Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Diplomová práce

Genetické algoritmy v úloze obchodního cestujícího

Martin Kučera

Vedoucí: Vlasta Radová

2021/2022

Prohlášení

Prohlašuji, že jsem diplomovou práci na téma "Genetické algoritmy v úloze obchodního cestujícího" zpracoval samostatně pod dohledem vedoucí diplomové práce. Veškerou použitou literaturu a další podkladové materiály uvádím v seznamu použitých zdrojů.

V Plzni dne ....
podpis autora

Poděkování

Chtěl bych poděkovat především své vedoucí Doc. Dr. Ing. Vlastě Radové, bez jejíž trpělivosti, odborných rad a ochoty pomoci by práce byla jen těžko dokončena. Dále bych chtěl poděkovat Doc. RNDr. Přemyslu Holubovi Ph.D. za pomoc při řešení problémů z teorie grafů. Mé poděkování patří také rodině a přátelům, jejichž psychická podpora byla při psaní práce neocenitelná.

Abstrakt

Hlavním zaměřením této práce jsou genetické algoritmy. Jedná se o typ náhodného prohledávání podpořený heuristikou ve formě fitness funkce imitující přírodní proces evoluce. V této práci budeme analyzovat jejich průběh na úloze obchodního cestujícího, obtížně řešitelném kombinatorickém problému ze skupiny NP-úplných úloh. Nejprve si představíme jednotlivé pojmy jako je chromozom, populace nebo křížení a zavedeme požadavky na jednotlivé procesy algoritmu, aby byl aplikovatelný na úlohu obchodního cestujícího. V dalších kapitolách prozkoumáme vliv velikosti populace N, míry elitismu E_R a míry mutace M_R na schopnost algoritmu konvergovat k optimálnímu řešení úlohy obchodního cestujícího. Různá nastavení algoritmu vyzkoušíme na více grafech včetně úloh *att48*, *eil101* nebo *tsp225* z *TSPLIB* a pokusíme se určit optimální hodnoty pro tyto parametry. V další kapitole si představíme různé operátory mutace a rekombinační operátory. Pokusíme se experimentálně ověřit jejich působení na chod algoritmu a určit takové operátory, které mají nejlepší vliv na schopnost algoritmu konvergovat k optimu.

Klíčová slova

Genetický algoritmus, úloha obchodního cestujícího, velikost populace, elitismus, míra mutace, operátory křížení, operátory mutace

Abstract

The subject of this diploma thesis is genetic algorithms. Genetic algorithms are a type of random search supported by a heuristic in the form of a fitness function that imitates the natural process of evolution. We will analyze their performance in the travelling salesman problem, a complex combinatorial problem belonging to the NP-complete class. Firstly, we will introduce concepts such as chromosome, population or crossover. We will then implement requirements for individual processes of the algorithm to apply to the travelling salesman problem. The effect of population size N, elitism rate E_R and mutation rate M_R on the ability of the genetic algorithm to converge to the optimal solution will be discussed in the following chapters. Multiple algorithm settings will be tested and evaluated on the *att48*, *eil101* and *tsp225* graphs from the *TSPLIB*. We will then try to determine the optimal values for the parameters above. In the next chapter, we will introduce various mutation and crossover operators. Experiments will be conducted to determine operators with the best effect on the performance of the genetic algorithm.

Keywords

Genetic algorithm, travelling salesman problem, population size, elitism, mutation rate, crossover operators, mutation operators

Obsah

1 Úvod **6**

2 Základní definice **7**

3 Úloha obchodního cestujícího **9**

4 Genetické algoritmy **12**
4.1 Chromozom 13
4.2 Populace 13
4.3 Fitness funkce 14
4.4 Selekce 15
4.5 Křížení 17
4.6 Mutace 18
4.7 Průběh algoritmu 19
4.8 Rozdíly oproti ostatním přístupům 20

5 Parametry genetických algoritmů **24**
5.1 Velikost populace 24
5.2 Elite rate 27
5.3 Mutation rate 31

6 Rekombinační operátory křížení a operátory mutace **35**
6.1 Operátory mutace 36
6.1.1 Mutace EM 36
6.1.2 Mutace CIM a CIM* 36
6.1.3 Mutace RSM 37
6.1.4 Mutace Throas 38
6.1.5 Mutace PSM 39
6.1.6 Mutace SSM 39
6.1.7 Porovnání vlivu operátorů mutace 40
6.2 Operátory křížení 44
6.2.1 Jednobodové křížení - 1P 45
6.2.2 Cycle crossover - CX 46
6.2.3 Position-based crossover - POS 47
6.2.4 Ordered crossover - OX 48
6.2.5 Partially mapped crossover - PMX 49
6.2.6 Porovnání vlivu rekombinačních operátorů 50

7 Závěr **53**

A Příloha

1 Úvod

Genetické algoritmy jsou heuristické metody inspirované Darwinovou evoluční teorií. Využívají procesy evoluci vlastní jako například křížení, mutaci nebo přežití silnějšího. Typicky se využívají pro nalezení dobré aproximace řešení NP-úplných úloh, pro které není znám postup, který by našel optimální řešení v polynomiálním čase. Genetické algoritmy pracují s celou množinou přípustných řešení, které různě kombinují, aby tak nalezly lepší řešení. V první kapitole si nejprve vysvětlíme základní fakta o úloze obchodního cestujícího, na které budeme testovat chování genetických algoritmů, jejichž principy si vysvětlíme v kapitole následující. Ve třetí kapitole se detailně budeme věnovat parametrům genetických algoritmů jako je velikost populace, míra elitismu a míra mutace a pokusíme se pro ně najít optimální hodnoty. V poslední kapitole se zaměříme na různé druhy operátorů mutace a křížení, otestujeme jejich vliv na chování algoritmu v různých úlohách a vyhodnotíme, které z nich mají nejlepší vliv na schopnost algoritmu konvergovat k optimu.

2 Základní definice

I přesto, že se práce zabývá řešením složité úlohy obchodního cestujícího, předpokládají se u čtenáře pouze základní znalosti z oblasti teorie grafů a pravděpodobnosti. Ty nejpodstatnější pro zkoumanou problematiku jsou uvedeny v této kapitole. Definice jsou převážně převzaty ze skript pro předmět "Diskrétní matematika"[1]

Definice 2.1 [1]. *Graf G je dvojice $G = (V, E)$, kde V je konečná množina a $E \subset \binom{V}{2}$, přičemž*

$$\binom{V}{2} = \{\{x, y\} : x, y \in V a\, x \neq y\}$$

je množina všech dvouprvkových množin prvků množiny V. Prvky množiny V nazýváme vrcholy (často také uzly), prvky množiny E pak hrany grafu G. Vrcholy $x, y \in V$ jsou sousední, pokud $\{x, y\} \in E$. Pokud je E množina uspořádaných dvojic ($\{x, y\} \neq \{y, x\}$), mluvíme o orientovaném grafu, pokud se jedná o množinu neuspořádaných dvojic ($\{x, y\} = \{y, x\}$), jedná se o neorientovaný graf.

Definice 2.2 [1]. *Nechť n je přirozené číslo a označme $[n] = \{1, ..., n\}$. Dále definovaný graf má množinu vrcholů $[n]$. Úplný graf na n vrcholech se značí K_n a jeho množina hran obsahuje všechny neuspořádané dvojice prvků $[n]$, takže*

$$V(K_n) = [n]$$
$$E(K_n) = \binom{[n]}{2}$$

Definice 2.3 [1]. *Sled (z vrcholu u do vrcholu v) v grafu G je posloupnost ($u = v_0, v_1, ..., v_k = v$), kde v_i jsou vrcholy grafu G. Číslo k je délka tohoto sledu. Říkáme, že sled prochází vrcholy $v_0, ..., v_k$ nebo že na něm tyto vrcholy leží.*

Definice 2.4 [1]. *Uzavřený sled v grafu G je sled $(v_0, ..., v_k)$, ve kterém platí $v_0 = v_k$. Kružnice v grafu G je uzavřený sled délky alespoň 3, ve kterém se vrchol v_0 objevuje právě dvakrát a každý ostatní vrchol grafu nejvýše jednou. Číslo k je délka dané kružnice. Kružnice, která prochází všemi vrcholy grafu, se nazývá hamiltonovská, a graf obsahující nějakou takovou kružnici je hamiltonovský graf.*

Definice 2.5 [1]. *Ohodnocený neorientovaný graf* (G, w) *je neorientovaný graf* G *spolu s reálnou funkcí* $w : E(G) \to (0, \infty)$. *Je-li hrana* e *grafu* G, *číslo* $w(e)$ *se nazývá její ohodnocení nebo váha.*

Definice 2.6 [1]. *Vážená matice sousednosti ohodnoceného neorientovaného grafu* (G, w) *s vrcholy* $v_1, ..., v_n$ *je matice* $W(G) = w_{ij}$, *kde*

$$w_{ij} = \begin{cases} w(v_i v_j) & \text{pokud } v_i v_j \in E(G), \\ 0 & \text{jinak}, \end{cases}$$

pro $i, j = 1, ..., n$.
Z definice je možné odvodit, že pro neorientovaný úplný graf bude matice $W(G)$ *čtvercová, symetrická a pozitivně semidefinitní.*

3 Úloha obchodního cestujícího

V úvodní kapitole se budeme věnovat *úloze obchodního cestujícího*. Stručně si uvedeme její principy a ukážeme si řešení problému na primitivním příkladu.

Úloha obchodního cestujícího je obtížný kombinatorický problém z teorie grafů a řadí se mezi *NP-úplné* úlohy. Definovat lze například takto: *Nalezněte nejkratší hamiltonovskou kružnici ve zkoumaném úplném ohodnoceném grafu.* Obecně je problém možné řešit na libovolném úplném ohodnoceném grafu, pro potřeby této práce se ale omezíme pouze na graf, který je úplný, neorientovaný a jehož matice sousednosti je symetrická. Vzdálenosti mezi vrcholy rovněž splňují trojúhelníkovou nerovnost.

Pro velmi malé počty vrcholů se jako vhodný postup nabízí řešení hrubou silou. Při zvyšujícím se počtu vrcholů ale stoupá počet možných hamiltonovských kružnic v grafu a narůstá do značných rozměrů. Podívejme se na složitost této úlohy. Uvažujme nyní graf, který má právě n vrcholů. Protože je námi zkoumaný graf úplný, existuje vždy z každého vrcholu v_i hrana do libovolného jiného vrcholu v_j. Začněme naši cestu ve vrcholu v_0. Podle předchozí úvahy existuje $n-1$ možností, kam se vydat v příštím kroku. Předpokládejme nyní, že jsme v následujícím kroku skončili ve vrcholu v_1. Z tohoto vrcholu opět vede $n-1$ hran, právě jednu z nich ale nelze použít, protože vede zpět do vrcholu v_0. Zbývá nám pouze $n-2$ možností. Takto můžeme pokračovat do té doby, dokud nevyčerpáme všechny vrcholy. Po návštěvě posledního vrcholu se vracíme do vrcholu počátečního (v našem příkladu se jedná o v_0) a dokončíme tak cestu grafem, která splňuje podmínky pro hamiltonovskou kružnici. Jednoduchou úvahou lze snadno odvodit, že složitost této úlohy je $O(n!)$. Formální zápis pro počet možných řešení je následující:

$$N_R(n) = n! \tag{1}$$

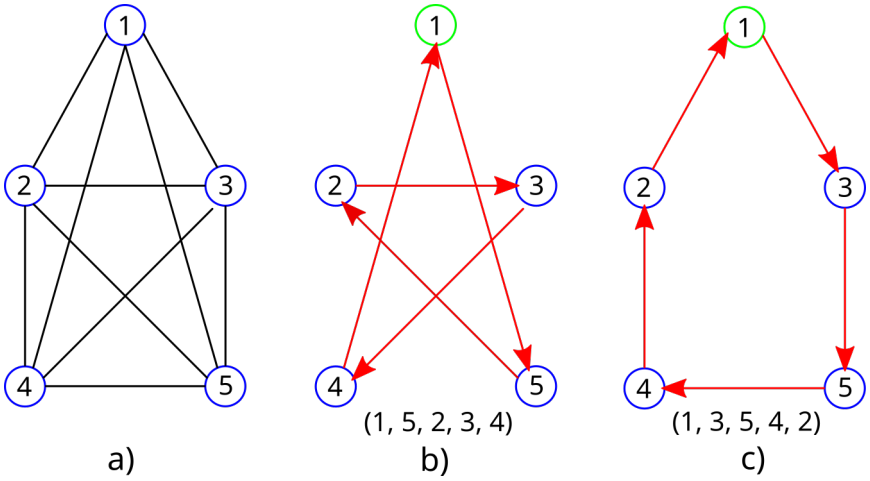
Uvažujme nyní případ s 20 městy. Počet různých cest, kterými lze města projít, je roven číslu $20! \approx 2.43 \times 10^{18}$. Je poměrně zřejmé, že řešit takovýto problém pomocí hrubé síly již není moudré. Pokud by vyhodnocení jedné z možných permutací trvalo $50\mu s$, což je podle simulací rozumný odhad, trvalo by vyhodnocení takovéto úlohy 92.5 milionu let. Pro větší počty vrcholů je proto nutné uvažovat sofistikovanější metody než je prosté řešení hrubou silou.

V současnosti neexistuje algoritmus, který by byl schopen v polynomiálním čase najít optimální řešení. Existují však algoritmy složitosti $O(n^2)$, které naleznou v praxi použitelná řešení. Dobrým příkladem může být např. algoritmus uvedený v učebních textech pro předmět "Teorie grafů a diskrétní optimalizace 2"[2]. Řešení dosažené tímto algoritmem lze považovat za polynomiální 2-aproximaci úlohy obchodního cestujícího, neboli dosažené řešení má v nejhorším možném případě 2x větší váhu než optimum. Dalším vhodným případem je Christofidův algoritmus [3]. Tento algoritmus poskytuje lepší odhad než předchozí 2-aproximace. Nalezené řešení

má maximálně 1,5x větší váhu než optimum. Nevýhodou této 1,5-aproximace oproti výše zmíněné 2-aproximaci je však větší výpočetní náročnost $O(n^3)$.

Uveďme si nyní velice jednoduchý příklad úlohy obchodního cestujícího, která bude řešena v průběhu práce.

Obrázek 1: Úloha obchodního cestujícího s 5 vrcholy



Na obrázku 1 můžeme vidět typickou ukázku úlohy obchodního cestujícího. Obecně platí, že úlohu má smysl řešit pouze pro ohodnocené grafy. V ukázce nejsou kvůli přehlednosti zobrazeny váhy jednotlivých hran, nicméně předpokládáme, že graf se nachází v rovině a platí zde eukleidovská metrika. Případ *a)* zobrazuje úplný graf K5. Povšimněme si, že z každého vrcholu vede hrana do všech ostatních vrcholů, což je jeden z předpokladů pro tuto úlohu. Naším cílem je najít nejkratší hamiltonovskou kružnici, neboli projít celý graf, navštívit každý vrchol právě jednou a urazit při tom co nejkratší cestu. Jedinou výjimkou je počáteční vrchol, ve kterém zároveň končíme. V reprezentaci cesty ale můžeme návrat do počátečního vrcholu vynechat. Případ *b)* ukazuje jedno z možných řešení. Posloupnost vrcholů pod obrázkem reprezentuje cestu, kterou jsme v grafu prošli. Pokud vezmeme v úvahu zmíněnou metriku, je na první pohled zřejmé, že existuje lepší řešení. Část *c)* zobrazuje optimální řešení.

Úloha obchodního cestujícího má v praxi mnoho využití. Typickým příkladem je logistika. Pokud je například nutné navštívit strategická místa na mapě kvůli zásobování, je rozumné navštívit tato místa v takovém sledu, který by ušetřil cenu paliva nebo času. Vrcholy v grafu poté reprezentují jednotlivá města nebo místa, hrany představují cesty mezi nimi. Hrany v tomto případě nemusí být ohodnoceny pouze na základě vzdálenosti mezi městy, dalším faktorem může být například čas potřebný k překonání vzdálenosti nebo spotřeba paliva. Všechny tyto faktory a mnohé další pak přispívají k celkovému ohodnocení hrany. V reálném případě nemusí existovat spojení mezi libovolnými městy, což lze řešit například přiřazením

velmi vysoké váhy konkrétní hraně. Sledy obsahující tyto hrany mají oproti ostatním sledům velmi vysoký součet vzdáleností a nejsou dále uvažovány jako vhodná řešení.

Úloha obchodního cestujícího má své využití i v astronomii. Pokud je potřeba se v určitém místě na obloze zaměřit na více cílů, je dobré natáčet teleskop optimálním způsobem, především kvůli ušetření času. Další významné uplatnění bychom nalezli při výrobě mikročipů, plánování nebo sekvenování DNA.

4 Genetické algoritmy

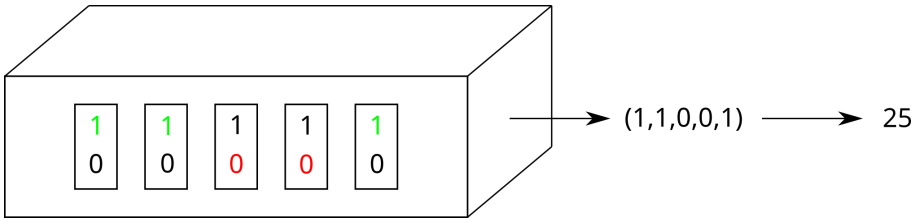
V této kapitole se budeme věnovat *genetickým algoritmům*. Stručně si vysvětlíme jejich vnitřní mechanismy a vysvětlíme si principy, na kterých jsou založeny. Na konec demonstrujeme jejich funkci na jednoduchých příkladech.

Genetické algoritmy (také známé jako evoluční algoritmy), jak již samotný název napovídá, imitují přírodní procesy. Darwinova evoluční teorie stručně řečeno tvrdí, že v dané populaci spíše přežijí silnější, lépe adaptovaní jedinci, než ti slabší, hůře přizpůsobení. Stejný princip adaptace se využívá u řešení úloh pomocí genetických algoritmů. Různou kombinací řešení dochází při správném chodu algoritmu ke zpřesnění odhadu a postupné optimalizaci.

Pro pochopení činnosti genetických algoritmů je klíčové si osvojit některé pojmy, které budou v práci později použity. Abychom zároveň nastínili podobu některých prvků a průběh operací, ukážeme si jednoduchý příklad převzatý od Goldberga [4].

Představme si nyní následující situaci. Máme k dispozici krabičku s pěti spínači. Každý spínač může být pouze v jednom ze stavů zapnuto a vypnuto. Pozice spínače v krabičce koresponduje s váhou bitu v binární reprezentaci a jeho stav s hodnotou daného bitu. Pokud je spínač sepnutý, má příslušný bit hodnotu 1, pokud není sepnutý, má bit hodnotu 0. Výstup naší krabičky je číslo v binární soustavě, které je vhodné převést do dekadické soustavy pro lepší čitelnost. Dejme tomu, že naším úkolem je maximalizovat hodnotu funkce $f(x) = x^2$, kde x je výše zmíněný výstup. Následující odstavce představí jednotlivé kroky genetických algoritmů a demonstrují jejich průběh na tomto příkladu.

