UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# DIPLOMOVÁ PRÁCE

2025 Bc. Nicholas Zahálka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Porovnání vybraných variant simulovaného žíhání na problému obchodního cestujícího

Diplomová práce

2025 Bc. Nicholas Zahálka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Nicholas Zahálka** |
| Osobní číslo: | **I23286** |
| Studijní program: | **N0613A140007 Informační technologie** |
| Téma práce: | **Porovnání vybraných variant simulovaného žíhání na problému obchodního cestujícího** |
| Zadávající katedra: | **Katedra softwarových technologií** |

## Zásady pro vypracování

Cílem práce bude porovnat různé varianty simulovaného žíhání na problému obchodního cestujícího. V teoretické části student popíše optimalizační problém obchodního cestujícího a problematiku simulovaného žíhání a jeho různých variant. V praktické části poté porovná vybrané varianty (případně navrhne a přidá vlastní variantu simulovaného žíhání). Student navrhne a provede experimenty a nakonec vyhodnotí jejich výsledky.

Rozsah pracovní zprávy: **50-60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

HYNEK, Josef. Genetické algoritmy a genetické programování. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.
MITCHELL, Melanie. An introduction to genetic algorithms. Cambridge, Mass.: MIT Press, c1996. ISBN 978-0262133166.
COOK, William. Po stopách obchodního cestujícího: matematika na hranicích možností. Zip. Praha: Argo, 2012. ISBN 978-80-7363-412-4.

Vedoucí diplomové práce: **Ing. Jan Merta, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **31. října 2024**
Termín odevzdání diplomové práce: **23. května 2025**

L.S.

**prof. Ing. Petr Doležel, Ph.D.** v.r.
děkan

**prof. Ing. Antonín Kavička, Ph.D.** v.r.
vedoucí katedry

V Pardubicích dne 29. listopadu 2024

Prohlašuji:

Práci s názvem Porovnání vybraných variant simulovaného žíhání na problému obchodního cestujícího
jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 19. 5. 2025

Bc. Nicholas Zahálka

# PODĚKOVÁNÍ

Chtěl bych poděkovat panu Ing. Janu Mertovi, Ph.D. za vedení diplomové práce a cenné rady při konzultacích, které pomohly k dokončení práce. Dále bych chtěl také poděkovat své rodině, kolegům a přátelům za veškerou podporu při celém studiu.

# ANOTACE

Cílem práce bude porovnat různé varianty simulovaného žíhání na problému obchodního cestujícího. V teoretické části student popíše optimalizační problém obchodního cestujícího a problematiku simulovaného žíhání a jeho různých variant. V praktické části poté porovná vybrané varianty (případně navrhne a přidá vlastní variantu simulovaného žíhání). Student navrhne a provede experimenty, a nakonec vyhodnotí jejich výsledky.

# KLÍČOVÁ SLOVA

Simulované žíhání, Varianty simulovaného žíhání, Optimalizační problém, Optimalizační metody

# TITLE

Comparison of selected variants of simulated annealing on the traveling salesman problem

# ANNOTATION

The aim of this work will be to compare different variants of simulated annealing on the business traveller problem. In the theoretical part, the student will describe the optimization problem of the business traveller and the problem of simulated annealing and its different variants. In the practical part, he/she will then compare the selected variants (or propose and add his/her own variant of simulated annealing). The student will design and conduct experiments and finally evaluate the results.

# KEYWORDS

Simulated annealing, Simulated annealing variants, Optimization problem, Optimization methods

# OBSAH

# SEZNAM ILUSTRACÍ A TABULEK

# SEZNAM ZKRATEK A ZNAČEK

SA – Simulované žíhání

TSA – Tradiční simulované žíhání

MSA – Modifikované simulované žíhání

PSA – Paralelní simulované žíhání

HPSA – Hybridní paralelní simulované žíhání

RL – Ranking selekce s použitím poměrového ruletového kola

RB – Ranking selekce s výběrem nejlepšího

TB – Tournament selekce s realizací všech soubojů

TL – Tournament selekce s použitím poměrového ruletového kola

TBL – Tournament selekce s omezeným počtem realizovaných soubojů s použitím poměrového ruletového kola

TBR – Tournament selekce s omezeným počtem realizovaných soubojů s použitím výběru nejlepšího

HPSA_RB – Hybridní paralelní simulované žíhání s Ranking selection s výběrem nejlepšího

HPSA_TBR – Hybridní paralelní simulované žíhání s Tournament selection s omezeným počtem soubojů s výběrem nejlepšího

GA – Genetický Algoritmus

# ÚVOD

Problém obchodního cestujícího není pouze teoretická knižní úloha, která by neměla příliš mnoho využití v praxi. S tímto problémem se může setkat každý v každodenním životě. Nejčastěji nastává tento problém v logistice, kdy se doručovací služby snaží optimalizovat rozvoz jejich balíků. Vzhledem k velkému množství, které musí doručit, chtějí samozřejmě ušetřit co možná nejvíce za náklady. Náklady v podobě pohonných hmot a času, který rozvoz zabírá. Jedním z dalších možných míst výskytu může být například strojní průmysl. Zde se výrobní stroje snaží optimalizovat sekvenci pohybů jednotlivých součástek stroje pro minimalizaci času vykonávání jednotlivých dílčích úkonů na celkovém produktu. Díky tomu mohou vyrobit větší množství produktů a snížit opotřebení součástek.

Cílem této diplomové práce je porovnat různé varianty simulovaného žíhání na problému obchodního cestujícího. Teoretická část popisuje různé varianty simulovaného žíhání, které jsou navrženy pro co nejvyšší zkvalitnění nalezeného řešení. Jednotlivé varianty se snaží provést rychlejší konvergenci ke globálně optimálnímu výsledku, nebo alespoň co nejblíže k němu. V této části budou též popsány individuální specifické modifikace původního tradičního simulovaného žíhání získané rozborem zmíněných variant. Implementace těchto modifikovaných variant se snaží vyhnout lokálnímu optimu. Různé modifikace pak umožňují lépe přizpůsobit řešení pro dané požadavky, které mohou zkvalitňovat hledání výsledného řešení.

Experimentální část se následně zabývá implementací vybraných variant simulovaného žíhání z teoretické části. Implementovány budou varianty založené na tradičním simulovaném žíhání, modifikované variantě simulovaného žíhání a paralelním simulovaném žíhání.

Práce si klade za cíl porovnat získané výsledky z implementovaných algoritmů, které budou testovány na stejném souboru dat s různými konfiguracemi.

# 1 PROBLÉM OBCHODNÍHO CESTUJÍCÍHO

Problém obchodního cestujícího je jeden z nejznámějších kombinatorických problémů, který je stále aktuální i v dnešní době. Vzhledem k tomu, že tento kombinatorický problém spadá do kategorie NP – těžkých problémů a je matematicky náročný, je mizivá šance, že bude dostupné deterministické řešení, které by bylo optimální v polynomiálně omezeném čase. [1]

Popis samotného problému je popsán následující definicí „*Problém obchodního cestujícího lze snadno slovně definovat tak, že obchodní cestující musí při své cestě navštívit každé město v přidělené oblasti právě jedenkrát a vrátit se do výchozího bodu. Kromě toho je třeba minimalizovat jeho cestovní náklady, které jsou definovány jako celková cena postupného přesunu mezi jednotlivými městy a obvykle závisí na vzdálenosti mezi městy. Cílem je tedy navrhnout takovou okružní cestu, kde součet nákladů nutných pro uskutečnění této cesty bude minimální.*“ [2]

Aplikace samotného problému je, že zasahuje do širšího spektra oborů než pouze do logistiky. Jedna z možností praktického využití je při vrtání desek plošných spojů. Při výrobním procesu je nutné propojit vodiče na různých vrstvách desek nebo vytvořit místo pro osazení pinů integrovaných obvodů. Jednotlivé díry mohou mít různé otvory, přičemž hlavička vrtáku musí vyvrtat veškeré otvory jedné velikosti, poté vyměnit vrták a pokračovat takto dále dokud nebudou vyvrtané všechny potřebné otvory s danými rozměry. Interpretací v původním znění problému obchodního cestujícího by města odpovídala otvorům v desce, které lze vyvrtat jedním vrtákem. Vzdálenosti odpovídají v původním znění přesunům mezi jednotlivými pozicemi otvorů. Cílem je minimalizovat dobu pohybu vrtáku. [3, 5, 30]

Dalším speciálním případem je generální oprava palivových turbínových motorů. S touto aplikací problému přišel v roce 1987 Plante a spol. [2] Tento problém nastal v momentě, kdy byly opravovány palivové turbínové motory letadel. Bylo nutné zajistit jednotné proudění paliva napříč turbínami. Turbína v každé fázi disponuje sestavou tryskových vodících lopatek. Sestava je složena z tryskových lopatek umístěných po obvodu a každou lopatku je nutné nahlížet individuálně, protože každá z nich má své vlastnosti. Správné umístění je zásadní pro získání dostupných výhod, jako jsou například snížení vibrací, zvýšení rovnoměrného proudění nebo snížení spotřeby paliva. [3, 4]

## 1.1 Matematická definice

Cílem úlohy je nalézt trasu, která spojí všechna města tak, aby každé z nich bylo navštíveno právě jednou a celkové náklady na realizaci cesty byly co nejnižší. Každý přesun mezi dvojící měst má určitou cenu a zároveň hledáme takové uspořádání cest, které minimalizuje součet těchto nákladů. Matematická funkce je dostupná v odborné literatuře. Vypadá poté následovně:

$$min = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Jednotlivé proměnné ze vzorce vyjadřují následující:

- $c_{ij}$ – náklady na přesun z města i do města j
- $x_{ij}$ – binární proměnná, která určuje, zda přesun z města i do města je součástí řešení $x_{ij}$ = 1, jinak $x_{ij} = 0$

Za podmínek:

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, \dots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \dots, n$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \; pro \; všechny \; S \subset \{1, \dots, n\}, S \neq \emptyset$$

$$x_{ij} \in \{0, 1\}, i, j = 1, \dots, n.$$

[1]

## 1.2 Tradiční algoritmy pro řešení problému

Mezi tradiční algoritmy patří explicitní výčty a deterministické metody. Zástupce explicitních výčtů je Brute force a Greedy algoritmus. Za deterministické metody je to dynamické programování, branch and bound a nearest neighbor. [6]

### 1.2.1 Brute force

Brute force (řešení hrubou silou) je algoritmus, který se snaží najít řešení problému pomocí vyčerpávajícího prohledávání všech možných řešení. Systematicky tedy prochází celý prostor řešení, dokud nenajde to nejlepší. [1]

Algoritmus nalezne globální řešení, avšak za cenu procházení celého prostoru možných řešení. Je velmi snadný na implementaci a s vysokou pravděpodobností nalezne správné řešení, avšak hlavním úskalím tohoto přístupu je jeho časová náročnost. Časová náročnost roste s počtem permutací jednotlivých měst *n*. Z toho důvodu je vhodné použít toto řešení pouze v případě malých problémů. [1, 6]

### 1.2.2 Algoritmus lokálního prohledávání

Lokální algoritmus je iterativní metoda, která je zahájena z náhodně vybraného řešení z dostupného stavového prostoru. Následně dochází k prohledávání okolních sousedů aktuálního řešení. Vyhledává řešení, které přináší zlepšení z pohledu cílové funkce. V případě, že je nalezeno lepší řešení, stává se novým aktuálním řešením a algoritmus pokračuje opět v hledání. Tento proces je opakován do doby, dokud nelze nalézt žádné lepší řešení, nebo pokud dojde ke splnění ukončovací podmínky. Výsledná kvalita nalezeného řešení se odvíjí od povahy problému a definice samotných sousedů. Vhodná volba sousedství může zajistit různé úrovně optimálnosti, případně pomoci nalézt řešení, které bude globální optimum nebo se mu bude co nejvíce blížit. Tento algoritmus typicky konverguje k lokálnímu optimu, tedy jím nalezené řešení nemusí být nutně globální optimum. Algoritmy založené na lokálním prohledávání je například Hill climbing (Horolezecký algoritmus). [8, 11, 13]

### 1.2.3 Algoritmus metropolis

Algoritmus byl vytvořen v roce 1953 třemi americkými vědci – Metropolis, Rosenbluth a Teller. Slouží k simulaci fyzikálního žíhání. Smysl tohoto algoritmu je napodobit změny struktury materiálu při pomalém ochlazování. Využívá metody Monte Carlo, která funguje na principu náhodného generování posloupnosti stavů systému. Algoritmus na počátku začíná ve výchozím stavu s určitou energií. Nový stav vzniká například pomocí změny polohy jedné částice stavového vektoru. V případě, že nový stav má nižší energii než původní, přechází do něj automaticky. Pokud naopak má vyšší nebo stejné množství energie, tak do tohoto stavu přechází pouze s určitou pravděpodobností dle vzorce Metropolis kritéria. Pokud je ochlazování dostatečně pomalé, tak se algoritmus ustálí a dosáhne rovnovážného stavu, který odpovídá dané teplotě. Vzorec pro pravděpodobnost přijetí je následný. [8]

$$\Pr\{Aktuální\ stav = j\} = e^{\left(\frac{E_i - E_j}{k_b * T}\right)}$$

Jednotlivé proměnné ze vzorce vyjadřují následující:

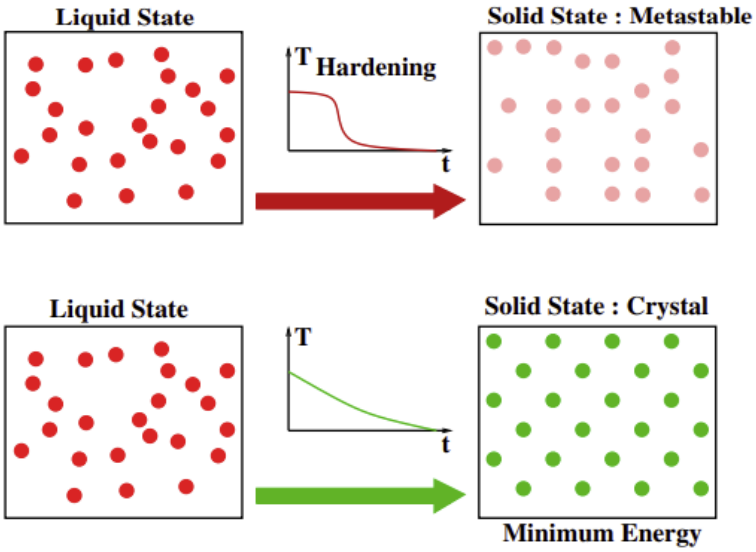
- $E_i$ – energií původního řešení
- $E_j$ – energii nového kandidátního řešení
- $k_b$ – Boltzmannovu konstantu
- $T$ – Teplotu systému

