NI-KOP

Úloha 4
Simulované ochladzovanie

Bc. Matej Šutý
ČVUT FIT 2021

Popis algoritmu simulovaného ochladzovania

Generovanie počiatočného stavu:
Algoritmus najprv generuje platný stav - naplnený batoh. Generuje ho pomocou náhodného výberu náhodného počtu predmetov, ktoré sa pokúsi vložiť do batohu. Ak má batoh prijateľnú hmotnosť, stav sa použije. Ak nie, generácie sa opakuje až kým počet pokusov neprekročí hranicu n*10, kde n predstavuje počet predmetov. Ak sa nepodarí vygenerovať platný stav, ako stav sa zvolí prázdny batoh.

Vstupné parametre:
Parametre, ktoré ovplyvňujú beh programu sú počiatočná teplota T, faktor ochladzovania k, finálna teplota t0 a počet iterácií vo vnútornom cykle I. Každý beh má nastavený maximálny počet iterácií rovný 100*(počet predmetov).

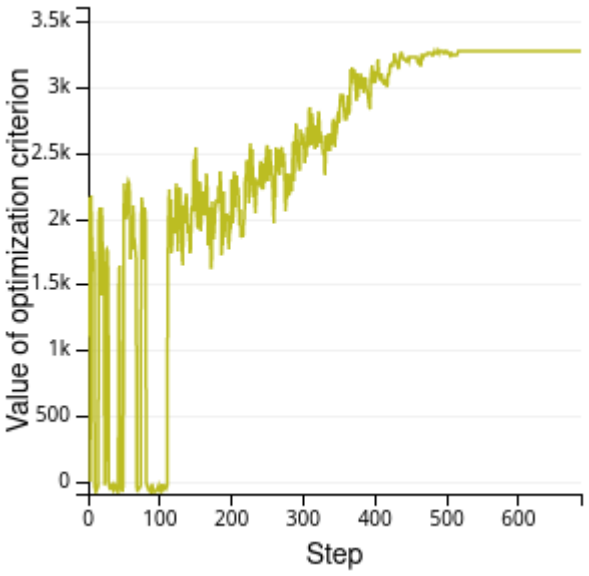
Prehľadávanie okolia:
Vo vnútornej slučke s I iteráciami sa generujú stavy v okolí súčasného stavu - náhodne sa zvolí jeden predmet, ktorý nie je v batohu v aktuálnom stave a pridá sa do batohu. Ak tento stav nie je platný - jeho hmotnosť prekročila kapacitu - stav sa opraví pomocou odoberania náhodných predmetov až kým nie je hmotnosť nižšia alebo rovná maximálnej kapacite. Ak je tento vygenerovaný stav lepší (cena batohu je vyššia) ako predchádzajúci stav, automaticky sa prijme a použije sa v ďalšej iterácii. Ak je cena batohu v novom stave nižšia, algoritmus s istou pravdepodobnosťou prijme aj zhoršujúci stav. Nazvime rozdiel v cene batohov d. Algoritmus vygeneruje náhodné číslo x v rozmedzí [0, 1]. Ak je $x < \exp(d / T)$, algoritmus prijme nový stav aj keď má nižšiu cenu. Vo vnútornej slučke algoritmus počíta počet prijatých stavov číslom a - pri prijatí zvýši a o jedna, pri odmietnutí stavu ho zníži o jedna. Ak sa a rovná I/2, vnútorná slučka sa opustí a nastáva proces ochladzovania. Táto slučka sa opustí aj vtedy, keď prejde I iterácií.

Chladenie:
Proces ochladzovania je znižovanie teploty podľa geometrického radu s koeficientom faktor ochladzovania k - pri každom ochladení sa aktuálna teplota vynásobí faktorom ochladzovania k ($T = k*T$). V celom procese simulovaného ochladzovania si algoritmus uchováva najlepšie nájdené riešenie, ktoré potom vráti ako riešenie.

