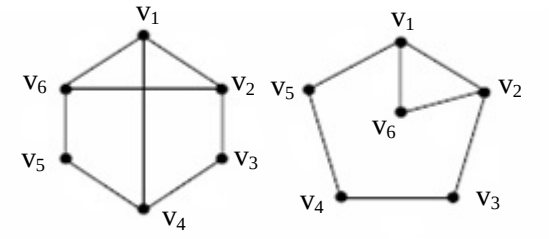
МАТЕМАТИЧКИ ОМНИБУС, 1 (2017), 101 – 113

НЕКОИ АЛГОРИТМИ ЗА РЕШАВАЊЕ НА ЗАДАЧАТА НА ПАТУВАЧКИОТ ТРГОВЕЦ

Ирена Стојковска [1]

Задачата на патувачкиот трговец е комбинаторна оптимизациона задача со едноставна формулација, широка примена, но тешка за решавање. Откако ќе ја дефинираме задачата, ќе дадеме краток историски осврт, неколку примени, доволни услови за егзистенција на нејзиното решение и малку повеќе ќе се задржиме на методите на нејзино решавање.

Дефиниција 1. Нека $G = (V, E)$ е граф, каде што V е множество од n темиња, E е множество од рабови и нека c_{ij} е тежина (трошок, должина) на работ (i, j). *Задачата на патувачкиот трговец (ЗПТ)* се состои во одредување на минимален циклус (тура) кој минува само еднаш низ секое теме на дадениот граф, т.е. одредување на минимален *Хамилтонов циклус*. Графот кој содржи Хамилтонов циклус се нарекува *Хамилтонов граф* (види Слика 1).

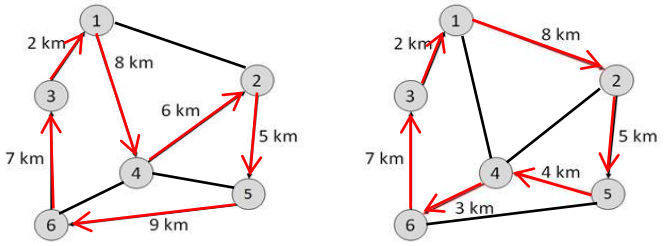


Слика 1. Примери за Хамилтонови графови, графови кои содржат Хамилтонови циклуси.

Дефиницијата 1 ја дополнуваме терминолошки со: ако $c_{ij} = c_{ji}$ за секои $i, j \in V$, $i \neq j$, тогаш ЗПТ е *симетрична*, а ако постои барем еден раб (i, j) за кој $c_{ij} \neq c_{ji}$, тогаш ЗПТ е *асиметрична*.

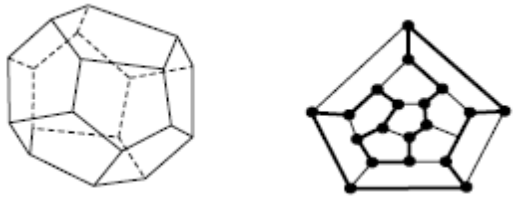
Најчестата практична интерпретација на ЗПТ е следната: *Патувачки трговец треба да најде оптимална тура низ n града т.е. да го помине секој град точно еднаш, да се врати од каде што го започнал патувањето и притоа трошокот за патување да биде минимален.*

На Слика 2 е даден пример за две (Хамилтонови) тури низ шест града, од кои втората тура е минималната тура со вкупна должина на патот од 29 km, во споредба со првата тура со вкупна должина од 37 km.



Слика 2. Две (Хамилтонови) тури низ шест града, од кои првата е со вкупна должина од 37 km, а втората 29 km.

Корените на ЗПТ се смета дека датираат од 1857 година, кога Сер Вилјам Хамилтон (Sir William Hamilton, 1805–1865), ирски математичар, физичар и астроном, ја предложил сложувалката наречена „Икозиска игра“ (Icosian game). Сложувалката се состоела од додекаедар на кој секое од 20-те темиња било именувано според некој главен светски град (Слика 3а)). Целта на играта била да се конструира пат по рабовите на додекаедарот кој поминува низ секој град точно еднаш и се враќа во градот од каде почнал. Со други зборови, требало да се конструира Хамилтонов циклус во граф кој соодветствува на додекаедар (Слика 3б)). Хамилтоновите графови го добиле своето име токму според Сер Вилијам Хамилтон.