# 2 SIMULOVANÉ ŽÍHÁNÍ

Simulované žíhání patří mezi nejpopulárnější metaheuristické metody, které jsou používané pro řešení obtížných optimalizačních problémů. „*Jde o pravděpodobnostní metodu, navrženou Kirkpatrickem, Gelettem a Vecchim (1983) a Černým (1985), pro nalezení globálního minima nákladní funkce, která může mít několik lokálních minim*“. [9, 20]

Koncept je založený na fyzikálním jevu zvaném žíhání materiálů. Při tomto procesu dochází k rapidnímu zvýšení teploty daného materiálu a následně k postupnému ochlazování. Při vysoké teplotě dochází k tavení struktury a následným ochlazením dojde opět k navrácení do pevného stavu. Během doby, kdy je materiál roztavený, jsou tak jeho částice neuspořádané a náhodně rozdělené. [8]

Ke snižování energie, tedy k rapidnímu ochlazení dochází, když je počáteční teplota dostatečně vysoká. Pro dosažení stavu s minimální energií musí být ochlazování dostatečně dlouhé. V případě, že nejsou provedeny tyto zmíněné kroky, dojde k tomu, že materiál bude v metastabilním stavu s neminimální energií. V případě metastabilního stavu bylo tedy nalezeno lokální optimum, nikoliv globální. [8]




*Obrázek 1 – Ilustrace vlivu rychlosti ochlazování [10]*

Na obrázku 1 je možné vidět, že pokud dojde k vysokému zahřátí, tak jsou jednotlivé částice uvnitř roztaveného materiálu neuspořádané. První varianta nastává ve zmíněném případě, pokud dojde k metastabilní verzi. Částice zde nejsou symetrické z důvodu, že došlo k nedostatečnému ochlazení. Druhá varianta s dostatečně dlouhým ochlazováním má částice symetricky uspořádané. [10]

## 2.1 Tradiční simulované žíhání

V simulovaném žíhání je využíván Metropolis algoritmus pro vytváření posloupnosti řešení v daném stavovém prostoru. Tímto přístupem se vytváří analogie mezi systémem s více částicemi a optimalizačním problémem pomocí následujících vztahů. První vztah, kdy body stavového prostoru představují možné stavy tělesa. Druhý vztah funkce, kterou chceme minimalizovat, představuje energii pevného tělesa. Je zaveden nový řídící parametr *c*, který reprezentuje teplotu. Parametr je reprezentován ve stejných jednotkách jako optimalizovaná cílová funkce. Důležitý předpoklad je, že uživatel definuje pro každé body stavového prostoru jeho okolí a mechanismus pro generování nového řešení. Na základě těchto podmínek je následně definováno přijímací pravidlo, které rozhoduje o tom, zda bude nové řešení přijato. [8]

Pokud má nové řešení nižší hodnotu cílové funkce než aktuální řešení, tak se přijímá automaticky. Pokud má hodnotu horší, tak stále může být přijato, nicméně s určitou pravděpodobností, která závisí na rozdílu hodnot a parametru *c* (teplotní hodnoty). Generování sousedních stavů odpovídá narušování stavů v původním Metropolis algoritmu. Obdobným způsobem je převzato i pravidlo pro rozhodování o přijetí nového stavu. [8]

Přechod je proces, kdy je stávající řešení nahrazeno jiným ze svého okolí. Skládá se ze dvou kroků, a to návrhu nového řešení a rozhodnutí o přijetí. [8]

Pseudokód algoritmu je následující.

1. Algoritmus na počátku provádí inicializaci, zvolí počáteční řešení, nastaví počáteční teplotu a počet pokusů pro první iteraci.
2. Hlavní smyčku algoritmus provádí do té doby, dokud teplota neklesne téměř k nule.
3. Algoritmus v každém kroku navrhne nové řešení z okolí aktuálního řešení a poté provede kontrolu přijímacím pravidlem.
4. Pokud má nové řešení lepší hodnotu cílové funkce než stávající, je automaticky přijato. Pokud má horší, je přijato s pravděpodobností získanou dle vzorce.
5. Po těchto přechodech je zvýšen čítač počtu iterací, je aktualizována teplota a dochází k opakovanému provádění. Vzorec pro přijetí horšího řešení je uveden níže. [8]

$$\Pr\{přijetí\ j\} = \begin{cases} 1 & pokud\ f(j) < f(i) \\ e^{(\frac{f(i)-f(j)}{c})} & jinak. \end{cases}$$

Jednotlivé proměnné ze vzorce vyjadřují následující:

- $f_i$ – cílová funkce aktuálního řešení
- $f_j$ – cílová funkce nového řešení
- $c$ – parametr teploty

Jednou z hlavních charakteristik simulovaného žíhání je jeho schopnost občas záměrně akceptovat horší přechod. Tedy řešení, které zvyšuje hodnotu cílové funkce. Na počátku algoritmu je teplota nastavena vysoko, což umožňuje přijímat i výrazně horší přechody. Díky těmto horším přechodům může algoritmus volně procházet různé oblasti stavového prostoru a zkoumat i horší řešení, která by jinak byla zavržena. [8]

S postupným klesáním teploty algoritmus zpřísňuje svá akceptační kritéria, dává tedy přednost převážně lepším řešením a je ochoten připustit jen mírné zhoršení. S teplotou blížící se k nule již není dovoleno zhoršení cílové funkce, v tento moment algoritmus podobá běžnému přístupu Greedy algoritmu. [8]

### 2.1.1 Počáteční řešení

Běžným přístupem je zvolení náhodného nastavení počátečního vektoru reprezentující řešení v rámci předem stanovených mezí. Platí zde pravidlo, že čím blíže jsou počáteční hodnoty ke skutečnému globálnímu optimu, tím rychlejší a efektivnější bude celý proces hledání optimálního řešení. [12]

### 2.1.2 Počáteční teplota

Počáteční teplota je zásadním parametrem, který ovlivňuje pravděpodobnost přijetí horšího řešení. Je úzce provázaná s aktuální teplotou, ale i se zvoleným ochlazovacím schématem. Korektní volba počáteční teploty ovlivňuje kompletní chování algoritmu. [15]

Zvolená počáteční hodnota musí být nastavena tak, aby algoritmus měl šanci překonat lokální minimum. Také je důležité, aby neztratil stabilitu v případě, že by se nacházel v blízkosti globálního optima. Tato stabilita úzce souvisí s pravděpodobností přijetí horšího řešení algoritmem. Tyto faktory musí být zvažovány při volbě počáteční teploty. [12]

### 2.1.3 Ochlazovací schéma

Výběr správného způsobu snižování teploty při běhu algoritmu hraje zásadní roli v jeho úspěšnosti. V odborné literatuře je možné nalézt velké množství různých ochlazovacích metod. Základní rozdělení je na dvě skupiny: monotónní a adaptivní. [12]

Jedná se o funkci, která určuje teplotu v dané iteraci. Nejčastěji vychází z teplotní hodnoty předchozího kroku a případně i z počáteční teploty. V případě monotónní strategie dochází ke snižování teploty dle předem definovaného pravidla bez ohledu na úspěšnost. [12]

V případě adaptivní strategie jde o přesný opak monotónní strategie. Tempo ochlazování je ovlivňováno kvalitou získávaných řešení. Při nacházení vhodných řešení může algoritmus provádět rychlejší ochlazování. Ve specifických případech je možné provádět i dočasné zvýšení teploty. Při tomto se algoritmus pokouší dostat z prohledávané oblasti a zkoumá jinou. [12]

Mezi dostupné varianty ochlazovacích schémat patři: [12]

1. lineární
2. geometrické
3. logaritmické
4. hybridní
5. exponenciální
6. adaptivní

### 2.1.4 Cost funkce

Nákladová funkce je jedním ze základních pilířů simulovaného žíhání. Vrací číselnou hodnotou, která vyjadřuje kvalitu daného řešení (náklady na cestu). U problému obchodního cestujícího je cílem naplánovat trasu tak, aby každé město bylo navštíveno pouze jednou a cestující se na konci vrátí zpět do počátečního města. Nákladová funkce poté představuje celkovou délku trasy. Úkolem algoritmu je tuto trasu minimalizovat pomocí optimalizace. [9]

### 2.1.5 Generování souseda

Vytváření nových sousedních řešení se provádí pomocí mutací. Mutace se používá v genetických algoritmech jako jeden z genetických operátorů. Volba konkrétní mutace vychází z řešeného problému a způsobu kódování do vektoru reprezentující řešení. S mírnou úpravou je možné mutace z genetického algoritmu využít v simulovaném žíhání. Pro tento specifický problém byly zvoleny tři varianty.

1. Swap mutace provádí záměnu části řešení (města) na libovolné pozici za jinou ve stávajícím řešení.
2. V Partial reverse dochází k vygenerování náhodného rozmezí, ve kterém dojde k otočení pořadí prvků. Minimální nutný rozsah musí být tři prvky pro korektní funkčnost.
3. Partial shuffle, obdobně jako předchozí, realizuje vygenerování náhodného rozmezí, ve kterém dojde k náhodnému promíchání pořadí prvků. [24, 25, 27]

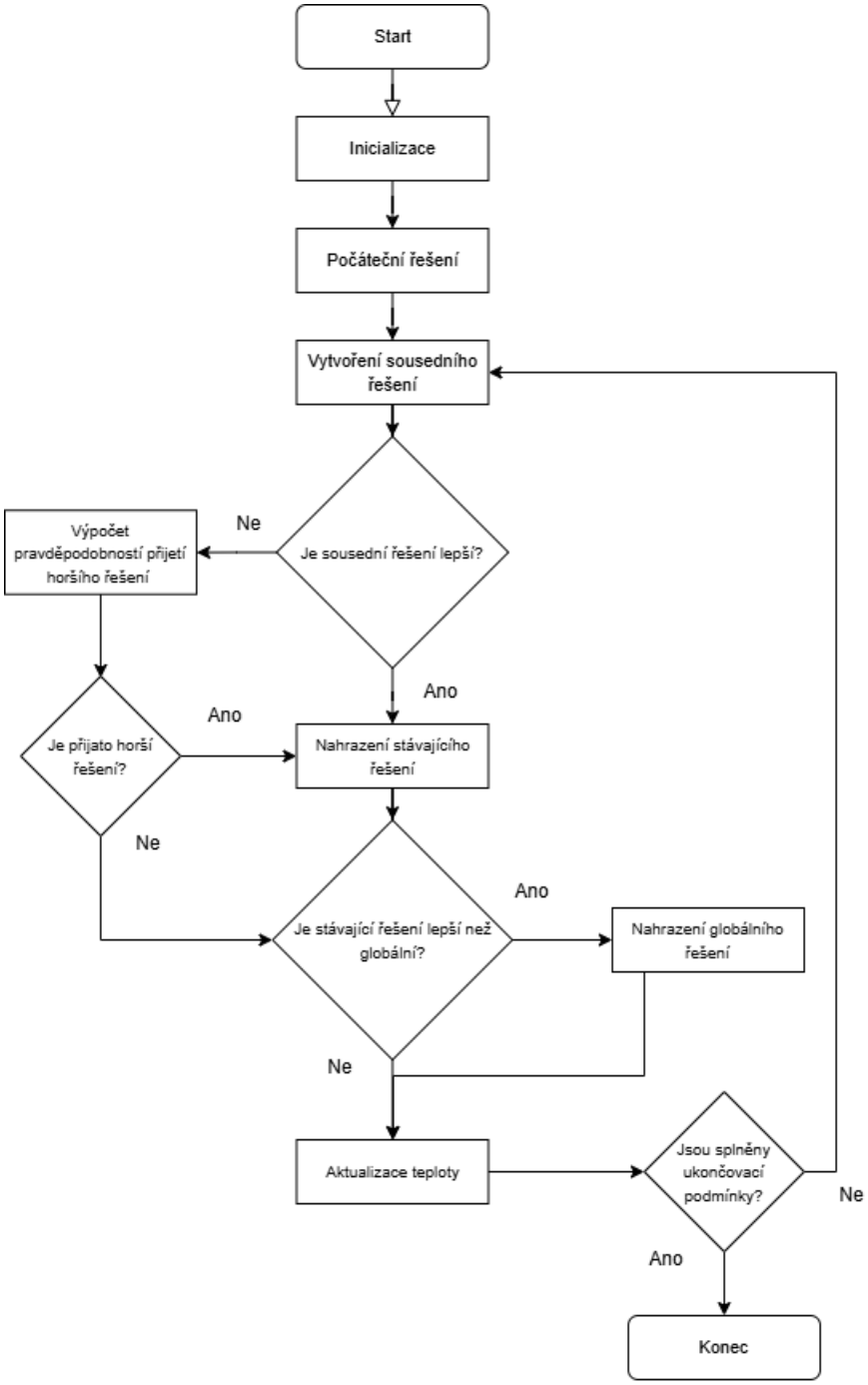
### 2.1.6 Ukončovací kritérium

Každý iterativní algoritmus vyžaduje ukončovací kritérium, které definuje, kdy má být ukončen. V případě simulovaného žíhání je dostupná celá řada různých ukončovacích kritérií, která jsou dostupná v literatuře. Základní myšlenka simulovaného žíhání je taková, že by měl být ukončen ve chvíli, kdy se systém ustálí natolik, že se přestane významně měnit a další iterace nepřináší významné zlepšení výsledků. [8, 14]

Jeden z oblíbených přístupů je pomocí monitorování řídícího parametru. Nejčastěji se jako tento monitorovaný řídící parametr volí například teplota. Pokud tento parametr klesne pod předem definovaný práh, tak algoritmus končí. Tento přístup vychází z fyzikálního simulovaného žíhání, kdy v případě nízké teploty již nedochází k žádným změnám. [8, 14]

Alternativní varianta je, že je algoritmus ukončen v případě, že několik po sobě jdoucích iteracích nepřináší nové výsledky. Dochází tedy k ukončení v momentě, kdy je algoritmus stabilizován a nepřináší dostatečně odlišné nebo nulové výsledky. Formálně se tato podmínka vyjadřuje jako kontrola, zda se rozdíl mezi aktuálním a předchozím odhadem optimálního řešení udržuje pod danou hranicí po určitý počet kroků. [8, 14]

Níže je dostupný vývojový diagram algoritmu Tradičního simulovaného žíhání na obrázku 2.