Analýza vstupných parametrov:

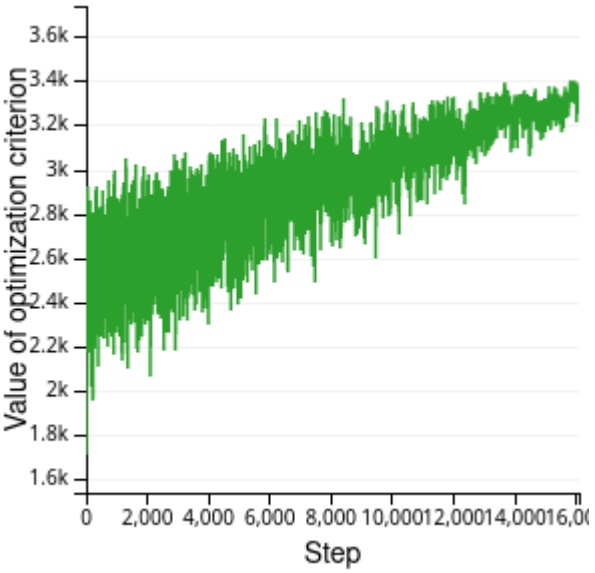
Vstupná teplota T

Vstupná teplota ovplyvňuje výber zhoršujúcich stavov - pri vyššej teplote je vyššia pravdepodobnosť výberu zhoršujúceho stavu. Vyššia teplota umožňuje lepšiu diverzifikáciu a prehľadávanie stavového priestoru. Na druhej strane môže spomaliť výpočet kvôli dlhému ochladzovaniu. Na grafe SA môžeme vidieť vplyv vysokej vstupnej teploty na začiatku grafu, kde sú veľké výkyvy medzi cenami stavov - algoritmus často prijíma horšie stavy.
Na obrázku je vidno, že algoritmus na začiatku častokrát prijímal aj “zbytočné” stavy, ktoré mali nulovú cenu.



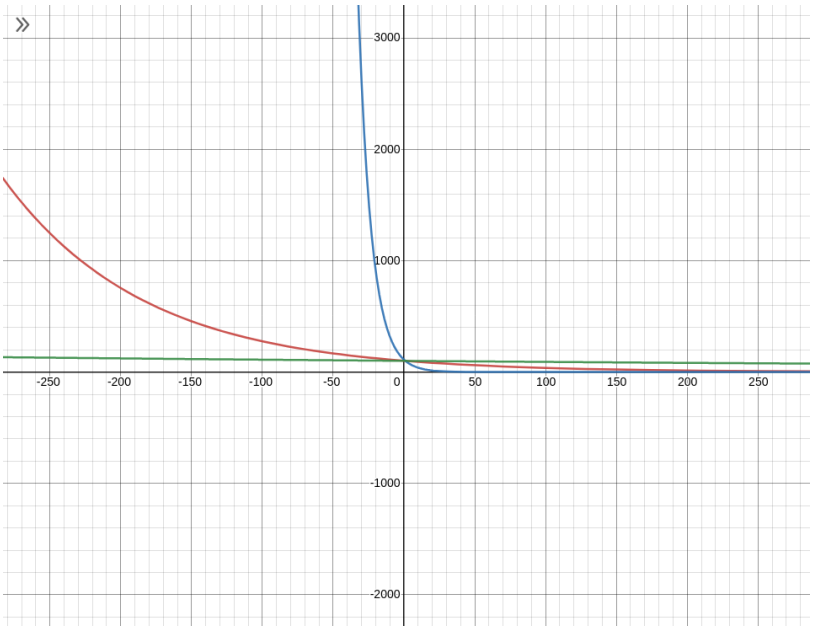
Koncová teplota TF

Algoritmus sa zastaví, keď sa súčasná teplota dostane pod koncovú teplotu. Koncová teplota ovplyvňuje dĺžku behu algoritmu. Keďže proces ochladzovania v mojom algoritme je geometrická postupnosť teplôt s koeficientom k, je dôležité si uvedomiť, že zníženie koncovej teploty o jeden stupeň napr. z 50 na 49 nie je to isté ako zníženie z 1 na 0. Pri vyšších teplotách sa chladí rýchlejšie ako pri nízkych. Pravdepodobnosť výberu zhoršujúceho riešenia závisí na aktuálnej teplote, na rozdiele v cene (optimalizačnom kritériu) stavov (podiel v exponente). To znamená, že ak ponecháme vysokú hodnotu koncovej teploty algoritmus môže aj v záverečnej fáze často vyberať zhoršujúce riešenie, čo nie je žiadané. Na priebehu SA to môžeme vidieť vtedy, keď cena stavov skáče často s veľkými rozdielmi aj na konci stavov - želaný priebeh je naopak konvergencia k nejakému optimu. Na obrázku je vidno, že algoritmus na konci ešte stále prehľadával široké okolie (diverzifikácia) a nestihol skúšať blízke okolie potencionálneho extrémenho stavu (intenzifikácia).



