VLIV VELIKOSTI POPULACE NA KVALITU ŘEŠENÍ ÚLOHY OBCHODNÍHO CESTUJÍCÍHO GENETICKÝM ALGORITMEM

IMPACT OF POPULATION SIZE USED IN GENETIC ALGORITHM ON QUALITY OF FOUND SOLUTION OF TRAVELLING SALESMAN PROBLEM

Ondřej Míča

Abstrakt

Úloha obchodního cestujícího je velmi známý a populární optimalizační problém. Protože se jedná o NP-těžkou úlohu, počet přípustných řešení je velmi vysoký – roste s faktoriálem počtu vrcholů v dopravní síti. Proto ani se soudobou výpočetní technikou není možné rozsáhlé úlohy obchodního cestujícího řešit exaktními metodami.

Genetický algoritmus patří mezi základní metaheuristické metody. Tento příspěvek se zaměřuje na experimentální ověření kvality nalezeného řešení v závislosti na velikosti populace použité při výpočtu.

Klíčová slova: *genetický algoritmus, úloha obchodního cestujícího, metaheuristiky, optimalizace*

Abstract

Travelling salesman problem is well known optimization problem. Because it is a NP-hard problem, it is usually solved with heuristic or metaheuristic methods.

The goal of this article is experimentally determine the impact of population size used in genetic algorithm on quality of found solution.

Key words: *genetic algorithm, travelling salesman problem, metaheuristics, optimization*

1 ÚVOD

Přestože výkon výpočetní techniky stále roste, některé optimalizační úlohy spadající do oblasti diskrétních úloh kombinatorického charakteru ještě pro svoji výpočetní složitost při větším rozsahu nemohou být vyřešeny v přijatelném časovém období. Proto se stále více prosazují komplexní sofistikované metody, obecně nazývané metaheuristiky, které nabízejí nalezení kvalitního (sub)optimálního řešení v relativně krátkém čase.

2 ÚLOHA OBCHODNÍHO CESTUJÍCÍHO

Úloha obchodního cestujícího (Travelling salesman problem, zkráceně TSP) patří mezi základní úlohy operačního výzkumu. Při řešení TSP je zadán graf reprezentovaný množinou uzlů a množinou hran. Cílem je najít Hamiltonovskou kružnici v grafu – nejkratší možnou cestou navštívit všechny vrcholy v grafu a vrátit se do výchozího vrcholu.

Jedná se o NP-těžký problém. Počet možných řešení N se určí podle vztahu (1), kde n je počet vrcholů v grafu.

$$N = (n-1)!/2 \tag{1}$$

V tabulce 1 je uveden počet možných řešení a délka exaktního výpočtu za předpokladu, že počítač by byl schopný vyhodnotit 1 000 000 000 řešení za sekundu. Z uvedených údajů je zřejmé, že exaktně lze řešit maximálně úlohy o jednotkách uzlů. Proto se v současné době TSP řeší heuristickými a metaheuristickými metodami.

Tabulka 1: Možnosti exaktního řešení úlohy obchodního cestujícího.

Počet uzlů	**Počet možných řešení**	**Trvání exaktního řešení [roků]**
18	177 843 714 048 000	0,006
19	3 201 186 852 864 000	0,102
20	60 822 550 204 416 000	1,929
21	1 216 451 004 088 320 000	38,573
22	25 545 471 085 854 700 000	810,042
23	562 000 363 888 804 000 000	17 820,915
24	12 926 008 369 442 500 000 000	409 881,037

Matematicky lze TSP popsat následujícím modelem [5]:

$$min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} x_{ij} \tag{2}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \text{ for } j = 1 \dots n \qquad \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \text{ for } i = 1 \dots n \tag{3}$$

$$y_i - y_j + n x_{ij} \leq n - 1 \text{ for } 2 \leq i \neq j \leq n \tag{4}$$

$$x_{ij} \in \{0,1\} \text{ for } i, j = 1 \dots n \qquad y_i \in \mathbb{N}_0 \text{ for } i = 2 \dots n \tag{5}$$

kde:
n počet vrcholů,
d_{ij} vzdálenost mezi vrcholy i a j,
x_{ij} bivalentní proměnná, která nabývá hodnotu 1, pokud hrana mezi vrcholy i a j patří do Hamiltonovské kružnice, jinak nabývá hodnotu 0.

