Semestrální práce předmětu a0m33eoa - TSP s několika cestujícími

Jakub Moravec

***Abstrakt*— V práci jsou popsány tři algoritmy pro řešení problému TSP s několika cestujícími - lokální prohledávání, evoluční a memetický algoritmus. Ty jsou následně porovnány na třech různých datasetech a výsledky porovnány.**

I. Zadání

Úkolem je vyřešit úlohu TSP s několika cestujícími pomocí heuristických algoritmů. Na vstupu je úplný neorientovaný graf o *N* uzlech. Cílem je nalézt takovou množinu cest pro *M* cestujících, které všechny vychází z počátečního uzlu (depotu) a zase v něm končí, všechny uzly (kromě depotu) jsou navštíveny právě jednou a délka nejdelší cesty je minimální. Žádná z cest nesmí mít nulovou délku.

II. Úvod

Úloha TSP s několika cestujícími známá také jako **VRP** (*Vehicle routing problem*) je kombinatorická optimalizační úloha odpovídající otázku "Jaké je optimální uspořádání hran pro několik cestujicích tak, aby byly výsledné cesty pokrývající všechny uzly nejkratší?". Určení optimálního řešení tohoto problému je *NP-těžká* úloha, proto bývá často řešena pomocí heuristik a aproximačních algoritmů. V tomto dokumentu budou představena řešení pomocí *lokálního prohledávání*, *evolučního algoritmu* a *memetického algoritmu* (kombinace předcházejících).

Fiteness funkce pro vyhodnocování optimality řešení byla převzata ze zadání úlohy: kvalita řešení je určena jako délka nejdelší cesty z M cest. Tuto fitness funkci se budeme snažit minimalizovat. Délka cesty je určena *Euklidovskou vzdáleností* měst.

Jako reprezentace řešení problému byla zvolena uspořádaná množnina identifikátorů měst, kde pořadí měst určuje pořadí jejich průchodu. Depot (id 0) je v množině obsažen *M +1* krát, všechna ostatní města právě jednou. Tato reprezentace byla zvolena proto, aby bylo možné stejným způsobem použitými algoritmy měnit pořadí měst v cestách jednotlivých cestovatelů i přesouvat města mezi cestovateli.[1] Příklad reprezentace úlohy s dvěmi cestovateli:

$$[0, 1, 3, 5, 7, 0, 9, 2, 4, 6, 8, 0]$$

III. Cíl práce

Cílem práce je implementovat *algoritmus lokálního prohledávání*, *evolučních algoritmus* a *memetický algoritmus* generující řešení problému taková, aby jejich hodnota fitness funkce byla minimální a to při co možná nejmenším počtu vyhodnocení definované fitness funkce.

TABULKA I
Parametry lokálního prohledávání

Počet sousedů	5
Pravděpodobnost prohození měst	0.5
Pravděpodobnost posunu *breakpointu*	0.5

IV. Experimenty

V této sekci budou představeny všechny implementované algoritmy, provedené experimenty a jejich výsledky.

A. Popis algoritmů

1) Lokální prohledávání: Reprezentace řešení pro tuto úlohu úlohu je odlišné, než u ostatních úloh. Řešení je reprezentováno dvěmi uspořádanými množinami, kde první určuje pořadí měst (neobsahuje depot), druhá obsahuje tzv. *breakpointy*, které definují, kdy je ukončena cesta jednoho cestovatele a kam má tedy být vložen depot. Příklad reprezentace úlohy s dvěmi cestovateli:

$$[1, 3, 5, 7, 9, 2, 4, 6, 8][4]$$

Jako iniciální řešení jsou přebrána města v pořadí na vstupu a *breakpointy* jsou rovnoměrně vygenerovány dle počtu cestovatelů.[2]

Pro lokální prohledávání je implementován pertubační algoritmus vybírající nejlepšího jedince z *k* vygenerovaných sousedů stávajícího řešení (*best neighbour*). Pertubační operátor generuje okolní řešení prohazováním náhodných měst a náhodným posouváním *breakpointů* o jednu pozici. Parametry algoritmu jsou uvedeny v tabulce I. Parametry byly určeny empiricky podle průběžných výsledků algoritmu.

