Evoluční Optimalizační Algoritmy

HW01 - Traveling Salesman Problem

Nejdříve představím požadavky pro splnění základní úlohy, pak věci navíc, které jsem nepokryl. Git projektu: https://github.com/jsvobo/evo_tsp

Representation:

Jako vhodnou reprezentaci jsem zvolil **seznam indexů měst**. Tento seznam představuje pořadí navštívených měst. [1 3 2 0] znamená, že salesman jde z města 1 do 3 do 2 do 0 a zpět do 1 (abych předešel mýlení se seznamem inverzí).

Initialization:

Funkce definované v “representation.py”. V základu dvě varianty:

random_init() vytvoří náhodnou permutaci indexů od 0 do n-1, pro problém s n městy.
better_init() vybere nejlepší permutaci z 50 náhodných, začnu tak z lepší pozice bez nutnosti dělat jakékoliv kroky.

Po zkoušení těchto inicializací se ukázalo, že je prakticky vždy vhodné použít better_init, jelikož nemá tak vysoké výpočetní nároky (max. sekundy pro problém se 400 městy pro populaci 1000 cest). Pokud tedy neřeknu jinak, tak jsem použil tuto lepší inicializaci.

Perturbation operations:

V základu využívám 3 základní perturbace, definované ve složce perturbations.
perturb_switch(i,j) prohodí dvě prvky na indexes i,j.
perturb_move(i,j) vloží prvek z pozice i na pozici j.
perturb_invert(i,j) “vysekne” podposloupnost mezi prvky i,j (i<j) a obrátí ji.

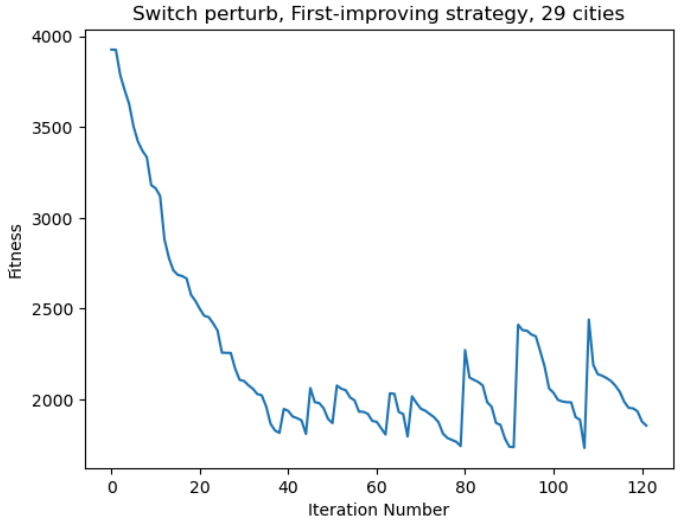
Perturbace jsou implementovány tak, abych jim akorát zadal indexy a “pustil” je na seznam měst. Indexy i, j vybírám uniformě z 0..n tak, že **i != j**. Pro případ best-improving local search se počítají perturbace pro každou takovou dvojici indexů (lehce enumerováno).

Local search:

Implementoval jsem několik variant lokálního prohledávání v “ls.py”.
first_improvement_ls používá first improvement strategy s náhodnou perturbací. K dispozici jsou 3 perturbace (výše). Také se mi osvědčilo, že pokud “dlouho” nemůžu zlepšit fitness momentálního řešení (např. po 500 evaluací), pak se vydám náhodným směrem (přijmu další perturbaci, i když nezlepšuje hodnotu řešení). Toto pomáhá dostat se z