а) б)

Слика 3. а) Додекаедар со 20 темиња,
б) Граф кој соодветствува на додекаедорот со впишан Хамилтонов циклус.

Примената на ЗПТ е разновидна и широка. Ќе наведеме неколку примени: оптимално дупчење дупки на електронските плочи, оптимален распоред на процеси на една машина, примена во рендген-

ската кристалографија, дистрибуција на производи, планирање на воени мисии, пренесување на податоци преку интернет, позиционирање на сателити за глобална комуникација, па дури и дизајнирање на табла за пикадо. Повеќе за примената на ЗПТ може да се прочита во [8].

Решението на ЗПТ секогаш постои во *комплетен граф* (граф во кој секои две темиња се поврзани со раб). Имено, секој комплетен граф е Хамилтонов граф. За постоењето на Хамилтонова тура во еден граф постојат повеќе тврдења, од кои можеби најпознати се следните ([2]):

Теорема 1. (Dirac 1952) *Секој граф* $G = (V, E)$ *со* $n \geq 3$ *темиња и степен на секое теме* $d(v) \geq n/2$ *за секој* $v \in V$ *има Хамилтонов циклус.*

Теорема 2. (Tutte 1956) *Секој 4-сврзан планарен граф има Хамилтонов циклус.*

Притоа, *степен на темето* $v \in V$ е бројот на рабови $(i, j) \in E$ со една крајна точка темето v. Степенот на темето $v \in V$ го означуваме со $d(v)$. Еден граф е *k-сврзан*, ако кои било две темиња на графот се поврзани со k различни патишта. Графот е *планарен*, ако може да се нацрта во рамнина и притоа никои два раба да не се сечат во точка која не е крајна точка.

Постоењето на Хамилтонова тура е услов за постоење на решение на ЗПТ. Но, добро познат факт е дека наоѓањето на Хамилтонова тура е NP-комплетна задача (истовремено NP и NP-тешка задача) ([3]), што повлекува дека ЗПТ е NP-тешка задача ([5]). Повеќе за ЗПТ како NP-тешка задача може да се прочита во [8]. Но, иако графот е комплетен, тогаш постојат $\frac{(n-1)!}{2}$ можни тури (во случај на симетрична ЗПТ), што претставува навистина голем број тури за испитување за да се најде минимална тура меѓу сите можни тури. Затоа, решавањето на ЗПТ бара користење на ефикасни алгоритми, точни или приближни ([4, 5, 6, 7]).

1. ТОЧНИ АЛГОРИТМИ

Најдобриот начин за објаснување и разбирање на точните алгоритми за решавање на ЗПТ е преку целобројното линеарно програмирање (ЦЛП) и алгоритмите за решавање на ЦЛП задачи.

1.1. ФОРМУЛАЦИЈА НА ЗПТ КАКО ЦЛП ЗАДАЧА

Во овој дел ќе изложиме неколку формулации на ЗПТ како ЦЛП задача ([5]). Една од првите формулации на ЗПТ како ЦЛП задача е предложена од Dantzig, Fulkerson и Johnson (DFJ) во 1954 година:

$$\min \sum_{i \neq j} c_{ij} x_{ij} \qquad (1)$$

при ограничувања

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \qquad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \qquad (3)$$

$$\sum_{i,\, j \in S} x_{ij} \leq |S| - 1, \; S \subset V, \; 2 \leq |S| \leq n - 2 \qquad (4)$$

$$x_{ij} \in \{0,1\}, \; i, j = 1, \ldots, n, \; i \neq j \qquad (5)$$

каде што x_{ij}, $i \neq j$ е бинарна променлива која прима вредност 1 ако и само ако работ (i, j) е дел од оптималното решение. Во оваа формулација, функцијата на целта (1) е вкупниот трошок на турата, ограничувањата (2) и (3) се *ограничувања на степенот* кои покажуваат дека во секое теме се влегува точно еднаш и се излегува точно еднаш. Ограничувањата (4) се *ограничувања за елиминација на поттури* со кои се попречува формирање на поттури, односно тури со помалку од n темиња.