2) Evoluční algoritmus: Evouční algoritmus používá následující operátory:

- **Inicializace**: První generace je inicializována zcela náhodnými řešeními. Každý jedinec je generován ze vstupních dat přidáním stejného počtu pivotů, jako je počet cestovatelů (jeden už je přítomen) a náhodným proházením pořadí měst.
- **Křížení**: Jako operátor křížení byl zvolen tzv. *ordered crossover*, který vybere náhodnou sekvenci měst jednoho

[1] Pro *lokální prohledávání* byla zvolena jiná reprezentace (popsaná v odstavci IV-A.1). Ta byla pro *evoluční a memetický algoritmus* upravena do této podoby, protože ta je pro uvedené algoritmy vhodnější.

[2] To bylo při odevzdání bodově penalizováno, protože generování iniciálního řešení je deterministické a neobsahuje žádné náhodné prvky.

TABULKA II
PARAMETRY EVOLUČNÍHO ALGORITMU

Velikost populace	500
Pravděpodobnost křížení	0.6
Pravděpodobnost mutace	0.005
Počet účastníků turnaje	10
Počet jedinců zachovaných do příští generace	100

z rodičů, ty vloží do potomka a doplní je o zbylá města v pořadí dle druhého rodiče. Tento algoritmus byl upraven tak, aby zachovával počet pivotů v řešení.

- **Mutace**: Jako mutace je použit algoritmus náhodně prohazující dvě města v rámci celého řešení (tedy i mezi různými cestovateli).
- **Selekce**: Vzhledem k tomu, že definovaná fitness funkce je minimalizována, byla pro výběr jedinců do nové generace zvolena *turnajová selekce*.

Parametry evolučního algoritmu jsou uvedeny v tabulce II. Parametry byly určeny empiricky podle průběžných výsledků algoritmu.

3) Memetický algoritmus: Memetický algoritmus kombinuje evoluční algoritmus a lokální prohledávání. Využívá následující operátory:

- **Inicializace**: Stejné jako u evolučního algoritmu (odstavec IV-A.2).
- **Křížení**: Stejné jako u evolučního algoritmu (odstavec IV-A.2).
- **Mutace**: Pro memetický algoritmus byla upravena mutace, ta náhodně vybere začátek či konec cesty (o náhodné délce) některého cestovatele a prohodí ho se začátkem či koncem cesty (o náhodné délce) jiného cestovatele. K této mutaci bylo přistoupeno proto, že na vygenerovaných řešeních bylo často patrné, že jsou vygenerovány špatně právě začátky a konce cest (cesty se proto často křížili apod.). Přidaná fáze lokálního prohledávání potom zaručí, že pokud takové prohození hodně "zpřehází"cesty, budou "narovnány".
- **Lokální prohledávání**: Lokální prohledávání bylo přidáno jako samostatná fáze algoritmu, která je spouštěna po křížení a mutaci nad nově vygenerovanými řešeními. Jako algoritmus lokálního prohledávání byl zvolen *2-opt* algoritmus. Ten je spouštěn dvěmi různými způsoby, buď to pouze na cestě jednoho cestovate (změna neovlivní cesty ostatních cestovatů), nebo na celém jedinci (mohou být prohazovány hrany i mezi různými cestovateli). K tomu bylo přistoupeno proto, že spouštění lokálního prohledávání pouze na cestách jednotlivých cestovatelů není dostatečné, na druhou stranu ale spouštění algoritmu vždy na celém jedinci vede k příliš násilným změnám jedince. *2-opt* algoritmus je tedy spouštěn vždy (s definovanou pravděpodobností) na cestách náhodných cestovatů a na celém jedinci pouze v případě, že se již několik iterací nepodařilo vygenerovat řešení lepší, než je nejlepší stávající. Aby algoritmus lokálního prohledávání nenavyšovavl počet ohodnocení *fitness funkce* (a tak příliš nenavyšoval časovou náročnost), je upraven tak, že je analyticky zjišťováno, které cesty se kříží, a ty jsou potom s definovanou pravděpodobností prohozeny.
- **Selekce**: Stejné jako u evolučního algoritmu (odstavec IV-A.2).