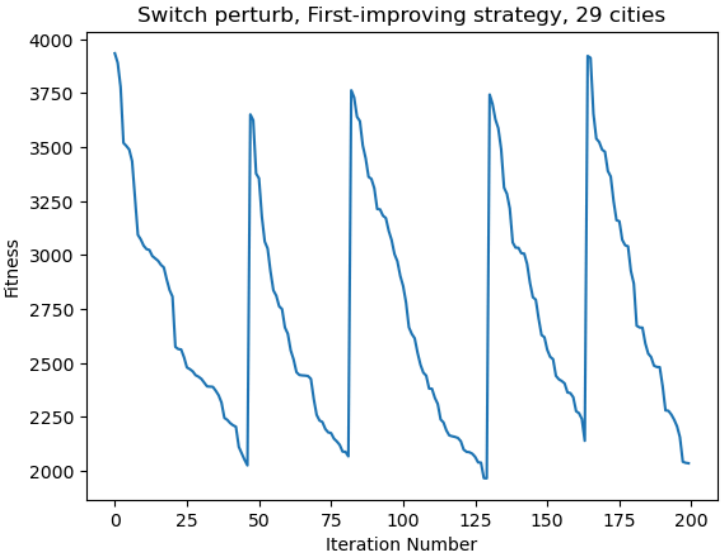
lokálních extrémů. Pozoroval jsem, že fitness tohoto blízkého řešení obvykle bývá jen o trochu horší. Pokud místo náhodné perturbace použiji náhodnou inicializaci, hodnota řešení se zpravidla zhorší drasticky.

Na obrázku: Fitness závislá na počtu udělaných kroků. Náhodná perturbace nastane po 500 neúspěšných evaluacích. Při tomto vyhodnocení byly evaluace fitness funkce omezeny na celkem 5000. Použitá perturbace je “switch”.



Obrázek 1. Zlepšující se hodnota řešení, kdy po 500 neúspěších proběhne náhodný krok.

Na druhém obrázku je vidět jak se local search chová, když reinicializujeme current solution při dlouhém neúspěchu namísto pouhé náhodné perturbace. Hodnota řešení velice klesne a “vyčerpá” spoustu evaluací na to, aby se dostala na podobnou hodnotu jako před restartem, ale hodně se vzdálí od tohoto lokálního minima, takže hypoteticky může snáze dosáhnout “lepšího regionu” prostoru řešení.



Obrázek 1. Zlepšující se hodnota řešení, kdy po 500 neúspěších proběhne úplná reinicializace.

V tabulce (níže) jsou uvedeny střední hodnoty a standardní odchylka přes 50 opakování LS s různým chováním, pokud dlouho nedojde ke zlepšení (500 evaluací). Tato úloha má

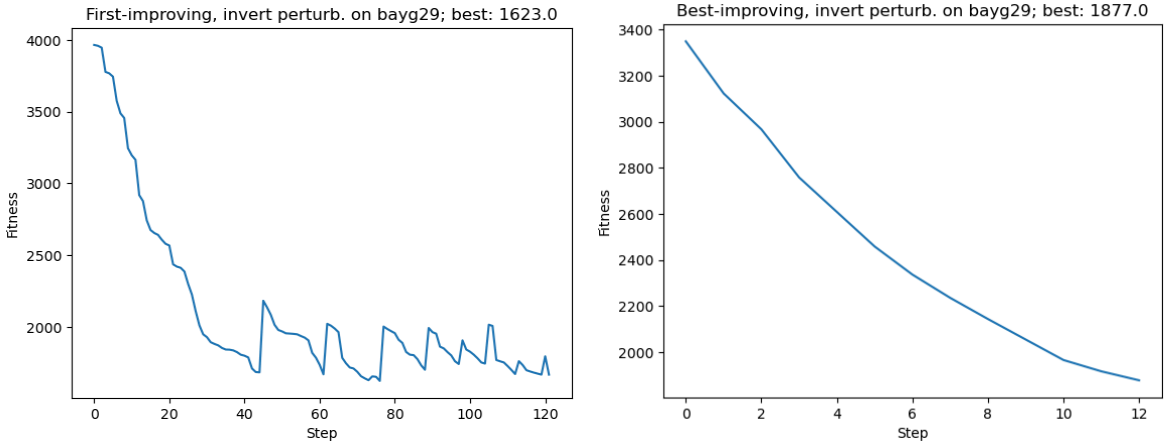
optimální řešení s hodnotou **1610**. Pro každý běh bylo nastaveno maximum na 10 000 evaluací.
Úplně nejlepších výsledků dosahuje **invert** perturbace bez reinicializace (pouze náhodně přeskočí). Také se ukazuje, že alespoň pro tuto úlohu je úplný restart zpravidla horší.