Еквивалентен облик на ограничувањата (4) се *ограничувањата на сврзаност* (4'):

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \geq 1, \; S \subset V, \; 2 \leq |S| \leq n - 2 \qquad (4')$$

каде што $\overline{S} = V \setminus S$, кои покажуваат дека во секое решение на ЗПТ, мора да има најмалку еден раб од S во неговиот комплемент, односно дека S не може да не биде сврзано.

Да забележиме дека DFJ формулацијата (1) – (5) содржи $n(n-1)$ бинарни променливи, $2n$ ограничувања на степенот и

$2^n - 2n - 2$ ограничувања за елиминација на поттури, што и за не толку големи вредности на n го прави скоро невозможно решавањето на DFJ како ЦЛП задача.

Затоа, во 1960 година, Miller, Tucker и Zemlin (MTZ) предлагаат воведување на дополнителни променливи u_i, $i = 2,...,n$ со кои се намалува бројот на ограничувања за елиминација на подтури, кои преминуваат во (6)–(7):

$$u_i - u_j + (n-1)x_{ij} \le n-2,\ i, j = 2,...,n,\ i \ne j \qquad (6)$$

$$1 \le u_i \le n-1,\ i = 2,...,n \qquad (7)$$

Ограничувањата (6) обезбедуваат решението да не содржи поттура на множеството $S \subseteq V \setminus \{1\}$, а со тоа и да не содржи поттура со помалку од n темиња. Ограничувањата (7) обезбедуваат променливите u_i, $i = 2,...,n$ да се едозначно одредени за секоја допустлива тура.

И покрај тоа што MTZ формулацијата е покомпактна, таа е послаба од DFJ формулацијата во поглед на оптималното решение на соодветната линеарна релаксација (задачата на линеарно програмирање добиена од соодветната ЦЛП задача од која се отстранети условите за целобројност на променливите). Имено, ако со $z^*(DFJ)$, односно $z^*(MTZ)$ ја означиме оптималната вредност на линеарната релаксација на DFJ, односно MTZ формулацијата, тогаш е точен следниот резултат покажан од Wong во 1980 година т.е $z^*(MTZ) \le z^*(DFJ)$. Овој резултат ја прави линеарната релаксација на DFJ формулацијата посилна, затоа што линеарна релаксација на ЦЛП задача на минимизација секогаш има оптимална вредност помала или еднаква на оптималната вредност на ЦЛП задачата, па линеарната релаксација на DFJ формулацијата дава подобра оценка за границата на оптималната вредност на ЦЛП задачата. Досега се предложени повеќе алтернативни формулации на ЗПТ како ЦЛП задача, но за ниедна од предложените формулации не е покажано дека има посилна линеарна релаксација од линеарната релаксација на DFJ формулацијата. Инаку, линеарните релаксации на ЦЛП задачите имаат важна улога во алгоритмите за решавање на ЦЛП задачите.

1.2. BRANCH-AND-BOUND АЛГОРИТМИ

Најчесто користените алгоритми за решавање на ЗПТ како ЦЛП задача се Branch-and-bound (BB) алгоритмите. Генерално, овие алгоритми ја решаваат задачата така што ја делат допустливата област (областа над која е дефинирана задачата на оптимизација) на помали подмножества, потоа ги наоѓаат границите на функцијата на целта над тие подмножества на тој начин што решаваат полесна (релаксирана) задача, а потоа со помош на најдените граници отфрлаат одредени подмножества и не ги разгледуваат понатаму. Постапката запира кога секое подмножество или дало допустливо решение или дало решение кое е подобро од дотогаш најденото најдобро решение. Најдоброто најдено решение со оваа постапка е глобален оптимум ([1]).