TABULKA III
PARAMETRY MEMETICKÉHO ALGORITMU

Velikost populace	500
Pravděpodobnost křížení	0.6
Pravděpodobnost mutace	0.005
Pravděpodobnost lokálního prohledávání	0.25
Počet účastníků turnaje	10
Počet jedinců zachovaných do příští generace	100

Parametry memetického algoritmu jsou uvedeny v tabulce III. Parametry byly určeny empiricky podle průběžných výsledků algoritmu.

TABULKA IV
VÝSLEDKY EXPERIMENTŮ PO 2000000 VYHODNOCENÍ FITNESS FUNKCE

Vstup	Algoritmus	Minimální	Medián	Průměr	Odchylka
M50	LS	4785.03	5014.32	5096.84	255.88
M50	EA	4846.80	5093.58	5127.64	257.36
M50	MA	4421.35	4996.18	4976.83	288.23
M100	LS	7679.63	8231.64	8326.89	371.50
M100	EA	8163.44	8821.02	8733.80	410.98
M100	MA	7596.20	7971.86	8133.59	466.24
M200	LS	18094.88	19836.41	19738.91	992.89
M200	EA	18936.75	20246.64	20561.97	1035.26
M200	MA	17146.45	19581.47	19865.19	1153.24

B. Konfigurace experimentů a výsledky

Každý z uvedených algoritmů byl tastován na poskytnutých sadách *50ti, 100 a 200 měst*, vždy pro *3* cestovatele. Každá z těchto konfigurací byla spuštěna *20krát*, v grafech 2, 3 a 4 jsou uvedeny průměry a směrodatné odchylky nejlepších výsledků experimentů po různé počty ohodnocení *fitness funkce*.

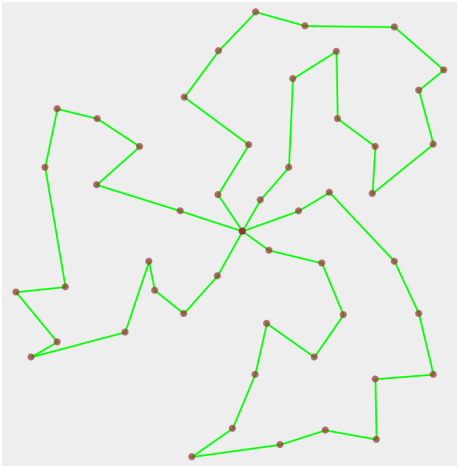
Tabulka IV zobrazuje výsledné hodnoty fitness funkce po 2000000 ohodnocení fitness funkce.

V. DISKUSE

Nejlepší průměrné výsledky na všech vstupech po 2000000 ohodnoceních fitness funkce má memetický algoritmus (kombinace evolučního algoritmu a lokálního prohledávání). Je ale patrné, že rozdíly po tomto počtu ohodnocení fitness funkce nejsou příliš velké a pokud by byly výsledky porovnávány po menším počtu ohodnocení, byl by úspěšnější algoritmus lokálního prohledávání, který konverguje k lokálnímu optimu výrazně rychleji. Naopak je předpoklad, že při testování výsledků po více ohodnoceních fitness funkce by memetický algoritmus dosálh výrazně lepších výsledků, než lokální prohledávání. Evoluční algoritmus také dosahuje dobrých výsledků, ale jeho konvergence k nim je pomalejší.