*Obrázek 2 – Vývojový diagram algoritmu simulovaného žíhání (autor)*

# 3 VARIANTY SIMULOVANÉHO ŽÍHÁNÍ

Kromě výše uvedené standardní verze simulovaného žíhání existuje i celá řada jeho modifikací, které nabízejí nový pohled a zlepšení konvergence optimalizačního algoritmu. Tyto varianty vycházejí z původního principu simulovaného žíhání a neustále přibývají nové. [14]

Jednou z variant je Coranavo simulované žíhání. Tato varianta využívá adaptivního generování nových kandidátních řešení podle jednotlivých souřadnicových směrů. Místo provádění změn všech proměnných současně, tato varianta mění pouze jednu souřadnici v každém kroku. To přináší výrazné zjednodušení výpočtu a zároveň zvyšuje flexibilitu algoritmu. [14]

Varianta Adaptivního simulovaného žíhání je považována za jednu z nejčastěji využívaných metod simulovaného žíhání. Vychází jako upravená verze algoritmu Fast Annealing a je charakterizována využitím funkce generující pravděpodobnostní funkci hustoty a akceptační funkci. Funkce generování pravděpodobnostní hustoty určuje, jakým způsobem bude vygenerováno nové kandidátní řešení, a akceptační funkce rozhoduje, zda bude toto řešení přijato. Obě funkce jsou závislé na stávajícím řešení, nově navrženém řešení a řídících parametrech, které mění chování těchto funkcí. [14]

Metoda threshold accepting je velmi podobná tradičnímu simulovanému žíhání. Oproti tradičnímu simulovanému žíhání tento algoritmus využívá deterministické kritérium, založené na prahové hodnotě zhoršení. Tedy v případě, že navržená úprava zlepší cílovou funkci, nebo její zhoršení nepřekročí zvolenou prahovou hodnotu, je tato změna přijata. V případě zhoršení cílové funkce o vyšší hodnotu, než je zvolená prahová hodnota, je odmítnuta. Hodnota prahu je tvořena prahovou posloupností. Pomocí této posloupnosti je toto kritérium tolerantnější v počátečních iteracích a postupně dochází ke zpřísňování v dalších iteracích. Díky této strategii může algoritmus asymptoticky konvergovat ke globálnímu optimu. [16, 17]

## 3.1 Modifikované simulované žíhání

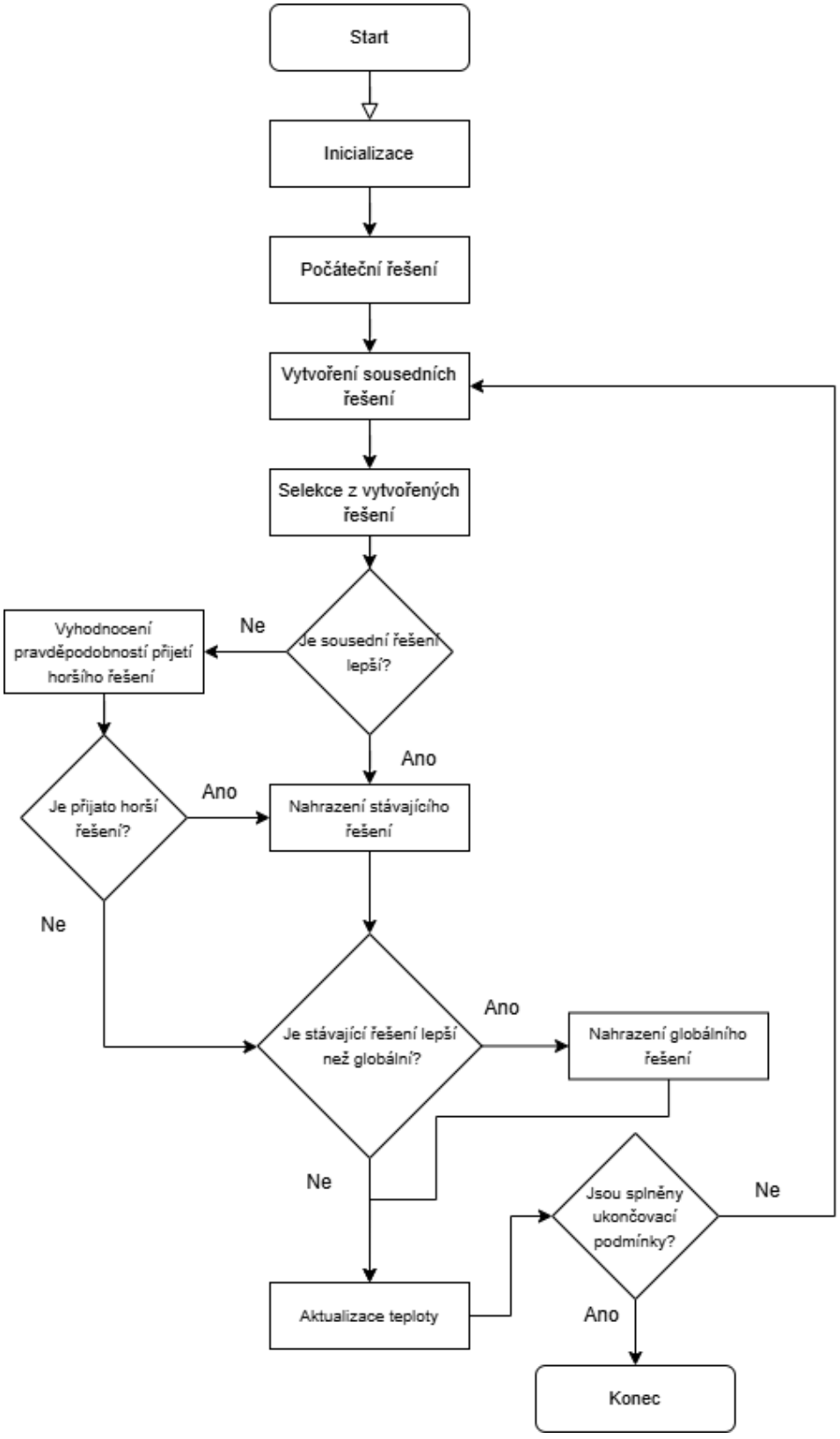
V této kapitole jsou popsány jednotlivé úpravy simulovaného žíhání. Modifikované simulované žíhání kombinuje SA s lokálním prohledáváním a selekcí z GA. Využívá upravené verze operátorů z genetického operátora – selekce. Selekce následně vybírá podle zvolené strategie, v praktické implementaci byly použity: [1]

1. Ranking selection [1, 21]
2. Tournament selection [22]
3. Proporcionální ruletové kolo [22, 23]

Modifikované simulované žíhání vůči základní verzi negeneruje jedno nové dílčí řešení (souseda), ale je zde vygenerovaný daný počet sousedních řešení. Počet řešení je zadán jako vstupní parametr při inicializaci. Další přidaný vstupní parametr je typ selekce, který slouží pro volbu selekce nového souseda (dílčího řešení) pro další krok optimalizace. Selekce byla přejata z GA, bylo ji nutné upravit, protože zde nedochází k výběru rodičů, ale pouze k výběru jednoho pokračujícího řešení. Přes vstupní parametr je také předávána velikost turnajového souboje, kolik tedy bude realizováno soubojů v případě limitovaného počtu. Zbylé vstupní parametry jsou stejné jako v tradiční verzi. Rozdíl mezi funkcí algoritmu je tedy pouze v generování rozdílného počtu dílčích řešení (sousedů).

Při Ranking selection [1, 21] a Tournament selection [22] dochází ke dvěma různým výběrům ze získaných výsledků. Jsou dostupné dvě výstupní varianty, první je přímý výběr nejlepšího řešení. V druhém výsledky z Ranking selection [1, 21] a Tournament selection [22] slouží jako podklad pro pravděpodobnostní výběr pomocí poměrového ruletového kola. Vybrané sousední řešení je poté přijato v případě zlepšení nebo dochází k vyhodnocení pravděpodobnostní podmínky přijetí.

Níže je dostupný vývojový diagram pro tento Modifikovanou verzi simulovaného žíhání na obrázku 3.




*Obrázek 3 – Vývojový diagram modifikovaného simulovaného žíhání (autor)*

### 3.1.1 Selekce

Selekce je využívána hlavně v genetických algoritmech jako další genetický operátor. Pomocí selekce jsou zde vybráni jedinci, kteří následně vytváří novou populaci pomocí křížení a mutací. Z původní populace se zvolí silní i slabší jedinci, kteří budou vytvářet potomky do nové populace. Na potomky budou přeneseny vlastnosti jeho rodičů. Převaha příliš silných jedinců může snížit rozmanitost řešení, a naopak příliš slabých jedinců způsobí velmi pomalou evoluci. Z tohoto důvodu je nutné mít vyvážené množství z obou skupin. [1, 21]

Využití v simulovaném žíhání byla realizována s mírnou modifikací. Zvolená řešení nevstupují již do žádného křížení. V rámci upravené selekce jsou generována nová dílčí sousední řešení pomocí mutace a z nich je následně vybráno jedno řešení, které pokračuje dále v optimalizaci.

### 3.1.2 Proporcionální ruletové kolo

Mechanismus imituje ruletového kola. Všem jednotlivcům se určí úměrná část z poměrového kola. Tato úměrná část je realizována vyhodnocením fitness funkce. Kvalitnější jedinci mají větší část výseče proti méně kvalitním. Všichni jedinci tedy mají šanci být zvoleni, i ti slabší. [22, 23]

Implementace ruletového kola je následovná.

1. Vyhodnocení fitness funkcí u všech jedinců z populace.
2. Sečtení získaných vyhodnocení.
3. Proporcionální rozdělení, výsečí dle kvality jednotlivých řešení.
4. Vygenerování náhodného čísla, které určí, v jaké části kola se nachází. Podle příslušné výseče dojde ke zvolení jedince.

Výpočet pravděpodobnosti pro přijetí je zobrazen ve vzorci níže. [22, 23]

$$p_i = \frac{f_i}{\sum_{j=i}^{N} f_i} \; i \in \{1, \dots, N\}$$

### 3.1.3 Turnajový výběr

Turnajový výběr je jedním z nejoblíbenějších způsobů selekce v genetických algoritmech. Oblíbený je především díky své výkonnosti a jednoduchosti implementace. Počet soubojů je vždy zvolen pomocí parametru. Při turnaji je zvoleno *n* jedinců z populace, kteří mezi sebou svedou souboj. Jedinci, kteří jsou nejsilnější na základě vyhodnocení fitness funkce, jsou dále vybráni při vytváření nové populace. Do samotného souboje může být vybrán libovolný jedinec, díky tomu je udržena rozmanitost, i když díky tomu může dojít k zpomalení konvergence. Turnajový výběr je možné realizovat dvěma způsoby – veškeré souboje nebo omezený počet. Následně dojde k výpočtu score pro jednotlivá řešení. Poté dojde k výběru buďto řešení s nejvyšším score nebo získané výsledky slouží jako podklad pro ruletové kolo. [22]

### 3.1.4 Ranking výběr

Hodnostní výběr je založen na principu, kdy všichni jedinci jsou vzestupně seřazeni dle výpočtu jejich fitness funkce. Vyhodnocení fitness funkce jedinců může přinést velké rozdíly, které následně příliš upřednostňuje jedince s mimořádnými výsledky. V případě použití pořadového čísla se provede rovnoměrnější pravděpodobnost výběru. Výhodou je snížení pravděpodobnosti uvíznutí v lokálním optimu, protože lepším jedincům nejsou přidělované enormně velké části celku. Samozřejmě přináší i nevýhody, může dojít k občasnému přijetí obzvlášť horšího řešení. [1, 21]

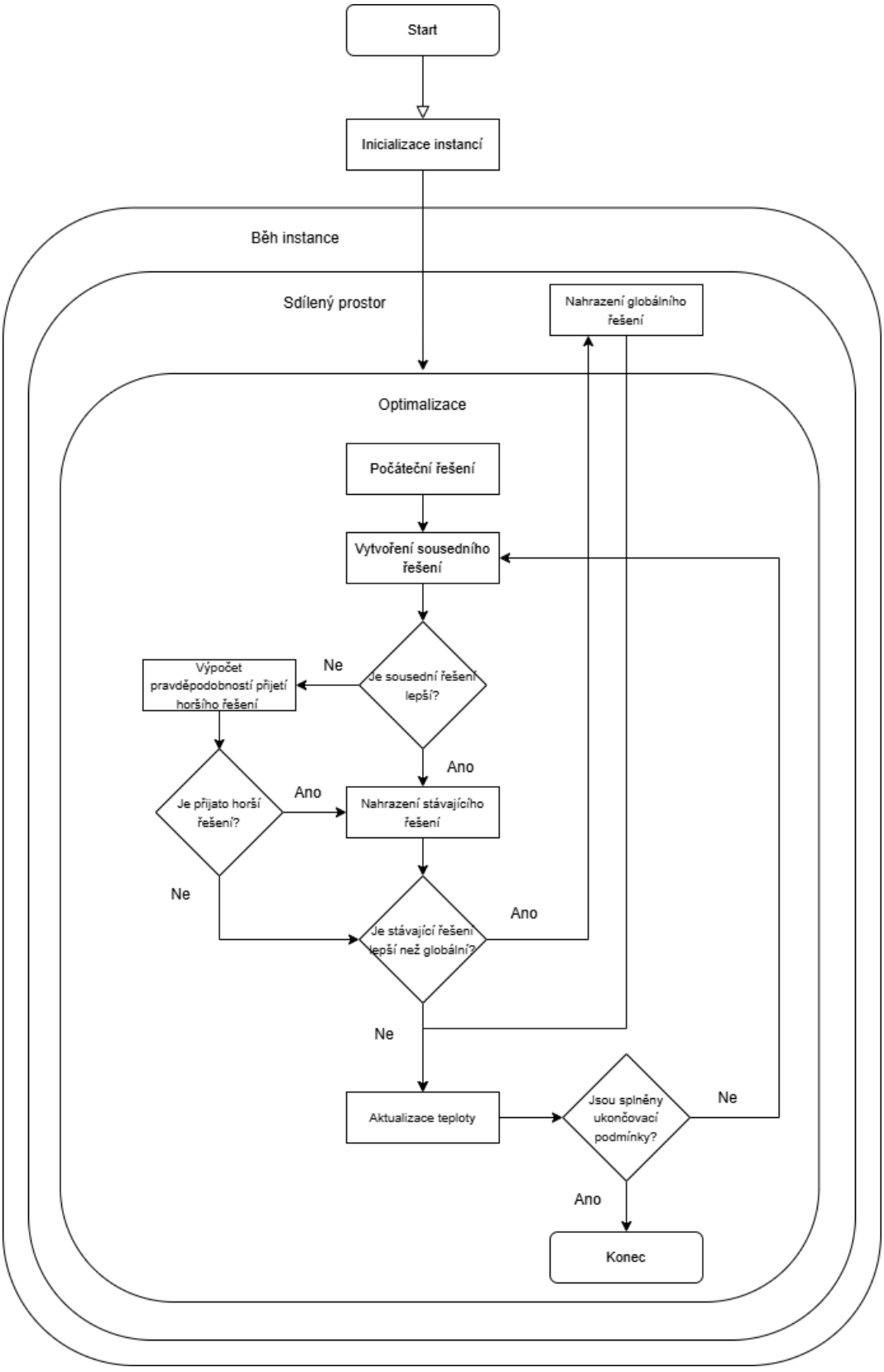
## 3.2 Parallel simulated annealing

Paralelní varianty vznikají za účelem zvýšení efektivity a účinnosti běžného simulovaného žíhání. Některé z přístupů jsou například: [9, 18, 28, 31]