Следниот алгоритам претставува една општа варијанта на BB алгоритмите, кој формира листа од активни подзадачи и притоа пред да вметне нова подзадача во листата, прво ја решава нејзината релаксација ([1]).

АЛГОРИТАМ 1. (Branch-and-bound алгоритам за ЗПТ)

Чекор 1. Реши ја релаксацијата на дадената ЗПТ (за да ја добиеш долната граница), стави ја ЗПТ на листата со подзадачи, зададi ја почетната вредност на горната граница $U = +\infty$ и оди на Чекор 2.

Чекор 2. Ако листата со подзадачи е празна, тогаш СТОП и турата придружена на горната граница U е оптимална (или ако $U = +\infty$, тогаш ЗПТ нема решение). Инаку, одбери подзадача $ЗПТ_i$ од листата, според правило за избор на подзадачи, отстрани ја $ЗПТ_i$ од листата и оди на Чекор 3.

Чекор 3. (незадолжителен) Со некоја евристика најди тура во ЗПТ. Ако е најдена подобра тура од до тогаш најдобрата, зачувај ја како најдобра и обнови ја горната граница U. Оди на Чекор 4.

Чекор 4. (незадолжителен) Отстрани ги од графот на $ЗПТ_i$ сите рабови чие вклучување во турата би ја покачиле вредноста на горната граница над U. Оди на Чекор 5.

Чекор 5. Според правилото за гранање (branching) дефинирај множество од подзадачи $ЗПТ_{i1}$, $ЗПТ_{i2}$, ... $ЗПТ_{iq}$ генерирани од подзадачата $ЗПТ_i$ и оди на Чекор 6.

Чекор 6. Ако сите подзадачи генерирани од $ЗПТ_i$ се веќе претходно разгледани, тогаш оди на Чекор 2. Инаку, премини на

следната подзадача $ЗПТ_{ij}$ и реши ја нејзината релаксација за да ја добиеш долната граница L_{ij} (bounding). Потоа,

- ако $L_{ij} \geq U$, врати се на Чекор 6,
- ако $L_{ij} < U$ и добиеното решение дефинира тура за ЗПТ, тогаш зачувај го решението како најдобро досега, стави $U = L_{ij}$ и оди на Чекор 6,
- ако $L_{ij} < U$ и добиеното решение не дефинира тура, стави ја подзадачата $ЗПТ_{ij}$ на листата и врати се на Чекор 6.

Друга варијанта на Алгоритмот 1, прво ги додава на листата подзадачите, а потоа ги решава ([1]). И во двата случаја процедурите може да се претстават во вид на дрво чии јазли се подзадачите чии релаксации се решаваат, а јазлите потомци на некој јазол се подзадачите кои се генерираат со Чекор 5 од Алгоритмот 1.

Квалитетот на BB алгоритмите е во директна врска со квалитетот на границата добиена со решавање на релаксацијата ([5]). Кај ЗПТ најчесто користена релаксација е онаа добиена од DFJ формулацијата со отстранување на ограничувањата за елиминација на поддури (4). Така добиената задача (1) – (3), (5) е *задача на доделување на задолженија* која се решава во полиномно $O(n^3)$ време. Исто така, постојат и повеќе правила за избор на подзадачи за решавање (Чекор 2) и повеќе правила за гранање (Чекор 5), предложени од разни автори ([1, 5]).

2. ПРИБЛИЖНИ АЛГОРИТМИ

Бидејќи ЗПТ е NP-тешка задача, природно се наметнува потребата да се решава со помош на евристични методи и приближни алгоритми. Приближните алгоритми не гарантираат оптимално решение, но затоа даваат решение близу до оптималното за разумно многу време.

2.1. ЕВРИСТИЧНИ ПРИСТАПИ

Во овој дел ќе изложиме неколку евристични пристапи кои обезбедуваат емпириски добри решенија на ЗПТ. Евристичните пристапи главно се делат на две групи според нивната намена и тоа

на: процедури за конструкција на тури и процедури за подобрување на тури ([5, 6]).

а) *Процедури за конструкција на тури*

Сите процедури за конструкција на тури стопираат кога турата е најдена без да се обидат да ја подобрат. Ќе разгледаме три процедури за конструкција на тури.