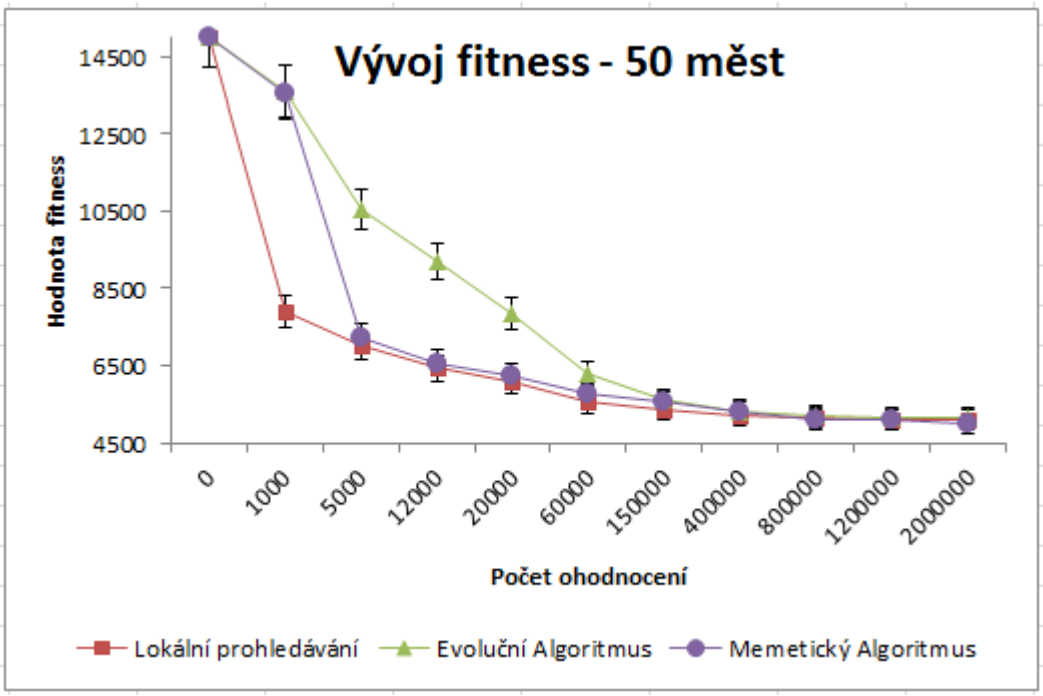
VI. ZÁVĚR

Byly implementovány tři algoritmy řešící problém *mTSP*, respektive *VRP - Vehicle Routing Problem*. Především pro menší vstupy algoritmy dosahují velmi dobrých výsledků blízkých globálnímu optimu. Příkladem je řešení nalezené