1) Nezávislé simulované žíhání – v tomto případě je realizováno více instancí samostatně. Na konci jejich realizace je zvolen nejlepší výsledek.
2) Kooperativní simulované žíhání – zde si jednotlivé instance v průběhu výpočtu periodicky vyměňují informace za účelem vzájemné pomoci při hledání lepšího řešení.
3) Parallel tempering – spouští vícero instancí simulovaného žíhání s různými teplotami a umožňuje mezi nimi vyměňovat řešení za účelem vyvážení prohledávání.
4) Island model, realizuje několik instancí na samostatných uzlech (ostrovech), které příležitostně migruje pro udržení různorodosti.
5) Hybridní přístup kombinující simulované žíhání s dalšími metaheuristikami. Například jako genetické algoritmy nebo Particle swarm optimization.

Zmíněné jednotlivé způsoby přístupu se snaží nalézt rovnováhu mezi přesností výpočtu cílové funkce, rychlostí, způsobem zpracování a množstvím dat, která se musí vyměnit [19].

Níže je možné vidět vývojový diagram tohoto základního PSA na obrázku 4.




*Obrázek 4 – Vývojový diagram tradičního paralelního simulovaného žíhání (autor)*

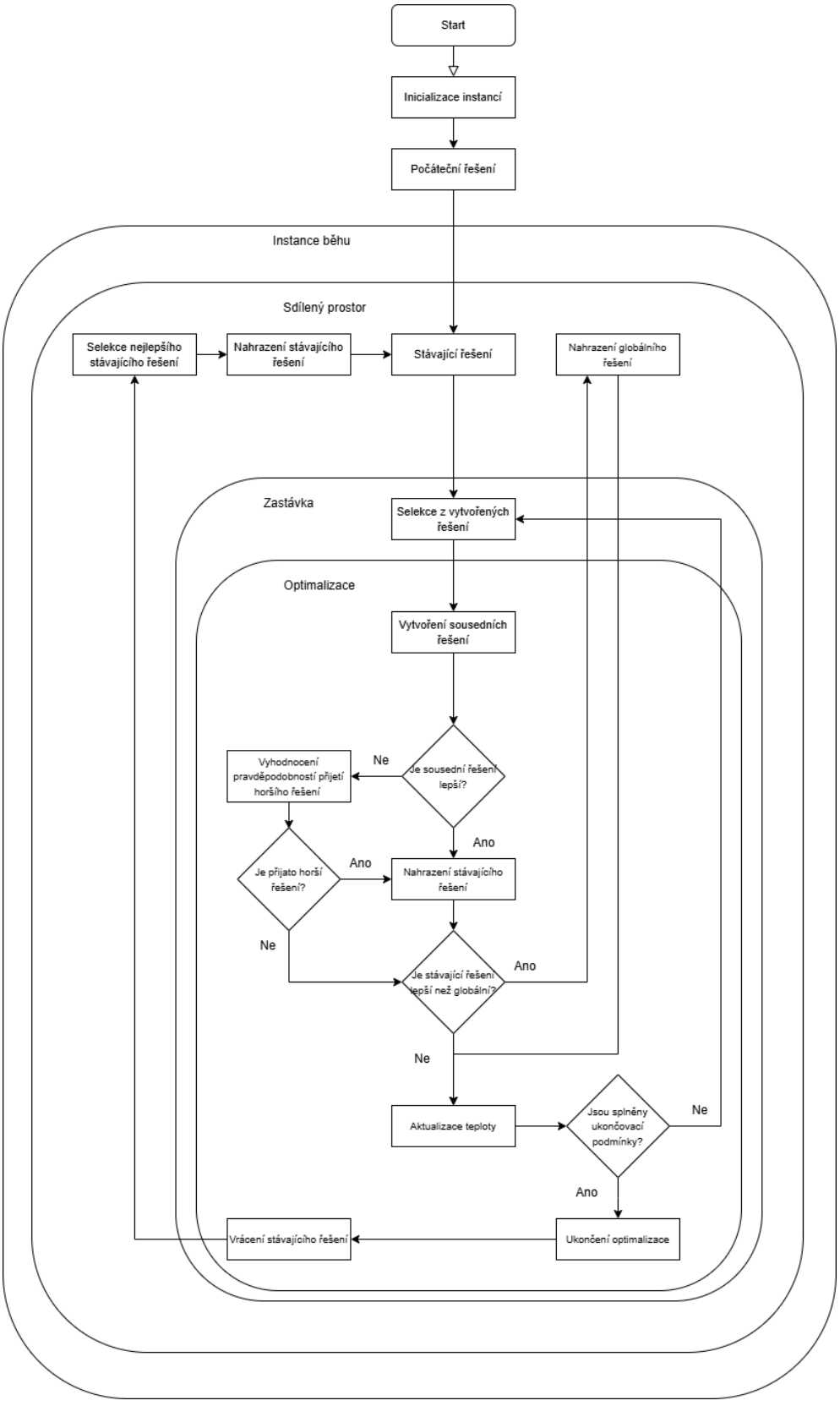
### 3.2.1 Parallel tempering

Algoritmus realizuje paralelní simulované žíhání, kde jednotlivá vlákna běží souběžně s rozdílnou teplotou $T_i$. Tyto jednotlivé instance průběžně optimalizují svá řešení, po určitém počtu kroků dojde ke koordinované výměně teplot mezi sousedními instancemi. Po provedení specifické počtu $N_{swap}$ kroků dojde k výběru dvojice sousedních instancí. S pokusem o záměnu konfigurací mezi sebou s určitou pravděpodobností přijetí $p$ a zamění si konfiguraci mezi sebou. [28]

### 3.2.2 Hybrid parallel simulated annealing

Na základě této myšlenky byl navrhnut upravený algoritmus hybridní algoritmus, kombinující paralelní simulované žíhání s modifikovanou verzí. Pozastavení je zde použité po specifickém počtu kroků, podobně jako v Parallel Tempering [28]. Dochází zde k volbě nejlepšího řešení, ze kterého dochází k pokračování běhu všech souběžných instancí algoritmu, a to do té doby, než dojde k opětovnému zastavení a opakování volby nejlepšího řešení. V případě zastavení jde o pomyslné restartování PSA. Při restartu dochází k pokračování z vybraného řešení pomocí selekce ze všech aktuální souběžných řešení. Algoritmus končí, jakmile dojde ke splnění definovaných počtu pozastavení.

Níže je možné vidět vývojový diagram tohoto algoritmu na obrázku 5.




*Obrázek 5 – Vývojový diagram modifikovaného paralelního simulovaného žíhání (autor)*

# 4 EXPERIMENTÁLNÍ ČÁST

Tato část se věnuje popis u jednotlivých implementací algoritmu simulovaného žíhání a jeho variant. Následuje provedení experimentů testujících jejich výkonnost při specifikovaných podmínkách a vstupních parametrů. Veškeré algoritmy byly vyvíjené v jazyce Java.

## 4.1 Implementace Tradičního simulovaného žíhání

Při praktické implementaci tradičního simulovaného žíhání jsou definované klíčové počáteční vstupní parametry.

1. InitialTemperature určuje počáteční teplotu
2. CoolingRate určuje rychlost ochlazovacího schématu
3. isSolutionRandom slouží k rozhodnutí, zda počáteční řešení je zcela náhodné nebo statické. Statické řešení je vytvořené pomocí parametru s názvem initialSolution, to je řešení vytvořené podle posloupnosti měst z načteného datového souboru. Při použití náhodného řešení je použito stejné řešení, ale je na něj použita metoda shuffle() ze standardní knihovní třídy java.util.Collections [32], která vytvoří náhodné řešení. Zvolené počáteční řešení je následně nastavené jako aktuální, a zároveň dosavadní nejlepší (globální optimum).
4. CountOfSteps určuje počet kroků optimalizace.
5. MutationType slouží pro zvolení způsobu generování nového souseda (dílčího řešení).

Implementace získávání nových dílčích řešení (sousedů) pomocí mutace je následující:

1. SWAP – Swap provádí záměnu pozice jednoho prvku z řešení za jiný. Pozice jsou generovány náhodně z celého rozsahu řešení.
2. PA_RE – Partial reverse vygeneruje náhodné rozmezí z řešení, ve kterém dochází k záměně pořadí prvků pomocí metody reverse() ze standardní knihovní třídy java.util.Collections [32], která má vstupní parametr sublist zvolených prvků. Zde je nutná podmínka nejmenšího nutného rozsahu, pro korektní funkčnost jsou nutné tři prvky. Bez této podmínky by jinak došlo k realizaci předchozí mutace.
3. PA_SH – Partial shuffle dodržuje stejné generování rozsahu jako předchozí varianta s odlišným přístupem. Zvolený rozsah prvků (měst) náhodně promíchá mezi sebou pomocí metody shuffle, která má vstupní parametr sublist zvolených prvků.

Postupné kroky algoritmu jsou následovné:

1. Inicializace vstupních parametrů, čítače provedených kroků optimalizací
2. Vygenerování nového souseda dle zvolené mutace
3. Rozhodnutí o přijetí nového dílčího řešení
4. Ověření, zda je stávající řešení lepší než globální optimum, případně aktualizace globálního optima
5. Aktualizace parametrů
6. Ověření splnění ukončovací podmínky

Při rozhodování, zda je nové dílčí řešení přijato, dochází k vygenerování náhodného čísla v rozmezí 0 až 1, které je porovnané s pravděpodobností přijetí řešení. Výpočet pravděpodobnosti přijetí je následovný. V případě že je nové dílčí řešení kvalitnější (lepší), pak je přijmutí automatické s pravděpodobností 1. V druhém případě, kdy je méně kvalitní (horší), dojde k výpočtu pomocí exponenciální funkce. V exponenciální funkci dojde k vypočtení rozdílu cost funkce mezi stávajícím a dílčím řešením, který je následně vydělen stávající teplotou (currentTemperature). Výsledná hodnota je poté porovnána s dříve vygenerovaným náhodným číslem. V případě přijetí řešení dochází k nahrazení stávajícího řešení za nové dílčí řešení.

Při porovnání globálního řešení je porovnaná cost funkce globálního řešení s aktuálním řešením. V případě že aktuální řešení má lepší vyhodnocení cost funkce, je nahrazeno (aktualizováno) globální řešení.

## 4.2 Implementace Modifikovaného simulovaného žíhání

Modifikované simulované žíhání je prakticky implementováno následovně. Při spuštění algoritmu jsou nastavené vstupní parametry stejné jako u tradiční verze. Nové přidané vstupní parametry jsou následovné:

1. NeighboursCount určuje počet generovaných dílčích řešení (sousedů).
2. SelectionType určuje typ selekce z dílčích řešení (sousedů).
3. TournamentSize v případě používání selekce pomocí turnajového výběru definuje počet realizovaných soubojů.

Postupné kroky algoritmu jsou následovné:

1. Inicializace vstupních parametrů, čítače provedených kroků optimalizací
2. Vygenerování nových sousedů
3. Zvolení z dílčích řešení pomocí selekce
4. Rozhodnutí o přijetí nového dílčího řešení
5. Ověření, zda je stávající řešení lepší než globální optimum, případně aktualizace globálního optima
6. Aktualizace parametrů
7. Ověření splnění ukončovací podmínky

### 4.2.1 Implementace ranking selection

Implementace ranking selection proběhla ve dvou variantách. Obě varianty provádí seřazení vygenerovaných dílčích řešení vzestupně podle jejich vyhodnocení cost funkce. Seřazené hodnoty jsou následně vložené do mapy, kde klíčem je dílčí řešení a hodnotou jeho pořadí. Vstupní parametry metody jsou následující:

1. List <Path> paths – seznam dílčích řešení.
2. ByBest – rozhoduje, zda se vyhledává následně nejlepší řešení nebo pomocí ruletového kola.

Kroky v algoritmu jsou následovné:

1. Seřazení dílčích řešení dle vyhodnocení jejich cost funkcí.
2. Vložení seřazených hodnot do mapy pomocí cyklu.
3. Selekce řešení podle zvoleného typu (přímým výběrem nejlepšího řešení nebo ruletovým kolem).

První varianta provádí závěrečný výběr pomocí přímé selekce, tedy vstupní parametr byBest je nastaven na hodnotu true.

Druhá varianta provádí závěrečný výběr poměrovým ruletovým kolem, tedy vstupní parametr byBest je nastaven na hodnotu false. Popis implementace ruletového kola byl již popsán v teoretické podkapitole, zde je pouze s rozdílem sumy hodností z mapy.