Можеби наједноставната процедура која конструира тура е *евристиката на најблизок сосед*. Суштината на овој пристап е секогаш да се посети најблискиот град. Комплексноста е полиномна од редот $O(n^2)$, каде што n е бројот на градови. Една модификација на оваа евристика наоѓа тури за секое од n-те темиња како почетно теме, а потоа ја одбира најдобрата. Комплексноста на модификацијата е $O(n^3)$, но најдената тура во општ случај е подобра.

Алчната евристика постепено конструира тура со повторувачко селектирање на најкраткиот раб и негово додавање на турата се додека не се креира циклус со помалку од N раба без да се зголеми степенот на некое теме да биде поголем од 2. Комплексноста на алчната евристика е $O(n^2 \log_2 n)$.

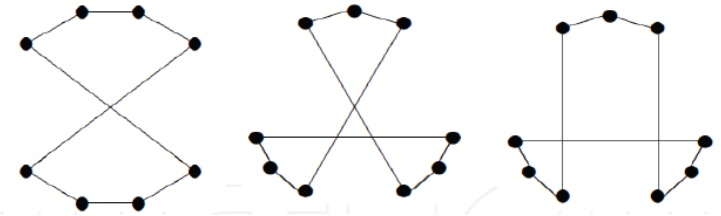
Општиот пристап на *евристиката на вметнување* е да почне со тура на некое подмножество од сите градови, најчеста почетна тура е триаголник (или почетната тура е само еден раб кој поврзува два града) и потоа ги додава останатите градови според некој критериум. Вметнувањето на теме може да биде според најразлични критериуми, на пример, се вметнува теме кое доведува до најмало нарастнување на должината на турата, или, се вметнува теме кое е најблиско до моменталната тура, или, теме кое формира најголем агол со две последователни темиња од моменталната тура и слично. Во зависност од критериумот на вметнување, комплексноста на оваа процедура варира меѓу $O(n^2)$ и $O(n^2 \log n)$.

б) *Процедури за подобрување на тури*

Откако е конструирана некоја тура со помош на некоја евристична процедура, најчесто се пристапува кон подобрување на турата. Едни од најчесто користените процедури за подобрување на тури се *k-opt алгоритмите*. Овие алгоритми, во секој чекор, отстрануваат k раба од веќе конструираната тура и ги преспојуваат k-те

преостанати ланци на сите можни начини. Ако некое од преспојувањата дава пократка тура, тогаш таа се зема за почетна тура во следниот чекор. Оваа постапка се повторува се додека има напредок. Добиената тура со овој алгоритам е *k-оптимална*. Најчесто користена вредност за k е 2 или 3, и тогаш станува збор за *2-opt* и *3-opt алгоритми*.

На Слика 4 е прикажано отстранувањето на 2 односно 3 раба од една тура и нивно повторно преспојување. Имено на Слика 4 а) е прикажан единствениот начин на преспојување по отстранување на 2 раба, додека на Слика 4 б) и в) прикажани се двата можни начини на преспојување по отстранувањето на 3 раба, при што повторно се добива тура низ сите темиња.



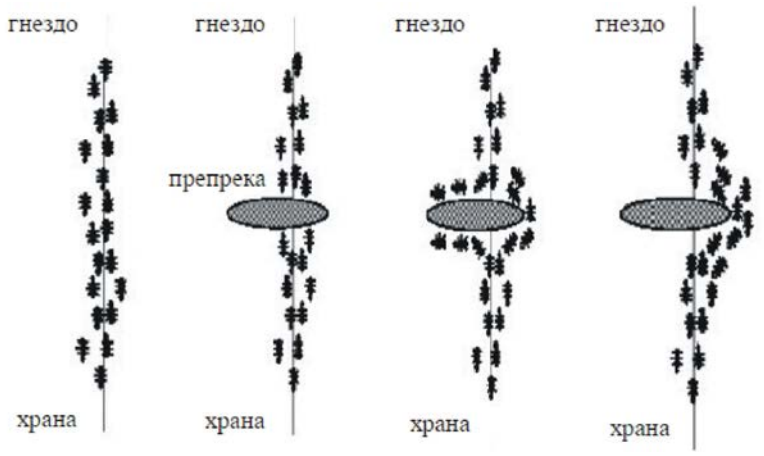
Слика 4. 2-opt и 3-opt преспојувања

Меѓу останатите алгоритми за подобрување на тури се вбројуваат: симулираното калење (simulated annealing) и табу пребарувањето (tabu search) (повеќе во: [5, 6]).