	Switch	Move	Invert
Re-inicializace	2020.0 ± 150.1	1965.0 ± 128.6	1675.7 ± 38.8
Random perturb.	1818.4 ± 74.2	1884.5 ± 128.2	**1642.9 ± 21.7**
Beze změny	1991.92 ± 154.9	1983.58 ± 133.3	1686.72 ± 36.1

Tabulka 1. Průměrné výsledky různých strategií reinicializace pro first-improving LS

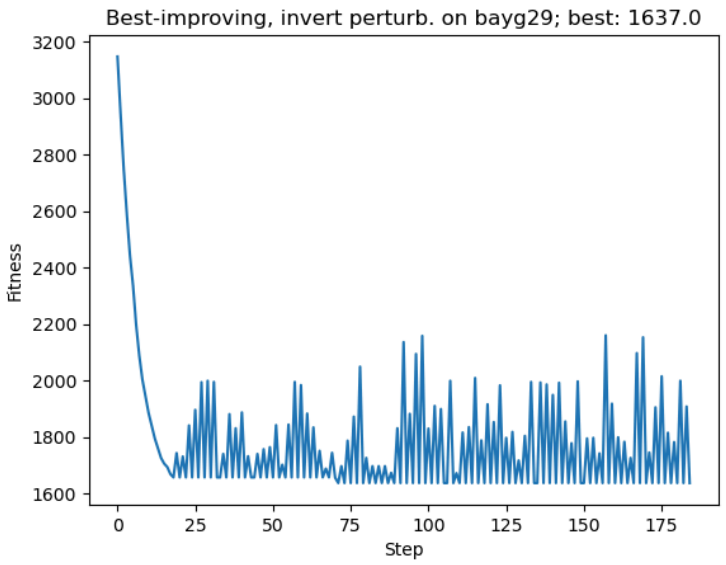
Alternativou je nedělat nic a doufat, že se časem najde lepší řešení náhodnou perturbací (třetí řádek tabulky). Tato metoda vykazuje horší výsledky než použití náhodné perturbace (někdy i horší než náhodná reinicializace). Myslím si, že přesné nastavení limitu vyhodnocení pro jeden krok může vést ke zlepšení prvních dvou metod.

Druhý typ ls, který zkouším je best-improving strategy (**best_improvement_ls** v kódu). Propočítám všechny perturbace z daného stavu a vyberu tu nejlepší. Pokud žádná permutace nevede ke zlepšení, tak jednu náhodně vyberu a opět zkouším vylepšit dalším krokem (tím se často vrátím do téhož minima, ale někdy vede k nalezení lepšího. Porovnání na více instancích bude na jednom místě později, po crossoverech a EA.



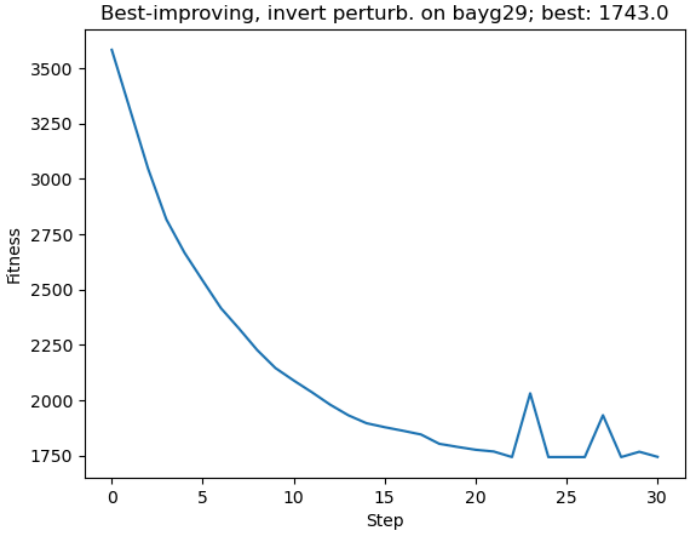
Obrázek 3. Porovnání zlepšujících kroků First-improving a best improving lokálních prohledávání

Bližší výsledky k best-improving vs first-improving strategy budou v tabulce s přehledem, ale zde chci zmínit, že pro mně best-improvement nedosáhlo tak dobrých výsledků. Celkově dojde k daleko méně iteracím, jelikož je každá velice náročná. Toto by se dalo částečně vyřešit pamatováním již prohledaných řešení. Na obrázcích nahoře je porovnání běhů těchto dvou lokálních prohledávání s perturbací invert a 10 000 evaluacemi.
Z pravé části obrázku 3 se zdá, že algoritmus se má ještě kam zlepšovat. pokud při úplně stejném nastavení spustíme algoritmus s max evaluacemi na 150 000 (15x tolik), pak výsledné zlepšení vypadá jako na obrázku 4 (níže).