Obrázek 2: Krabička se spínači



4.1 Chromozom

Chromozom neboli jedinec je nejjednodušší stavební kámen genetických algoritmů. Sám o sobě představuje jedno z možných řešení problému. Ve většině případů je vhodné chromozom reprezentovat jako posloupnost bitů, případně znaků. Význam jednotlivých bitů však striktně závisí na zkoumané úloze a na implementaci návrháře. Pro výše zmíněnou úlohu se spínačem by vhodná reprezentace mohla vypadat takto:

$$C_1 = (1, 1, 0, 0, 1)$$

Každý bit reprezentuje aktuální nastavení korespondujícího spínače. Pokud bychom tento řetězec považovali za pravý biologický chromozom, použili bychom místo výrazu bit označení *gen*. Pozice genu v řetězci se nazývá *lokus*, v této práci se nicméně omezíme na prostý výraz pozice. Forma konkrétního genu se nazývá *alela*, v práci budeme tento výraz volně zaměňovat za označení hodnota. Další příklady chromozomů pro danou úlohu mohou vypadat například následovně:

$$C_2 = (0, 1, 0, 1, 1), C_3 = (0, 0, 0, 0, 1), C_4 = (1, 0, 0, 0, 0)$$

Všimněme si, že všechny chromozomy mají stejnou délku a že všechny představují konkrétní řešení problému. Během chodu algoritmu nesmí dojít k takové změně, která by z chromozomu vytvořila nevyhovující řešení, např.: $C_N = (2, 1, 0, 0, 1)$. Hodnota 2 na pozici 1 v chromozomu C_N nedává v úloze smysl, protože spínače mohou nabývat pouze hodnot $(0, 1)$. Při návrhu jednotlivých částí algoritmu je klíčové použít pouze takové postupy, které garantují, že chromozom bude po jejich průběhu stále reprezentovat validní řešení problému.

Pro správný chod genetického algoritmu je potřeba začínat s množinou vyhovujících řešení. Tuto množinu nazýváme populace.

4.2 Populace

Populace představuje soubor chromozomů a je nezbytná pro další chod algoritmu. Pro předchozí zavedené chromozomy by populace vypadala například takto:

$$P_0 = \{C_1 = (1, 1, 0, 0, 1), C_2 = (0, 1, 0, 1, 1), C_3 = (0, 0, 0, 0, 1), C_4 = (1, 0, 0, 0, 0)\}$$

Je dobré si uvědomit, že v této reprezentaci nezáleží na pořadí chromozomů, pouze na jejich počtu. Jeden chromozom se v populaci může vyskytovat vícekrát. V tomto případě na něj hledíme jako na dva individuální případy. Ve standardním průběhu genetického algoritmu zůstává velikost populace neměnná. Mění se pouze chromozomy v ní obsažené. Vlivem změn během chodu algoritmu, které si vysvětlíme v dalších částech práce, může dojít k obměně některých jedinců v populaci. Po jedné iteraci může mít populace vlivem změn následující podobu:

$$P_1 = \{C_5 = (1, 1, 0, 0, 1), C_6 = (1, 1, 0, 1, 1), C_7 = (0, 0, 1, 0, 1), C_8 = (1, 0, 0, 0, 0)\}$$

Zaměřme se nyní na rozdíly v populaci. První chromozom zůstal beze změny. U druhého chromozomu došlo ke změně v hodnotě bitu na pozici nejvýznamnějšího bitu (pozice 1). U třetího chromozomu nastala změna na pozici 3. Poslední jedinec zůstal beze změny. Je proto patrné, že v průběhu jedné iterace mohou chromozomy změnit svoji podobu nebo mohou iterací projít beze změny.

4.3 Fitness funkce

V úvodu kapitoly jsme se zmínili, že genetické algoritmy využívají dobře adaptované jedince na řešení konkrétního problému. Abychom mohli jedince ohodnotit, potřebujeme tzv. *fitness funkci*. Fitness funkci lze obecně popsat jako zobrazení

$$F : C \to \Re, C \in SS,$$

kde $\Re$ je soubor reálných čísel a SS je stavový prostor úlohy, tj. množina všech možných vyhovujících řešení. C je prvek této množiny a reprezentuje jeden konkrétní chromozom. Úlohou fitness funkce je přiřadit každému jedinci reálné číslo. Hodnota čísla, které je přiřazeno chromozomu, se nazývá *fitness*. Po ohodnocení všech jedinců lze pomocí standardní metriky seřadit jedince podle velikosti jejich fitness. Obecně platí, že čím vyšší má jedinec hodnotu fitness, tím lepší je řešení, které reprezentuje. Pro náš příklad se nabízí použít fitness funkci ve tvaru $f(x) = x^2$, která je shodná s funkcí, jejíž hodnotu chceme maximalizovat. Nyní ilustrujeme použití fitness funkce na výše zmíněnou populaci P_0.

Tabulka 1: Hodnota fitness v úloze maximalizace funkce $f(x) = x^2$

Chromozom	**x**	$\boldsymbol{f(x) = x^2}$
$\boldsymbol{C_1 = (1,1,0,0,1)}$	25	625
$\boldsymbol{C_2 = (0,1,0,1,1)}$	11	121
$\boldsymbol{C_3 = (0,0,0,0,1)}$	1	1
$\boldsymbol{C_4 = (1,0,0,0,0)}$	16	256

Při pohledu na tabulku 1 je patrné, že některé chromozomy poskytují podstatně lepší řešení než jiné. Chromozom C_1 dosahuje řádově lepších výsledků než např. chromozom C_3. Pro další krok, kterým je selekce, je nejprve nutné jednotlivé chromozomy seřadit sestupně podle jejich hodnoty fitness. Pro zkoumanou populaci bychom je seřadili následovně:

$$\bar{P}_0 = (C_1 = (1,1,0,0,1), C_4 = (1,0,0,0,0), C_2 = (0,1,0,1,1), C_3 = (0,0,0,0,1))$$

Pro seřazenou populaci zavedeme nové označení $\bar{P}_0$. Zatímco k prvotní populaci P_0 můžeme přistupovat jako k množině, protože nezáleží na pořadí prvků, k seřazené populaci $\bar{P}_0$ je již nutné přistupovat jako k uspořádané n-tici.

4.4 Selekce

Bezprostředně po ohodnocení dochází k *selekci*. Proces selekce by mohl být považován za součást křížení, kterému se věnuje další odstavec, nicméně v této práci je pro větší přehlednost odloučíme. Při selekci dochází k výběru jedinců, kteří se budou účastnit procesu křížení. Způsobů, kterými můžeme vybírat jednotlivce pro další fázi algoritmu existuje mnoho, zatím si uvedeme pouze ty základní. Nejjednodušší proces selekce se nazývá *ruleta*. Pokud bychom použili analogii se skutečnou ruletou, má každý chromozom svůj slot v kole rulety. Všechny sloty jsou stejně velké a proto i pravděpodobnosti, že daný chromozom bude vybrán, mají stejnou hodnotu. Tento způsob je velice jednoduchý, nicméně v žádném ohledu nerespektuje fitness jedinců, kterou jsme počítali v minulém kroku. Zavádí se proto modifikace tohoto procesu, která se nazývá *vážená ruleta*. Velikosti jednotlivých slotů jsou proporcionálně rozděleny podle hodnoty fitness podle následující funkce:

$$P(C_i) = \frac{F(C_i)}{\sum_{j=0}^{n} F(C_j)} \tag{2}$$

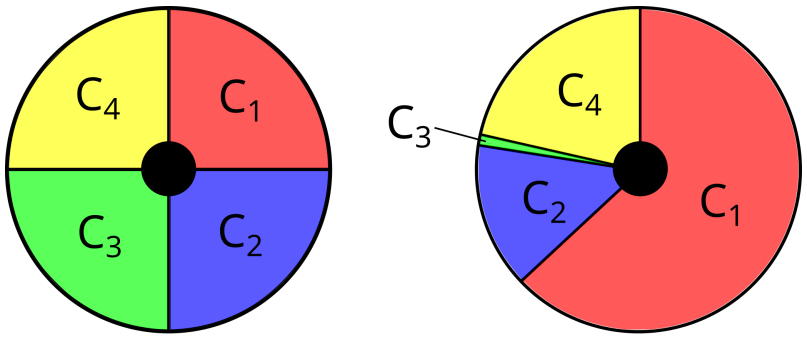
Při pohledu na vztah 2 je patrné, že součet všech pravděpodobností pro výběry jednotlivých chromozomů je roven 1. Pomocí této funkce jsou přiděleny relativní váhy každému chromozomu podle toho, jaký podíl tvoří fitness konkrétního chromozomu a celkový součet fitness všech chromozomů. Pro náš případ bychom dostali následující tabulku.

Tabulka 2: Relativní podíly fitness v úloze maximalizace funkce $f(x) = x^2$

Chromozom	$\boldsymbol{f(x) = x^2}$	**% z celku**
$\boldsymbol{C_1}$	625	62.3
$\boldsymbol{C_2}$	121	12.1
$\boldsymbol{C_3}$	1	0.1
$\boldsymbol{C_4}$	256	25.5

Čím vyšší má chromozom fitness, tím větší část zabírá jeho slot v kole rulety. Pravděpodobnost, že kulička přistane v příslušném slotu, tím pádem narůstá a chromozom má vyšší šanci účastnit se procesu křížení. Je důležité si uvědomit, že proces selekce (a vlastní genetický algoritmus) není deterministický, nýbrž stochastický. Díky pravděpodobnostnímu rozdělení můžeme očekávat jisté výsledky, může ale docházet i k neočekávaným krokům.

Obrázek 3: Znázornění velikosti slotů s ohledem na hodnotu fitness v tabulce 2



Proces selekce využívající váženou ruletu ilustrovanou na obrázku 3 b) dosahuje poměrně dobrých výsledků a je dostatečně účinný pro vyřešení jednodušších problémů. Nevýhodou tohoto postupu je riziko příliš častých výběrů několika nejlepších chromozomů, což může vést k předčasné konvergenci. V našem příkladu by se jednalo o chromozom C_1. Tomuto problému je možné se vyhnout použitím sofistikovanějšího procesu výběru, který se nazývá *turnajová selekce* a prokazatelně dosahuje lepších výsledků [6]. Při selekci je namísto roztočení kola rulety náhodně vybráno k chromozomů z populace, kde k je tzv. *velikost turnaje*. V této práci budeme uvažovat variantu turnajové selekce nazývanou deterministická turnajová selekce. Během turnaje se porovnává fitness všech zúčastněných jedinců a vybírá se ten nejlepší. Jiné postupy uvažují pravděpodobnost výběru libovolného jedince účastnícího se turnaje např. podle následujícího předpisu ($k \geq 3$):

$$\begin{aligned} P(C_1) &= p \\ P(C_2) &= p(1-p) \\ P(C_3) &= p(1-p)^2 \\ &\vdots \\ P(C_k) &= 1 - p(1 + (1-p) + \ldots + (1-p)^{k-2}) \end{aligned}$$

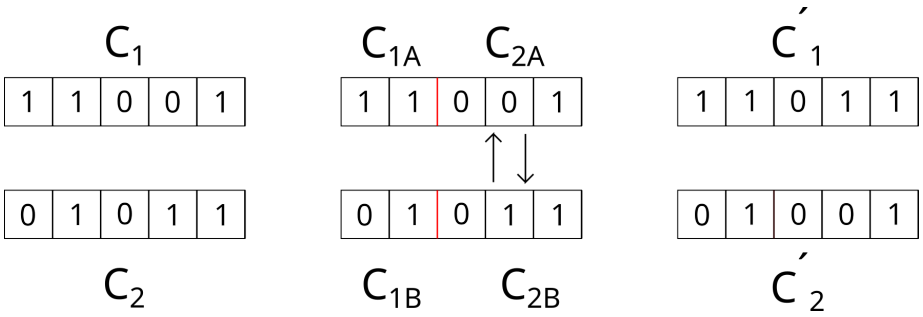
V průběhu let se pro parametr k zkoumaly různé hodnoty. Běžný postup je výběr parametru k z intervalu $[2, 5]$. Miller s Goldbergem ve své práci [15] popisují vliv zvyšující se hodnoty parametru k na chod algoritmu. Vyšší hodnoty vedou na zpřísnění tzv. *selekčního tlaku*, což může mít dva následky. Obecně vzato vyšší selekční tlak vede ke zkvalitnění výsledného řešení, na druhou stranu ale způsobuje častější uvíznutí algoritmu v lokálních minimech, protože pro větší k je pravděpodobnost účasti chromozomů s nízkou fitness v procesu křížení prakticky nulová. Volba menší hodnoty v intervalu kolem $k \in [2, 5]$ zlepšuje chování algoritmu, není příliš náročná na výpočetní čas [7] a nevytváří příliš velký selekční tlak. Pro potřeby této práce

budeme respektovat ověřené hodnoty a vybírat parametr k z intervalu $[2, 5]$.

4.5 Křížení

Po selekci jedinců dochází k procesu *křížení*, který má největší vliv na podobu jedinců v další iteraci. Při křížení dochází k výměně jedinců původní populace, rodičů, za jedince z populace nové, děti. K procesu dochází v jednotlivých krocích. V prvním kroku jsou vybráni dva jedinci dle předchozího postupu selekce. Tyto chromozomy, nazývané mateřské chromozomy nebo rodiče, jsou posléze rekombinovány pomocí stochastických pravidel, která nazýváme *rekombinační operátory*. Přesnějšímu popisu těchto operátorů se budeme věnovat v pozdějších kapitolách, nyní si uvedeme pouze nejjednodušší případ tzv. *jednobodového křížení*. Při jednobodovém křížení dochází k výběru jednoho bodu uvnitř chromozomu, který určuje, kde dojde k rozdělení chromozomu na dvě části C_{1A} a C_{1B}. Ke stejnému rozdělení na C_{2A} a C_{2B} dochází i u druhého chromozomu na totožném místě. Část C_{1A} je poté spojena (rekombinována) s částí C_{2B}. Ke stejnému procesu dojde s fragmenty C_{1B} a C_{2A}. Ze dvou původních chromozomů se tak opět stávají dva chromozomy C_1' a C_2', nazývané jako potomci, děti nebo dceřiné chromozomy. Pro lepší pochopení je proces ilustrován na následujícím obrázku za použití dříve definovaných chromozomů C_1 a C_2.

Obrázek 4: Jednobodové křížení



Tento proces se opakuje pro všechny další dvojice chromozomů, které byly vybrány v průběhu selekce, dokud nemá nová populace stejnou velikost jako populace původní. Po skončení křížení sestává populace z nových jedinců, jejichž fitness již není známa. Nová populace proto není seřazena.

Jednobodové křížení je poměrně jednoduché na implementaci a u chromozomů s krátkou délkou je jeho použití dostačující. Existují ale i procesy křížení, kdy dochází k rozdělení chromozomu na fragmenty na více místech. Obecně se označují jako *k-bodová křížení*. Mnoho autorů se v minulosti zabývalo problematikou rozdělení chromozomu ve více bodech. De Jong se ve své práci [8] k-bodovému křížení věnuje a ukazuje, že pro zvyšující se hodnoty k dochází k přílišnému narušení stavby

jednotlivých chromozomů a přenos důležité informace na další generaci je narušen.

U procesu křížení se jako dobrá metoda ukazuje postup, kdy někteří jedinci přecházejí do další populace bez účasti v procesu křížení. V následující populaci se tak vyskytují chromozomy z populace předchozí, aniž by prošly jakoukoliv změnou. Tomuto přístupu se říká *elitismus* a budeme se mu detailně věnovat v dalších kapitolách.

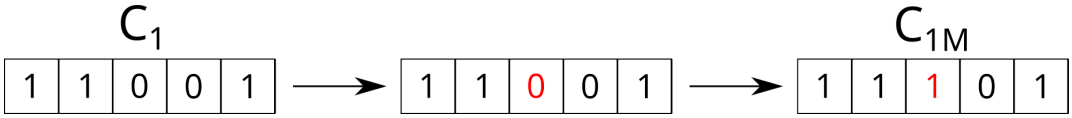
U křížení existují různé přístupy, které modifikují jeho funkci. Jeden ze základních rozdílů v těchto přístupech je pravděpodobnost P_K, že po vybrání dvou jedinců pro křížení k němu skutečně dojde. V této práci budeme uvažovat $P_K = 1$, jinými slovy ke křížení dojde vždy. Odlišné přístupy mohou uvažovat $P_K < 1$. Hodnoty $P_K < 1$ mají obecně negativní vliv na rychlost konvergence algoritmu a musí se kompenzovat dalšími mechanismy. Mísení jednotlivých genotypů je zmírněno, čímž dochází k pomalejšímu prohledávání stavového prostoru úlohy. Tento nedostatek lze eliminovat použitím dodatečných přístupů, kterým se ale v této práci nebudeme příliš věnovat.

V příkladu na obrázku 4 se křížení účastní dva rodičovské chromozomy a výslední potomci jsou také dva. V jiných variacích procesu křížení nemusí být rodiče ani potomci vždy v párech. Edmondson ve své práci [11] popisuje přístup, kdy se procesu křížení účastní 3 rodičovské chromozomy, které plodí 6 dceřiných jedinců. Různými postupy jsou poté vybrány tři chromozomy, které přecházejí do další generace, aby byla zachována stálá velikost populace. Jiný přístup, který jsme zvolili i této práci, uvažuje dva rodičovské chromozomy a vznik pouze jednoho jedince. Druhému dceřinému chromozomu, který by vznikl podle ilustrace na obrázku 4, není umožněn přesun do budoucí generace.

4.6 Mutace

Posledním klíčovým operátorem v průběhu chodu genetického algoritmu je proces *mutace*. Stejně jako v přírodě, kdy dochází ke změnám v chromozomu organizmu kvůli radiaci, toxinům nebo jiným vlivům, dochází i v umělém chromozomu k občasné mutaci. Nejjednodušší případ mutace, který je vhodný pro náš příklad, je změna hodnoty bitu. Pokud má bit na začátku hodnotu 1 a na příslušném genu dojde k mutaci, změní se hodnota bitu na 0 a naopak. Populace se prochází jedinec po jedinci a u každého se rozhoduje, zda dojde k mutaci. Pokud má dojít k mutaci, náhodně se vybere pozice, ve které dojde ke změně bitové hodnoty.

Obrázek 5: Bitová mutace



Proces se opakuje pro všechny jedince, ke změně ale dochází pouze v malém zlomku chromozomů. Po skončení této operace je dokončena jedna iterace algoritmu, která je také nazývaná *generace*. Pokud byla splněna zastavovací podmínka, typicky nalezení jedince s ohodnocením lepším než určitá mez nebo překročení maximálního povoleného počtu generací, algoritmus končí svůj chod. V opačném případě dochází k návratu do kroku, kde se provede ohodnocení chromozomů pomocí fitness funkce, a cyklus se opakuje.