### 4.2.2 Implementace tournament selection

Implementace tournament selection proběhla ve dvou variantách s dvěma možnými kombinacemi. Obě varianty vygenerovaná dílčí řešení vkládají do mapy, kde klíčem je dílčí řešení a počet vyhraných soubojů, na počátku je inicializována na 0. Vstupní parametry metody jsou následující:

1. List <Path> paths – seznam dílčích řešení
2. ByBest – rozhoduje, zda se vyhledává následně nejlepší řešení nebo pomocí ruletového kola.
3. PickCount – celočíselná proměnná (V případě, že je omezen počet realizovaných soubojů je zde navíc jeden parametr určující počet provedených soubojů)

Kroky v algoritmu jsou následovné:

1. Vložení dílčích řešení do mapy
2. Rozlišení dle počtu realizovaných zápasů
    a. Realizaci veškerých zápasů, přičtení hodnoty k příslušnému dílčímu řešení v případě výhry.
    b. Realizace zvoleného počtu unikátních zápasů dle proměnné pickCount, přičtení hodnoty k příslušnému dílčímu řešení v případě výhry.
3. Selekce řešení
    a. Přímým výběrem nejlepšího řešení
    b. Ruletovým kolem

Implementace první varianty realizovala veškeré souboje. Vzhledem k symetrii matice došlo k zrychlení výpočtů, bylo nadbytečné provádět duplicitní výpočty. Získávány byly pouze hodnoty pod diagonálou pomocí dvou zanořených for cyklů. Cykly zanořené v sobě při porovnání vždy vyhodnocují cost funkci dvou dílčích řešení, kterou se porovnají. Tento přístup přináší značné urychlení v případě velkého množství dílčích řešení.

Implementace druhé varianty je podobná první variantě, souboje jsou prováděné pod diagonálou pouze v omezeném počtu. Je zde nutné zavést ošetřující podmínku v případě, že počet soubojů by byl vyšší, než je množství dostupných řešení. V této situace dojde k převedení na způsob první varianty. Vygenerované náhodné pozice jedinců, kteří budou porovnáváni, jsou ukládány do HashSetu. Generování je ošetřené, aby nedošlo k porovnávání řešení sama se sebou a zároveň vícenásobnému provedení stejných soubojů.

Veškeré implementované varianty následně byly testované jako variace všech dostupných kombinací. V tabulce 1 níže jsou vypsány veškeré realizované kombinace.

*Tabulka 1 – Výčet dostupných kombinací modifikovaného simulovaného žíhání*

| Typ selekce | Mutace | Výběr řešení | Počet soubojů v případě turnajového souboje |
|---|---|---|---|
| Ranking selection | Swap | Přímý výběr nejlepšího | X |
| Ranking selection | Partial reverse | Přímý výběr nejlepšího | X |
| Ranking selection | Partial shuffle | Přímý výběr nejlepšího | X |
| Ranking selection | Swap | Ruletové kolo | X |
| Ranking selection | Partial reverse | Ruletové kolo | X |
| Ranking selection | Partial shuffle | Ruletové kolo | X |
| Tournament selection | Swap | Přímý výběr nejlepšího | Realizace všech soubojů |
| Tournament selection | Partial reverse | Přímý výběr nejlepšího | Realizace všech soubojů |
| Tournament selection | Partial shuffle | Přímý výběr nejlepšího | Realizace všech soubojů |
| Tournament selection | Swap | Ruletové kolo | Realizace náhodných soubojů |
| Tournament selection | Partial reverse | Ruletové kolo | Realizace náhodných soubojů |
| Tournament selection | Partial shuffle | Ruletové kolo | Realizace náhodných soubojů |

## 4.3 Implementace Paralelního simulovaného žíhání

Implementace paralelní simulované žíhání byla realizována ve dvou variantách.

Implementace obou variant je pseudoparalelní (souběžné kroky paralelních instancí SA probíhají za sebou v jednom vlákně). Získané výsledky jednotlivých instancí byly ukládány do seznamu. Veškeré běhy byly na sobě nezávislé. Algoritmus korektně dodržuje sdílení globálního optima v proměnné globalBestSolution mezi všemi spuštěními.

První varianta je pomocí paralelizace tradičního simulovaného žíhání na předem definovaný počet instancí. Při spuštění algoritmu jsou nastavené vstupní parametry stejné, jako u tradičního simulovaného žíhání. Nově přidaný vstupní parametr je threadsCount – definuje počet použitých instancí.

Postupné kroky algoritmu jsou následovné:

1. Inicializace vstupních parametrů, čítače provedených kroků optimalizací
2. Vygenerování nového souseda
3. Rozhodnutí o přijetí nového dílčího řešení
4. Ověření, zda je stávající řešení lepší než globální optimum, případně aktualizace globálního optima
5. Aktualizace parametrů
6. Ověření splnění ukončovací podmínky

Implementace druhé varianty je značně komplexnější, tvoří ji hybridní algoritmus mezi první variantou a modifikovaným simulovaným žíháním.

Nově přidané vstupní parametry jsou

1. ThreadsCount – definuje počet souběžných instancí.
2. NeighboursCount určuje počet generovaných sousedů (dílčích řešení).
3. SelectionType určuje typ selekce ze sousedů (dílčích řešení).
4. TournamentSize v případě používání selekce pomocí turnajového výběru definuje počet realizovaných soubojů

Postupné kroky algoritmu jsou následovné:

1. Inicializace vstupních parametrů, čítače provedených kroků optimalizací, čítač počtu zastávek.
2. Vygenerování nových sousedů ze stávajícího řešení.
3. Zvolení z dílčích řešení pomocí selekce.
4. Rozhodnutí o přijetí nového dílčího řešení.
5. Ověření, zda je stávající řešení lepší než globální optimum, případně aktualizace globálního optima.
6. Aktualizace parametrů.
7. Ověření splnění ukončovací podmínky.
8. Pozastavení, při které dojde ke zvolení nejkvalitnějšího dílčího řešení, které nahradí stávající řešení pro všechny instance.
9. Zvýšení počtu realizovaných zastávek.
10. Ověření splnění realizování počtu zastávek.

Pro korektní porovnání rozdílného počtu zastávek a jednotlivé varianty nakonfigurovat, tak aby měly, pokud možno stejný počet vyhodnocení cost funkce (víc zastávek znamená častější generování sousedů. Algoritmus by tak měl skončit po menším počtu kroků, protože se stejným počtem kroků jako konfigurace s menším počtem zastávek by měl nefér výhodu).

Při výpočtu celkového množství vyhodnocení cost funkcí je nutné na počátku zvolit čtyři parametry. Celkový počet kroků optimalizace, počet vláken, počet generovaných sousedů a počet zastávek. Při samotném výpočtu na počátku je již definované celkové množství kroků optimalizace. Kroky značí počet vyhodnocení cost funkcí. Tyto kroky je nutné vydělit počtem zastávek, počtem vláken a počtem generovaných sousedů, které budou realizované. Získaný výsledek udává počet kroků v jedné zastávce s automatickým zaokrouhlením na celé číslo. Pro zjištění celkového počtu vyhodnocení cost funkcí je nutné celkový počet kroků vynásobit množstvím generovaných sousedů, zastávek a počtem souběžných instancí SA. Vzorový výpočet je dostupný v tabulce 2.

*Tabulka 2 – Vzorový výpočet počtu vyhodnocení cost funkce v zastávce*

| Počet generovaných sousedů | Počet instancí | Počet zastávek | Počet kroků v zastávce | Celkový počet vyhodnocení cost funkcí |
|---|---|---|---|---|
| 10 | 3 | 3 | 250 | 22 500 |
| 10 | 3 | 10 | 90 | 27 000 |

## 4.4 Konfigurace algoritmů

Definování konfigurace pro jednotlivé algoritmy byla pro testování jeden z klíčových bodů. Vzhledem k podobnosti a stejným parametrům algoritmů jsou v mnoha aspektech stejné počáteční vstupní parametry. Testování bylo provedené na implementovaných algoritmech, rovněž byly otestovány i veškeré dostupné kombinace, tedy různé druhy mutací pro generování sousedů, a různé druhy selekce. Cílem tohoto testování bylo zvolit nejvhodnější kombinace, které jsou vhodné pro další porovnávání s ostatními algoritmy a zhodnocení získaných výsledků.

Testování bylo realizováno na náhodném počátečním řešení.

Společné počáteční parametry jsou dostupné v tabulce níže, jednotlivé specifikované jsou dále popsané v jednotlivých kapitolách u konkrétních konfigurací algoritmů. Testovací dataset pro tyto konfigurace byl o velikosti 1979 měst [29]. V získaných grafech byly porovnávány nalezená globální optima, kterých jednotlivé algoritmy dosáhly. Shrnutí je dostupné v tabulce 3 níže.

*Tabulka 3 – Počáteční společná konfigurace*

| | |
|---|---|
| Počáteční teplota | 1000 |
| Ochlazovací hodnota | 0.999 |
| Počet testovaných spuštění algoritmu | 100 |
| Počet měst v datovém souboru | 1979 |

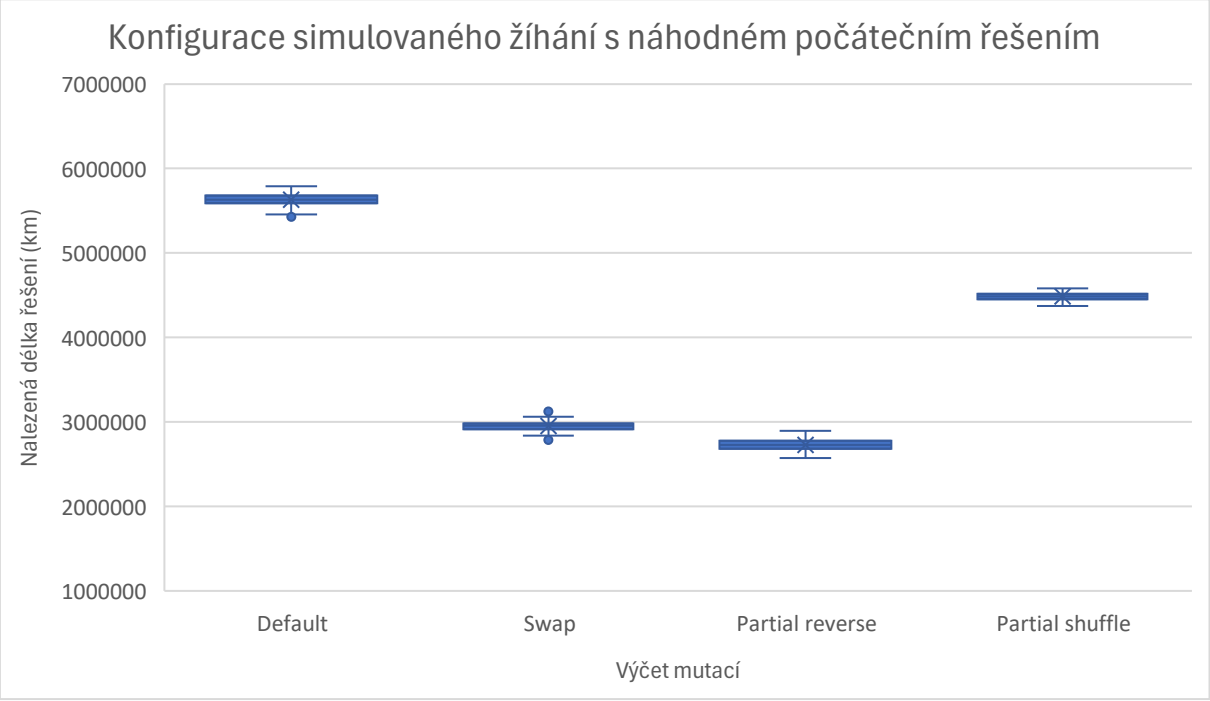
## 4.5 Konfigurace simulovaného žíhání

Testování tradičního simulovaného žíhání bylo realizováno na statickém a na náhodném počátečním řešení. Počet vyhodnocení cost funkce bylo stanoveno na 10000. Získané výsledky jsou porovnané dle mutací, následně zanesené do krabicového grafu. Cílem bylo nalézt vhodnou mutaci pro další porovnání.

### 4.5.1 Náhodné počáteční řešení

Získané výsledky z realizace náhodného počátečního řešení jsou dostupné v obrázku 6, zde je možné vidět porovnání jednotlivých mutací.

Pro porovnání byly zvolené průměrné hodnoty jednotlivých variant. Průměrné počáteční řešení tohoto náhodného řešení bylo 5 667 738 km, tedy velmi vysoké. V grafu na obrázku 6 je možné vidět, že zlepšení je poměrně úspěšně i v nejslabší variantě. Spektrum získaných hodnot je poměrně ustálené ve všech mutacích. Porovnání jednotlivých mutací je možné vidět v tabulce 4, kde je dostupné průměrné nalezené řešení a procentuální zlepšení vůči počátečnímu náhodnému řešení. Při použití náhodného počátečního řešení je nejlepší mutace partial reverse, která přináší průměrné zlepšení o 51,54 %.