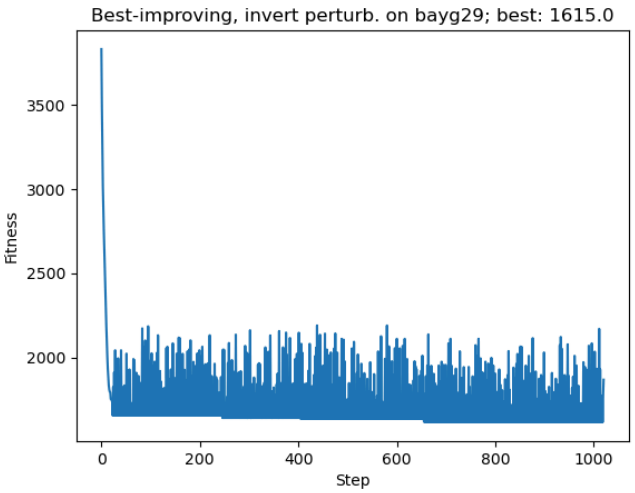
Obrázek 4. Zlepšující se hodnota řešení best-improving strategy se 150k eval.

Zde je vidět, jak eventuálně vyjedeme z lokálního minima a přeskočíme na trochu lepší minimum, ze kterého se už nepodaří vyskočit.
Dopsal jsem vylepšení, kdy si zaznamenám fitness pro již prošlá řešení a nemusím je znovu počítat. To mi dovolí udělat více kroků, než mi “dojdou” pokusy. Na obrázku 5 jsem opakoval experiment s vylepšenou verzí best-improvement search a 10 000 evaluacemi. došel jsem do 30 kroků, kd jsem předtím “ušel” jen 12 kroků. Ve finálním porovnání budu používat pouze tuto verzi.



Obrázek 5. Zlepšující se hodnota řešení, Best-improving strategy s pamatováním.
10k evaluací, optimum 1610.

Tato lepší verze je schopná udělat 1000 kroků se 150k evaluacemi, kde předtím udělala necelých 200 kroků (Obrázek 6 vs obrázek 4)



Obrázek 6. Zlepšující se hodnota řešení, Best-improving strategy s pamatováním. 150k evaluací, optimum 1610.

Crossover:

Používám dva crossovery implementované ve složce “cross/crossovers.py”.
První je **sequential constructive crossover (SCX)** , který jsem ve více článcích viděl jako nejlepší (https://www.mecs-press.org/ijisa/ijisa-v7-n11/IJISA-V7-N11-3.pdf). Tento crossover konstruuje cestu po jednom skoku, kde přednostně bere hrany z obou rodičů (nebo i více, u mně ze 2). Např. vyberu startovní město 3. Poté se kouknu v rodičích, co následuje po 3. jeden z rodičů obsahuje 3->5 (délka 3) a druhý 3->8 (délka 6). vyberu tedy hranu 3->5. opět koukám, jaká hrana vede v obou rodičích z 5 dále a vybírám levnější. Pokud je nějaký z vrcholů použitý, vezmu hranu do dalšího z měst v rodiči. Takto může tento crossover přidat hrany, které nejsou v žádném z rodičů, ale vylepšují celkovou cenu.

Druhý je **partially mapped crossover (PMX)** ,který vybere dvojici indexů i,j **(i!=j)**, zkopíruje jednoho z rodičů a mezi těmito indexy se pokusí nahradit úsek cesty z druhého rodiče. Pokud by mělo dojít k duplikaci nějakého města, algoritmus použije výměnu měst uvnitř potomka, aby zajistil, že se město, které by jinak bylo přemazáno pouze vymění s tím vloženým. Tento crossover je de-facto upravený dvoubodový crossover, ale zachová validitu řešení díky prohození měst.
Detaily například zde:
https://user.ceng.metu.edu.tr/~ucoluk/research/publications/tspnew.pdf

EA:

Inicializace:
Na začátku vyberu populaci pomocí lepší z mých inicializačních metod (vyberu nejlepší z 50 náhodných inicializací).