Počet vnútorných iterácií l
Algoritmus prehľadáva svoje okolie vo vnútornej slučke pri konštantnej teplote. Podľa počtu vnútorných iterácií môže algoritmus prehľadávať svoje okolie do široka (pridávať desiatky predmetov) alebo iba v menšom okolí (postupné pridávanie jedného predmetu za druhým). Teplota sa znižuje až keď algoritmus opustí vnútornú slučku (prešiel všetky iterácie alebo prijal veľa zlepšujúcich stavov).

Faktor ochladzovania k
Faktor ochladzovania určuje rýchlosť chladenia - pri nižšom faktore sa teplota znižuje rýchlejšie. Rýchlejšie chladenie znamená kratší beh algoritmu ale tiež menší priestor na prehľadávanie okolia, čo môže spôsobiť uviaznutie v lokálnom extréme.
Modrá k = 0.9, červená k = 0.99, zelená k = 0.999.



Popis instancií:

White box fáza

Instancia 100-3:
Generovaný problém so 100 predmetmi, ktorých sumárna váha je 500, kapacita batohu 200, a maximálna cena predmetu 100, optimálna cena batohu je 3429 (vyrátané pomocou cenovej dekompozície).

Instancia 40-1:
Problém z datasetu ZKC, o ktorom nie je nič známe. Optimálna cena batohu 32413.

Black box fáza

100-BB (10 instancií):
Generované problémy so 100 predmetmi, sumárna váha neznáma, kapacita batohu 500, a maximálna cena predmetu 100.

Instancia 40-BB (10 instancií):
Problémy z datasetu ZKC, o ktorých nie je nič známe.

Vplyv faktoru ochladzovania

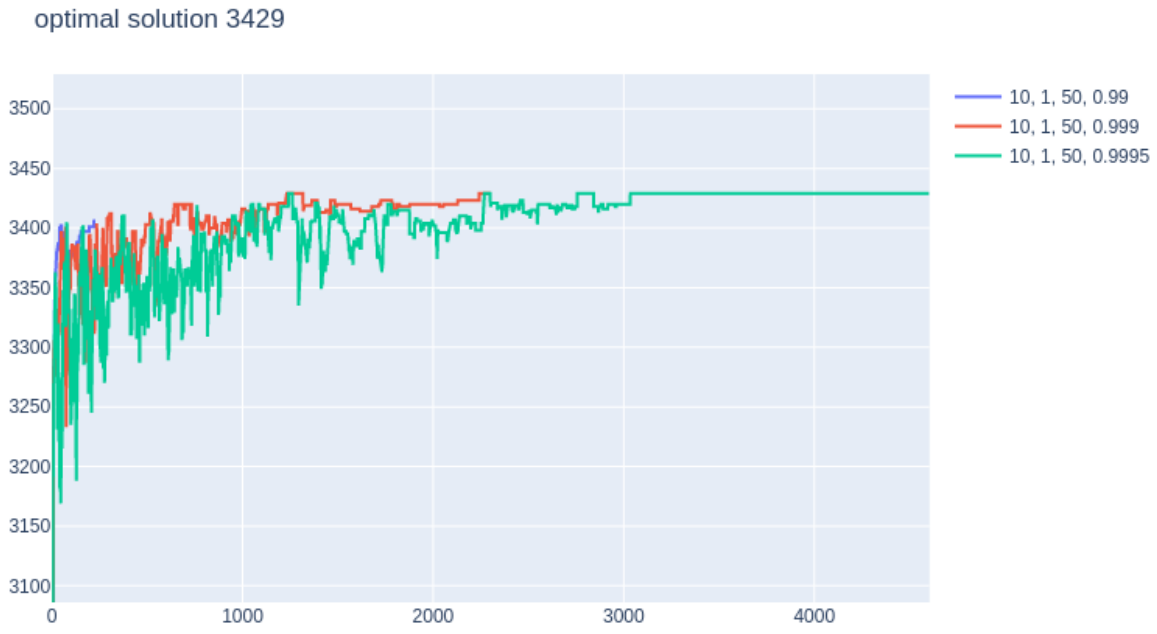
V tabuľke nižšie sú zaznamenané experimenty z white-box fázy, kde sledujem správanie heuristiky a popr. upravujem algoritmus, ak pre to nájdem dobrý dôvod. Počas tohto experimentu som zistil, že by mohlo pomôcť ak implementujem niečo ako *equilibrium* funkciu, ktorá zabráni príliš pomalému ochladzovania. Do kódu som preto pridal podmienku, že ak je veľa prijatých zlepšených riešenie vo vnútornej slučke, slučka sa predčasne opustí a teplota sa zníži.