*Obrázek 6 – Graf porovnání konfigurace nalezených řešení jednotlivých mutací na tradičním simulovaném žíhání s náhodným řešením (autor)*

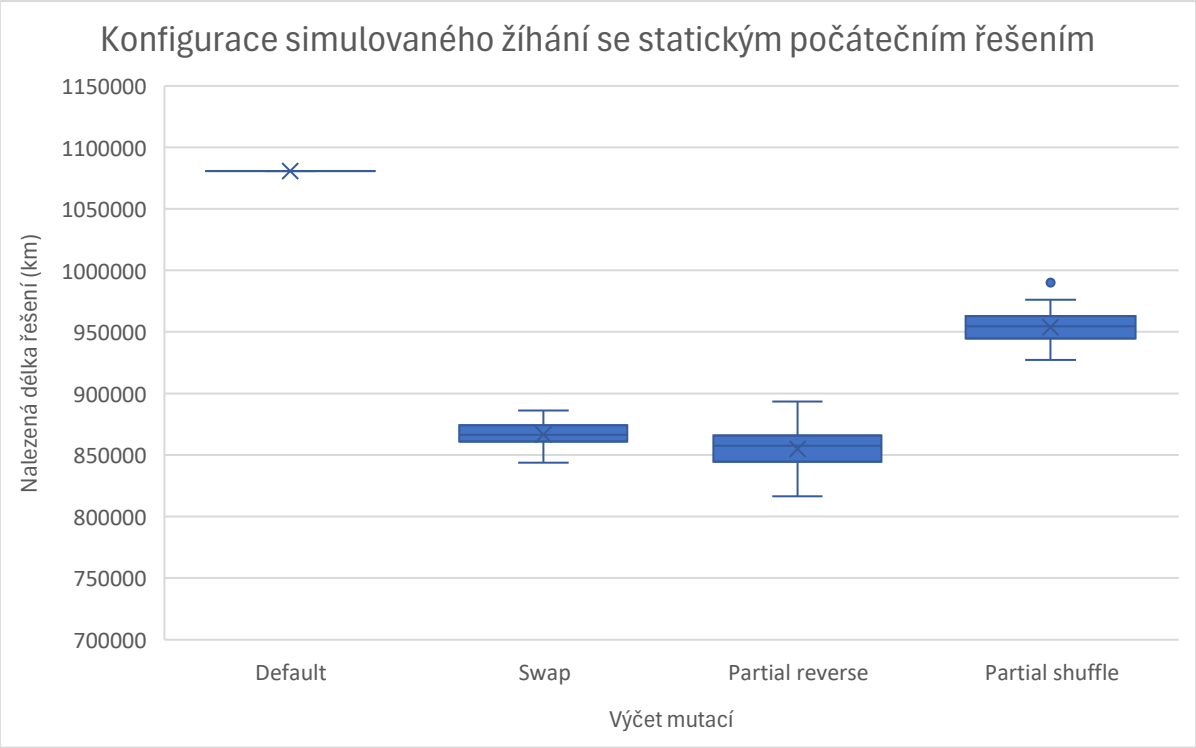
*Tabulka 4 – Porovnání nalezených řešení dle mutací na tradičním simulovaném žíhání s náhodným řešením*

| Typ mutace | Průměrné řešení (km) | Průměrné zlepšení (%) |
|---|---|---|
| Swap | 2 950 752,874 | 47,59 |
| Partial reverse | 2 728 250,669 | 51,54 |
| Partial shuffle | 4 485 966,643 | 20,32 |

### 4.5.2 Statické počáteční řešení

V případě statického počátečního řešení byly nalezeny velmi podobné výsledky jako v náhodném počátečním řešení. Počáteční hodnota byla 1080605, tedy několikanásobně menší než předchozí řešení.

V grafu na obrázku 7 je možné vidět, že zlepšení již nejsou tak rozdílná jako při náhodném počátečním řešení. Spektrum získaných hodnot je poměrně ustálené ve všech mutacích. Porovnání jednotlivých mutací je možné vidět v tabulce 5, kde je dostupné průměrné nalezené řešení a procentuální zlepšení vůči počátečnímu statickému řešení. Při použití statického počátečního řešení je nejlepší mutace partial reverse, která přináší průměrné zlepšení o 51,54 %. Nejefektivnější mutací je opět partial reverse.




*Obrázek 7 – Graf porovnání konfigurace nalezených řešení dle mutací na tradičním simulovaném žíhání se statickým řešením (autor)*

*Tabulka 5 – Porovnání nalezených řešení dle mutací na tradičním simulovaném žíhání se statickým řešením*

| Typ mutace | Průměrné nalezené řešení (km) | Průměrné zlepšení vůči počátečnímu řešení (%) |
|---|---|---|
| Swap | 866 687,27 | 19,80 |
| Partial reverse | 854 985,86 | 20,88 |
| Partial shuffle | 953 673,81 | 11,74 |

Při závěrečném zhodnocení těchto dvou přístupu, kdy je použité zcela náhodné řešení a statické tak v obou případech působí nejefektivněji mutace partial reverse, jak je možné vidět v tabulce 6.

Zároveň je možné vidět že v obou případech je nejefektivnější mutace partial reverse. Na druhém místě je swap mutace, která ale i přes její drobné změny v podobě jedné pozice přináší velmi dobré výsledky.

Z časového hlediska je ovšem nejrychlejší mutace swap v případě statického řešení, nicméně rozdíly jsou zanedbatelné. To samé platí i pro náhodné počáteční řešení, zde je však nejrychlejší mutace partial shuffle, která ovšem přináší velmi slabé výsledky oproti partial reverse.

*Tabulka 6 – Porovnání nalezeného zlepšení u tradičního simulovaného žíhání*

| Typ mutace | Statické počáteční řešení | | Náhodné počáteční řešení | |
|---|---|---|---|---|
| | Zlepšení řešení (%) | Časové trvání (s) | Zlepšení řešení (%) | Časové trvání (s) |
| Swap | 19,79 | 35,123 | 47,59 | 39,947 |
| Partial reverse | 20,88 | 39,130 | 51,54 | 43,368 |
| Partial shuffle | 11,75 | 40,893 | 20,32 | 38,578 |

## 4.6 Konfigurace Modifikovaného simulovaného žíhání

Tato modifikovaná verze přináší velké množství dostupných kombinací selekcí. Vzhledem ke generování několika sousedů nikoliv pouze jednoho, jak tomu je u tradiční varianty je nutné otestovat. Vyhodnocení získaných výsledků bude testováno s mutací partial reverse, která se dle předchozího testování zdá být jako nejpřívětivější volbou pro tento dataset. V modifikovaném simulovaném žíhání bylo nutné otestovat různý počet generovaných sousedů. Pro otestování, jaký počet bude nejvíc vyhovovat, které selekci případně kompromis, jaký počet bude vyhovovat všem.

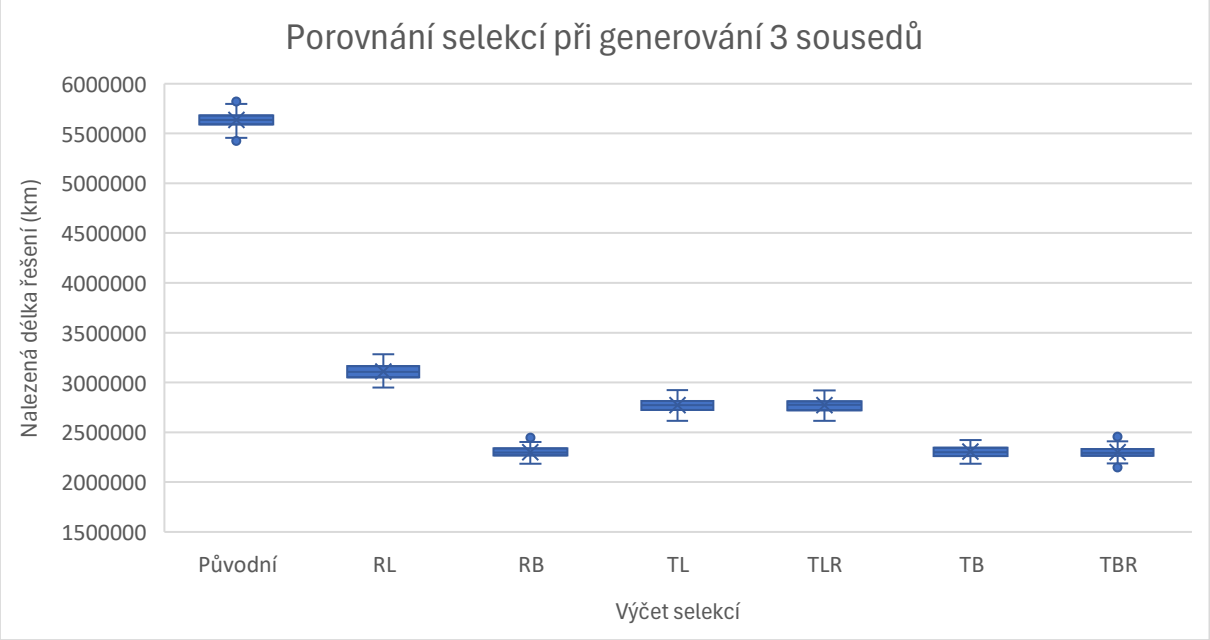
Při tomto testování došlo také k vyčlenění selekce, které nejsou příliš efektivní a které mají potenciál být nejvhodnější. Z důvodu přehlednosti jsou jednotlivá porovnání v jednotlivých podkapitolách selekcí.

Podrobná konfigurace sousedů je dostupná níže v tabulce 7.

*Tabulka 7 – Konfigurace testovaných sousedů*

| Celkový počet vyhodnocení cost funkce | 15 000 |
|---|---|
| Počet generovaných sousedů | Počet vyhodnocení cost funkce jednotlivce |
| 3 | 5000 |
| 5 | 3000 |

Při prvotním testování bylo nutné zvolit, která ze selekcí bude nejvíc efektivní a dále dle toho na vybrat na, které bude testováno rozdílné množství generovaných sousedů. Z grafu na obrázku 8 níže je možné vidět, že při generování 3 sousedů s vyhodnocením 5000 cost funkcí přináší přívětivé výsledky.




*Obrázek 8 – Graf porovnání selekce u 3 generovaných sousedů (autor)*

Nicméně volba selekce, která je vhodná pro další porovnání je nutné rozhodnout mezi několika kandidáty.

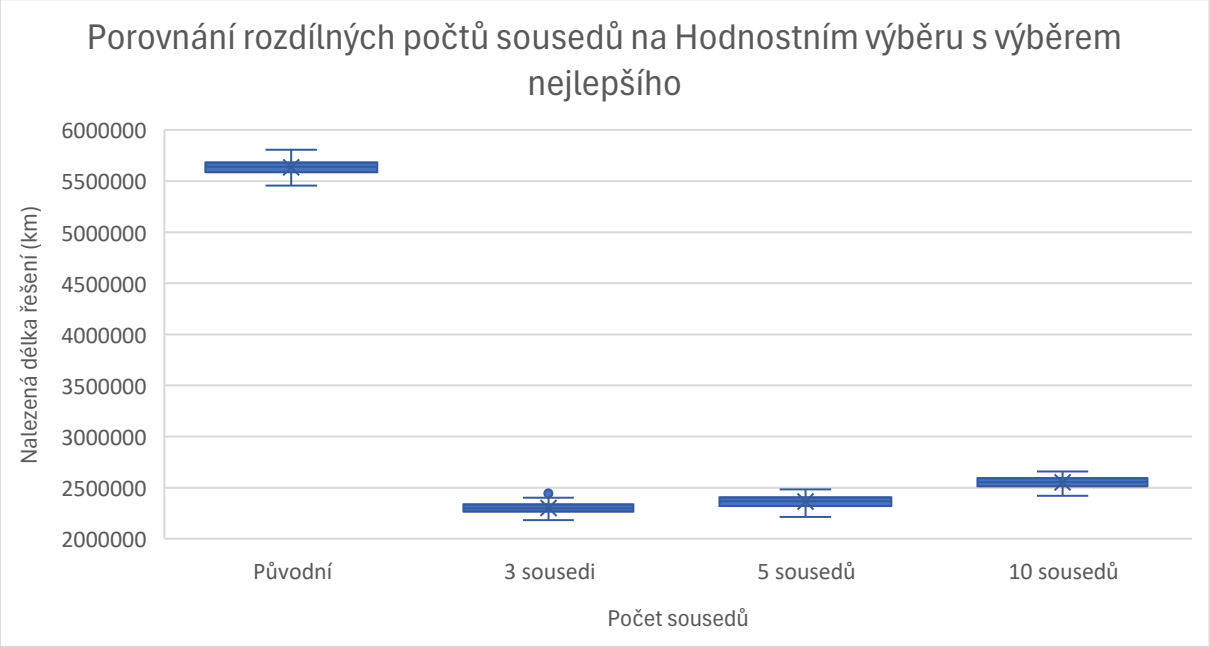
1. Hodnostní výběr s výběrem nejlepšího
2. Turnajový výběr s výběrem nejlepšího
3. Turnajový výběr s omezeným počtem soubojů s výběrem nejlepšího

V tabulce 8 je možné vidět, že výsledky jsou opravdu velmi podobné. Pro otestování změn při generování různého počtu sousedů byl zvolen Hodnostní výběr s výběrem nejlepšího.

*Tabulka 8 – Porovnání nejvýkonnějších selekcí u 3 generovaných sousedů*

| Selekce | Nalezené průměrné řešení (km) | Průměrné zlepšení (%) | Časové trvání (s) |
|---|---|---|---|
| Hodnostní výběr s výběrem nejlepšího | 2 301 339,458 | 59,16 | 38,792 |
| Turnajový výběr s výběrem nejlepšího | 2 305 035,775 | 59,09 | 38,690 |
| Turnajový výběr s omezeným počtem soubojů s výběrem nejlepšího | 2 298 352,758 | 59,21 | 38,906 |
| Hodnostní výběr s výběrem ruletovým kolem | 3 111 132,919 | 44,79 | 37,480 |
| Turnajový výběr s výběrem ruletovým kolem | 2 771 907,671 | 50,81 | 37,942 |
| Turnajový výběr s omezeným počtem soubojů s výběrem ruletovým kolem | 2 771 979,517 | 50,81 | 38,102 |

Při testování dalšího různého množství generovaných sousedů je možné vidět zjištěné výsledky v grafu níže na obrázku 9. Zde je možné vidět, že nejlepší výsledky jsou v případě generování třech sousedů.




*Obrázek 9 – Graf porovnání počtu sousedů na Hodnostním výběru s výběrem nejlepšího (autor)*

*Tabulka 9 – Porovnání průměrných nalezených řešení*

| Počet sousedů | Průměrné řešení (km) | Průměrné zlepšení (%) | Časové trvání (s) |
|---|---|---|---|
| 3 | 2 301 339,458 | 59,17 | 38,792 |
| 5 | 2 364 088,365 | 58,05 | 34,886 |
| 10 | 2 551 252,557 | 54,73 | 30,248 |

V tabulce 9 výše je možné vidět, že nejvyšší průměrné zlepšení přináší generování 3 sousedů. Z časového hlediska lehce zaostává o necelé 4 sekundy, nicméně i přes toto delší trvání, zda se být nejvhodnější provést testování jednotlivých selekcí s 3 sousedy. Dle těchto nalezených výsledků je pro další ověření výkonnosti jednotlivých selekcí zvoleno generování 3 sousedních řešení.