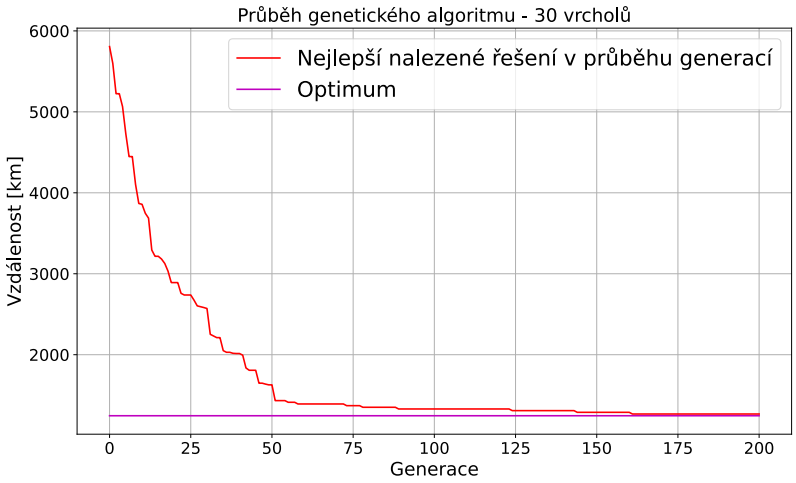
4.7 Průběh algoritmu

Po seznámení s jednotlivými součástmi a procesy algoritmu je vhodné si uvést jeho průběh v několika krocích.

1. Inicializace prvotní populace - náhodný výběr N chromozomů ze stavového prostoru
2. Ohodnocení jedinců pomocí fitness funkce, sestupné seřazení podle fitness
3. Selekce jedinců pro křížení podle zvoleného kritéria (turnaj, vážená ruleta)
4. Křížení, mutace
5. Pokud je splněna zastavovací podmínka, konec, jinak zpět na 2.

Vykonání všech výše zmíněných kroků právě jednou představuje jednu iteraci neboli generaci. Počet generací budeme v této práci značit g. Genetické algoritmy potřebují ke svému běhu typicky stovky generací. Při nižším počtu ($g < 100$) nemá algoritmus dostatek času na to, aby prozkoumal postačující počet bodů stavového prostoru, které jsou dostupné díky rozmanitosti genotypu v chromozomech. Naproti tomu pro vyšší počty generací ($g \doteq 200$) dochází ke zmenšení genetické rozmanitosti z důvodu vymizení chromozomů s nedostatečnou fitness a jejich nahrazením kvalitnějšími, ale mezi sebou příbuznými jedinci.

Obrázek 6: Typický průběh genetického algoritmu - 30 vrcholů, $N = 200$



Graf 6 zobrazuje postupné vylepšování odhadu minima nejkratší možné cesty grafem. V prvních generacích můžeme vidět strmý pokles, ke kterému dochází díky mísení genotypů chromozomů, které nejsou na začátku běhu algoritmu příbuzné. Postupným křížením chromozomů ale dochází ke ztrátě genetické rozmanitosti a proces konvergence se zpomaluje. Po uplynutí 100. generace můžeme pozorovat stagnaci na jedné hodnotě po několik generací. Z tohoto lokálního minima algoritmus typicky vysvobodí náhodná mutace.

4.8 Rozdíly oproti ostatním přístupům

Po seznámení s jednotlivými procesy, které se odehrávají uvnitř genetického algoritmu můžeme porovnat výhody, nevýhody a rozdíly s jinými standardními přístupy. Uvedeme si také, proč je vhodné řešit úlohu obchodního cestujícího pomocí genetických algoritmů. Ze všeho nejdříve se zaměříme na to, do které rodiny prohledávacích algoritmů patří algoritmy genetické. Prohledávací algoritmy bychom mohli rozdělit do tří skupin. Typickým příkladem první skupiny, vyčerpávajícího vyhledávání, je prohledávání hrubou silou, o kterém jsme se již zmínili. Tento typ prohledávání je vhodný pouze pro jednoduché úlohy s malým počtem možných řešení. Dalším typem je prohledávání založené na matematické analýze. Pro optimalizace kriteriální funkce se typicky používá derivace této funkce a následná úprava aktuálního řešení pomocí informace získané z gradientu. Metody využívající diferenciální počet byly v minulosti důkladně zkoumány a jsou dnes hojně využívány ve všech oblastech optimalizace. Poslední rodina prohledávacích algoritmů, do které spadají i genetické algoritmy, je založena na náhodném prohledávání. Algoritmy z této skupiny pracují s řešením nebo řešeními, která jsou nejprve získána náhodně. Iterativními úpravami poté dochází k postupnému zlepšování řešení, dokud není nalezeno vyhovující.

Výhodou těchto metod je fakt, že ke svému chodu nepotřebují gradient kriteriální funkce, lze je proto nasadit i na problémy s nespojitými nebo nediferencovatelnými kriteriálními funkcemi.

Nespornou výhodou genetických algoritmů je jejich robustnost. V této práci se věnujeme řešení úlohy obchodního cestujícího, nicméně se s jejich pomocí dá řešit množství jiných úloh, často s diametrálně odlišnými stavovými prostory. Pokud je potřeba genetický algoritmus použít na jiný typ úlohy, je nutné pozměnit strukturu chromozomu tak, aby reprezentoval jeden bod stavového prostoru v konkrétní úloze. Někdy může být nezbytné upravit i metody pro křížení a mutaci takovým způsobem, aby byly kompatibilní s novou podobou chromozomu. Ostatní procesy a principy však zůstávají neměnné a účinně pracují stejným způsobem pro různé úlohy.

Další výhodou genetických algoritmů je stochastický přechod mezi stavy. Zatímco u deterministických přechodů by získané řešení bylo pro zafixované vstupní parametry vždy stejné, genetické algoritmy poskytují různá řešení. Stochastický přístup zde umožňuje výskyt jevů, které by se na první pohled zdály nelogické nebo nepravděpodobné, mohly by ale vést k důležitým změnám v celém procesu a dosažení lepšího řešení.

Jeden z často zmiňovaných kladů genetických algoritmů oproti postupům, které využívají matematickou analýzu a derivace kriteriální funkce, je simultánní prohledávání více bodů stavového prostoru. Obvyklé metody pracují pouze s jedním bodem stavového prostoru reprezentujícím odhad hledaného řešení a postupně ho zpřesňují. Genetické algoritmy pracují s celou populací jedinců, kde každý z nich reprezentuje jedno validní řešení. Po skončení chodu algoritmu obsahuje poslední populace řadu jedinců, přičemž více z nich může představovat dostatečně kvalitní, ale odlišná řešení.

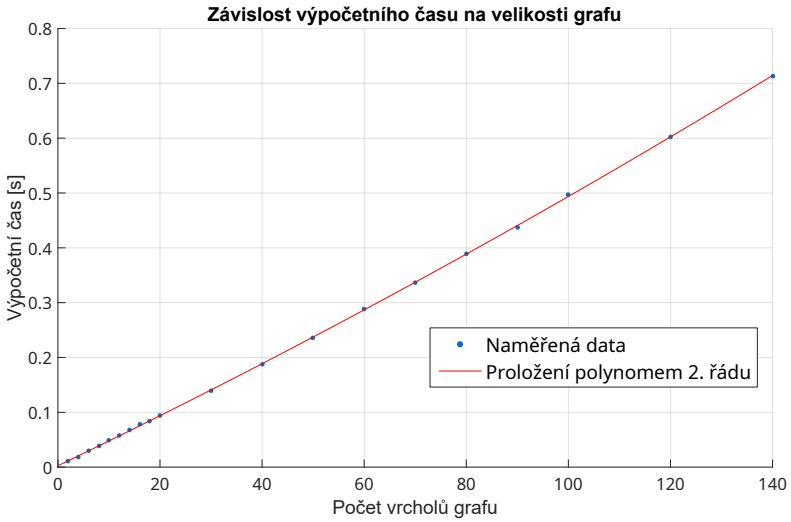
Paralelizace je další předností genetických algoritmů. Pokud řešíme úlohu, která je složitá, ale lze rozdělit do jednodušších částí, můžeme každý fragment úlohy řešit samostatně. Rozdělením na jednotlivé části se podstatně zmenší stavový prostor a je snazší nalézt optimální řešení pro podúlohy. Kombinací těchto částečných řešení je pak možné nalézt optimální řešení pro kompletní úlohu.

Genetické algoritmy mají na druhou stranu i své stinné stránky. Pro svůj chod potřebují jasně definovanou fitness funkci. Sestrojení takové funkce, která by byla vhodná pro danou úlohu, může být často velmi nelehký úkol. Fitness funkce musí standardně uvažovat více faktorů důležitých pro konkrétní problém. Identifikace faktorů, jejich vliv na řešení úlohy a zohlednění v rámci fitness funkce je kritické pro úspěšné řešení úlohy.

Nevýhodou je i velký nárůst výpočetního času potřebného k vyřešení složitějších úloh. Je důležité si uvědomit, že teoretická závislost výpočetní složitosti na velikosti

chromozomu je pouze lineární $O(n)$. Pokusy ukazují, že naměřená závislost odpovídá spíše polynomu 2. řádu, nicméně lineární aproximace pro počty vrcholů $n < 500$, kterými se v této práci budeme zabývat, je téměř totožná s křivkou získanou proložením dat polynomem 2. řádu.

Obrázek 7: Výpočetní čas v závislosti na počtu vrcholů grafu, $N = 100$



Obrázek 7 zobrazuje nárůst doby potřebné k výpočtu pro rostoucí počet vrcholů. Data byla naměřena ze 100 realizací algoritmu, aby se průměrováním vyhladily náhodné vlivy, které mohly vzniknout uvnitř výpočetní soustavy. Již od pohledu je experimentálně získaná křivka ve zkoumaném intervalu téměř lineární funkce.

Úskalí ale spočívá v potřebě zvyšovat velikost populace nebo použít výpočetně náročné heuristiky pro složitější problémy. Pokud by nám například stačila populace o 50 jedincích pro vyřešení úlohy obchodního cestujícího s 20 vrcholy, nemusí nám stejná velikost populace nutně zaručit vyřešení úlohy se 30 vrcholy. Závislost výpočetní doby na velikosti populace je již kvadratická $O(n^2)$ a je detailněji prozkoumána v následujících kapitolách. V případě použití heuristik opět narůstá doba potřebná k výpočtu obecně s alespoň druhou mocninou velikosti populace.

Genetické algoritmy jsou obecně vhodné pro řešení NP-úplných problémů typu úlohy obchodního cestujícího, které by jinak ke svému vyřešení potřebovaly neúnosné množství výpočetního času. Genetické algoritmy na jednu stranu negarantují nalezení optimálního řešení, jsou ale schopné nalézt řešení blízké tomu optimálnímu v rozumném čase. Tato schopnost je dána především vlastností algoritmů analyzovat mnoho odlišných řešení najednou a možností je vhodně kombinovat. Z tohoto důvodu jsou genetické algoritmy jedním z často uvažovaných přístupů k řešení úlohy obchodního cestujícího.

V další kapitole si ukážeme důležité parametry genetických algoritmů a vliv těchto parametrů na jejich činnost. Vysvětlíme si také vliv parametrů na čas potřebný k dokončení úlohy. Nedílnou součástí volby vhodné hodnoty pro parametry bude kritérium doby výpočtu, bude proto nutné zvolit rozumný kompromis mezi výpočetní náročností a kvalitou nalezeného řešení.

5 Parametry genetických algoritmů

V této kapitole se budeme věnovat klíčovým parametrům, které mají významný vliv na správný chod genetického algoritmu. Každému parametru věnujeme vlastní kapitolu a dopodrobna si vysvětlíme, jaký mají na chod algoritmu vliv. Jeden z cílů této práce je pokusit se zjistit optimální hodnoty pro tyto parametry při řešení úlohy obchodního cestujícího. Uvedeme zde proto také experimenty, ze kterých lze vyvodit rozumné hodnoty pro zkoumané parametry.

5.1 Velikost populace

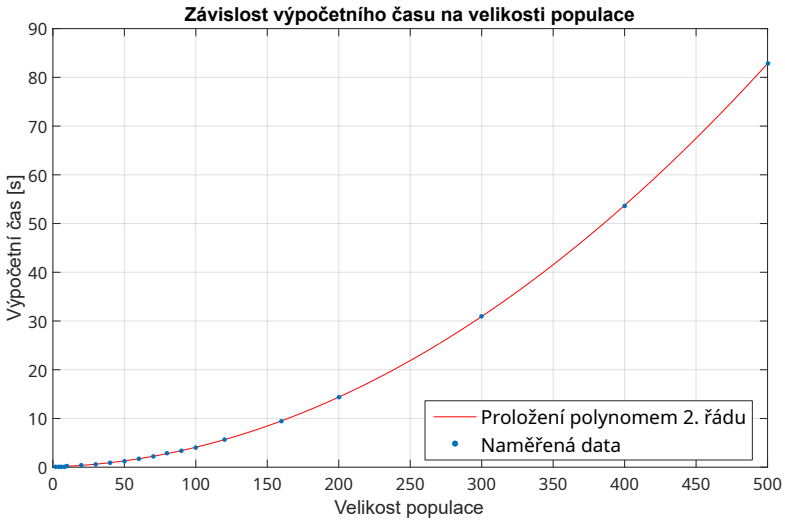
Jedním z nejdůležitějších parametrů pro chod algoritmu je velikost populace, se kterou algoritmus pracuje. V této práci budeme parametr značit jako N. Velikost populace představuje počet chromozomů, o kterých se uvažuje jako o možných řešeních. V průběhu chodu algoritmu se standardně uvažuje neměnný počet jedinců, i když to není pravidlo. Populace může ubývat nebo nabývat na velikosti, tento proces by v přírodním systému odpovídal migraci. V této práci nebudeme uvažovat proměnnou velikost populace, rozmanitost prvotní populace je proto nesmírně důležitá. Stejně jako v přírodě je i u genetických algoritmů potřeba zabezpečit bohatou genetickou diverzitu. Nízká rozmanitost má za následek předčasnou konvergenci algoritmu v lokálním minimu, ze kterého je možné uniknout pouze díky náhodné mutaci. Pokud bychom například uvažovali pouze dva jedince, došlo by v první iteraci k jejich zkřížení a každá další iterace by již pravděpodobně nevedla k podstatnému zlepšení, protože oba chromozomy by byly příbuzné a měly podobné znaky.

Při výběru N je potřeba si uvědomit, že ze všech ostatních parametrů a operátorů má velikost populace **největší vliv** na dobu chodu. Na tento fakt je třeba myslet při volbě hodnoty parametru, protože pro příliš velké populace narůstá výpočetní náročnost nad únosnou mez. Vetší populace zároveň nutně negarantuje zisk lepšího řešení. Volba by tedy měla reflektovat kompromis mezi schopností algoritmu najít zajímavé řešení a dobou potřebnou k jeho získání.

Obdobný vliv má na výpočetní čas také použití heuristik. Zapojením heuristiky do chodu algoritmu vzniká *hybridní (memetický) genetický algoritmus*, který obecně dosahuje lepších výsledků, protože využívá i lokální informaci o řešené úloze. V závislosti na použité heuristice významně stoupá výpočetní náročnost, nicméně v této práci se až na výjimky heuristikami nebudeme zabývat, protože jsou specifické pro řešené úlohy, nikoliv pro genetické algoritmy.

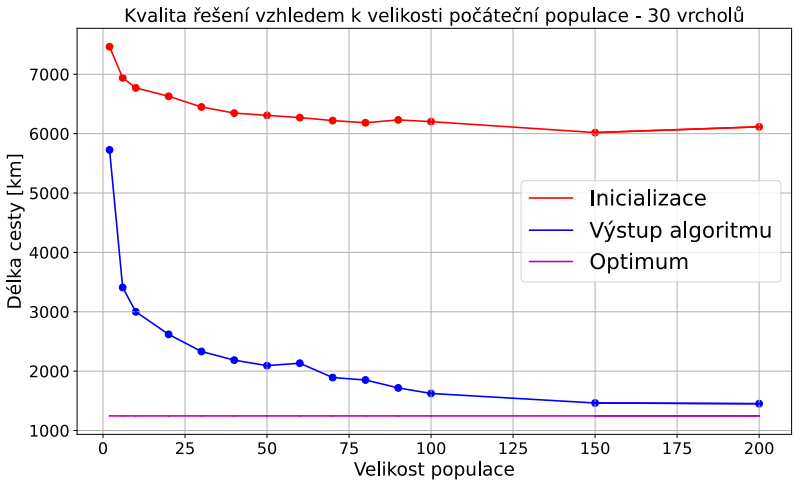
Zaměřme se nyní závislost mezi časem potřebným k výpočtu a velikostí populace. Následující obrázek ilustruje výpočetní náročnost algoritmu pro různé velikosti populace.

Obrázek 8: Výpočetní náročnost algoritmu pro různě velké populace, $g = 100$



Pokud proložíme získaná data křivkou, lze snadno ověřit, že výpočetní čas je přímo závislý na druhé mocnině N. Začátek kapitoly se zmiňuje o nutnosti populace obsahovat geneticky rozmanité jedince. Ukážeme si, že existuje rozumný odhad pro velikost populace, který ve většině případů poskytuje dobré řešení a není příliš výpočetně náročný. Všechny pokusy byly provedeny s náhodnou volbou počáteční populace a zafixovanými parametry. Výsledky byly následně zprůměrovány ze 100 realizací algoritmu pro každou hodnotu populace, aby se zmírnily vlivy stochastických jevů.