Selekce:

Zvolil jsem jednoduše turnajovou selekci. velikost turnaje přizpůsobuji velikosti generage. Větší turnaj spíše vybere jednoho/ více z top řešení, proto tímto můžu zvyšovat tlak na přežití nejsilnějšího.

Replacement:
Vybral jsem steady state replacement, kdy přimíchám potomky ke staré generaci, seřadím je a vyberu novou populaci z nejlepších.

Crossovery vysvětleny v sekci výše. Možné mutace pouze jako perturbace zmíněné v LS. Po experimentech s LS se ukázalo, že **inverze subsekvence** měst performuje nejlépe, proto jsem zvolil jako jedinou perturbaci pro EA.

Upozorňuji, že pro účely následujícího porovnání proběhne ladění různých aspektů mých EA, speciálně velikosti generace, prob_crossover, prob_mutation, chtěný počet potomků (resp. kolik dvojic rodičů vytvořím), velikost turnaje. Tyto parametry jsem ladil na prostřední úloze (jelikož je velikostí cca ve středu), nesahám na ně pak pro jednotlivé úlohy.

Porovnání LS a EA:

v následující tabulce jsou uvedené hodnoty dosažených řešení pro různé algoritmy (ve sloupci) pro tři různé (pečlivě vybrané) instance z tsplib. “brg180” má velice nízké optimum, ale při náhodné inicializaci dosahuje délka cesty cca 800k. Některé cesty mají délku stovek tisíc (a neměly by být zahrány, brg180 představuje hru bridge, to ale mé modely apriori neví). Proto je výsledná fitness pro některé řešení extrémně daleko od optima.

Problém:	**bayg29**	**brazil58**	**brg180**
Optimum	1610	25 395	1950
Počet měst	29	58	180
First-Improvement * switch	1 765 ± 61	34 929 ± 2339	67 694 ± 10560
First-Improvement * move	1 798 ± 69	33 362 ± 2139	54 253 ± 15442
First-Improvement * invert	1 629 ± 14	26 240 ± 442	5 366 ± 1660
Best-Improving ** switch	1 941 ± 96	56 935 ± 3191	735 379 ± 25 631
Best-Improving ** move	1 884 ± 79	62 224 ± 3381	742 400 ± 26 936
Best-Improving ** invert	1 660 ± 37	50 205 ± 1978	770 439 ± 18641
EA - SCX	1 695 ± 35	32 981 ± 1189	135 549 ± 12 113

EA - PMX	1 709 ± 37	38 808 ± 1721	282 894 ± 19 638

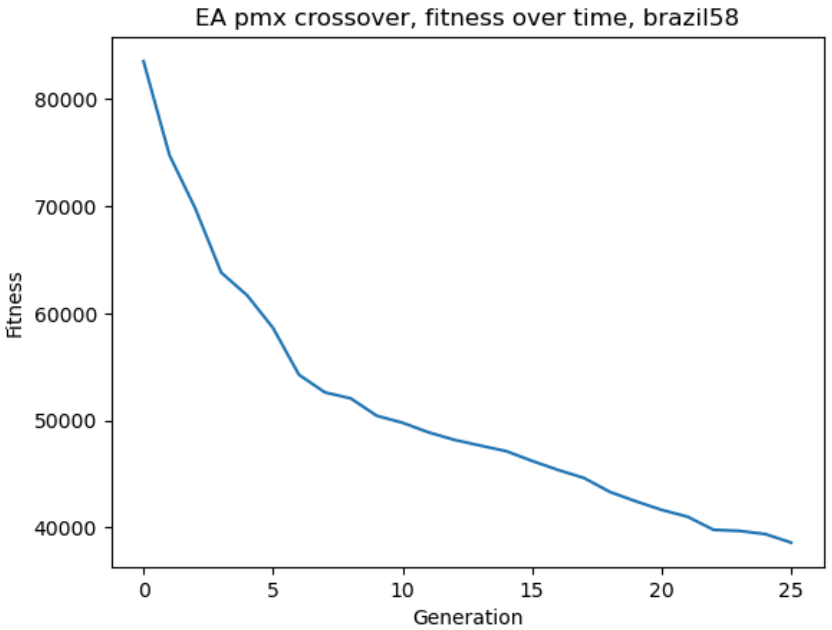
*Pro first improvement lokální prohledávání beru v potaz, že po 500 krocích kdy nedojde ke zlepšení přijmu další perturbaci (probráno výše). Hodnota nastavena na základě krátkých testů.