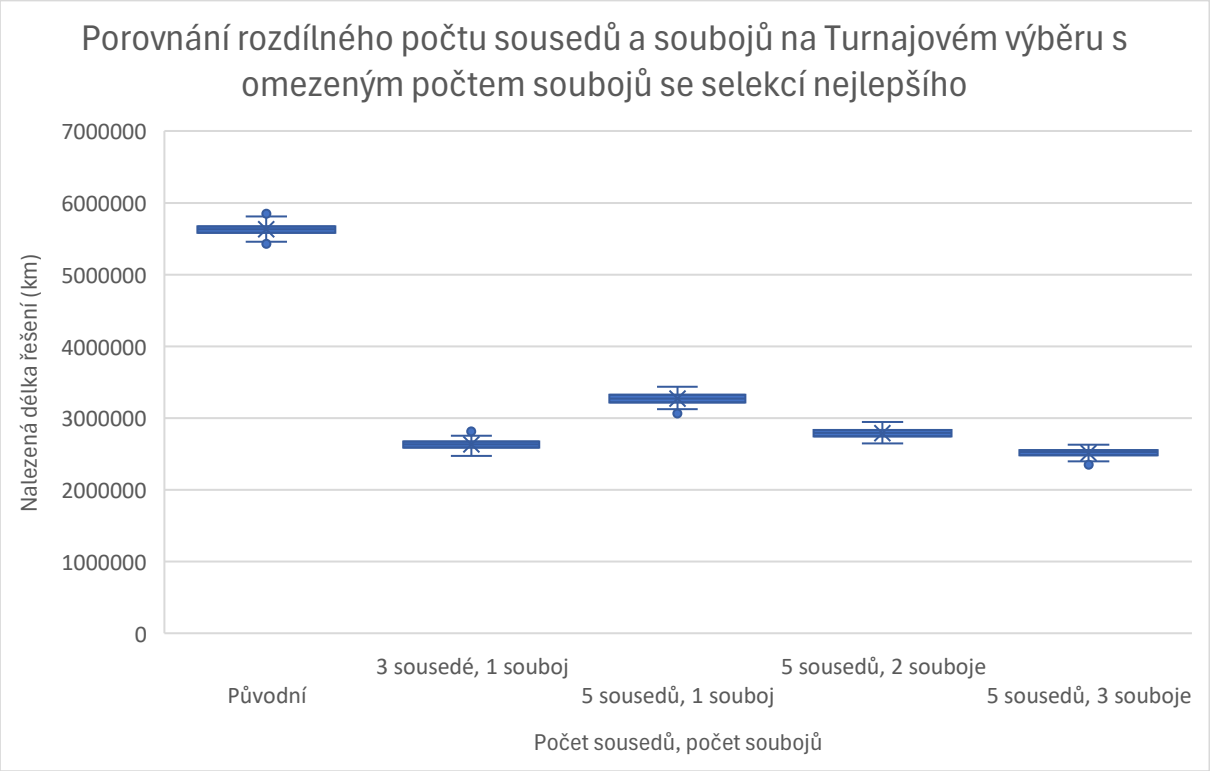
Ještě je vhodné provést jednou testování pro Turnajový výběr s omezeným počtem soubojů. Jedná se o speciální případ. Zde je použita znalost z předchozího testování, kde lepší byla varianta této selekce s výběrem nejlepšího oproti proporcionálnímu ruletovému kolu.

Na této selekci tedy byly ještě ověřené různé počty realizovaných soubojů. V tabulce 10 níže jsou jednotlivá testování soubojů shrnuta.

*Tabulka 10 – Seznam generovaných kombinací soubojů*

| Počet generovaných sousedů | Počet realizovaných soubojů |
|---|---|
| 3 | 1 |
| 5 | 1 |
| 5 | 2 |
| 5 | 3 |

V grafu níže na obrázku 10 je možné vidět získané výsledky po testování odlišného počtu soubojů na různém počtu generovaných sousedů. Průměrné počáteční řešení je 5 630 025 km Nejpřívětivější výsledky v této selekci přináší generování 5 sousedů s následným provedením 3 turnajových soubojů.




*Obrázek 10 – Graf porovnání rozdílného počtu sousedů a soubojů na Turnajovém výběru s omezeným počtem soubojů s výběrem nejlepšího (autor)*

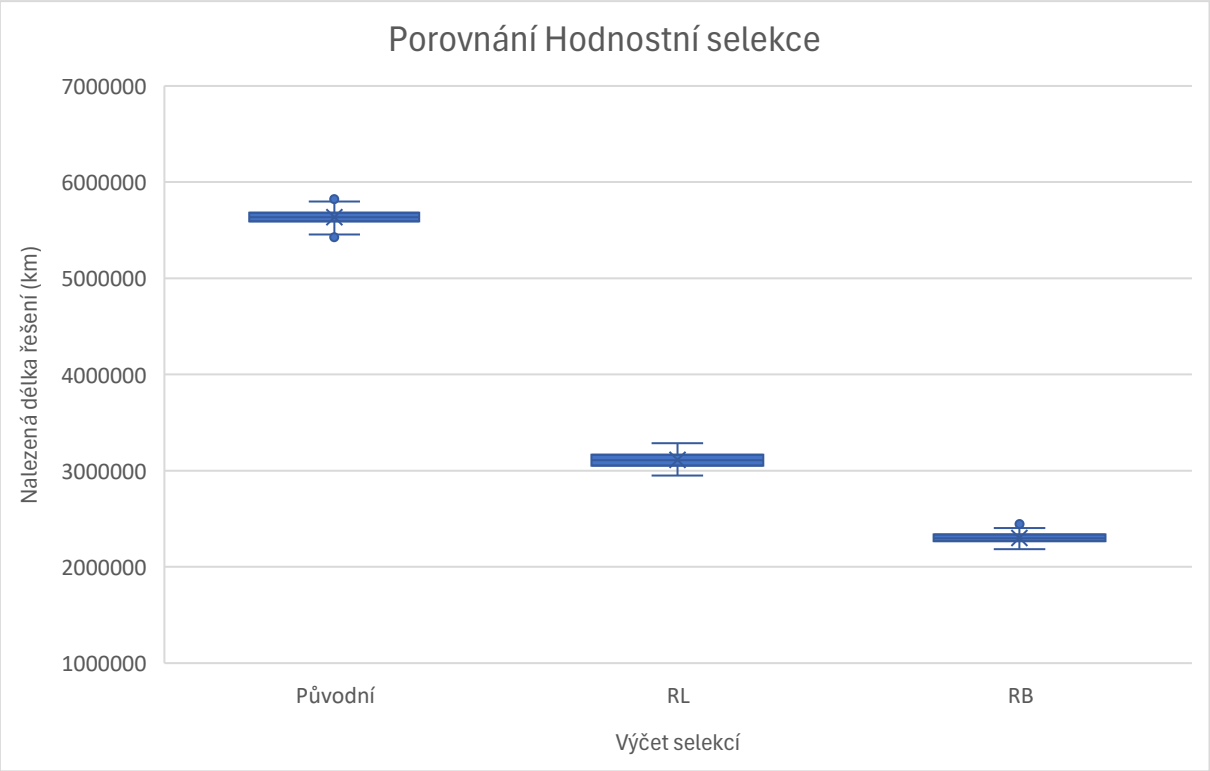
Tabulka 11 níže popisuje nalezená průměrná řešení. Zde průměrná nalezená řešení potvrzují, že nejvhodnějším počtem realizovaných soubojů u Turnajového výběru s omezeným počtem je vhodné použít 5 generovaných sousedů a 3 souboje. Z časového hlediska jsou výsledky velmi podobné.

*Tabulka 11 – Porovnání průměrných nalezených řešení u Turnajového výběru s omezeným počtem soubojů s výběrem nejlepšího*

| Počet sousedů | Počet soubojů | Průměrné řešení (km) | Průměrné zlepšení (%) | Časové trvání (s) |
|---|---|---|---|---|
| 3 | 1 | 2 634 906,865 | 53,20 | 38,189 |
| 5 | 1 | 3 274 156,453 | 41,84 | 36,671 |
| | 2 | 2 791 228,666 | 50,42 | 35,672 |
| | 3 | 2 515 009,405 | 55,33 | 35,658 |

### 4.6.1 Hodnostní selekce

Vzájemné porovnání Hodnostní selekce bylo realizováno na dvou variantách s následnou selekcí z dostupných řešení pomocí poměrové ruletového kola a výběrem nejlepšího řešení. Zde v grafu na obrázku 11 je možné vidět porovnání Hodnostní selekce. Zde velmi přesvědčivě vyhrává varianta s výběrem nejlepšího řešení. Průměrné počáteční řešení bylo 5 634 838 km. Porovnané výsledky jsou dostupné, v tabulce 12. Varianta s výběrem nejlepšího řešení přináší relativně dobré výsledky. Průměrné zlepšení je o 59,16 % zatímco u poměrového ruletového kola pouze 44,79 %. Rozdíl časového trvání je v obou variantách odlišný pouze minimální, zhruba o 1 sekundu.




*Obrázek 11 – Graf porovnání Hodnostní selekce (autor)*

*Tabulka 12 – Porovnání Hodnostní selekce*

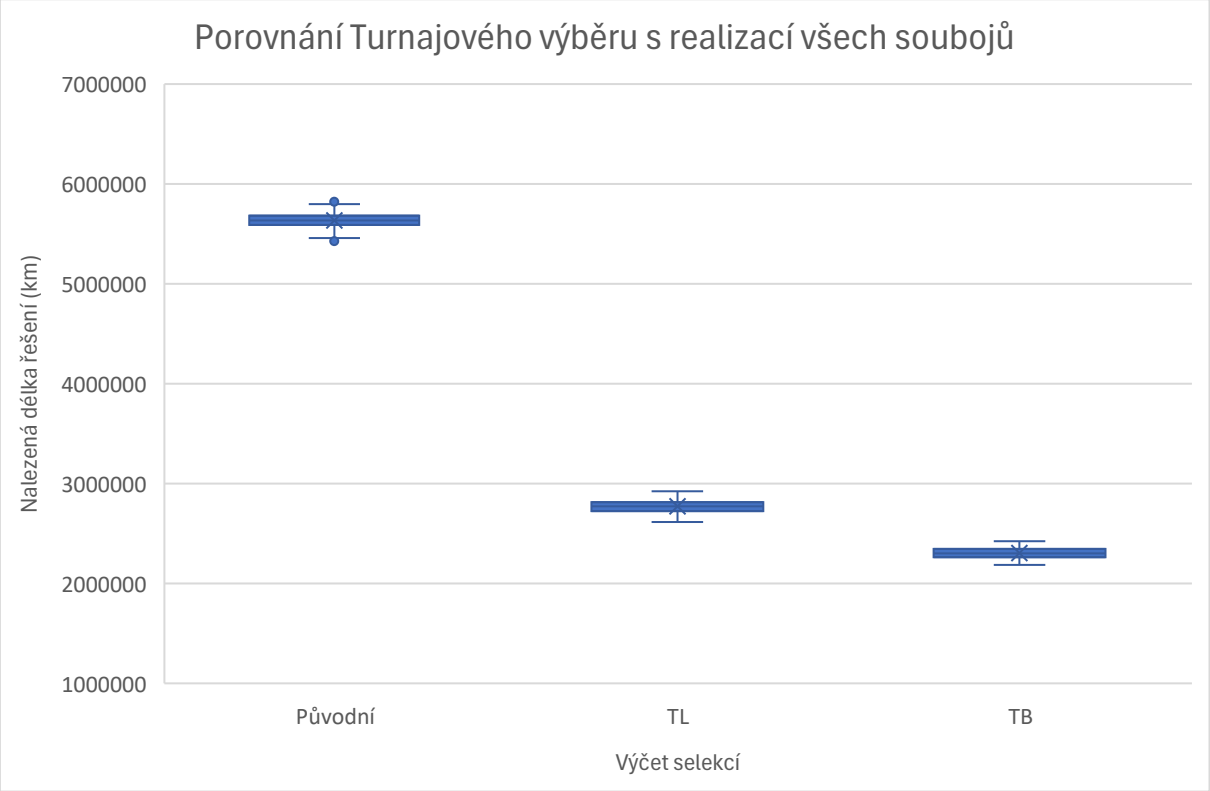
| Varianta | Průměrné nalezené řešení (km) | Průměrné zlepšení (%) | Časové trvání (s) |
|---|---|---|---|
| Ruletové kolo | 3 111 132,919 | 44,79 | 37,480 |
| Výběr nejlepšího | 2 301 339,458 | 59,16 | 38,792 |

### 4.6.2 Turnajový výběr

Testování Turnajového výběru bylo realizováno ve dvou variantách. Jednotlivé varianty jsou analyzovány v podkapitolách. Obě testované varianty byly realizovány ve dvou variantách následného výběru z dostupných řešení. Pomocí poměrového ruletového kola a výběrem nejlepšího řešení.

#### 4.6.2.1 S realizací veškerých zápasů

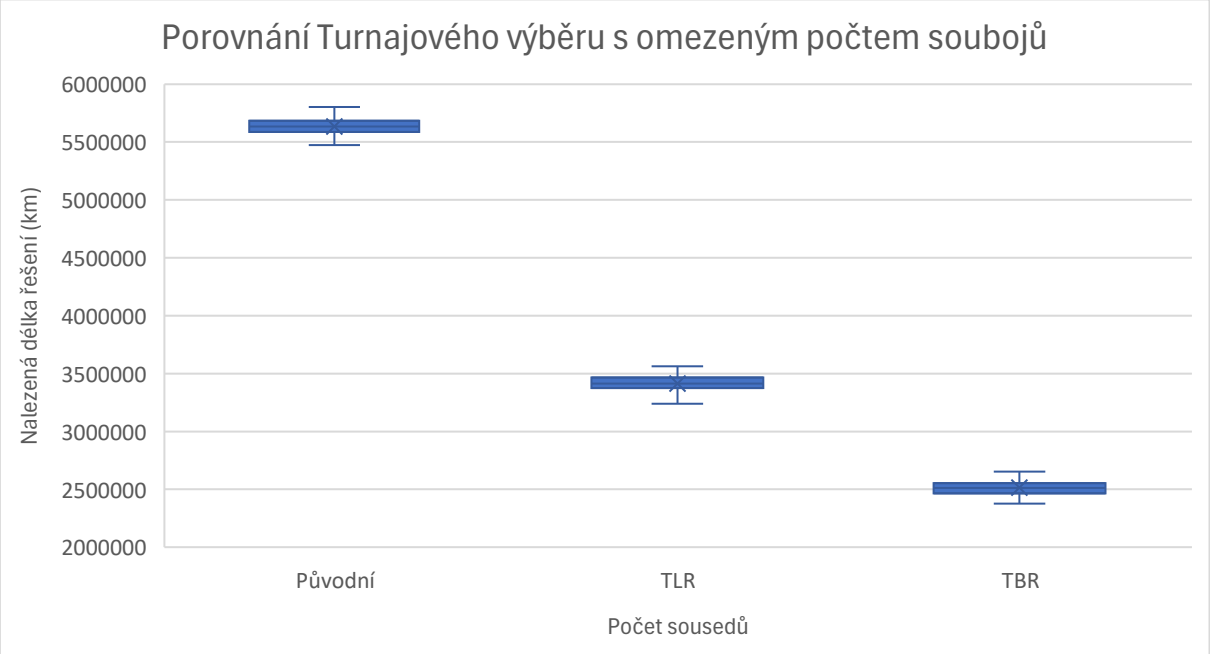
V grafu na obrázku 12 je možné vidět porovnání Turnajové selekce realizující veškeré souboje. Průměrné počáteční řešení je 5 634 838 km. Úspěšnější je varianta výběru nejlepšího řešení, průměrné zlepšení je o 59,09 %, poměrové ruletové kolo přináší zlepšení průměrně o 50,81 %.