Obrázek 9: Graf se 30 vrcholy, 100 generací

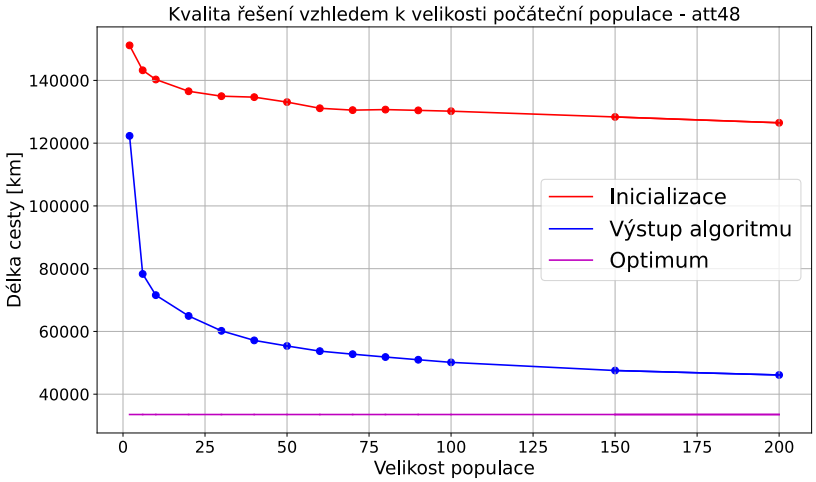


Podívejme se nyní na obrázek 9 zobrazující nejlepší řešení nalezené algoritmem

v závislosti na volbě velikosti počáteční populaci. Červená křivka reprezentuje fitness nejlepšího jedince v počáteční populace před začátkem chodu algoritmu. Povšimněme si, že bod reprezentující populaci o velikosti $N = 150$ má lepší ohodnocení než bod reprezentující větší populaci o velikosti $N = 200$. Tento jev je způsoben náhodným generováním jedinců do počáteční populace a není neobvyklý. Modrá křivka zobrazuje nejlepší řešení, kterého algoritmus dosáhl po skončení svého chodu. Rozdíl mezi délkou náhodného průchodu při inicializaci a nalezeného řešení není pro menší populace tak rozdílný, protože počáteční chromozomy nejsou dostatečně geneticky rozmanité. Pro větší populace se tento rozdíl zvětšuje z důvodu výskytu genotypů, které předtím byly dosažitelné pouze díky nepravděpodobné, náhodné mutaci. Fialová přímka představuje délku optimálního řešení. Je patrné, že zvyšování populace v intervalu $N \in [2; 100]$ podstatně vylepšuje kvalitu nalezeného řešení. Na intervalu $N \in [100; 200]$ stále dochází ke zlepšení, nýbrž zkrácení délky nalezeného řešení již není tak výrazné. Pro stále větší populace se algoritmus dostává blíže k optimu, zároveň ale narůstá výpočetní čas. V tomto konkrétním případě v úloze se 30 vrcholy lze volbou relativně velké populace $N \doteq 500$ dosáhnout optima (experimentálně ověřeno) nebo alespoň výsledku, jehož délka je pouze o jednotky % delší.

Ukažme si nyní jiný případ. Uvažujme graf, který místo 30 vrcholů obsahuje vrcholů 48. Jedná se o graf *att48* z volně dostupné knihovny zaměřené na úlohu obchodního cestujícího *TSPLIB* [13] poskytnuté univerzitou v Heidelbergu, jehož polohy vrcholů odpovídají skutečným geografickým polohám hlavních měst Spojených států amerických. V této práci budeme na tomto konkrétním grafu často vyhodnocovat simulace, protože odpovídá reálnému logistickému problému. Dle vzorce (1) v kapitole 3 je zřejmé, že množství možností, kterými lze projít graf att48, je nesrovnatelně vyšší než počet cest v grafu s 30 vrcholy.

Obrázek 10: Graf att48, 100 generací



Při pohledu na obrázek 10 je dobře patrné, že se zvyšující se populací dochází ke stejnému jevu jako na obrázku 9. Zajímavé je, že charakter poklesu křivky představující nejlepší řešení je i přes významné rozdíly v množství možných průchodů podobný. Pro menší populace dochází ke znatelnému zlepšení odhadu řešení a rychlost konvergence odhadu řešení k optimu se opět zpomaluje pro vyšší populace.

Při volbě velikosti populace se musíme zamyslet nad časovými nároky, které chceme splnit. Pokud je naším cílem najít co nejlepší řešení v krátkém čase, například v řádu vteřin, je dobré volit menší počáteční populaci o velikosti $N \in (100, 200)$. Pokud se náš časový plán pohybuje ve vyšších řádech, například hodin, můžeme si dovolit uvažovat populace o velikostech mnoha tisíců, která za cenu mnohonásobně delšího výpočetního času poskytne lepší řešení.

Velikost populace ale není jediný parametr, který ovlivňuje konvergenci algoritmu. V dalších kapitolách si ukážeme způsoby, jak podstatně zlepšit vlastnosti genetického algoritmu za cenu minimálního navýšení časové náročnosti.

5.2 Elite rate

Elitismus jsme předběžně zmínili již v kapitole 4.5. Pokud v našem genetickém algoritmu použijeme elitismus, uvažujeme přežití některých rodičovských chromozomů z původní populace do té následující. *Elite rate* určuje míru, s jakou elitismus v průběhu algoritmu uplatňuje svůj vliv. V této práci budeme elite rate značit jako E_R. V procesu křížení je v závislosti na hodnotě parametru E_R vybráno několik jedinců s nejlepší fitness a ti jsou beze změny zařazeni do následující populace. Protože požadujeme zachování velikosti populace v průběhu algoritmu, musí dojít k omezenému počtu opakování křížícího mechanismu. Po skončení procesu křížení pak nová populace sestává zároveň z nejlepších jedinců z předchozí generace a z jedinců, kteří vznikli procesem křížení.

Elitismus prokazatelně zlepšuje schopnost algoritmu dosáhnout kvalitnějších řešení [8]. Určité riziko však spočívá v tendenci elitismu způsobovat uvíznutí algoritmu v lokálních minimech. Představme si nyní případ, kdy má jeden z chromozomů v počáteční populaci i přes náhodný proces inicializace mnohem větší fitness než ostatní jedinci. Díky elitismu jistě přejde do další generace a zároveň existuje poměrně velká pravděpodobnost, že se účastní křížení. Ve většině případů projde tento jedinec a chromozomy s ním příbuzné do další generace beze změny díky nízké hodnotě míry mutace M_R, které se budeme detailně věnovat v další kapitole. V další generaci se pravděpodobně vyskytují chromozomy s vysokou fitness příbuzné s výše zmíněným chromozomem, které budou opět přeneseny do další generace díky elitismu a křížení. Z populace následně vymizí chromozomy s odlišným genotypem, který mohl být potenciálně důležitý. Vymizením genotypu dochází k ochuzení rozmanitosti populace a je pravděpodobnější, že algoritmus uvízne v lokálním minimu.

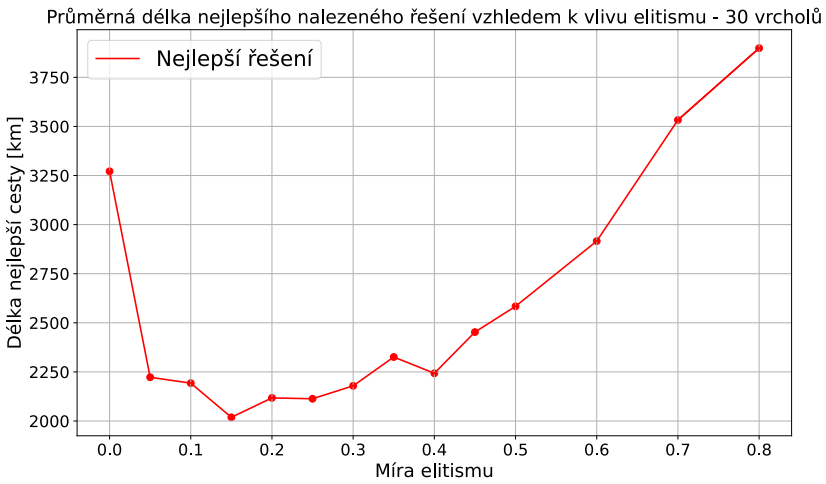
Jeden z postupů, který slouží ke zmírnění rizika uvíznutí v lokálním minimu způsobeného elitismem, je omezení životnosti jednotlivých chromozomů [14]. Tato metoda zabraňuje přežívání jedinců beze změny během příliš velkého počtu generací. Zavedením maximální délky života pro chromozomy zamezíme zablokování části populace starými, neměnnými chromozomy a podpoříme změny v genotypu pomocí operátoru křížení.

Pokud chceme určit optimální hodnotu pro E_R, musíme si nejprve uvědomit, že parametr představuje podíl populace. Z tohoto faktu logicky vyplývá, že hledaná hodnota bude ležet v intervalu $[0, 1]$. Příliš vysoké hodnoty (blízko 1) by znamenaly, že podstatnou část nové populace budou tvořit jedinci z té předchozí. Zůstává tak málo prostoru pro zdokonalení zbývajících jedinců pomocí křížení. Diverzita počáteční populace by v takovém případě ztratila svůj původní smysl, kterým je poskytnout dostatečný počet nepříbuzných chromozomů a umožnit tak prohledání více bodů stavového prostoru.

Pojďme nyní prozkoumat vliv hodnoty E_R na kvalitu nalezeného řešení. Nejprve se podíváme na stejný případ se 30 vrcholy jako v předchozí kapitole. Pro zmírnění vlivu kvalitní počáteční populace, byla její volba zafixována pro všechny simulace. Pro každou hodnotu elitismu bylo provedeno 100 simulací, jejichž výsledky byly poté zprůměrovány. Pokusy byly provedeny pro následující hodnoty parametru E_R:

$$E_R = \{0; 0,05; 0,1; 0,15; 0,2; 0,25; 0,3; 0,35; 0,4; 0,45; 0,5; 0,6; 0,7; 0,8\}$$

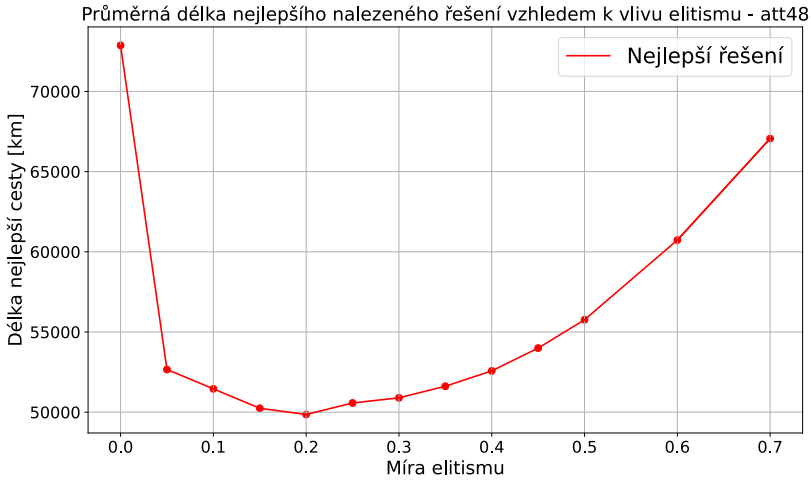
Obrázek 11: Úspěšnost algoritmu v řešení úlohy



Obrázek 11 ilustruje vliv míry elitismu na kvalitu nalezeného řešení. Povšimněme si strmého pádu v hodnotě nejlepšího nalezeného řešení již pro nejmenší testovanou hodnotu elitismu. Při zachování pouhých 5% populace dochází v tomto případě ke zlepšení odhadu nejkratší cesty o 30%. Pro zvyšující se hodnoty elitismu v intervalu $E_R \in [0,1; 0,3]$ dochází k dalšímu zlepšení, přičemž nejlepších výsledků dosáhl algoritmus s hodnotou $E_R = 0,15$. Pro velmi vysoké hodnoty elitismu $E_R > 0,7$ dochází k přílišnému ochuzení genotypu z důvodu kopírování příliš velké části populace. Operátor křížení potom nemá dostatek prostoru pro průzkum stavů úlohy a algoritmus uvízne v lokálním minimu. Připomeňme si, že elitismus nemá negativní vliv na výpočetní náročnost, ve skutečnosti je to právě naopak. Jistý počet jedinců, který proporcionálně odpovídá $E_R \times N$ je přímo zkopírován do další populace. Protože je část nové populace obsazena ještě před započetím procesu křížení, dochází ke sníženému počtu výpočetně náročnějších rekombinací chromozomů a algoritmus je tak paradoxně rychlejší.

Ukažme si nyní chování algoritmu v závislosti na různých hodnotách elitismu v úloze att48. Uvažované hodnoty parametru E_R jsou stejné jako v předchozím případě.

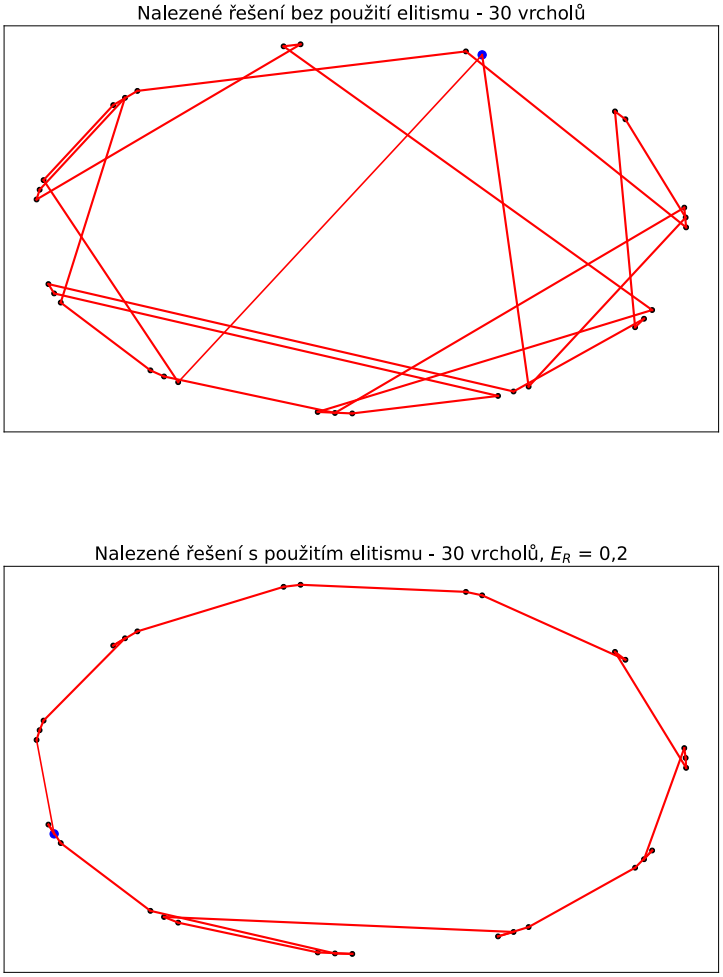
Obrázek 12: Úspěšnost algoritmu v řešení úlohy



Na obrázku 12 můžeme vidět velmi podobné chování jako na předchozím obrázku 11. Opět pozorujeme vysoké zlepšení kvality řešení pro minimální hodnoty parametru E_R. V tomto případě byla nejlepší řešení v průměru nalezena pro $E_R = 0,2$, nicméně všechny hodnoty E_R v intervalu $E_R \in [0.1; 0.4]$ poskytují velice dobré zpřesnění nejlepšího odhadu minima délky cesty. Pro vyšší hodnoty parametru opět pozorujeme zhoršení kvality řešení.

Pro lepší představu vlivu elitismu je vhodné zobrazit si nalezené řešení jako skutečnou cestu grafem, ne pouze jeho délku. Následující obrázky demonstrují kvalitu nalezené cesty.

Obrázek 13: Úspěšnost algoritmu v řešení úlohy, $N = 100, M_R = 0, g = 100$



Obrázek 13 znázorňuje účinek elitismu na výslednou podobu nalezeného řešení. Ostatní parametry algoritmu, velikost populace N a počet generací g, byly nastaveny na hodnoty $N = 100, g = 100$. Mutace nebyla prozatím uvažována, jejímu vlivu se budeme věnovat v další kapitole. Pro takovéto nastavení je pro algoritmus poměrně obtížné nalézt optimum, zejména kvůli nedostatečně velké populaci, která tak nezajistí uspokojivou genetickou rozmanitost. Hodnoty byly takto zvoleny, aby byl dobře viditelný vliv elitismu na chod algoritmu a byly minimalizovány dopady ostatních parametrů. Pro další pokusy již budeme uvažovat hodnotu $E_R \in [0,1; 0,3]$, abychom dosahovali lepších výsledků.

5.3 Mutation rate

Mutation rate je další důležitý parametr s vlivem na chod genetického algoritmu. V kapitole 4.6 jsme si vysvětlili, co je to mutace. Mutation rate M_R představuje pravděpodobnost, že u chromozomu dojde ke změně během procesu mutace. Pokud bychom například zvolili $M_R = 0,5$, došlo by v posledním kroku algoritmu u každého chromozomu k mutaci s pravděpodobností $0,5$. Mutation rate představuje pravděpodobnost jevu, proto budeme optimální hodnotu pro parametr opět hledat v intervalu $[0; 1]$.

Hodnoty blížící se $M_R = 1$ představují téměř jistou změnu ve všech chromozomech. Pro takovéto hodnoty je vliv mutace tak výrazný, že algoritmus ztrácí svůj původní účel a prohledávání stavového prostoru se do velké míry stává náhodným. Flegr [12] ve svých textech zabývajících se genetikou, konkrétně mutacemi a jejich důsledky na organizmus, zmiňuje, že vliv mutace na organizmus je ve většině případů žádný nebo negativní, pozitivních mutací je pouze zlomek. Pokud bychom se inspirovali přírodními procesy, bude lepší volit $M_R << 1$, aby se omezil nežádoucí vliv nadměrného počtu mutací.

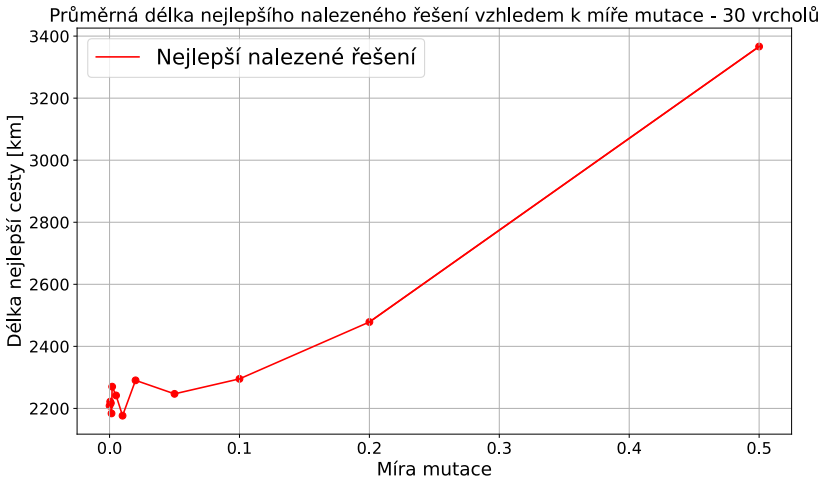
Mnoho vědců se již zabývalo otázkou vhodné volby pro tento parametr, velmi často ale pro jiné úlohy. V publikované literatuře se můžeme setkat s různými přístupy a doporučeními pro hodnoty parametru M_R. Dobré srovnání poskytl ve svém článku Hassanat a spol. [5]. De Jong ve své disertaci [8] udává optimální hodnotu pro $M_R = 0,001$. Grefenstette [9] uvádí optimální pravděpodobnost mutace při uvažování málo rozmanité populace. Udává horní mez $M_{R\_MAX} = 0,05$, protože vyšší hodnoty dle jeho práce zhoršují činnost algoritmu. Řádově vyšší hodnoty pro M_R zkoumali například Deb a Aggraval [10]. Abychom lépe pochopili rozdíly v názorech na optimální hodnoty mutation rate, musíme si uvědomit, že postupy v ostatních krocích algoritmů se článek od článku liší. Zatímco De Jong uvažuje neměnný parametr, rozmanitou populaci a jistotu křížení $P_K = 1$, Greffenstette, Deb a Aggraval uvažují menší, méně bohatou populaci a nejistou pravděpodobnost, že dojde k rozmnožení dvou jedinců $P_K < 1$. Mnoho dalších autorů uvažuje různé přístupy k této problematice, proto se můžeme setkat s odlišnými názory na optimální hodnotu, které se pohybují v celém uvažovaném intervalu. Je také nutné dodat, že většina vypracovaných prací se nutně nezabývá úlohou obchodního cestujícího. Hodnoty nalezené předchozími autory proto nemusí odpovídat výsledkům této práce, protože stavové prostory řešených úloh jsou odlišné.