Pre porovnanie som rovnaké inštancie spustil aj na webovom rozhraní https://ddd.fit.cvut.cz/KOP_App/#/ a ich riešenie (s rovnakými parametrami) som nazval referencia web. Keďže nepoznám detaily implementácie, použil som rovnaké vstupné parametry ako pri mojom riešení. V ďalších experimentoch je aj porovnanie výpočtovej zložitosti (počet navštívených stavov).
Z tabuľky je vidno, že počet navštívených stavov závisí najmä na faktore ochladzovania, lineárne na počtu vnútorných iterácií (násobky nie sú presné, pretože algoritmus občas predčasne opustí slučku).

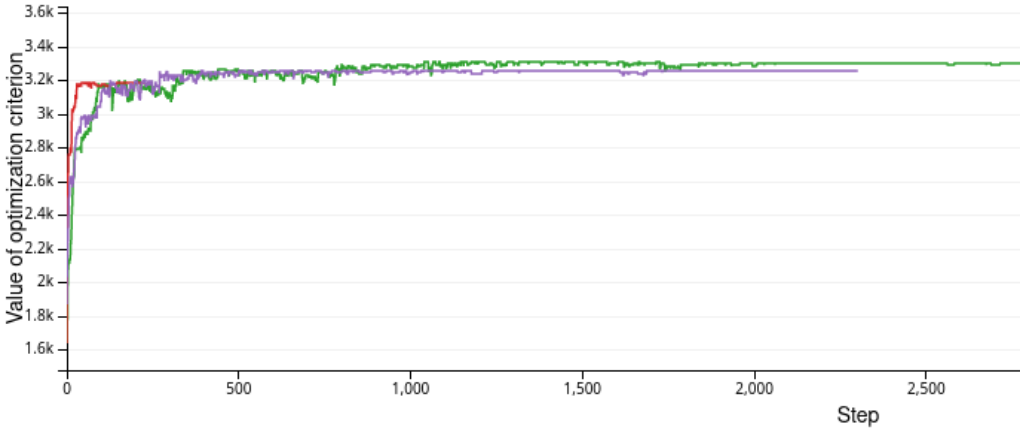
Ako som už vysvetlil vyššie, počet navštívených stavov je viac citlivý pre nízke hodnoty teploty (počiatočnej aj koncovej) ako pre vysoké z dôvody charakteristiky geometrického radu ochladzovania (porovnaj prvé dva riadky). Pre porovnanie som pridal aj počet navštívených stavov referenčným algoritmom z webu do zátvorky. Hodnoty sú porovnateľné, v mojom algoritme občas dôjde k predčasnému opusteniu slučky.

Názov instancie (optimálna hodnota)	T, T final	I	Faktor ochladzovania	Najlepšia cena (referencia web)	Počet navštívených stavov (referencia web)
100-3 (3429)	10, 0.1	50	0.99	**3429** (2733)	22950 (23300)
100-3 (3429)	10, 1	50	0.99	**3429** (3182)	11475 (11500)
100-3 (3429)	10, 1	50	0.999	**3429** (3257)	115075
100-3 (3429)	10, 1	50	0.9995	**3429** (3297)	230233
100-3 (3429)	10, 1	100	0.99	**3429** (3182)	22994
100-3 (3429)	10, 1	100	0.999	**3429** (3257)	230200
100-3 (3429)	10, 1	100	0.9995	**3429** (3297)	460500
100-3 (3429)	50, 1	50	0.99	3420 (3271)	19361 (19500)
100-3 (3429)	50, 1	50	0.999	**3429** (3355)	194433
100-3 (3429)	50, 1	50	0.9995	3423 (3310)	389122 (391150)
100-3 (3429)	**50**, 1	100	0.99	**3429** (3182)	38912
100-3 (3429)	**50**, 1	100	0.999	**3429** (3257)	389906
100-3 (3429)	**50**, 1	100	0.9995	**3429** (3297)	779812

Už pri počiatočnej teplote 10 algoritmus prehľadal dostatočne veľké okolie aby dokázal objaviť optimum. Najlepšou voľbou faktoru ochladzovania sa ukázal 0.999. Pri 0.99 boli výsledky nestabilné, algoritmus často vracia neoptimálne riešenie, naopak pre 0.9995 bol výpočet zbytočne dlhý.