Rovnice (3) zaručují, že do každého vrcholu se může vstoupit jen jednou a z každého vrcholu se může jen jednou odejít. Podmínka (4) zaručuje existenci jedné souvislé Hamiltonovské kružnice a vylučuje existenci podcyklů. Podmínky (5) jsou pouze obligatorními podmínkami.

Úloh obchodního cestujícího je celá řada:

- symetrická: vzdálenost mezi vrcholy *i* a *j* a *j* a *i* je stejná,
- asymetrická: vzdálenost mezi vrcholy *i* a *j* a *j* a *i* není stejná,
- s časovými okny: každý vrchol lze navštívit pouze v předem daném čase,
- vícenásobná úloha obchodního cestujícího: v grafu je více obchodních cestujících,
- Sequential Ordering Problem: uzly lze navštívit pouze v určitém pořadí,
- atd.

V tomto příspěvku bude uvažována pouze symetrickou úlohou obchodního cestujícího.

3 GENETICKÝ ALGORITMUS

První návrhy evolučních algoritmů se objevily již v šedesátých letech minulého století. Velmi důležitou roli ve vývoji genetických algoritmů sehrál John Henry Holland, který ve své knize *Adaptation in Natural and Artificial Systems* poprvé použil označení genetický algoritmus [3].

Genetické algoritmy jsou inspirovány přírodou, konkrétně Darwinovou teorií evoluce založené na přirozeném výběru. V genetickém algoritmu každé jedno řešení představuje jednoho jedince v populaci (nebo též chromozom). A každý jedinec je zároveň potomkem dvou rodičů – část jeho chromozomu je od prvního rodiče a zbytek od druhého rodiče[1]. Základní myšlenkou algoritmu je vytvářet stále silnější jedince křížením dvou vhodných rodičů. Naopak slabí, nebo defektní jedinci se dále nereprodukují.

Na rozdíl od metod tabu search, či simulovaného žíhání, genetický algoritmus nehledá řešení na dopravní síti, ale pracuje s vlastní množinou řešení – jednotlivými chromozomy, které lze dále dělit na geny.

Na začátku algoritmu je nutné vytvořit výchozí populaci jedinců, ze které se odstartuje běh algoritmu. Existují tři možnosti, jak získat počáteční populaci. První možností je generaci náhodně vygenerovat, ale je důležité, aby populace obsahovala dostatek rozdílných jedinců, z důvodu pokrytí co možná největšího prostoru možných řešení. Nebo lze jako počáteční populaci nasadit množinu kvalitních řešení získaných jinými optimalizačními metodami. Výhodou tohoto přístupu je velká pravděpodobnost rychlejšího nalezení velmi kvalitního řešení, nevýhodou je eventuální nebezpečí předčasné konvergence algoritmu. Třetí možnost je kombinací obou uvedených přístupů – několik kvalitních jedinců dosadit a zbytek generace vygenerovat [3].

Velikost populace v algoritmu *n*, která je většinou nastavitelným parametrem úlohy, má velký dopad na kvalitu výsledného řešení. Malá populace může vést k uváznutí v lokálním extrému, za to velká populace neúměrně zvyšuje výpočetní čas úlohy.

V každé iteraci algoritmu se nejprve ohodnotí všichni jedinci z rodičovské generace – spočítá se tzv. fitness každého jedince. Podle tohoto fitness pak dále probíhá výběr jedinců pro vytvoření potomků.

Možností výběru rodičů je několik. V [6] na straně 166 je uvedeno, že původně navrženým a nejrozšířenějším způsobem je mechanismus výběru s pravděpodobností přímo úměrnou fitness jedince (*Fitness proportionate selection*). Možnou implementací takového přístupu je tzv. ruletová selekce (*Roulette-wheel selection*). Pro každého jedince v populaci se určí pravděpodobnost, se kterou může být vybrán podle vztahu (6).