Ukažme si, jaký má hodnota M_R vliv na chod algoritmu. Běžným postupem může být zkoumání mnoha průběhů algoritmu pro jednotlivé hodnoty M_R. Z většího počtu průběhů je poté možné určit relativní poměr konvergence algoritmu, který udává, v kolika případech bylo s konkrétní hodnotou mutation rate nalezeno optimum. Jednotlivé poměry můžeme porovnat a určit hodnotu nebo interval pro M_R, pro které algoritmus nejčastěji dosahuje konvergence k optimu. V této práci vyzkoušíme od-

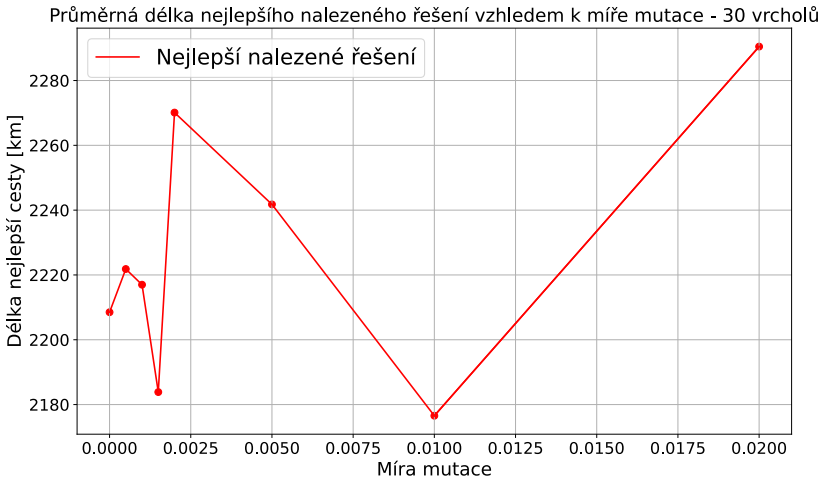
lišný přístup. Budeme nyní předpokládat, že úloha, u které chceme najít optimální řešení, je při vybrané populaci a vlivu elitismu velmi obtížně řešitelná, neboli nalezení optima s danými parametry je nepravděpodobné. Pokud bychom zvolili dostatečně velkou velikost populace, mohlo by dojít k příliš časté konvergenci algoritmu a vliv mutace by tak byl obtížně rozpoznatelný. Namísto hledání optima se zaměříme na minimalizaci kriteriální funkce, která v tomto případě odpovídá fitness funkci. Chod genetického algoritmu prozkoumáme na několika grafech, vždy s fixovanými hodnotami pro všechny parametry s výjimkou zkoumaného M_R. Pro každou hodnotu M_R spustíme 100 chodů algoritmu, abychom mohli výsledky zprůměrovat a vyloučit přílišný vliv ostatních stochastických částí algoritmu. Při tomto druhu testování je nesmírně důležité zafixovat prvotní generaci. Proces inicializace prvotní generace je silně náhodný a rozdíly v různých realizacích algoritmu způsobené křížením by silně zastínily vliv mutace, který zkoumáme především. Velikost populace $N = 100$ a míra elitismu $E_R = 0.25$ je pro všechny následující simulace stejná z důvodu zachování konzistence pokusů. Zkoumané hodnoty pro M_R jsou následující:

$$M_R = \{0; 0,0005; 0,001; 0,0015; 0,002; 0,005; 0,01; 0,02; 0,05; 0,1; 0,2; 0,3; 0,4\}$$

Obrázek 14: Míra mutace - 30 vrcholů



Obrázek 15: Míra mutace - 30 vrcholů, detail

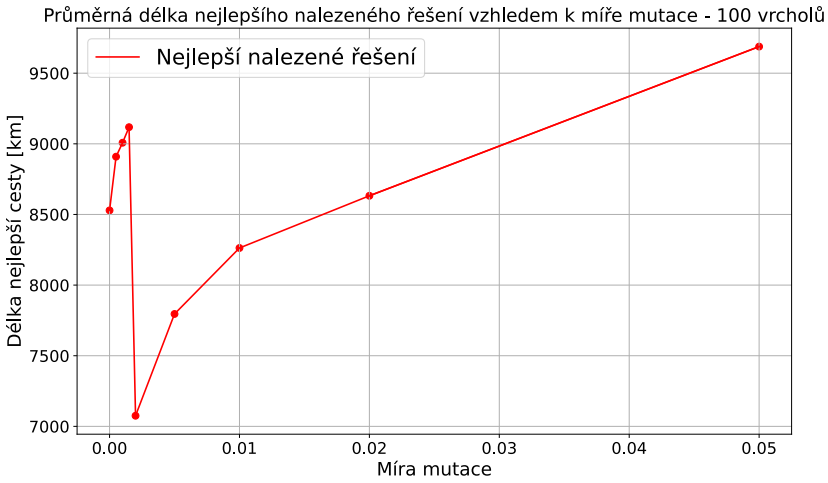


Obrázek 14 ukazuje vliv míry mutace na schopnost algoritmu nalézt nejkratší cestu. Prozatím se zaměřme na vyšší hodnoty mutation rate. Je dobře patrné, že hodnoty vyšší než $M_R = 0,1$ mají za následek zhoršení schopnosti algoritmu nalézt rozumné řešení. Vliv mutace je tak velký, že zastíní činnost ostatních důležitých chodů algoritmu, zejména procesu selekce. Selekce vybírá nejvhodnější kandidáty na rozmnožení dle jejich fitness ohodnocení. Velká míra mutace ale způsobuje, že charakteristiky, díky kterým byly chromozomy dobře adaptovány a měli vysokou fitness, budou s velkou pravděpodobností narušeny díky náhodné mutaci.

Interval $M_R = 0 \wedge M_R <= 0,01$ vykreslený v detailu na obrázku 15 je pro naše účely mnohem zajímavější. Nižší pravděpodobnost mutace má za následek větší celistvost genotypů napříč generacemi. Nedochází k častým změnám genů, které by příliš ovlivnili podobu chromozomu. Mutace s negativním vlivem mohou být odstraněny z příští generace díky křížení, protože je vyšší poměr jedinců, kterým nebyla snížena *fitness* díky nevhodné mutaci. Naopak jedinci, u kterých došlo k mutaci s pozitivním vlivem mají větší šanci se dále propagovat beze změny v klíčových částech genotypu. Malé rozdíly v nejlepším nalezeném řešení ve zkoumaném intervalu jsou způsobeny konkrétní podobou grafu, na kterém byla mutace testována. Zlepšení chodu algoritmu při hodnotě $M_R = 0,01$, které nastává i při průměrování přes velký počet simulací, je pravděpodobně způsobeno náhodným charakterem mutací a je spíše neobvyklé. Tato skutečnost pouze podtrhuje stochastický charakter algoritmu a schopnost mutací pozitivně ovlivnit chod algoritmu. Relativní zlepšení nalezeného řešení se pro tento případ pohybuje pouze v jednotkách %.

Podívejme se nyní na případ složitější úlohy o 100 vrcholech. Velikost populace a míra elitismu jsou stejné jako v předchozím případě.

Obrázek 16: Míra mutace - 100 vrcholů



Obrázek 16 poskytuje lepší pohled na vliv mutace na chod algoritmu. Zatímco v předchozím případě bylo zlepšení v řádu jednotek %, pro složitější graf je vliv mutace významnější. Pro hodnotu $M_R = 0,002$ dosahuje relativní zlepšení 20%, což je významný rozdíl oproti případu, kdy by mutace nebyla využita. Charakter vyobrazené křivky pro vyšší hodnoty mutation rate ($M_R > 0,002$) je podobný jako v předchozím případě. Častější případy mutace mají opět za následek zhoršení činnosti algoritmu z důvodu opakovaného narušení genotypu.

Příklady pro další testované úlohy jsou uvedeny v příloze, nicméně všechny nalezené charakteristiky mají společné rysy. Míra mutace v řádu tisícin má pozitivní vliv na chod algoritmu ve smyslu nalezení kratší cesty, než jakou by algoritmus našel bez přítomnosti mutace. Vyšší hodnoty mají za následek příliš časté změny v genotypu chromozomů a vedou ke zhoršení schopnosti algoritmu najít dobré řešení.

Velice zajímavý je fakt, že nalezené hodnoty jsou podobné těm, které De Jong [8] a Grefenstette [9] nalezli ve svých pracích, které se ale věnovaly optimalizacím funkcí a ne úloze obchodního cestujícího. Obě tyto úlohy mají nesrovnatelné stavové prostory, nicméně optimální hodnoty pro parametr M_R jsou téměř stejné.

6 Rekombinační operátory křížení a operátory mutace

Dosud jsme se věnovali pouze parametrům genetického algoritmu, které lze vyjádřit pomocí čísla. Ukázali jsme si, že vhodné nastavení parametrů má významný vliv na dobu potřebnou k nalezení přijatelného řešení a podstatně ovlivňuje schopnost algoritmu najít dobré řešení. V této kapitole se budeme věnovat rekombinačním operátorům. Jednotlivé operátory si můžeme představit jako soubory pravidel, kterými se řídí procesy křížení a mutace. V kapitole 4.5 a 4.6 jsme si zmínili nejjednodušší případy operátoru křížení a mutace. Příklad, na kterém jsme operátory ilustrovali, předpokládal reprezentaci chromozomu ve tvaru řetězce nul a jedniček. Tuto reprezentaci ale není vhodné použít v úloze obchodního cestujícího, pro kterou je vhodné uvažovat chromozomy ve tvaru posloupnosti indexů vrcholů grafu. Každý index se v reprezentaci může vyskytovat právě jednou, aby pro cestu grafem byla splněna základní podmínka pro hamiltonovskou kružnici. Operátory křížení a mutace musí respektovat tuto podmínku a jejich předpisy musí zabezpečit, aby dceřiné, respektive zmutované chromozomy reprezentovaly hamiltonovské kružnice.

Samotné operátory, podobně jako v předchozích kapitolách zkoumané parametry, mají na chod algoritmu významný vliv. Jsou zodpovědné za vhodnou kombinaci genů dvou nebo více rodičovských chromozomů do dceřiných chromozomů. Rekombinační operátory musí zachovávat jistou genetickou strukturu v chromozomech, jinými slovy nesmějí příliš rozkládat struktury, které vznikly v průběhu předchozích generací a které silně korelují s vysokou fitness chromozomu. Pokud by operátory příliš často narušovaly zmíněné struktury, jako v případě k-bodového křížení pro vysoká k zmíněném v kapitole 4.5, změnil by se charakter algoritmu na náhodné prohledávání.

Nejprve si zmíníme některé druhy rekombinačních operátorů, které se dají použít pro řešení úlohy obchodního cestujícího, a ukážeme si jejich vliv v procesu křížení na jednoduchém příkladu. Poté demonstrujeme použití jednotlivých operátorů křížení na stejných úlohách, na kterých jsme vyhodnocovali vliv parametrů velikosti populace atd. Na konec porovnáme vliv rekombinačních operátorů na chod algoritmu a pokusíme se najít takový operátor, který poskytuje lepší výsledky než ty ostatní.

Rekombinační operátory křížení a operátory mutace fungují na velice podobném principu. Hlavním rozdílem je počet chromozomů vstupujících do procesu křížení, respektive mutace. Zatímco křížení se téměř vždy účastní alespoň dva chromozomy, mutace se účastní pouze jeden. Křížení proto umožňuje mísení různých genotypů chromozomů s vysokou fitness a představuje klíčový mechanismus genetického algoritmu, díky kterému může algoritmus navštívit body stavového prostoru s potenciálně vysokou fitness. Mutace naproti tomu slouží jako pomocný mechanismus, který umožňuje únik algoritmu z lokálního minima díky obohacení genotypu náhodnou

změnou v chromozomu.

V první části kapitoly se budeme věnovat rekombinačním operátorům, ve druhé části se poté zaměříme na vliv různých druhů mutace na chod algoritmu.

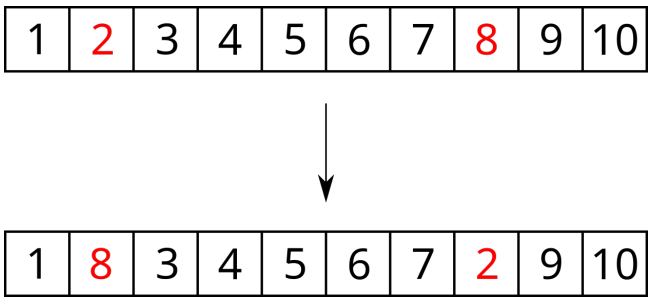
6.1 Operátory mutace

V této sekci se budeme věnovat operátorům mutace. Nejprve si zmíníme mechanismy skrývající se pod jednotlivými typy mutace, předvedeme jejich průběh na jednoduchém příkladu a na konec porovnáme účinnost algoritmu při použití konkrétních typů mutací. Larrañaga a spol. [17] a Abdoun a spol. [16] ve svých článcích zmiňují některé základní typy mutací, které zde budeme zkoumat.

6.1.1 Mutace EM

Mutace označovaná jako *exchange mutation* je asi nejzákladnější případ pro mutaci chromozomu v úloze obchodního cestujícího. Její princip je velice jednoduchý. Na začátku jsou v chromozomu náhodně vybrány dva geny, které si následně vymění pozici.

Obrázek 17: Exchange mutation

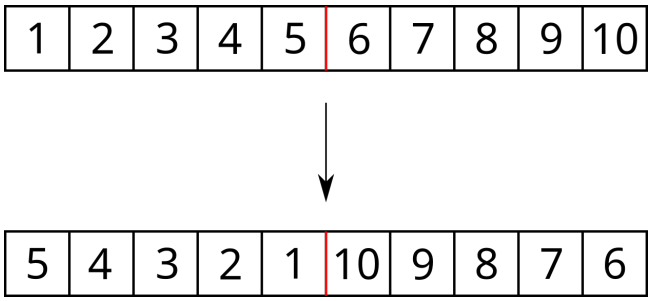


Exchange mutation na obrázku 17 je oproti ostatním typům mutací poměrně neinvazivní. Výměnou 2 měst v průchodu grafu dochází ke změně minimálně 2 a maximálně 4 hran. Ke změně dvou hran však dochází pouze v případě, kdy spolu v původním chromozomu vybraná města sousedí a zároveň řešíme symetrickou úlohu obchodního cestujícího. Pro jakýkoliv jiný případ, kdy spolu dvě vybraná města nesousedí, dochází ke změně 4 hran.

6.1.2 Mutace CIM a CIM*

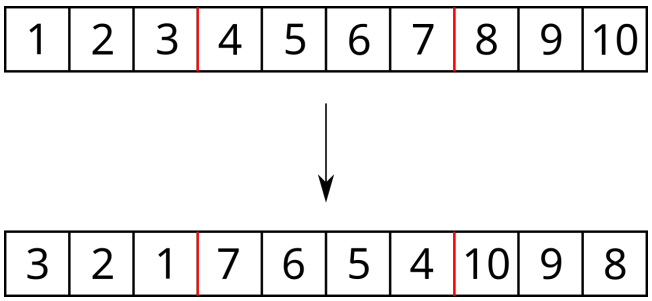
Mutace CIM neboli *central inverse mutation* v chromozomu provádí změny podobného počtu hran jako výše zmíněna exchange mutation. Chromozom je nejprve rozdělen v libovolném místě na dvě části. Každá z těchto částí je poté procházena v opačném sledu, než ve kterém byla procházena původně.

Obrázek 18: Mutace CIM



Podívejme se nyní důkladně na výsledný chromozom na obrázku 18. Při bližším pohledu na chromozom vzniklý po aplikaci operátoru CIM zjistíme, že se jedná o stejnou reprezentaci jako v původním případě. Začátek cesty je posunut z města 1 do města 5 a celá cesta je procházena pozpátku, nicméně za předpokladu symetrické úlohy obchodního cestujícího se jedná o reprezentaci stejného průchodu grafem, pouze jinak zapsaného. Mutaci CIM proto obměníme zvýšením počtu segmentů, na které původní chromozom rozdělíme. Nově získanou mutaci označíme CIM*.

Obrázek 19: Mutace CIM*

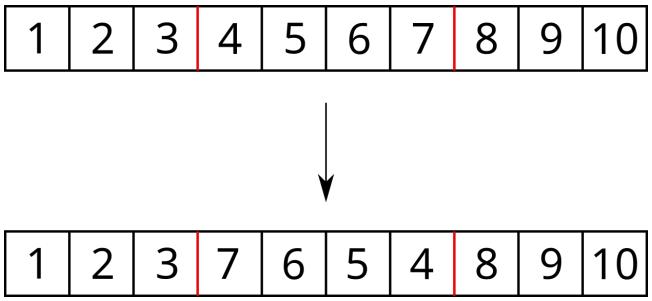


Povšimněme si rozdílu mezi operátorem CIM na obrázku 18 a operátorem CIM* na obrázku 19. Jednotlivé segmenty jsou opět procházeny pozpátku, zvýšený počet dělících bodů má ale za následek vznik zcela nového chromozomu. Struktury uvnitř jednotlivých segmentů jsou zachovány, dochází tak k porušení a vzniku nových hran pouze na rozhraní segmentů.

6.1.3 Mutace RSM

Reverse sequence mutation je další poměrně neinvazivní druh mutace. Je velice podobná modifikovanému operátoru CIM* s tím rozdílem, že pozpátku se projde pouze centrální segment. Zbylé dva segmenty jsou procházeny stejně jako v původním chromozomu. V poslední kapitole si ukážeme, že tento druh mutace je totožný s typem heuristiky, která je vhodná pro řešení úloh obchodního cestujícího.

Obrázek 20: Mutace RSM

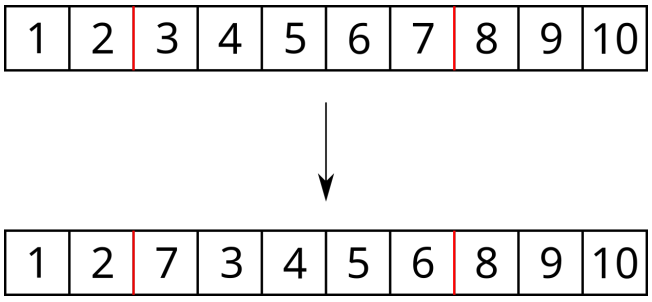


Při mutaci RSM může dojít k jednomu ze dvou případů. Pokud se hrany, které vycházejí z měst na okraji segmentů kříží, dojde k jejich rozuzlení. Pokud se hrany naopak neprotínají, dojde k jejich překřížení. Z trojúhelníkové nerovnosti lze poměrně snadno odvodit, že v grafu, kde platí eukleidovská metrika, lze dosáhnout kratší cesty rozuzlením překřížených hran. Ideálním případem je proto mutace, která odstraní překřížené hrany.