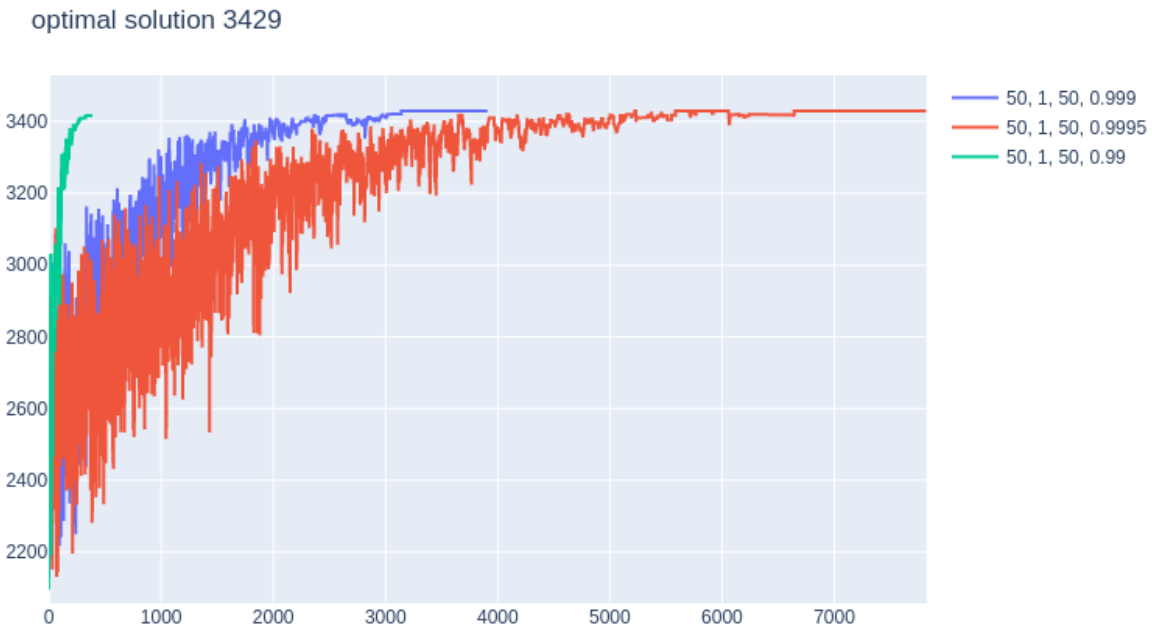


Graf 1.1: Vstupná teplota 10, koncová teplota 1. Faktor ochladzovania podľa legendy. Algoritmus dokázal prehľadať dostatočne veľké okolie. Pri k=0.99 boli výsledky nestabilné, záviseli od náhodne vygenerovaného prvého stavu. Pre k=0.9995 vidíme, že algoritmus konvergoval ale dlho sa neukončil.

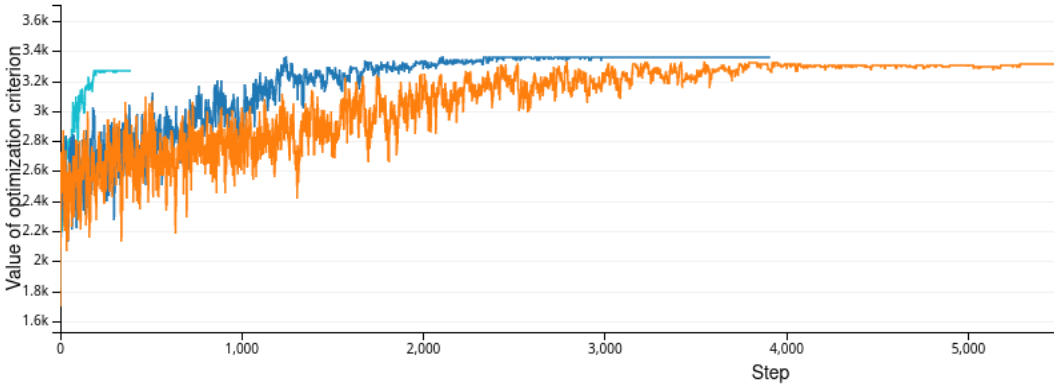


Graf 1.2: Priebeh SA z webového rozhrania. Algoritmus konvergoval pre všetky hodnoty k, nájdené hodnoty boli neoptimálne.