2.2. ОПТИМИЗАЦИЈА СО КОЛОНИЈА НА МРАВКИ

Често пати истражувачите се обидуваат да ја имитираат природата при решавањето на посложените проблеми. Така, *генетскиот алгоритам (ГА)* ја имитира природната селекција на видовите на тој начин што најсилните видови имаат поголеми шанси да ги пренесат своите гени на следните генерации по пат на размножување. Од друга страна, *оптимизацијата со колонија на мравки (ОКМ)* го имитира социјалното однесување на мравките и нивното делување за доброто на колонијата. Ние овде ќе се задржиме на ОКМ како метод за решавање на ЗПТ.

Мравките се инсекти кои делуваат независно, но живеат во колонии и комуницираат меѓусебно (со останатите мравки од колонијата) со помош на хемикалијата *феромон*. При истражување на нови територии тие ја испуштаат оваа хемикалија по патот по кој се движат, но исто така се и осетливи на оваа супстанца. Можеби најпознат пример е примерот со ставање на препрека на патот по кој одат мравките од нивното гнездо до храната и назад. На почетокот мравките ја заобиколуваат препреката од обете страни, но подоцна тие се „одлучуваат“ за пократкиот пат (Слика 5).



Слика 5. Мравките го одбираат пократкиот пат.

Ова однесување на мравките се должи на начинот на кој тие комуницираат. Секоја мравка го чувствува феромонот кој го испуштиле мравките кои поминале пред неа по патеката и така таа открива каде да се движи. Но, со текот на времето феромонот испарува, па јачината на ферономот е пресудна за одлуката на мравката за тоа по кој пат да тргне. Затоа, во случајот на препреката, мравките „одлучуваат“ да се движат по пократкиот пат, затоа што јачината на феромонот е посилна по тој пат.

Начинот на кој мравките го решаваат проблемот со препреката, инспирирало многу научници да предложат алгоритми за наоѓање на минимална тура при решавањето на ЗПТ. Тука ќе го изложиме алгоритмот на Dorigo, Maniezzo и Colorni од 1996 година ([7]). Според нивниот алгоритам, на почетокот m (вештачки) мравки се поставуваат секоја во различен случајно избран град. Во секој чекор се применува веројатносно правило според кое секоја од мравките одлучува кон кој град ќе се придвижи. Така, веројатноста k-тата

мравка која е во градот i да одбере да отиде во градот j во времето t е:

$$p_{ij}^{k}(t)=\frac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}[\tau_{il}(t)]^{\alpha}\cdot[\eta_{il}]^{\beta}},\quad \text{ако } j\in N_i^k, \tag{8}$$

каде што N_i^k се градовите кои k-тата мравка која се наоѓа во i-тиот град не ги посетила, $\tau_{ij}(t)$ е јачината на феромонот на лакот (i,j) во времето t и $\eta_{ij}=1/c_{ij}$ е видливоста на градот j од градот i. Параметрите α и β го одредуваат релативното влијание на патеката одредена од феромонот и евристичката информација. Така, ако $\alpha=0$, најверојатно ќе се изберат најблиските градови, што соодветствува на класичниот стохастички алчен алгоритам со повеќе почетни точки. Ако $\beta=0$, само феромонот одлучува кој град ќе се избере, што доведува до стагнација на алгоритмот, кога сите мравки следат ист пат и конструираат исто решение.