6.1.4 Mutace Throas

Poslední z mutací, která nemá příliš velký vliv na stavbu chromozomu, se nazývá *Throas* mutace. Chromozom je stejně jako v předchozích případech rozdělen na tři části. Autoři v původní práci [16], ze které je podoba mutace převzata, uvažují centrální segment, který obsahuje přesně tři geny. V této diplomové práci je mutace generalizována a centrální segment může být libovolně dlouhý. Mutace spočívá v posunutí jednoho genu v centrálním segmentu. Po rozdělení na části a utvoření centrálního segmentu je poslední gen segmentu přesunut na jeho začátek. Zbylé dva segmenty zůstávají beze změny. Průběh Throas mutace je ilustrován na obrázku 21.

Obrázek 21: Mutace Throas

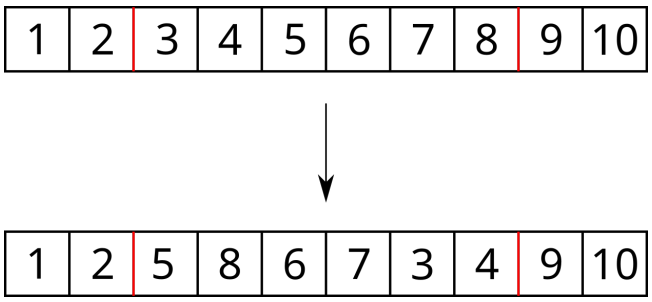


Opět pozorujeme změnu pouze 4 hran. Nedochází proto k podstatné změně genotypu. Mutace Throas lze více zobecnit. Zatím předpokládáme posun genů uvnitř segmentu právě o jednu pozici. Vygenerováním náhodného čísla lze ale učinit náhodným i proces, který rozhoduje o počtu pozic, o které se jednotlivé geny mají posunout. Tento postup však v této práci nebudeme uvažovat.

6.1.5 Mutace PSM

Další druh mutace, který si v této práci zmíníme, je *partial shuffle mutation*. Na rozdíl od ostatních typů mutace je tento druh poměrně invazivní ke genové struktuře chromozomu. Chromozom, stejně jako v předchozích případech, náhodně rozdělíme na tři části. Dva krajní segmenty zůstávají beze změny. Centrální segment, jehož délka je náhodná, je poté nahrazen jeho libovolnou permutací. Následkem permutace dochází k náhodné podobě centrálního segmentu, která může být naprosto odlišná od původní struktury. Výhodou této mutace je přínos v podobě nové genetické informace, který však může být zastíněn přílišným zásahem tohoto operátoru do struktury chromozomu.

Obrázek 22: Mutace PSM

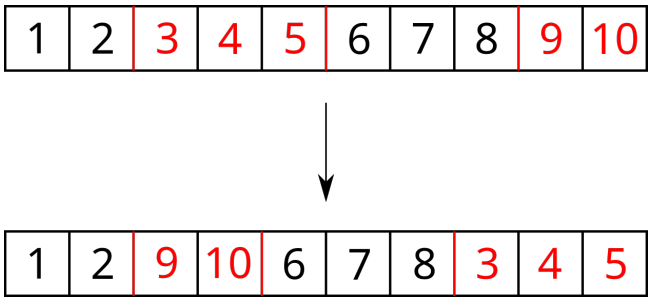


Zásah mutace do genotypu chromozomu vykreslený na obrázku 22 je závislý především na šířce centrálního segmentu. Pokud jsou dělící body relativně na blízkých pozicích v chromozomu, permutace centrálního segmentu nemá za následek přílišné přeskupení hran v cestě grafem. Pokud se však dělící body nacházejí daleko od sebe a centrální segment zabírá podstatnou část chromozomu, dochází k významným změnám uvnitř segmentu a k přeskupení velkého počtu hran. Mutace PSM může v nejhorším případě zcela znehodnotit chromozom přílišným snížením jeho fitness.

6.1.6 Mutace SSM

Posledním typem mutace, který budeme zkoumat, se nenachází ani v jednom z článků [16][17]. Byl navržen čistě pro potřeby této diplomové práce s cílem otestovat alternativní přístupy k již existujícím operátorům. Nazveme ho *segment switch mutation*, protože následkem jeho působení je výměna dvou segmentů v chromozomu. Jedná se o zobecnění operátoru exchange mutation s tím rozdílem, že nedochází k výměně pozic dvou měst, ale dvou celých částí chromozomu.

Obrázek 23: Mutace SSM



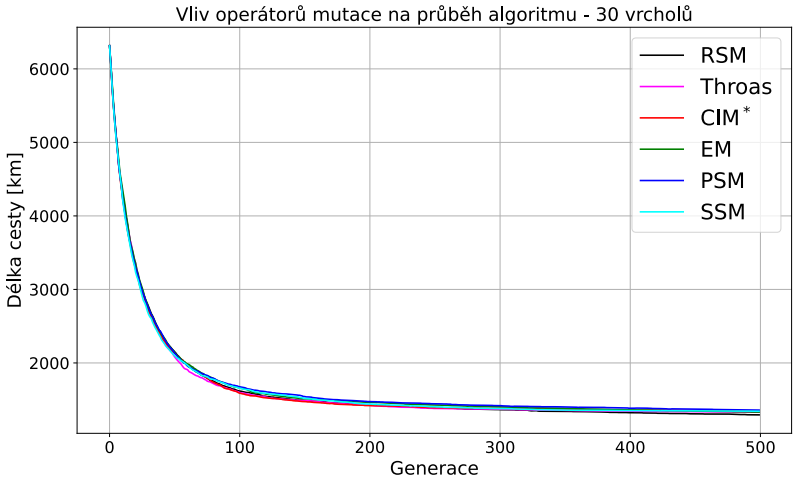
Výměnou dvou segmentů opět dochází k obměně celkem 8 hran, přičemž 4 zanikají a 4 vznikají. Hrany uvnitř jednotlivých segmentů nejsou pozměněny, v celém chromozomu proto nedochází k přílišnému narušení genotypu.

6.1.7 Porovnání vlivu operátorů mutace

V této části si ukážeme vliv jednotlivých operátorů mutace na chod algoritmu. V průběhu této práce byl použit postup, který uvažuje jednotkovou pravděpodobnost procesu křížení, jinými slovy ke křížení dojde vždy. Míra mutace M_R má hodnotu v řádu pouhých tisícin, nemá proto podstatný vliv na výsledek algoritmu. Mutace u tohoto postupu slouží pouze jako pomocný operátor, který může pomoci algoritmu k občasnému úniku z náhodného lokálního minima.

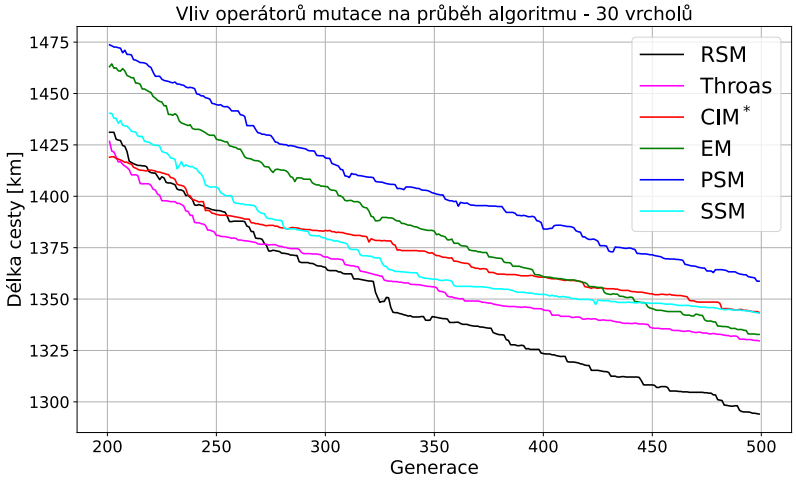
Původní pokusy, které byly provedeny s podobnými parametry jako v předchozích kapitolách ($N = 100, g = 100, E_R = 0, M_R = 0.002,$), ukázaly, že rozlišování mezi jednotlivými operátory mutace nemá příliš velký smysl. V průběhu algoritmu s výše uvedenými parametry bychom totiž očekávali přibližně $M_R \times N \times g$ mutací, což by pro zvolené hodnoty odpovídalo 20 mutacím v průběhu celého algoritmu. Takto malý počet mutací neposkytuje dostatečné rozdíly ve výsledných průbězích a znemožňuje rozlišit rozdílné vlivy jednotlivých operátorů. Články [5] a [18] pojednávají o vlivu mutace na průběh algoritmu, nicméně autoři předpokládají nesrovnatelně vyšší hodnoty míry mutace M_R než uvažujeme v této práci. Pro zajištění lepších výsledků je nutné zvýšit počet mutací, které se uskuteční během jednoho průběhu algoritmu. Zároveň je ale důležité, abychom zachovali původně zamýšlenou úlohu rekombinačních operátorů a operátorů mutace. Přílišným zvýšením hodnoty míry mutace M_R zaniká účel operátoru mutace jakožto podpůrného mechanismu pro únik z lokálního minima a operátor se postupně stává hlavním prohledávacím mechanismem algoritmu. Postup, který byl zvolen v této práci uvažuje stále stejnou populaci o velikosti $N = 100$. Míra mutace je nastavena na mez definovanou Grefenstettem [9] na $M_R = 0.05$. Počet generací je navýšen na $g = 500$. Výsledky byly opět průměrovány ze sta realizací a z fixované počáteční populace.

Obrázek 24: Porovnání operátorů mutace - 30 vrcholů



Obrázek 24 ilustruje použití jednotlivých operátorů mutace a jejich vliv na chod algoritmu. Při zobrazení celého průběhu není vliv rozdílných operátorů patrný, nicméně pro úplnost je nutné jej zobrazit. Při bližším pohledu je zřejmé, že pro vyhodnocení není příliš vhodné uvažovat vyšší počet generací, protože proces konvergence je v posledních generacích velmi pozvolný. Podívejme se nyní na chod algoritmu v posledních 300 generacích ve větším detailu.

Obrázek 25: Porovnání operátorů mutace - 30 vrcholů, detail

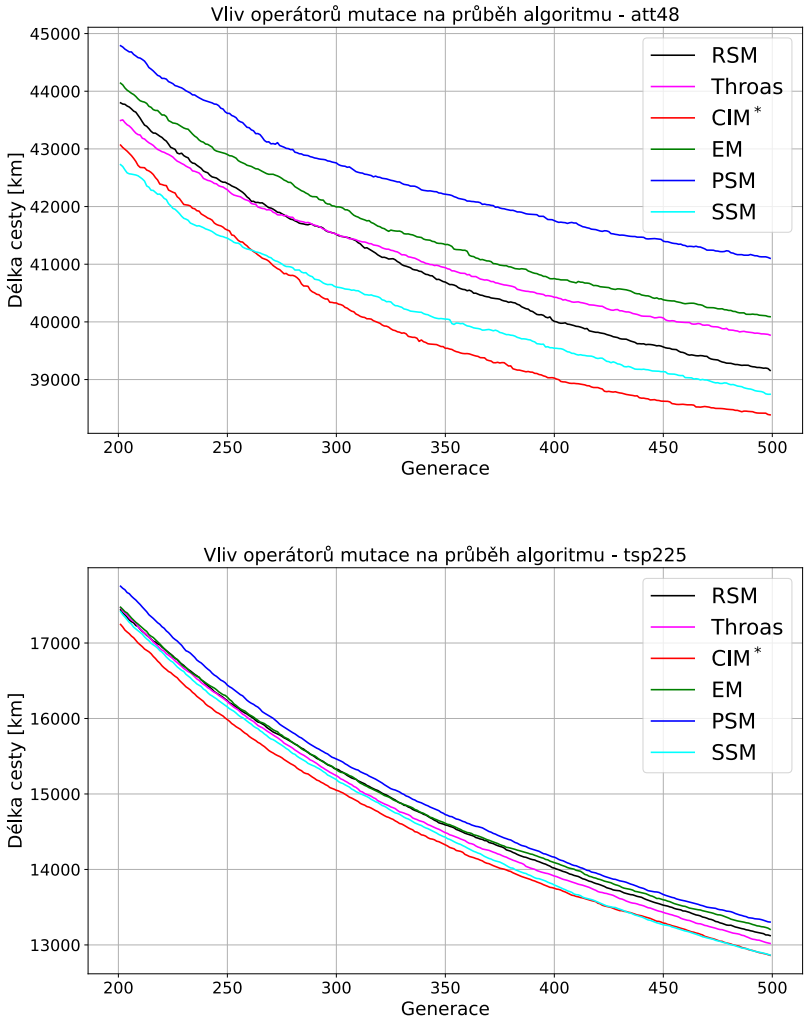


Při pohledu na obrázek 25 již můžeme rozeznat rozdílné chování algoritmu při použití odlišných operátorů. Díky odstranění prvních generací z vykreslení můžeme

mnohem lépe pozorovat rozdíly mezi jednotlivými křivkami. Mějme však na paměti, že relativní rozdíly v délce nalezené cesty jsou pouze v jednotkách %. Nejlepšího výsledku dosahuje pro zkoumaný případ se 30 vrcholy operátor RSM. Nejhorší průběh měl algoritmus, který jako operátor mutace použil PSM.

Na základě jednoho grafu je obtížné určit, který operátor je nejlepší. Výsledek může být ovlivněn velikostí grafu, rozmístěním jeho vrcholů nebo konkrétní počáteční populací. Podívejme se nyní na průběh algoritmu při řešení složitějších úloh. Jeden případ bude zobrazovat průběh na již známém grafu att48, zaměříme se ale i na podstatně složitější úlohu s 225 vrcholy *tsp225* převzatou opět z TSPLIB [13]. Počáteční generace nebudeme zahrnovat do vykreslení, protože významný pokles délky nejkratší nalezené hamiltonovské kružnice je způsoben rekombinačními operátory, a nikoliv operátory mutace.

Obrázek 26: Porovnání operátorů mutace - att48, tsp225, detail



Při porovnání obrázků 25 a 26 již můžeme vyvodit některé závěry. Ve všech zkoumaných úlohách si nejhůře vedl algoritmus používající operátor PSM. Pravděpodobnou příčinou je přílišný vliv operátoru na genotyp díky náhodné permutaci jednoho segmentu chromozomu a změně příliš velkého počtu hran v cestě grafem. Operátor RSM si vedl nejlépe na úloze se 30 vrcholy, jeho efektivita ale pro složitější úlohy klesá. Operátory EM a Throas si lépe vedly na jednodušší úloze, zatímco ve složitějších úlohách zaostávají. Nejlépe si vedly algoritmy využívající operátory CIM* a SSM. U jednodušší úlohy lehce zaostávaly, nicméně ve složitějších úlohách tyto typy mutace pomohly algoritmu nalézt kratší řešení než ostatní operátory. Obecně si lépe vedly algoritmy s operátory, které svým působením měnily v grafu větší počet hran a prováděly tak významnější změny v genotypu. Porovnání s relativně špatnými výsledky algoritmu využívajícího PSM ve všech úlohách ale naznačuje, že existuje

rozumný odhad pro limit počtu hran vzhledem k velikosti grafu, které by operátor měl svým působením změnit.

6.2 Operátory křížení

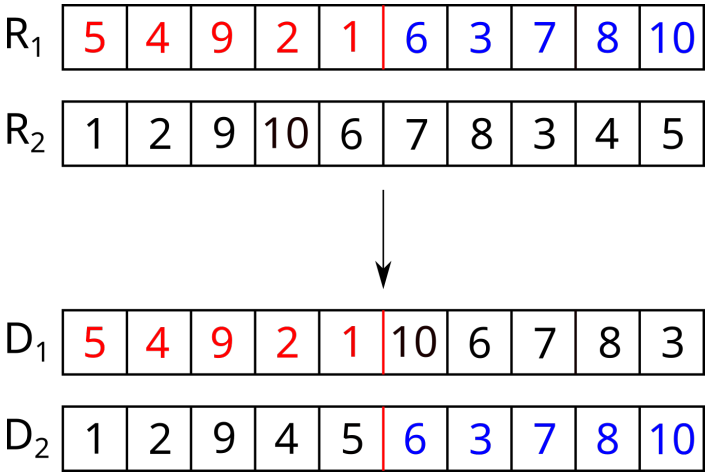
V poslední kapitole se budeme věnovat rekombinačním operátorům. Rekombinační operátory slouží v genetickém algoritmu jakožto hlavní prohledávací mechanismus stavového prostoru. Jsou zodpovědné za proces křížení a určují, jakým způsobem budou z rodičovských chromozomů vytvořeny chromozomy dceřiné. Stejně jako v předchozí kapitole 6.1, která se věnovala operátorům mutace, i v této kapitole si nejprve ukážeme jednotlivé druhy rekombinačních operátorů. Mějme na paměti, že obyčejné rekombinační operátory pro bitové řetězce není možné použít, protože by během procesu křížení vznikaly jedinci, kteří by neprezentovali validní průchod grafem. Po seznámení s rekombinačními operátory použitými během experimentů vyzkoušíme jejich vliv na chod algoritmu. Pro zajímavost si uvedeme i jeden typ heuristiky, jejíž použití je vhodné v úloze obchodního cestujícího.

Mnoho autorů se v minulosti zabývalo operátory křížení. Pro potřeby této práce je důležité zmínit především práce Hollanda [19], Hussaina a spol. [20] a Larrañagy a spol. [21]. Pro porovnání vlivu operátorů na konvergenci genetického algoritmu jsme čerpali z těchto uvedených prací. Připomeňme si, že na rozdíl od mutace se procesu křížení účastní alespoň dva chromozomy. Podívejme se nyní na jednotlivé typy operátorů použité v této práci.

6.2.1 Jednobodové křížení - 1P

Ze všech uvažovaných rekombinačních operátorů si nejprve zmíníme nejjednodušší *jednobodové křížení*. Nejprve je náhodně zvolen dělící bod v jednom z rodičů. Jeden ze vzniklých segmentů je přímo zkopírován do jednoho dceřiného chromozomu, druhý segment je zkopírován do druhého potomka. Do volných pozic v potomcích jsou popořadě zkopírovány geny z druhého rodiče tak, aby vzniklý chromozom reprezentoval hamiltonovskou kružnici v grafu.