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \quad pro\ i = 1..n \tag{6}$$

kde:

p(i) pravděpodobnost výběru *i*-tého jedince,
f(i) fitness *i*-tého jedince,
n velikost rodičovské populace.

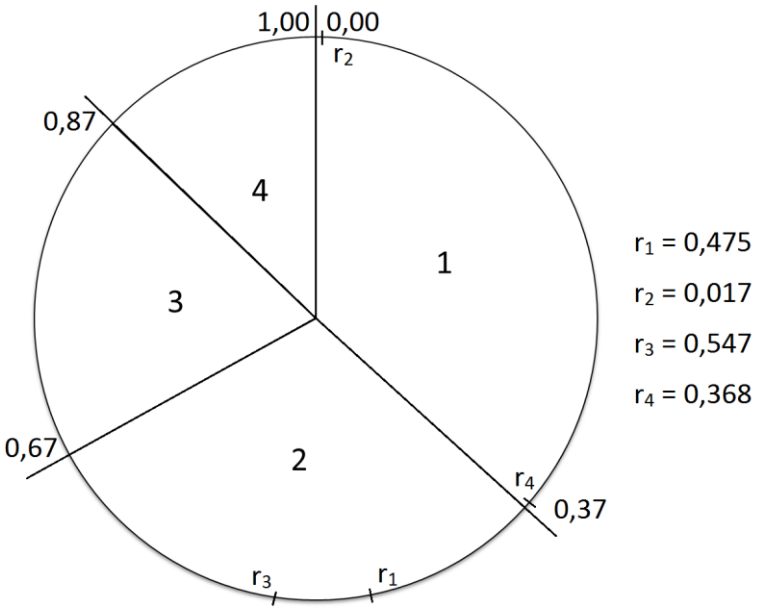
Vlastní ruletové kolo se pak vytvoří součtem těchto pravděpodobností a poté se každému jedinci přiřadí kruhová výseč proporcionálně odpovídající velikosti pravděpodobnosti jeho vybrání za rodiče.

[1] V tuto chvíli záměrně pomíjím možnost mutace, která bude popsána dále.

Následně se vygenerují náhodná čísla r_i pro $i = 1..k$ s rovnoměrným rozdělením pravděpodobnosti z intervalu ⟨0; 1⟩, kde *k* je počtem jedinců vstupujících do křížení. Konkrétní jedinci pro reprodukci se vyberou podle vztahu (7) [4].

$$\sum_{j=1}^{l-1} p(j) < r_i < \sum_{j=1}^{l} p(j) \tag{7}$$

Příklad ruletové selekce je uveden na obrázku 1. V tomto případě mají být ke křížení vybráni 4 jedinci. Proto byla vygenerována 4 náhodná čísla, na základě kterých byli k další reprodukci dvakrát vybráni jedinci 1 a 2.



Obrázek 1: Princip fungování ruletové selekce. Zdroj: [4]

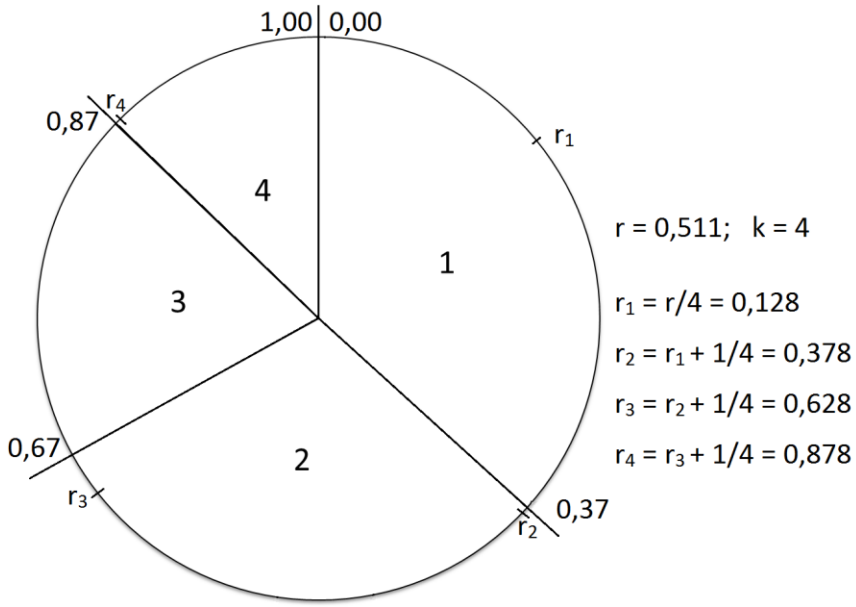
Výše uvedený přístup má nevýhodu, že pokud je v populaci několik opravdu silných jedinců, výrazně roste pravděpodobnost jejich výběru a tím i hrozba předčasné konvergence (i v uvedeném příkladu byly dvakrát vybráni oba nejsilnější jedinci). Tomuto se snažil předejít Baker [1] úpravu ruletové selekce. Tato modifikovaná metoda se nazývá stochastický univerzální výběr (*Stochastic Universal Sampling*).