** Pro best-improving lokální prohledávání používám vylepšenou variantu s pamatováním předchozích vyhodnocení.

Tabulka 2. Průměrné výsledky pro všechny obsáhlé algoritmy na třech instancích TSP.
V zelené je nejlepší alg. pro každou úlohu, v červené nejhorší.

Pro každý běh je k dispozici **20 tis.** evaluací. Výsledek je střední hodnota a std. odchylka z 10 nezávislých běhů algoritmu. 10 je relativně málo, ale kvůli dlouhému běhu EA jsem musel mírně slevit (např. z 50). Pro všechny inicializace (local search i EA) jsem použil “lepší” inicializaci (výše), kdy vyberu nejlepší instanci z 50 náhodných.

Vybral jsem 3 úlohy tak, aby měly různé velikosti. Různé algoritmy reagují různým způsobem na tyto větší úlohy. Například best-improvement extrémně zaostává s velikou maticí (2. a speciálně 3. sloupec),asi tím, že vyplítvá spoustu vyhodnocení na bezcenná řešení. First improving LS (s invert perturbací) si vcelku dokáže poradit i s velikou maticí. Další pozorování: PMX crossover je daleko rychlejší než SCX, ale performuje hůře. Myslím si, že v ideálním případě bych zvládl automaticky optimalizovat hyperparametry EA pro každou úlohu zvlášť. K mému smutku neperformují EA lépe než koncepčně i technicky jednodušší (a také hodně rychlejší) first-improving LS. Myslím si, že je to částečně tím, že LS má daleko méně hyperparametrů, které jsem nezvládl u EA dobře vybrat. Také se zdá, že 20 tis. je málo vyhodnocení pro velký EA algoritmus.

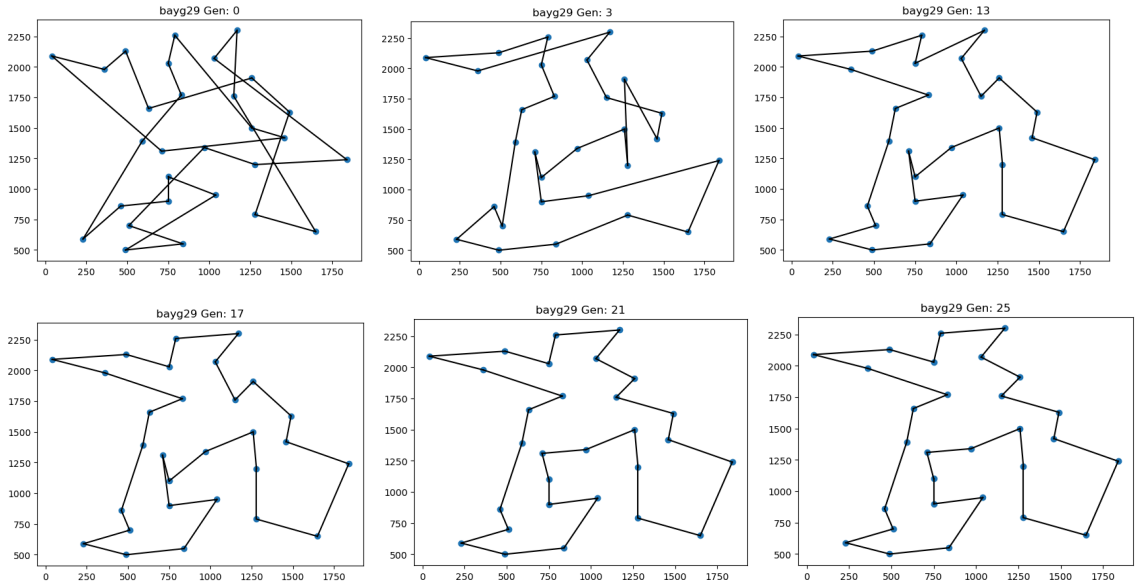


Obrázek 7. Zlepšující se fitness s rostoucí generací. EA s PMX crossoverem a invert perturbací.