Obrázek 27: Operátor 1P - jednobodové křížení

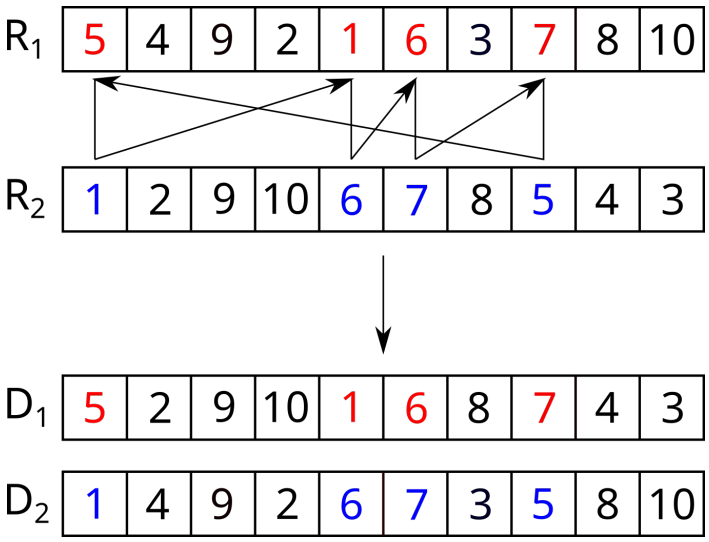


Rekombinační operátor jednobodového křížení garantuje, že velká část genetické informace jednoho z rodičů bude přenesena do další generace téměř beze změny. Nevýhodou tohoto operátoru je ale slabá schopnost prozkoumávat body stavového prostoru, protože takovýto druh křížení neumožňuje složitější kombinace genotypů. Po zkřížení jsou si rodiče a potomci velmi podobní, což může být nežádoucí jev z hlediska schopnosti algoritmu konvergovat k optimu.

6.2.2 Cycle crossover - CX

Druhý rekombinační operátor, který si v práci zmíníme, nese název *cycle crossover*. Funkce operátoru spočívá v identifikaci cyklu, pomocí metody, kterou si popíšeme v následujícím odstavci. Schéma funkce operátoru CX je ilustrováno na obrázku 28.

Obrázek 28: Operátor CX - cycle crossover

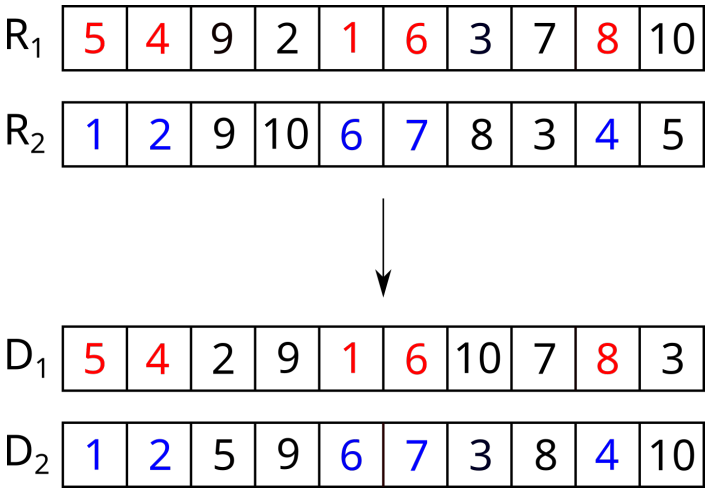


V prvním z rodičů je náhodně vybrán gen. V našem příkladu na obrázku 28 byl vybrán gen na pozici 1 s hodnotou 5. K tomuto genu je nalezen korespondující gen ve druhém rodiči, který sdílí stejnou pozici v chromozomu. V příkladu tomu odpovídá gen 1 na pozici 1 v chromozomu R_2. V prvním rodiči je poté vyhledán gen se stejnou hodnotou (nikoliv pozicí). V R_1 by tomu odpovídal gen s hodnotou 1 na pozici 5. Postup se opakuje, dokud není dokončen cyklus. V příkladu by byl při použití operátoru nalezen cyklus $5 - 1 - 6 - 7 - 5$. Nalezený cyklus je přenesen do obou potomků se stejnými pozicemi, ve kterých se nachází v jednotlivých rodičích. Volné geny v potomkovi D_1 jsou poté doplněny geny z rodiče R_2 a geny v potomkovi D_2 jsou získány z R_1. Díky charakteru průběhu operátoru nemůže dojít ke kolizím dvou genů se stejnou hodnotou v jednom chromozomu. Pozice zkopírovaných genů jsou proto v potomcích totožné s pozicemi genů u rodičů. Může nastat situace, kdy má cyklus stejnou délku jako celý chromozom. V takovém případě sice dochází ke křížení, potomci ale mají stejnou podobu jako rodiče.

6.2.3 Position-based crossover - POS

Position-based crossover je další z rekombinačních operátorů, kterým se v této práci budeme věnovat. Na začátku procesu křížení je náhodně vygenerován seznam pozic v chromozomu, který je pro oba rodiče shodný. Geny nacházející se na těchto pozicích jsou zkopírovány popořadě z rodičů R_1, R_2 do potomků D_1, D_2. Zbývající pozice v D_1 jsou poté popořadě doplněny z chromozomu R_2 tak, aby se v potomkovi nevyskytovaly duplikátní geny. Volné pozice v D_2 jsou stejným způsobem zaplněny geny z R_1. Průběh křížení za pomoci operátoru POS je znázorněn na obrázku 29.

Obrázek 29: Operátor POS - position-based crossover

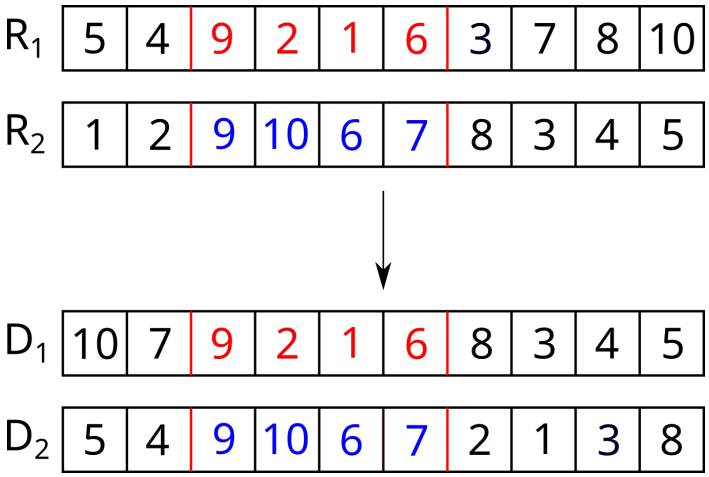


Proces výběru pozic, které zůstanou zachovány, je zcela náhodný. V průměru bychom ale očekávali zachování 50% genů. Operátor umožňuje přenesení velkého počtu osamocených genů, nicméně potomci pravděpodobně zdědí i větší množství kratších segmentů, které se původně vyskytovaly v rodičích.

6.2.4 Ordered crossover - OX

Operátor *ordered crossover* poprvé zmíněný Davisem v jeho práci [22] má za úkol zachovat relativní pořadí vrcholů v chromozomu. Nejprve jsou náhodně vybrány dvě pozice, ve kterých jsou rodiče rozděleni na tři segmenty. Centrální segmenty jsou poté zkopírovány z rodičů na potomky. Zbývající volné pozice jsou vyplněny stejným způsobem jako u operátorů 1P a POS. Opět platí, že volné pozice dceřiného chromozomu D_1, který obsahuje segment rodiče R_1, jsou zaplněny geny z rodiče R_2. Analogicky pozice v D_2 jsou zaplněny geny z rodiče R_1.

Obrázek 30: Operátor OX - ordered crossover

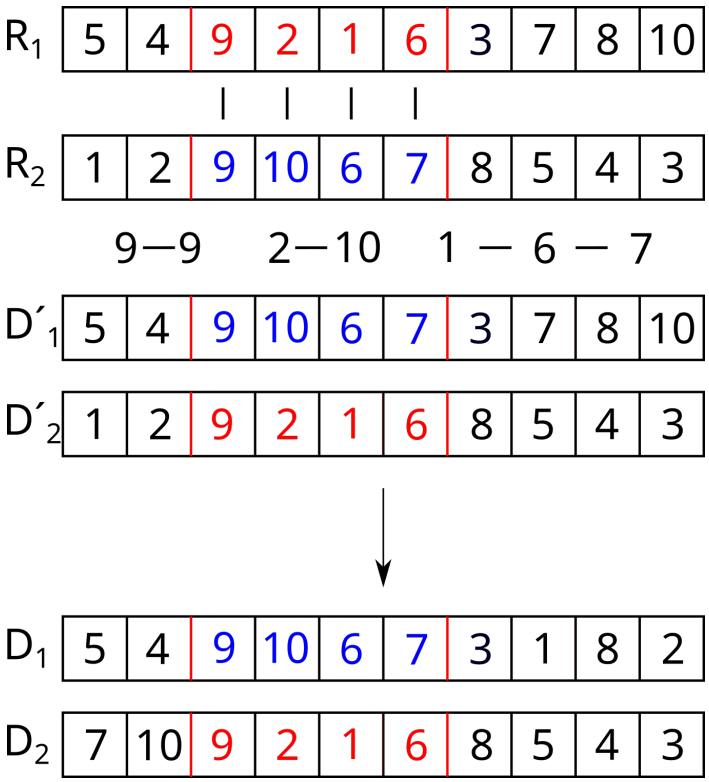


Obrázek 30 ukazuje proces křížení za použití operátoru OX. Oba centrální segmenty přecházejí do další generace beze změny. Chromozomům je tak umožněno zachovat potenciálně důležitý kus genotypu a přenést ho do dalších generací. Zachováním alespoň části relativního řádu genů z druhého rodiče dochází k vytvoření potomka, který má značnou šanci na vysokou fitness.

6.2.5 Partially mapped crossover - PMX

Partially mapped crossover, neboli PMX, byl poprvé zmíněn Goldbergem a Linglem [23]. Jejich cílem bylo navrhnout operátor, který by zabezpečil přenos co nejvíce jednotlivých segmentů, které způsobují vysokou fitness chromozomu, do další generace. Výsledkem jejich snažení je PMX.

Obrázek 31: Operátor PMX - partially mapped crossover



Podívejme se nyní blíže na obrázek 31, který ukazuje proces křížení pomocí operátoru PMX. Na první pohled je vidět, že na rozdíl od ostatních rekombinačních operátorů, obsahuje PMX jeden mezikrok navíc. Stejně jako v předchozích případech jsou zvoleny dva body, které rozdělují rodiče na tři segmenty. Mezi centrálními segmenty se vytvoří mapování hodnot mezi geny, které leží na stejných pozicích v chromozomu. Jedna hodnota se může mapovat na více jiných stejně jako v našem příkladu. Po výběru pozic pro vytvoření segmentů bychom dostali následující mapování:

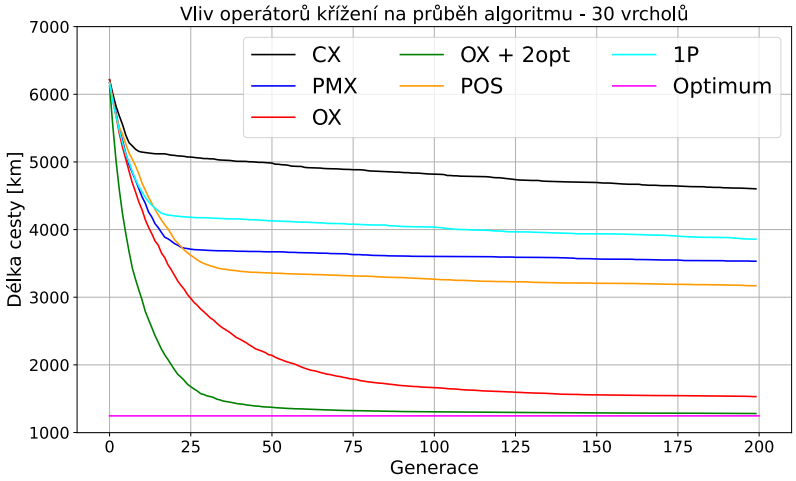
$$9 \Leftrightarrow 9, \quad 2 \Leftrightarrow 10, \quad 1 \Leftrightarrow 6 \Leftrightarrow 7$$

V mezikroku jsou nejprve vytvořeny kopie rodičovských chromozomů, u kterých jsou následně prohozeny centrální segmenty, čímž vznikají *protodceřiné* chromozomy D'_1, D'_2. Všechny ostatní geny jsou poté transformovány na D_1, D_2 pomocí dříve vytvořeného mapování na validní hodnoty.

6.2.6 Porovnání vlivu rekombinačních operátorů

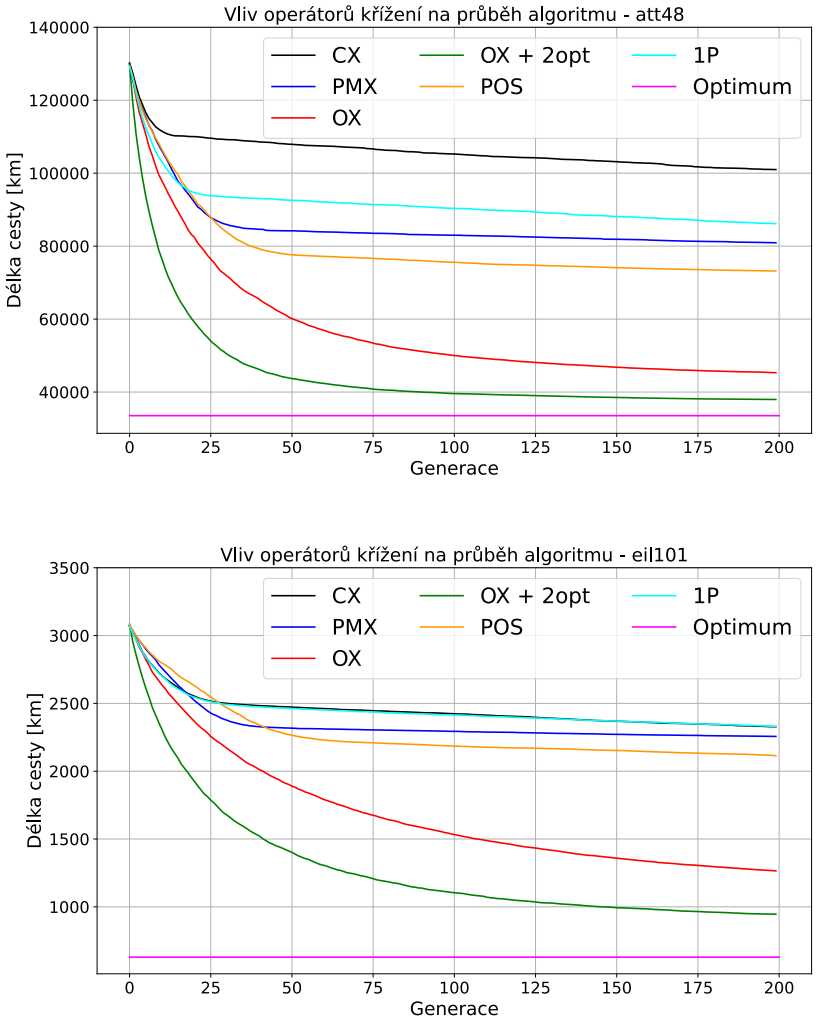
Po seznámení se všemi typy operátorů křížení, které v práci použijeme, si ukážeme účinnost algoritmu při použití těchto operátorů na různých úlohách. Pro simulace budeme uvažovat optimální hodnoty pro parametry, které jsme v této práci dříve určili. Všechny simulace byly opět průměrovány ze 100 realizací algoritmu, v zájmu rozumné výpočetní náročnosti proto byla zvolena populace o velikosti $N = 100$. Míra elitismu a míra mutace byly zvoleny s hodnotami $E_R = 0.2, M_R = 0.002$. Pro srovnání je uvedena i verze algoritmu, která využívá *2-opt heuristiku* [24]. Jedná se o jednoduchou heuristiku původně vyvinutou pro řešení úlohy obchodního cestujícího. V chromozomu se hledá posloupnost genů, která při průchodu grafem vytváří překřížené hrany. Pokud je taková posloupnost nalezena, je v chromozomu přeskládána tak, aby byla procházena v opačném směru. Tento přístup je shodný s funkcí operátoru RSM.

Obrázek 32: Porovnání rekombinačních operátorů - 30 vrcholů



Obrázek 32 ilustruje očekávané průběhy genetických algoritmů při výběru jednotlivých rekombinačních operátorů. Nejhůře si jednoznačně vedl algoritmus s operátorem CX. Prvních 10 generací konvergoval podobně rychle jako ostatní varianty, poté ale rychlost konvergence významně zpomalila. Lépe si vedly algoritmy využívající operátory jednobodového křížení, PMX a POS. Nicméně i tyto algoritmy poměrně rychle ztratily schopnost rychle konvergovat k optimu. Ze zkoumaných operátorů poskytuje nejlepší výsledky jednoznačně operátor OX, který se dostatečnou rychlostí přibližoval k optimu až do 100. generace. Schopnost konvergence algoritmu lze značně posílit použitím heuristiky. Zelená křivka představuje algoritmus, který ke svému chodu využil operátor OX a zároveň 2-opt heuristiku. Již po 75 generacích našel algoritmus řešení velice blízko optima. Podívejme se nyní na další příklady z knihovny TSPLIB [13].

Obrázek 33: Porovnání rekombinačních operátorů - att48, eil101



Typické průběhy algoritmu ve složitějších úlohách jsou vykresleny na obrázku 33. Problémy att48 a eil101 se 101 vrcholy představují podstatně těžší úlohy, zejména kvůli nesrovnatelně většímu stavovému prostoru. U problému att48 můžeme pozorovat téměř totožné chování jednotlivých verzí algoritmu jako na obrázku 32. Operátor CX rychle ztrácí rychlost konvergence a zůstává daleko od optima. Rekombinační operátory 1P, PMX a POS dosahují lepších výsledků, stále se ale nacházejí příliš daleko od optima. Jediný algoritmus, který se k optimu alespoň přiblížil byl algoritmus využívající operátor OX. Hybridní algoritmus s 2-opt heuristikou opět poskytl podstatně lepší výsledky.

Úloha eil101 se svými 101 vrcholy již pro algoritmy s parametry uvedenými na začátku kapitoly představovala obtížně řešitelný problém. Operátor CX dosahuje

oproti ostatním zkoumaným úlohám lepšího výsledku, nicméně ze všech zkoumaných alternativ stále poskytuje nejhorší odhad optimální cesty. Operátory 1P, POS a PMX poměrně brzy vyčerpají dostupný genotyp a proces konvergence je silně utlumen. Operátor OX opět předčil všechny ostatní typy rekombinačních operátorů. V tomto případě se ale ani hybridní algoritmus využívající 2-opt heuristiku nepřiblížil k optimu.