Základ metody je stejný, pro každého jedince v populaci se určí pravděpodobnost výběru podle vztahu (6). Bakerův návrh spočívá v tom, že čísla r_i pro výběr jedinců ke křížení nejsou generovány náhodně, ale mají rovnoměrné rozložení po celém ruletovém kole. Nejprve se vygeneruje náhodné číslo *r* z intervalu ⟨0; 1⟩ s rovnoměrným rozdělením pravděpodobnosti, které se vydělí počtem jedinců vstupujících do křížení *k*. Takto vznikne číslo r_1, které značí prvního vybraného jedince podle vztahu (7). Číslo r_i pro výběr *i*-tého jedince se určí podle vztahu (8).

$$r_i = r_{i-1} + \frac{1}{k} \quad pro\ i = 2..k \tag{8}$$

Vlastní výběr jedince pak probíhá dosazením čísla r_i do vztahu (7). Výhodou tohoto přístupu je zachování větší rozmanitosti populace.

Příklad metody stochastického univerzálního výběru je znázorněn na obrázku 2. Ruletové kolo je stejné, jako v předchozím příkladu na obrázku 1. Náhodně bylo vygenerováno číslo $r = 0{,}511$. Číslo pro výběr prvního jedince r_1 bylo získáno vydělením vygenerovaného čísla r počtem jedinců, kteří se mají vybrat ke křížení k (v tomto případě 4). Další čísla r_i byla určena podle vztahu (8). V tomto konkrétním příkladu byli ke křížení vybráni jedinec 1, dvakrát jedinec 2 a jedinec 4.