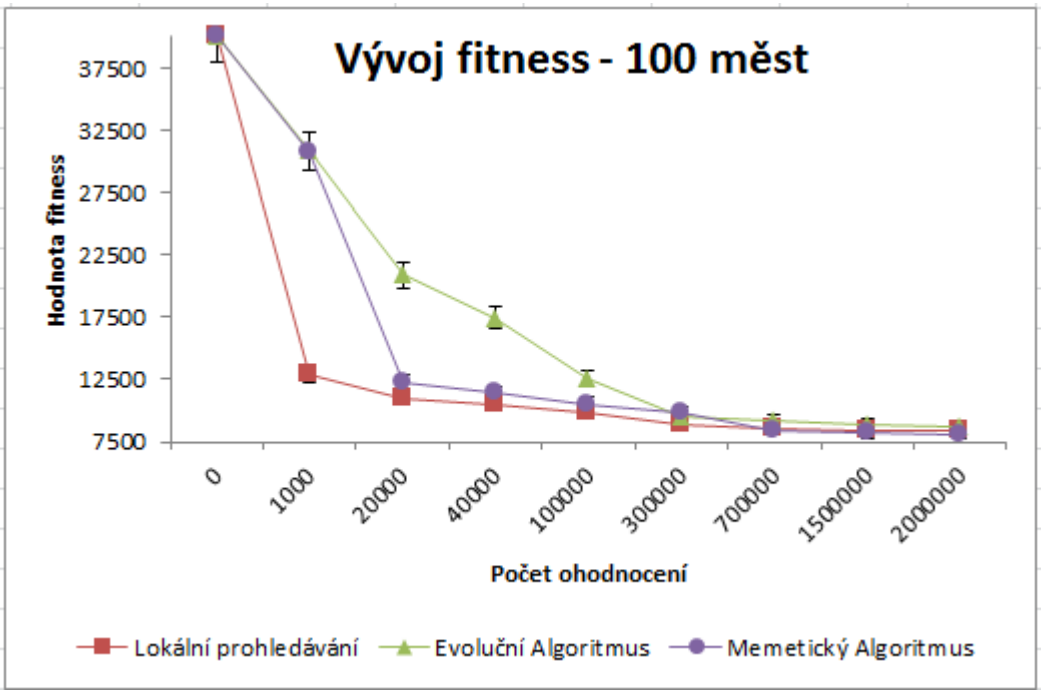


Obr. 1. Řešení vygenerované memetický algoritmem pro vstpu s 50 městy.

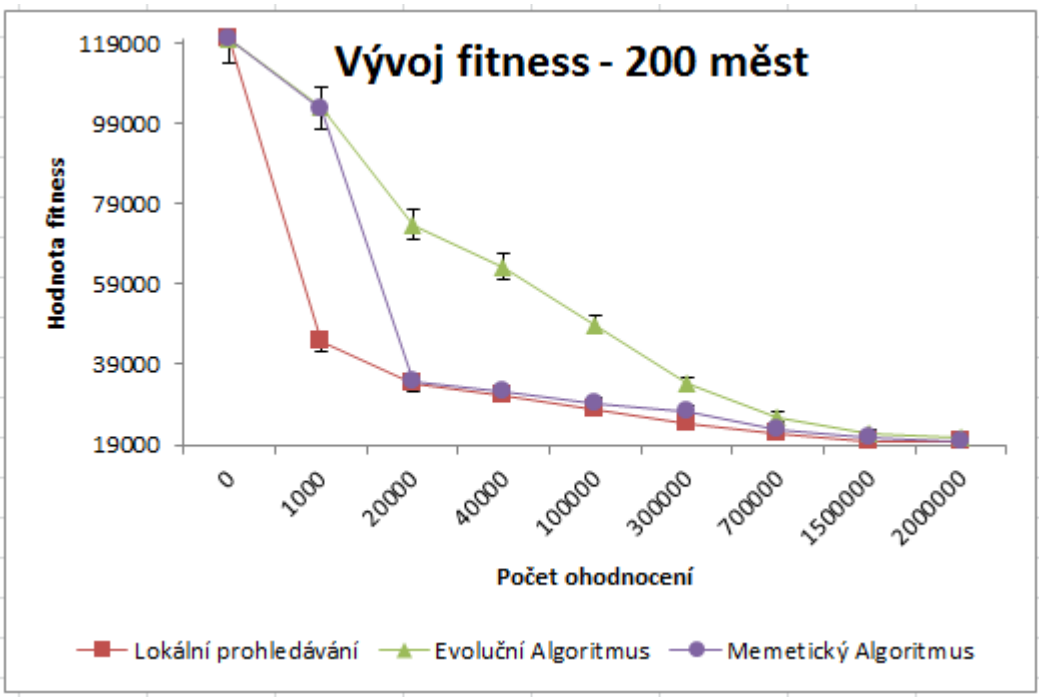
memetickým algoritmem pro 50 měst (obrázek 1). Memetický algoritmus kombinující operátor křížení *ordered crossover* a *2-opt* lokální prohledávání je také průměrně nejlepším algoritmem při uvedené konfiguraci experimentů. Další výzkum by se mohl věnovat optimálnějšímu určení parametrů algoritmů - ty mohou být určeny například evolučním algoritmem.



Obr. 2. Výsledky algoritmů pro 50 měst



Obr. 3. Výsledky algoritmů pro 100 měst



Obr. 4. Výsledky algoritmů pro 200 měst