Bonusové úkoly (které zatím nikde nejsou):

Vizualizace:

Vizualizace zlepšování jednotlivých řešení jsou již obsaženy výše (Obrázky 1,2,3,4,5,6). Dále jsem také zpracoval vizualizaci řešení na mapě, speciálně pro TSP instance, které představují geografii. Pro úlohu bayg29 jsem v jupyter notebooku visualisations_ea.ipynb sestavil skript co ukáže nejlepší řešení v každé generaci. Výsek ze začátku a konce ukážu zde v Obrázku 8 (níže):



Obrázek 8. nejlepší cesta generací 0, 3, 13, 17, 21, 25

Je vidět, že k velkému zlepšení dochází v prvních pár generacích, poslední generace mají stejného nejlepšího jedince. Pro SCX crossover s parametry jako pro vypočtení tabulky (20 tis. evaluací. etc.). Výsledná fitness tohoto běhu je 1707, cca 100 od optima.

Konstruktivní heuristika:

S informací z distance_matrix můžu vyzkoušet různé věci. Napadlo mně konstruovat počáteční řešení podobně jako funguje SCX crossover v EA. Budu vybírat nejlevnější dostupný edge z posledního přidaného města sekvence.

	bayg29	**brazil58**	**brg180**
Random init	4 692 ± 337	123 851 ± 6 880	914 259 ± 45 305
Nejlepší z 50 random	3 934 ± 150	106 546 ± 3 412	801 193 ± 17 460
Konstruktivní inicializace	2 064 ± 91	29 927 ± 2 226	12 907 ± 441

Tabulka 3. Porovnání různých inicializací na stejných úlohách. Průměr přes 50 inicializací

Tato inicializace se sama chová jako jeden z nejlepších algoritmů pro vyhledávání. Tím že začnu z náhodného indexu, tak není výsledek inicializace deterministický. Dala by se zavést náhodnost do výběru z nejlevnějších cest, pak by tato inicializace byla vhodnější i pro inicializaci veliké populace. takto s ebudou často vracet stejná řešení. Chtěl jsem, aby se jednotlivé algoritmy projevily a nezačínaly rovnou z tak dobré pozice, proto jsem nepoužíval tuto incializaci během celého vyhodnocení. Myslím si, že k nejlepší kombinaci povede tato inicializace s nejlepším LS (first improvement, random perturbace po 500 nezlepšujících krocích, invert perturbace). Výsledek na těchto třech problémech v Tabulce 4 (níže).

	bayg29	**brazil58**	**brg180**
Optimum	1610	25 395	1950
First-improving LS	1629 ± 8	26111 ± 390	4472 ± 1708
Best-Improving LS	1650 ± 17	26018 ± 315	3349 ± 1697

Tabulka 4. Výsledky pro vylepšenou inicializaci pro LS se stejnými podmínkami jako v tabulce 2. Průměr a std. z 10 běhů.

Výsledky pro First-improvement LS jsou srovnatelné s ostatními běhy tohoto algoritmu (s horší inicializací). ve finále alg. pouze začne z trochu lepší pozice, takže před 20 tis. kroků dojdou do podobných hodnot lokálního minima. Tato inicializace ale velice pomůže best-improving LS, kde se normálně spoustu evaluací spotřebuje na velice špatná řešení daleko od optima. Best-Improvement na poslední úloze 5x z 10 dosáhla skóre 1960 (téměř optimum!). Oproti původní hodnotě minima (cca 700 tisíc pro třetí úlohu) je toto minimum skvělé. Dokonce nejlepší ze všech.

Pro přehlednost jsem sestavil tabulku se všemi výsledky na konci dokumentu.

Nesymetrické problémy:

Chtěl jsem jen velice krátce prozkoumat asymetrické problémy. Myslím si, že se pertrubace pro LS budou chovat velice jinak, protože projití měst v opačném směru může radikálně změnit fitness. Např. invert perturbace (která performovala zdaleka nejlépe) zároveň obrátí cesty v celé podposloupnosti, která je vybrána, namísto pouhé změny hodnoty na krajích podposloupnosti.
Navrhuji alternativní perturbaci pro LS na asymetrických TSP. Pokud omezím indexy, mezi kterými může proběhnout inverze podposloupnosti, mohl bych zamezit velkým odskokům od vhodnějších řešení. Toto můžu udělat více způsoby, např. vybrat první index a druhý omezit na rozmezí okolo. Já zvolil samplování z binomického rozdělení okolo prvního indexu (vybraného uniformně). Takto s větší pravděpodobností vyberu blízké dvojce indexů a změna jedné perturbace nebude tak velká. V tabulce 5 (níže) porovnávám 3 základní perturbace + moji novou na jedné úloze ATSP.