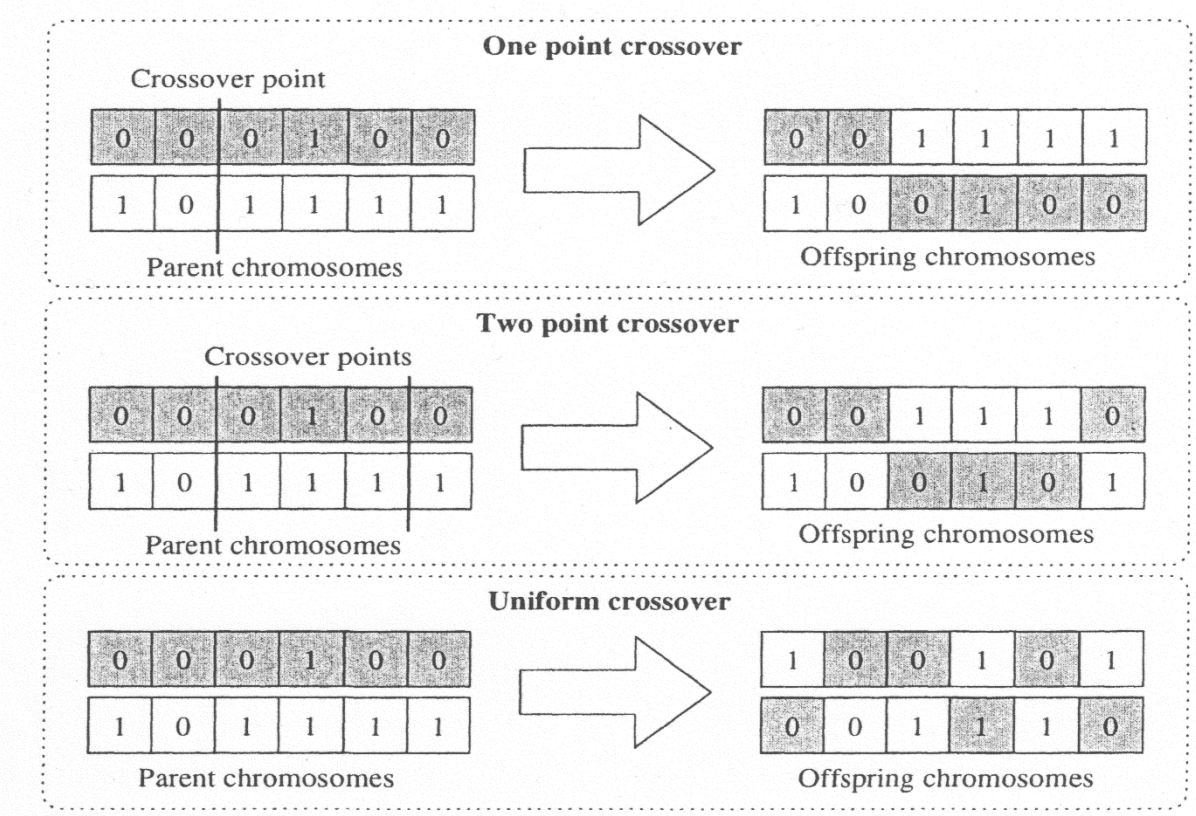
Obrázek 2: Princip fungování stochastického univerzálního výběru. Zdroj: [4]

Jinou možností výběru jedinců je turnajová selekce (*Tournament selection*). Z populace se náhodně vybere x jedinců, z nichž je vítězem turnaje, který je dále vybrán k reprodukci, ten s nejvyšší hodnotou fitness. Čím je vyšší hodnota parametru x, který udává počet jedinců v turnaji, tím je vyšší selektivní tlak. Proto se obvykle používá hodnota $x = 2$. Pro výběr k jedinců je nutné uspořádat k turnajů [4].

Ze dvou vybraných rodičů jsou vytvořeni typicky dva potomci. Pokud je jedinec specifikován množinou objektů, křížení nejčastěji probíhá podle jednoho z následujících způsobů. Buď je vygenerován jeden bod zlomu a oba rodiče si prohodí geny za tímto bodem zlomu, čímž vzniknou dva noví jedinci. Nebo se vygenerují dva body a oba rodiče si vymění geny mezi těmito body. Třetí způsob je, že se náhodně vygeneruje šablona, podle které si rodiče budou měnit konkrétní geny. Všechny popsané možnosti křížení jsou znázorněny na obrázku 3.

Pokud je jedinec specifikován pořadím objektů, je operace křížení náročnější. Je třeba aplikovat pokročilejší přístupy ke křížení, např. Partially Matched Crossover, nebo Cycle Crossover, aby zůstala zachována přípustnost řešení. Oba uvedené přístupy jsou podrobněji popsané v [2].

Častým přístupem je, že se vytvoří více potomků, než je potřebná velikost populace n. Pak probíhá selekce vytvořených potomků do nové generace, přičemž se klade důraz na to, aby se do populace dostali nejsilnější potomci, a zároveň aby nová populace byla dostatečně rozmanitá.



Obrázek 3: Možné přístupy ke křížení dvou jedinců. Zdroj: [2]

Pseudokód základního genetického algoritmu by mohl vypadat takto:

```
GenAlgoritmus(n)
      VytvorPocatecniPopulaci(n)
      repeat
            VyhodnotPopulaci
            VyberRodicu(k)
            ProvedKrizeni(n)
      until PravidloZastaveni
end;
```

Parametrem metody může být velikost populace *n*, která bude pro všechny generace společná. Procedura `VytvorPocatecniPopulaci` vygeneruje počáteční populaci jedinců. Parametrem této procedury je velikost populace *n*. Při implementaci je kladen důraz na to, aby počáteční populace byla dostatečně rozmanitá a tím pokryla co nejvyšší počet možných řešení.