Откако сите мравки ќе ги конструираат своите тури, се преминува на обновување на јачината на ферономот на патеките. Обновувањето се прави така што прво се намалува јачината на феромонот на сите рабови со скалирање со константен фактор, а потоа ѝ се дозволува на секоја мравка да додаде феромон на рабовите кои ги посетила:

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t), \tag{9}$$

каде што $0<\rho\le 1$ е параметарот на испарување на феромонот на патеките. Улогата на параметарот ρ е да се избегне неограничена акумулација на феромон на патеките, но и да му помогне на алгоритмот да ги „заборави“ претходно донесените лоши одлуки. $\Delta\tau_{ij}^{k}(t)$ е количината на фероном кој k-тата мравка го става на работ (i,j) кој го посетила и се дефинира со:

$$\Delta\tau_{ij}^{k}(t)=\begin{cases}1/L^{k}(t), & \text{ако работ } (i,j) \text{ е посетен од } k-\text{тата мравка}\\ 0, & \text{инаку}\end{cases} \tag{10}$$

каде што $L^{k}(t)$ е должината на турата на k-тата мравка. Според фор-

мулата (10), колку е подобра турата на мравката, рабовите кои се дел од турата примаат поголема количина на феромон. Генерално, рабовите кои се користат од повеќе мравки и кои се дел од пократките тури, ќе примат повеќе феромон, па значи ќе биде поголема веројатноста да бидат избрани во следните итерации на алгоритмот. Овој процес се повторува сè додека не се најде доволно кратка тура.

Интересно е да се напомене дека ОКМ алгоритмите се едни од најкористените алгоритми за решавање на ЗПТ, токму заради својата ефикасност, брзо да најдат задоволително приближно оптимално решение. На пример, ОКМ е имплементитан во OptiMap софтверот за наоѓање на најбрза тура меѓу неколку селектирани града со користење на Google Maps ([9]).

3. ЗАКЛУЧОК

Задачата на патувачкиот трговец отсекогаш претставувала предизвик за решавање. Таа е NP-тешка задача, што значи дека не постои начин да се предвиди колку долго би ја решавале без да пробаме да ја решиме. Но денес, со помош на ефикасни алгоритми успешно се решаваат задачите со неколку стотици темиња, а и оние со над 2000 темиња веќе не претставуваат проблем. Евристичните алгоритми се тие кои сè повеќе земаат замав и носат во себе голем потенцијал кој треба да се искористи.

ЛИТЕРАТУРА

[1] E. Balas, P. Toth, *Branch and Bound Methods for the Traveling Salesman Problem*, Management Science Research Report No. MSRR 488, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, Pennsylvania, 1983.

[2] R. Diestel, *Graph Theory*, Springer-Verlag, New York, 2000.

[3] M. R. Garey, D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, Freemen, San Francisco, 1979.

[4] S. A. Haroun, B. Jamal, E. H. Hicham, *A Performance Comparison of GA and ACO Applied to TSP*, International Journal of Computer Applications, 117(19) (2015) 28 – 35.

[5] G. Laporte, *The Traveling Salesman Problem: An overview of exact and approximate algorithms*, European Journal of Operational Research, 59 (1992) 231 – 247.

[6] R. Matai, S. P. Singh, M. L. Mittal, *Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches*, in Traveling Salesman Problem, Theory and Applications, InTech, 2010.

[7] T. Stäutzle, M. Dorigo, *ACO Algorithms for the Traveling Salesman Problem*, in Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications, John Wiley & Sons, 1999.

[8] И. Стојковска, *Задачата на патувачкиот трговец - еноставна формулација, широка примена, тешка за решавање,* Портал ПОИМ на Институтот за математика, ПМФ, Скопје, 25 јули 2017, http://poim-pmf.weebly.com/zadacata-na-patuvackiot-trgovec.html

[9] OptiMap - Route Planner for Google Maps, http://gebweb.net/optimap/

[1] Универзитет „Св. Кирил и Методиј“, Скопје
Природно-математички факултет,
Институт за математика
ул. Архимедова 3, 1000 Скопје, Р. Македонија
e-mail: irenatra@pmf.ukim.mk

Примен: 8.07.2017
Поправен: 12.08.2017
Одобрен: 13.08.2017