	ftv38 (má ale 39 měst)

Optimum:	1530
invert	2240.8 174
speciální invert	2678.1 211
move	1959.1 96
switch	1926.6 69

Tabulka 5. Porovnání invert perturbací s různou generací indexů pro asymetrický problém. First improvement LS s jinými perturbacemi

Pro ukázku rozdílů mezi operacemi jsem použil **úplně náhodnou inicializaci**, 20 tis. kroků. průměr a std. přes 10 běhů alg., jako v ostatních tabulkách. Zkoušena pouze first improving strategy LS, kteýr vezme náhodnou perturbaci po 500 neúspěšných. Ukazuje se, že invert pro tuto úlohu opravdu není vhodnou perturbací, naopak performuje nejhůře ze standardních perturbací. Můj perturbační operátor (invert subsequence s menším rozptylem indexů) ale nepracuje lépe než standartní invert. Stále si myslím, že stojí za prozkoumání nějak manipulovat indexy, ale např. méně omezovat samplig.

Závěr: Za co si zasloužím body?

Popis	Počet bodů
Základní zadání (1 LS, 1 EA, 3 TSPLIB,dokumentace - zde i jupytery)	5 bodů
3 jiné perturbace pro LS (invert, switch,move)	+1
Prozkoumání a porovnání vylepšení pro first-improvement LS (různé chováním při opakovaném neúspěchu, Tabulka 1; heuristická inicializace)	+1
Prozkoumání vylepšení pro best-improvement LS (verze s pamatováním minulých fitness evaluací, heuristická inicializace)	+1 (ano, chci 3 jiné bonusové body za LS, ale myslím si, že jsem dal hodně práce do porovnání variant a zpracování jednotlivých vylepšení pro každý alg, které jsem i vícekrát přepsal, včetně 3 jiných perturbací a ještě jsem je porovnal i s nejlepší inicializací)
2 jiné EA (zadané dvěma crossovery)	+1
Vizualizace zlepšování řešení + vizualizace	+1

jednotlivých řešení u geografických TSP	
Konstruktivní heuristika (a porovnání s ostatními inicializacemi jako samostatná metoda řešení úloh)	+1
Navržení a prozkoumání perturbace pro asymetrický problém (na nejlepším LS).	+1 (i když můj operátor nebyl lepší, alespoň jsem se nad tím zamyslel?)
Suma:	**5+7 bodů**

Přehled složené tabulky výsledků:

Problém:	**bayg29**	**brazil58**	**brg180**
Optimum	1610	25 395	1950
Počet měst	29	58	180
First-Improvement switch	1 765 ± 61	34 929 ± 2339	67 694 ± 10560
First-Improvement move	1 798 ± 69	33 362 ± 2139	54 253 ± 15442
First-Improvement invert	1 629 ± 14	26 240 ± 442	5 366 ± 1660
Best-Improving switch	1 941 ± 96	56 935 ± 3191	735 379 ± 25 631
Best-Improving move	1 884 ± 79	62 224 ± 3381	742 400 ± 26 936
Best-Improving invert	1 660 ± 37	50 205 ± 1978	770 439 ± 18641
EA - SCX	1 695 ± 35	32 981 ± 1189	135 549 ± 12 113
EA - PMX	1 709 ± 37	38 808 ± 1721	282 894 ± 19 638

Avg přes 50 inicializací:

Random init	4 692 ± 337	123 851 ± 6 880	914 259 ± 45 305
Nejlepší z 50 random	3 934 ± 150	106 546 ± 3 412	801 193 ± 17 460
Konstruktivní inicializace	2 064 ± 91	29 927 ± 2 226	12 907 ± 441

Avg přes 10 běhů s nejlepší inicializací:

First-improving s konstruktivní inicializací, invert	1629 ± 8	26 111 ± 390	4472 ± 1708
Best-Improving s konstruktivní inicializací,invert	1650 ± 17	26 018 ± 315	3349 ± 1697