Procedura `VyhodnotPopulaci` určí podle daného kritéria fitness každého jedince, na jehož základě pak probíhá výběr rodičů pro vytvoření potomků. Současně se provede kontrola, zda se v populaci nenachází jedinec s hodnotou fitness vyšší, než má zatím nejlepší nalezené řešení.

V proceduře `VyberRodice` se podle výše popsaných přístupů vybere *k* jedinců – rodičů, ze kterých bude vytvořena nová populace možných řešení.

Procedura `ProvedKrizeni` z vybraných rodičů vytvoří *n* nových potomků. Po vytvoření potomků může stejně jako v přírodě proběhnout mutace, která na náhodně vybraném potomku (potomcích) změní náhodně vybraný gen (geny). I v této proceduře je dobré dbát na to, aby nová generace byla dostatečně rozmanitá, jinak může dojít k předčasné konvergenci algoritmu.

Podmínkou zastavení pro genetický algoritmus může být vytvoření předem daného počtu nových generací, vytvoření určitého počtu generací bez aktualizace nejlepšího nalezeného řešení, nebo uplynutí času výpočtu. Také se doporučuje zastavit algoritmus, pokud se sníží rozmanitost nově vytvořených populací.

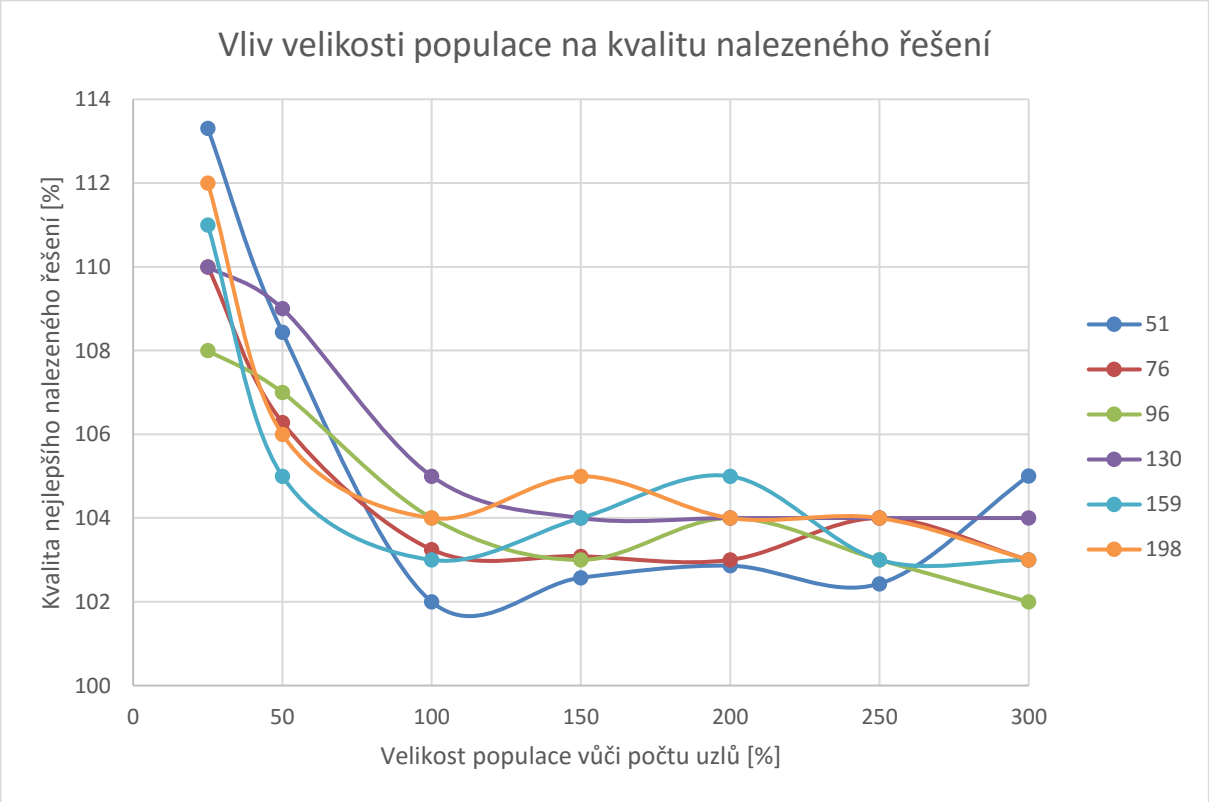
4 VÝSLEDKY EXPERIMENTU

Pro experimentální ověření vlivu velikosti populace na kvalitu řešení úlohy obchodního cestujícího genetickým algoritmem v praxi byla použita knihovna TSPLIB [7], kterou provozuje Ruprecht-Karls-Universität v Heidelbergu. Jedná se o soubor 112 příkladů úloh obchodního cestujícího společně s jejich optimálními řešeními, z nichž řada je založena na problémech z reálného světa. Tato knihovna je veřejně přístupná a slouží výzkumníkům z celého světa pro porovnání vlastních výsledků řešení.

Pro vlastní řešení byl genetický algoritmus implementován v programovacím jazyce Java a pomocí této implementace bylo řešeno 6 úloh prezentovaných v TSPLIB, každá s jiným počtem uzlů (51, 76, 96, 130, 159 a 198 uzlů).

Pro každou instanci TSP docházelo k postupnému zvyšování velikosti populace až do trojnásobku počtu uzlů v instanci Pro kažkou velikost populace bylo spočítáno 1 000 replikací. Celkem bylo vypočítáno 42 000 iterací TSP. Pro každou sledovanou velikost populace bylo porovnáno nejlepší vypočtené řešení s nejlepším řešením uvedeným v TSPLIB [7].

