and TOVEY, C. 2004. *On honey bees and dynamic server allocation in internet hosting centers.* U.S.A : Sage Publications, 2004.
- [9] TEODOROVIĆ, D. 2008. *Swarm intelligence systems for transportation engineering: Principles and applications.* Srbsko : Elsevier, 2008.
- [10] WEDDE, F. H., FAROOQ, M. and ZHANG, Y. 2004. *Beehive: An efficient faulttolerant routing algorithm inspired by honey bee behavior.* Berlín : Springer, 2004.
- [11] WONG, L. P. and CHONG, Ch. S. 2009. *An Efficient Bee Colony Optimization Algorithm for Traveling Salesman Problem using Frequency-based Pruning.* Wales : IEEE International Conference, 2009. ISBN: 978-1-4244-3759-7.

Ing. Marián Šedivý, PhD.
marian.sedivy@gmail.com
0907 305 014