Vyššia teplota prekvapivo nepriniesla stabilnejšie výsledky, objavili sa optimálne a takmer optimálne výsledky. Podobne aj tu pri k=0.9995 bol výpočet zbytočne dlhý (na grafe je orezaný priebeh).



Graf 2.1: Vstupná teplota 50, koncová teplota 1. Faktor ochladzovania podľa legendy. Vyššia vstupná teplota umožnila prehľadávanie veľkého okolia. Graf je orezaný a nie je vidno zbytočne dlhý priebeh SA s k=0.9995. Riešením by bolo pridať ukončovaciu podmienku ak sa dlho nemení stav.



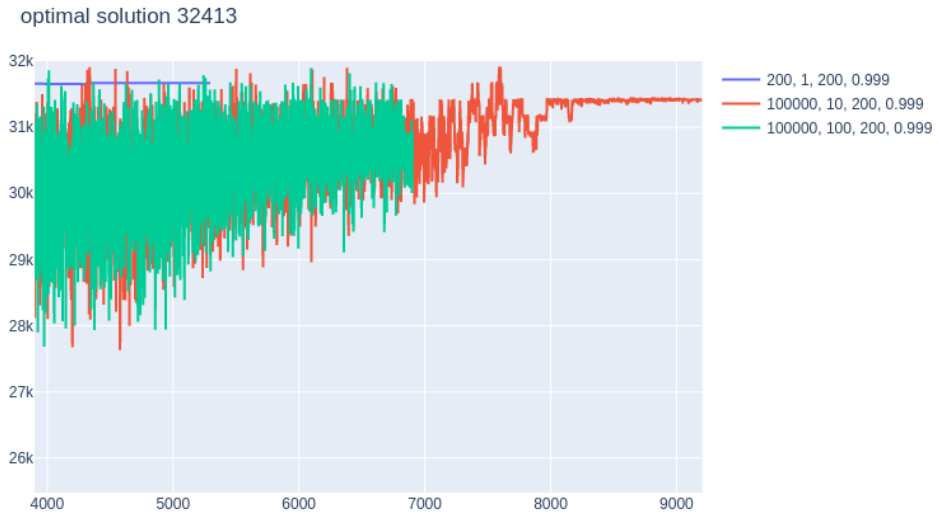
Graf 2.2: Priebeh SA z webového rozhrania. Algoritmus konvergoval pre všetky hodnoty k, výsledky boli neoptimálne.

Po analýze priebehu SA tvrdím, že algoritmus je správne nasadený a jeho priebeh sa veľmi podobá na priebeh referenčného SA z webu.

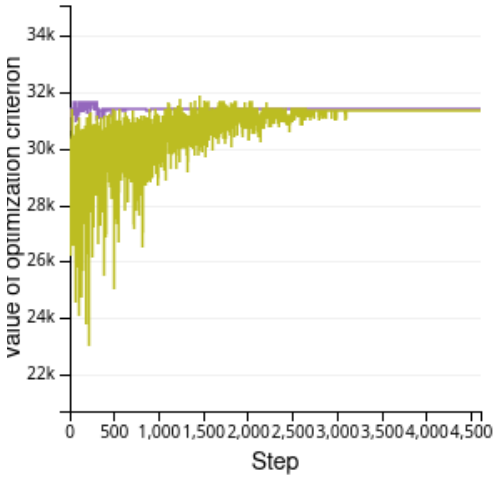
Vplyv vstupnej teploty

O instancii **40-1** nie je nič známe. Priebeh SA vyzerá zvláštne, akoby tam bola veľmi hlboká optimálna jama - algoritmus do nej vošiel a už nevyšiel, iba skúša malé okolie. Treba nastaviť veľmi vysokú počiatočnú teplotu aby sa algoritmus dostal z tejto jamy a skúšal aj zhoršujúce stavy - pre tento dataset nastavím značne vyššiu teplotu.

Názov instancie (optimálna hodnota)	T, T final	l	Faktor ochladzovania	Najlepšia cena (referencia web)	Počet navštívených stavov (referencia web)
40-1 (32413)	10, 1	50	0.99	31167 (30166)	11476 (11500)
40-1 (32413)	10, 1	50	0.999	31670 (31182)	115075 (115100)
40-1 (32413)	200, 1	200	0.999	31905 (31348)	1059200 (1059200)
40-1 (32413)	100000, 10	200	0.999	32164	1398499
40-1 (32413)	100000, 100	200	0.999	**32167** (31395)	938141 (1381000)



Graf 3.1: Vysoká teplota a pomalé ochladzovanie dokázali nájsť riešenia blízke globálnemu optimu. Nižšia konečná teplota červeného grafu dokázala konvergovať.