Vypočtený rozdíl nejlepšího vypočteného řešení s nejlepším řešením uvedeným v TSPLIB je znázorněn na obrázku.



Obrázek 4: Vliv velikosti populace na kvalitu řešení

5 ZÁVĚR

Tento příspěvek je věnován problematice určení velikosti populace při řešení úlohy obchodního cestujícího genetickým algoritmem. Experimentálním ověřením na 6 různých instancích TSP bylo zjištěno, že pro velikost populace větší jak 150 % uzlů v instanci TSP již nedochází k výraznému zlepšení kvality nalezeného řešení, nýbrž pouze stoupá čas potřebný pro výpočet.

Použitá literatura

1. BAKER, James E., 1987. Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the Second International Conference on Genetic Algorithms and their application*. New York: Lawrence Erlbaum Associates, s. 14-21. ISBN 0-8058-0158-8
2. BURKE, Edmund a KENDALL, Graham, 2005. *Search methodologies: introductory tutorials in optimization and decision support techniques*. New York: Springer, 620 s. ISBN 978-0-387-28356-2.
3. GENDREAU, Michel a POTVIN, Jean-Yves, 2010. *Handbook of metaheuristics*. 2nd ed. New York: Springer, 648 s. ISBN 978-1-4419-1663-1.
4. HYNEK, Josef, 2008. *Genetické algoritmy a genetické programování*. Praha: Grada, 182 s. ISBN 978-80-247-2695-3.
5. MILLER, C. E., A. W. TUCKER a R. A. ZEMLIN. Integer Programming Formulation of Traveling Salesman Problems. In: *Journal of the ACM*. s. 326-329. DOI: 10.1145/321043.321046. ISSN 00045411.
6. MITCHELL, Melanie, 1998. *An introduction to genetic algorithms*. Cambridge: The MIT Press, 209 s. ISBN 0-262-13316-4.
7. TSPLIB. *Ruprecht-Karls-Universität Heidelberg* [online]. 2001 [accessed 2016-11-20]. Dostupné z: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Kontaktní údaje

Ing. Ondřej Míča
Univerzita Pardubice
Dopravní fakulta Jana Pernera
Katedra informatiky v dopravě
Studentská 95, 532 10 Pardubice
Tel: 466 036 428
email: ondrej.mica@upce.cz