Mějme na paměti, že schopnost konvergence algoritmů byla ovlivněna volbou velikosti počáteční populace $N = 100$. Zvětšením populace dojde ke zlepšení procesu konvergence pro všechny varianty rekombinačních operátorů. Výsledky pokusů jasně ukazují, že pro symetrickou úlohu obchodního cestujícího je ze zkoumaných operátorů nejlepší volbou pro operátor OX. Při porovnání chování čistých genetických algoritmů a hybridního algoritmu, který využívá heuristiku, je zřejmé, že hybridní algoritmy mohou se stejnými parametry dosáhnout daleko lepších výsledků. Použitá 2-opt heuristika je jedna z nejjednodušších heuristik pro úlohu obchodního cestujícího. Existují další, sofistikovanější přístupy, jako například 3-opt [25] heuristika. 3-opt heuristika se řídí velmi podobnými principy jako 2-opt. Odstraněním 3 hran v grafu je chromozom rozdělen na 3 segmenty. Segmenty lze poté spojit dohromady 7 novými uspořádáními a vzniká tak 7 nových chromozomů. Pokud je fitness některého z nových chromozomů vyšší než fitness původního chromozomu, dojde k nahrazení původního chromozomu jedincem s vyšší fitness. V současnosti jedna z nejlepších heuristik pro řešení úlohy obchodního cestujícího, Lin-Kernighanova heuristika [26], iterativně vylepšuje řešení postupným přeskupením hran v grafu. Pokud je nalezeno lepší řešení než řešení původní, je původní chromozom nahrazen novým a proces se opakuje, dokud není dosaženo lokálního minima fitness funkce. Hybridizací genetických algoritmů za použití heuristik lze dosáhnout mnohem lepších výsledků než při použití samotných genetických algoritmů. Úskalí ale spočívá ve výpočetní složitosti heuristik, které na rozdíl od genetického algoritmu s teoretickou výpočetní složitostí vzhledem k počtu vrcholů $O(n)$, mají výpočetní složitost alespoň kvadratickou.

Z poznatků získaných v průběhu práce můžeme určit strategii, která je vhodná pro nalezení optimálního řešení, nebo alespoň řešení blízkého optimu. Vhodnou volbou velikosti počáteční populace $N \doteq 100$ pro problémy, které potřebujeme vyřešit okamžitě, nebo N v řádu tisíců pro problémy, kde nejsme v časové tísni, zabezpečíme dostatečnou genetickou rozmanitost. Použitím podpůrných mechanismů selekce, například turnajové selekce, zrobustníme proces výběru jedinců s vysokou fitness pro křížení. Aplikací elitismu s mírou $E_R \in (0,1; 0,3)$ významně posílíme pravděpodobnost, že se dobře adaptovaní jedinci dostanou do následující generace beze změny. Vhodnou volbou míry mutace $M_R \doteq 0,002$ poskytneme algoritmu mechanismus úniku z lokálního minima, který zároveň negativně neovlivňuje schopnost algoritmu konvergovat. Výkonnost algoritmu lze značně posílit použitím vhodné heuristiky, která je ale jedinečná pro konkrétní typ úlohy. V úloze obchodního cestujícího jsme si pouze jako příklad uvedli heuristiky 2-opt, 3-opt a Lin-Kernighanovu heuristiku.

7 Závěr

Cílem této práce bylo seznámit se s principy genetických algoritmů a demonstrovat jejich funkci na úloze obchodního cestujícího. V první kapitole jsme si nejprve představili samotnou úlohu obchodního cestujícího a vymezili požadavky na symetričnost a eukleidovský prostor, ve kterém jsme úlohu řešili. V následující kapitole jsme si vysvětlili důležité pojmy a principy genetických algoritmů a seznámili se s jednotlivými mechanismy a operátory, které algoritmus ke svému chodu využívá. Na závěr kapitoly jsme si uvedli výhody genetických algoritmů oproti ostatním přístupům a vysvětlili si důvody, proč je vhodné využít genetické algoritmy právě v úloze obchodního cestujícího.

Ve třetí kapitole jsme se zaměřili na tři důležité parametry: velikost populace N, míru elitismu E_R a míru mutace M_R. Provedli jsme pokusy s cílem určit optimální hodnoty pro tyto parametry zejména s ohledem na schopnost algoritmu konvergovat k optimu a na výpočetní čas. Experimentálně jsme určili výrazné zlepšení chování algoritmu pro postupné zvyšování populace až do hodnoty $N \doteq 100$. Vyšší hodnoty vedou ke zlepšení schopnosti algoritmu konvergovat, nicméně vliv již není tak výrazný a použití velkých populací je vhodné spíše pro zpracování, které není nutné dokončit v reálném čase. Pro míru elitismu a pro míru mutace jsme určili optimální hodnoty na základě experimentů provedených na několika úlohách s rozdílnou obtížností. Míra elitismu $E_R \in (0,1; 0,3)$ se ukázala jako optimální pro různé úlohy a podstatně zlepšila schopnost algoritmu najít lepší řešení než bez použití elitismu. Vysoké hodnoty míry elitismu vedly k přílišné degradaci genotypu a algortimus uvízl v lokálních minimech. Míra mutace M_R má podobný vliv na konvergenci jako míra elitismu. Vhodná hodnota míry mutace $M_R \doteq 0,002$ umožňuje algoritmu vysvobození z lokálního minima a nalezení lepšího řešení, nicméně v nastavení, ve kterém mutace slouží pouze jako pomocný operátor, není zlepšení příliš významné. Na druhou stranu zvýšení hodnoty nad $M_R = 0,05$ vede k pozvolnému zhoršení chování algoritmu. Při dalším zvyšování hodnoty M_R algoritmus zcela ztrácí schopnost konvergovat a jeho charakter se mění na náhodné prohledávání.

V poslední kapitole jsme se věnovali jednotlivým operátorům mutace a rekombinačním operátorům. Nejprve jsme si představili některé operátory mutace, vysvětlili jejich principy a ukázali si jejich průběh na příkladech. Poté jsme porovnali jejich vliv na genetický algoritmus v sérii pokusů se zafixovanou vstupní populací, abychom co nejvíce omezili vliv rekombinačních operátorů. Ve všech pokusech dosahoval nejhorších výsledků algoritmus s operátorem PSM, který svým působením příliš zasahuje do struktury chromozomu. Nejlepší chování vykazovaly algoritmy využívající operátory mutace SSM a CIM. Je nutné si připomenout, že pro získání výsledků bylo nutné zvětšit počet generací a hlavně podstatně navýšit hodnotu míry mutace až na únosnou mez $M_R = 0,05$. Pokud za hlavní prohledávací mechanismu považujeme rekombinační operátory, a operátory mutace slouží jen jako pomocný mechanismu

proti uvíznutí v lokálním minimu, není příliš významné, jaký typ mutace algoritmus používá. Mnohem důležitější je správná volba hodnoty míry mutace M_R, kterou jsme si uvedli v předchozím odstavci.

Ve druhé části poslední kapitoly jsme se seznámili s rekombinačními operátory. Opět jsme si uvedli jednotlivé typy, vysvětlili jejich principy a předvedli na jednoduchých příkladech. Algoritmy využívající různé typy křížení jsme poté podrobili experimentům, s cílem identifikovat nejlepší ze zkoumaných operátorů. Na rozdíl od operátorů mutace, mají rekombinační operátory významný vliv na schopnost algoritmu konvergovat. V průběhu pokusů byly použity parametry s hodnotami, které byly zjištěny v průběhu této práce. Po vyhodnocení experimentů se jako nejlepší volba pro typ křížení jeví operátor OX. Pro zajímavost byl použit i algoritmus využívající 2opt heuristiku, který ze všech ostatních verzí algoritmu poskytuje nejlepší výsledky.

Jako nejlepší postup pro řešení problému obchodního cestujícího se jeví použití hybridního genetického algoritmu, který využívá heuristiku charakteristickou pro danou úlohu, s parametry získanými v této práci. Použití heuristiky v kombinaci se správně zvolenými parametry významně posiluje schopnost genetického algoritmu nalézt dobré řešení. Budoucí práce by se mohly zabývat hybridizací genetických algoritmů a použitím heuristik vhodných pro konkrétní problémy. Rozvinutí této práce by se mohlo zabývat například použitím heuristických rekombinačních operátorů nebo implementací heuristik vhodných pro řešení úlohy obchodního cestujícího, zvláště pak Lin-Kernighanovy heuristiky.

Reference

[1] Čada R., Kaiser T., Ryjáček Z., (2004), Diskrétní matematika. Plzeň : Západočeská univerzita, 170s. ISBN: 80-7082-939-7.

[2] Ryjáček Z., (2001); Teorie grafů a diskrétní optimalizace 2. Plzeň: Západočeská univerzita, 89s. *Dostupné na:* http://home.zcu.cz/∼ryjacek/students/TGD/TGD2.pdf.

[3] Christofides, N., (1976), Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, *Dostupné na:* https://apps.dtic.mil/sti/pdfs/ADA025602.pdf

[4] Goldberg D. E., (1989), Genetic Algorithms in Search, Optimization and Machine Learning (1st. ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[5] Hassanat, A.; Almohammadi, K.; Alkafaween, E.; Abunawas, E.; Hammouri, A.; Prasath, V.B.S., (2019) Choosing Mutation and Crossover Ratios for Genetic Algorithms — A Review with a New Dynamic Approach. Information 2019, 10, 390.

[6] Zhong J., Hu X., Zhang J. and Gu M., (2005), Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms, International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), pp. 1115-1121

[7] Lavinas Y., Aranha C., Sakurai C. and Ladeira M., (2018), Experimental Analysis of the Tournament Size on Genetic Algorithms, 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3647-3653

[8] De Jong, K.A. (1975), An analysis of the behavior of a class of genetic adaptive systems, (Doctoral dissertation, University of Michigan). Dissetation Abstracts International 36(10), 5140B. (University Microfilms No. 76-9381)

[9] Grefenstette, J., (1986), Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern., 16, 122–128.

[10] Deb, K.; Agrawal, S.,(1999), Understanding interactions among genetic algorithm parameters. Found. Genet. Algorithms, 5, 265–286.

[11] Vincent Edmondson L., (1993). Genetic algorithms with 3-parent crossover. (Doctoral dissertation, University of Missouri/Rolla, USA), Order Number: UMI Order No. GAX94-01909.

[12] Flegr J., (2007), Úvod do evoluční biologie. Praha: Academia, 544s., ISBN: 978-80-200-1539-6

[13] Reinelt, G. (1991). TSPLIB–A Traveling Salesman Problem Library. ORSA Journal on Computing, 3(4), 376–384.

[14] Sarma J., De Jong, K.A., (2003), C2.7 Generation gap methods. *Dostupné na:* www.shorturl.at/fsxGI

[15] Miller, B. L., Goldberg, D. E., (1995), Genetic algorithms, tournament selection, and the effects of noise. Complex systems, 9(3), 193-212.

[16] Abdoun, O., Abouchabaka, J., Tajani, C., (2012), Analyzing the performance of mutation operators to solve the travelling salesman problem. *Dostupné na:* https://doi.org/10.48550/arXiv.1203.3099

[17] Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., Dizdarevic, S., (1999), Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial intelligence review, 13(2), 129-170.

[18] Alkafaween, E., Hassanat, A., (2018), Improving TSP solutions using GA with a new hybrid mutation based on knowledge and randomness. *Dostupné na:* https://doi.org/10.48550/arXiv.1801.07233

[19] Larranaga, P., Kuijpers, C. M., Murga, R. H., Yurramendi, Y., (1996), Learning Bayesian network structures by searching for the best ordering with genetic algorithms. IEEE transactions on systems, man, and cybernetics-part A: systems and humans, 26(4), 487-493.

[20] Hussain A., Muhammad Y. S., Sajid M. N., Husssain I., Shoukry A. M., Gani S., (2017), Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator, Computational Intelligence and Neuroscience, Volume 2017, *Dostupné na:* https://doi.org/10.1155/2017/7430125

[21] Holland, J. H. (1992). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

[22] Davis, L. (1985, August). Applying adaptive algorithms to epistatic domains. In IJCAI (Vol. 85, pp. 162-164).

[23] Goldberg, D.E., Lingle, R., (1985), Alleles, Loci and the Traveling Salesman Problem. Proceedings of the First International Conference on Genetic Algorithms, 154-159.

[24] Croes, G. A. (1958). A method for solving traveling-salesman problems. Operations research, 6(6), 791-812.

[25] S. Lin, "Computer solutions of the traveling salesman problem,"in The Bell System Technical Journal, vol. 44, no. 10, pp. 2245-2269, Dec. 1965, *Dostupné na:* https://doi.org/10.1002/j.1538-7305.1965.tb04146.x

[26] Lin, S., Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research, 21(2), 498–516. *Dostupné na:* http://www.jstor.org/stable/169020

A Příloha

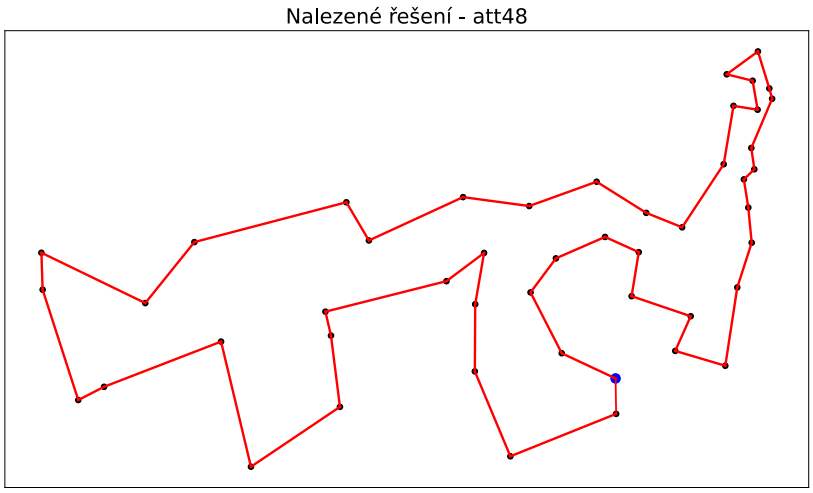
Tato sekce obsahuje ukázky grafů z *TSPLIB*, na kterých byly testovány genetické algoritmy. Jedná se o grafy att48, eil101 a tsp225. Na následujících grafech jsou zobrazena optimální řešení a pro ilustraci řešení nalezená genetickým algoritmem. V průběhu práce bylo ukázáno, že genetické algoritmy bez heuristik mohou dobře vyřešit úlohy, které svou obtížností odpovídají úloze att48. Pro graf tsp225 je vyobrazeno pouze optimální řešení, protože pro algoritmus je bez heuristiky nebo velmi velké populace téměř nemožné najít rozumné řešení v přijatelném čase a vyobrazení nalezeného řešení by bylo příliš nepřehledné.

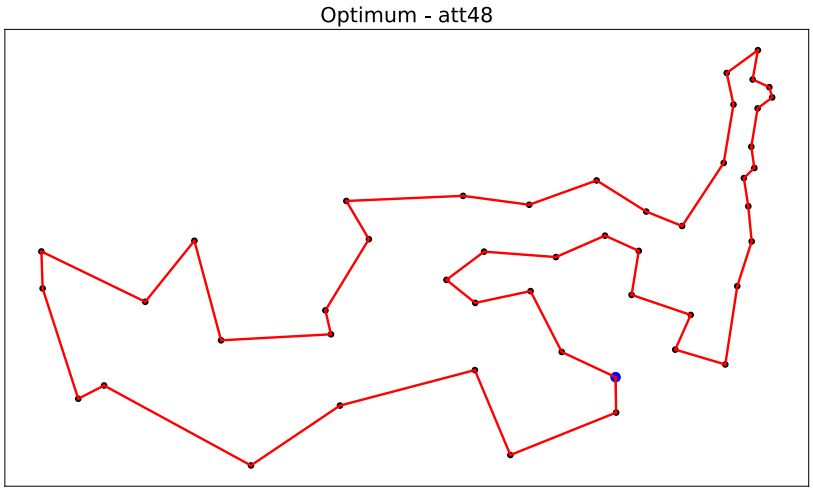
Pro graf att48 bylo nalezeno řešení, které je velmi blízko optimu a i při pohledu na přílohu A je zřejmé, že nalezené řešení je rozumné.

Příloha B zobrazuje graf eil101 se 101 vrcholy. Nalezené řešení se od optima už liší, nicméně je možné pozorovat poměrně rozumné struktury.

V příloze C můžeme pozorovat rozmístění měst v grafu tsp225. Tento graf je již příliš složitý pro nalezení řešení v rozumném čase, své uplatnění ale měl při optimalizaci parametrů.

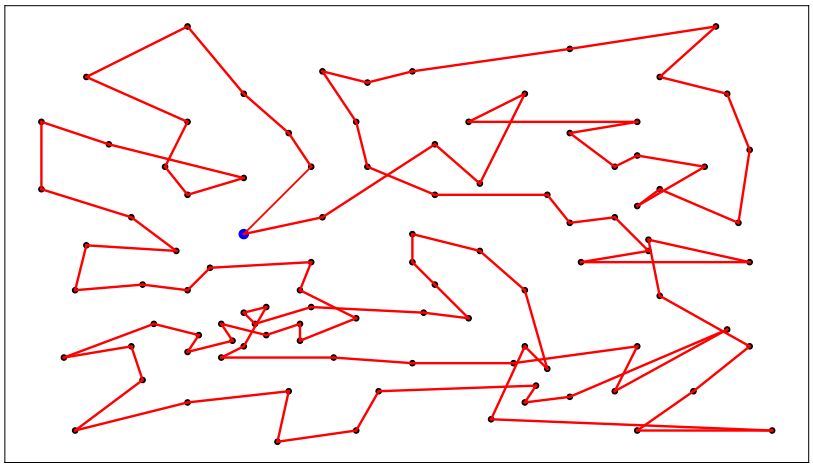
Příloha A - att48



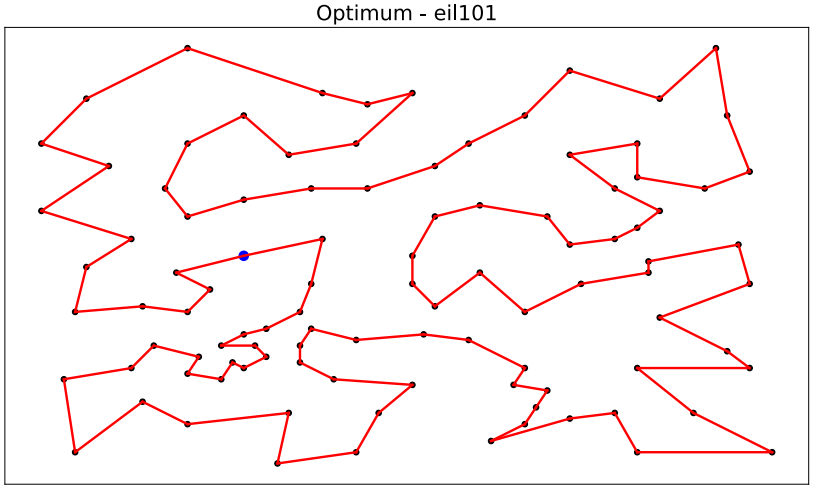


Příloha B - eil101

Nalezené řešení - eil101



Optimum - eil101



Příloha C - tsp225