Graf 3.2: Webové rozhranie, vstupná teplota 200 (modrá) a 1000(žltá), k=0.999, l=200.

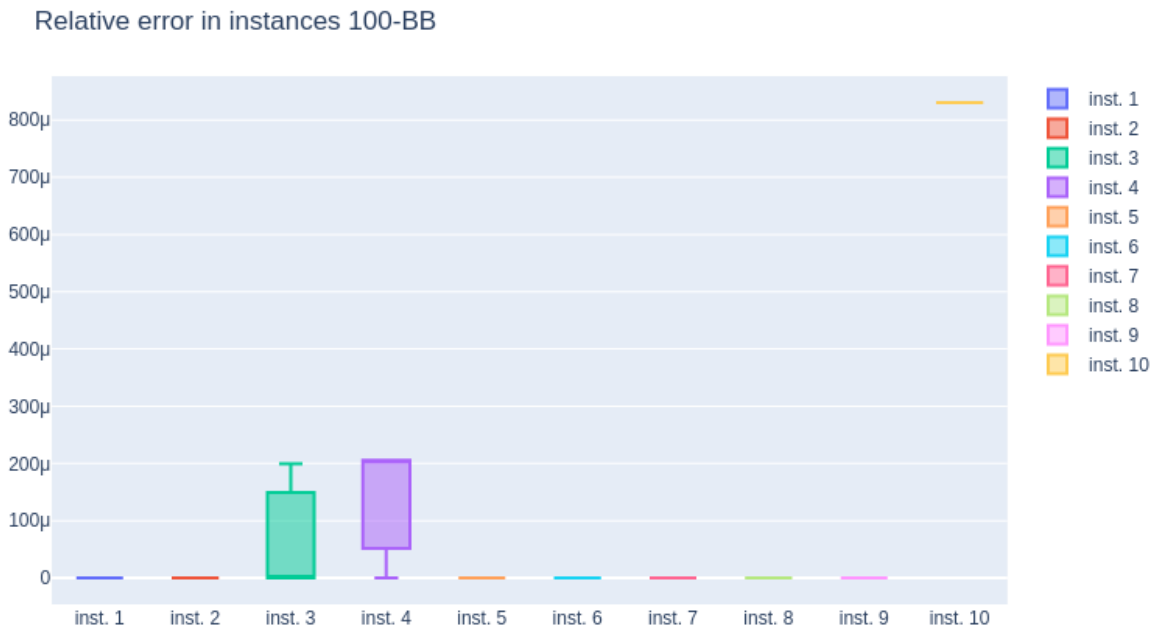
Podľa správania sa algoritmu zvolím vyššiu počiatočnú teplotu aby algoritmus mohol objavovať dostatočne veľké okolie.

Black box testovanie

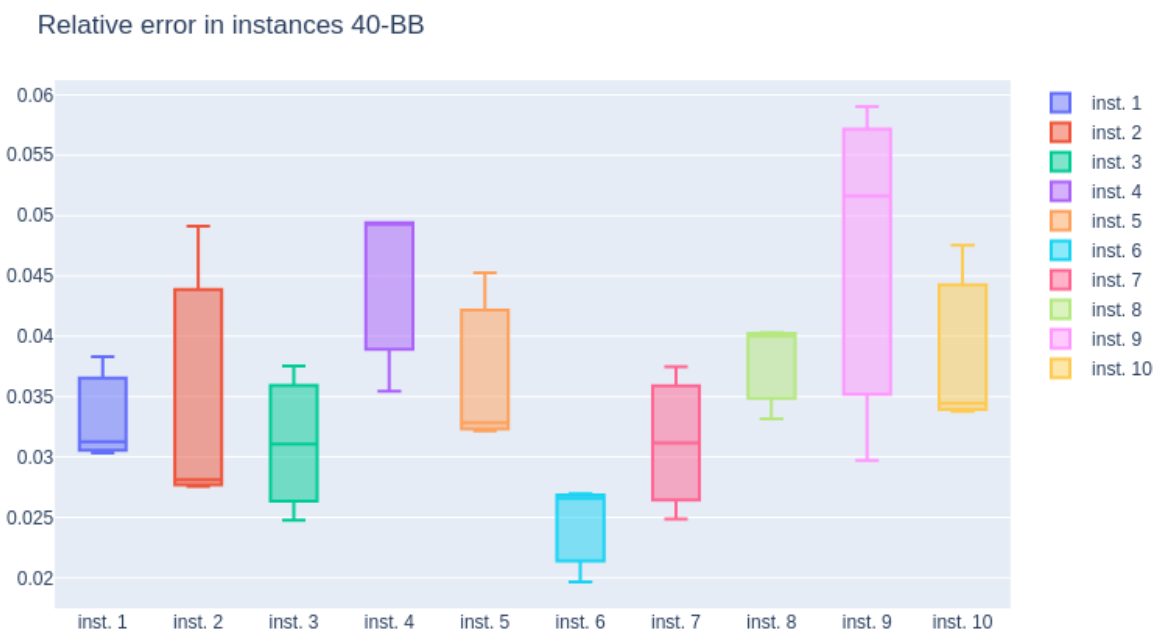
V black box testovaní som už neupravoval algoritmus a podľa zistených parametrov z predchádzajúcih podobných instancií som testoval úspešnosť algoritmu.

V tabuľke je vidno, že vyššia teplota naozaj pomohla v zložitom datasete 40-BB (instancie zo ZKC). V jednoduchšom vygenerovanom datasete 100-BB postačila aj nízka teplota.
Časová náročnosť SA je oveľa vyššia ako pri iných metódach.
SA som spustil 5x pre každú instanciu a vypočítal som priemernú chybu ako je vidno v box-plot grafoch 5.1, 4.1, 4.2. Priemernú chybu na celom datasete som vložil do tabuľky.
Chyba redukovanej heuristiky a cenovej dekompozície nie je ovplyvnené opakovaným spúšťaním. Všetky metódy som opakovane spustil 5x a meral som priemerný výpočtový čas.

Dataset (10 instancii)	SA avg error, std.	REDUX error, std.	SA výpočtový čas	REDUX výpočtový čas	Cenová dekompozácia výpočtový čas
40-BB T = 10	3.59%, **+- 0.06**	**1.19%,** +- 0.12	28.58s	**0.0001s**	0.59s
40-BB T = 1000	**1.0%,** **+- 0.04**	1.19%, +- 0.12	62.66	**0.0001s**	0.59s
100-BB T = 10	**0.10%,** **+- 0.25**	0.22%, +- 0.32	76.94s	**0.00014s**	0.08s



Graf 5.1: Jednoduchý náhodne vygenerovaný dataset má asi podobné vlastnosti ako instancia 100-3, ktorá slúžila na prehľadávanie vhodných parametrov. Aj napriek dvom instanciám bola chyba takmer nulová.



Graf 4.1: Pri nízkej počiatočnej teplote záviseli výsledky SA od náhodného spustenia, algoritmus nemal šancu dostatočne prehľadať stavový priestor.

Graf 4.2: Vyššia počiatočná teplota umožnila lepšie prehľadávať stavový priestor a algoritmus nebol až tak závislý na počiatočnom riešení. Výsledky sú lepšie (nižšia chyba aj rozptyl) ako pri redukovanej heuristike ale výpočet je oveľa dlhší.

Záver

Preskúmanie vlastností instancií je dôležité pri výbere vhodných vstupných parametrov. Najväčší vplyv na presnosť mali počiatočná teplota a faktor ochladzovania, ktoré tiež značne ovplyvňovali výpočetný čas - nemá zmysel bezhlavo nastavovať vysoké hodnoty. Pre ďalšiu úlohu sa pokúsim nájsť spôsob ako odhadnúť vstupné parametre - z literatúry, ktorú som aktuálne čítal som našiel veľmi zložité matematické prístupy, ktoré som sa neodvážil implementovať. Výber parametrov som preto spätne porovnával s priebehom SA a s webovým rozhraním. Myslím si, že som SA implementoval správne, pretože má veľmi podobný priebeh ako SA na webovom rozhraní, ktoré sme mali ako referenciu alebo inšpiráciu.

Algoritmus SA pri správnom nastavení dokázal priniesť optimálne výsledky alebo výsledky blížiace sa optimu. Vypočetný čas tohto algoritmu je veľmi dlhý. Výhodou je, že sa môže použiť na viacero NP problémov, ktoré nemusia mať FPTAS algoritmus.