*Obrázek 12 – Graf porovnání Turnajového výběru s realizací všech soubojů (autor)*

#### 4.6.2.2 Turnajový výběr s omezeným počtem soubojů

V grafu na obrázku 13 je dostupné porovnání Turnajové selekce realizující omezený počet soubojů. Průměrné počáteční řešení je 5 634 838 km. Úspěšnější je varianta výběru nejlepšího řešení, průměrné zlepšení je o 55,39 %, poměrové ruletové kolo přináší zlepšení průměrně o 39,40 %.




*Obrázek 13 – Graf porovnání Turnajového výběru s omezeným počtem soubojů (autor)*

#### 4.6.2.3 Srovnání obou variant

Při porovnání obou testovaných variant s omezeným a neomezeným počtem soubojů nacházíme zajímavé výsledky.

V tabulce 13 zobrazené níž je možné vidět porovnání obou variant, tedy v případě, že je realizován omezený počet soubojů a v případě, že jsou realizovány všechny. Z výsledků je dobře viditelné, že realizace všech soubojů s výběrem nejlepšího přináší nejlepší výsledky. Z výsledků je zde možné vidět průměrné zlepšení o 59,09 %, nicméně v případě použití omezeného počtu soubojů dochází rovněž k dobrým výsledkům 55,39 %. Ve srovnání je možné pozorovat, že použití ruletového kola s omezeným počtem soubojů přináší nejrychlejší dokončení algoritmu z časového hlediska. Nicméně získané zlepšení je pouhých 39,40 %.

*Tabulka 13 – Porovnání Turnajového výběru*

| Výběr | Počet soubojů | Průměrně nalezené řešení (km) | Průměrné zlepšení (%) | Časové trvání (s) |
|---|---|---|---|---|
| Ruletové kolo | Omezený počet soubojů | 3 414 590,543 | 39,40 | 34,089 |
| Ruletové kolo | Všechny souboje | 2 771 907,671 | 50,81 | 37,942 |
| Výběr nejlepšího | Omezený počet soubojů | 2 513 631,655 | 55,39 | 36,424 |
| Výběr nejlepšího | Všechny souboje | 2 305 035,775 | 59,09 | 38,690 |

### 4.6.3 Vyhodnocení výsledků konfigurací modifikovaného simulovaného žíhání

V jednotlivých podkapitolách byly porovnány získané výsledky z realizovaného testování algoritmu. Při testování byla použitá mutace partial reverse, která byla zvolena z předchozího testování na tradičním simulovaném žíhání.

Z variant Hodnostní selekce byly získány velmi dobré. V případě ruletového kola došlo k zaostávání proti výběru nejlepšího, i přes své rychlejší vykonání ruletového kola je vhodné použít pro další testování zvolit variantu s výběrem nejlepšího řešení.

Srovnání kombinací Turnajového výběru přineslo zajímavější výsledky. Pro testování všech dostupných soubojů byli vygenerováni 3 sousedé, pro omezený počet soubojů bylo vygenerováno 5 sousedů s dvěma souboji. Porovnání různých početních kombinací je dostupné v grafu na obrázku 8.

V obou testovaných variantách přináší nejlepší výsledky použití výběru nejlepšího, přestože ruletové kolo přináší přívětivější výsledky z časové trvání algoritmu. Toto zlepšení není dostatečné, aby byla upřednostněna selekce s ruletovým kolem. I přes slabší výsledky u varianty s omezeným počtem při výběru nejlepšího řešení je vhodné pro další testování. Rychlejší dokončení algoritmu se může projevit v případě testování velkého množství vyhodnocení cost funkce.

V tabulce 14 je možné vidět všechny porovnané výsledky. Dle těchto výsledků je nejvhodnější pro další testování použít selekce s výběrem nejlepšího řešení.

*Tabulka 14 – Porovnání výsledků jednotlivých selekcí*

| Typ selekce | Typ výběru | Zlepšení cesty | Zlepšení řešení (%) | Časové trvání algoritmu (s) |
|---|---|---|---|---|
| Hodnostní výběr | Ruletové kolo | 3 111 132,919 | 44,79 | 37,480 |
| | Nejlepší řešení | 2 301 339,458 | 59,16 | 38,792 |
| Turnajový výběr se všemi souboji | Ruletové kolo | 2 771 907,671 | 50,81 | 37,942 |
| | Nejlepší řešení | 2 305 035,775 | 59,09 | 38,690 |
| Turnajový výběr s omezeným počtem soubojů | Ruletové kolo | 3 414 590,543 | 39,40 | 34,089 |
| | Nejlepší řešení | 2 513 631,655 | 55,39 | 36,424 |

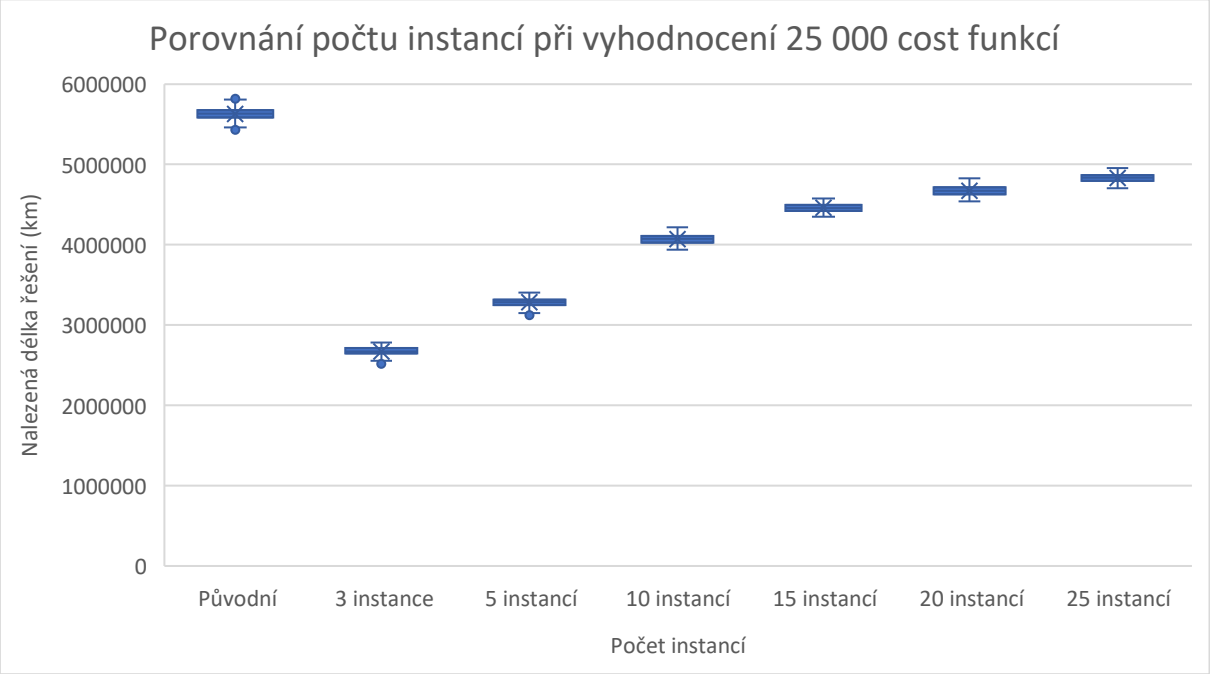
## 4.7 Konfigurace PSA

### 4.7.1 Testování rozdílných počtu paralelizovaných řešení

Při testování první varianty bylo nutné provést, jak budou rozdílné počty paralelizovaných běhů ovlivňovat získané výsledky. Nejen z výkonnostního hlediska, ale i z doby vykonávání algoritmu. Testování bylo provedené na 3, 5, 10, 15, 20, 25 paralelních vykonáváních. Testovaná počáteční řešení byla náhodná i statická. Celkový počet vyhodnocení cost funkcí byl v obou testovaných scénářích 25 tisíc. Testování bylo realizováno na všech třech typech mutace pro zjištění, která bude nejlepší.

V následujícím grafech na obrázku níže jsou dostupné porovnání výsledků jednotlivých počtu vláken při použití počátečního řešení, které bylo průměrně 5 629 556 km. Graf popisuje rozdílné výsledky, které byly nalezené při porovnání s počtem vláken.

V grafu níže na obrázku 14 je možné vidět porovnání rozdílného množství vláken na mutaci partial reverse. Je opět zřejmé, že vyšší počet vyhodnocení cost funkcí s menším počtem instancí je mnohem efektivnější.




*Obrázek 14 – Graf porovnávající rozdílný počet instancí (autor)*

V tabulce 15 zobrazené níže je možné vidět časové trvání algoritmus s jednotlivými počty vláken. S rostoucím počtem vláken dochází k prodlužování časového trvání algoritmu. Zde je nejkratší časové trvání při použití 3 vláken, nicméně rozdíly nejsou příliš zásadní.

*Tabulka 15 – Porovnání časového trvání paralelního simulovaného žíhání s náhodným řešením*

| Počet vláken | Časové trvání algoritmu (s) |
|---|---|
| 3 | 104,964 |
| 5 | 108,766 |
| 10 | 112,006 |
| 15 | 105,108 |
| 20 | 114,689 |
| 25 | 114,728 |

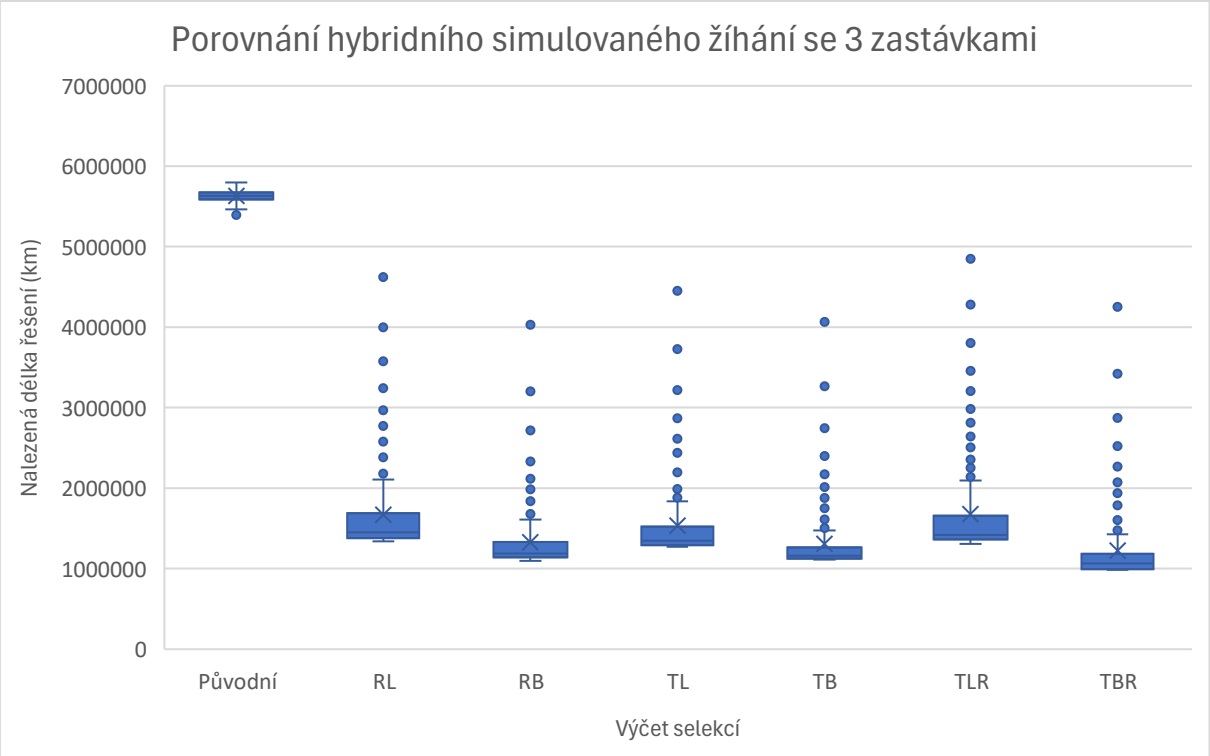
### 4.7.2 Testování rozdílného počtu zastávek

Při testování hybridní varianty s postupným zastavováním a pevně daným počtem optimalizací v každé zastávce bylo nutné provést investigaci, jak se budou lišit výsledky. Testovaný počet zastávek byl 3, 5, 10. Při testování byl stanovený celkový počet na 10800 vyhodnocení cost funkce.

Na základě testování předchozího případu rozdílného počtu vláken bylo zvoleno testování na 3 instancích. Počet generovaných sousedů byl převzat z testování konfigurace modifikovaného simulovaného žíhání v případě Hodnostní selekce a Turnajového výběru s realizací všech soubojů jsou generováni 3 sousedé. Pro Turnajový výběr s omezeným počtem soubojů bylo zvoleno 5 sousedů a 3 provedené souboje.

**3 zastávky**

Z grafu na obrázku 15 je možné vidět výsledky získané v případě třech zastávek. V případě těchto výsledků je možné vidět, že veškeré porovnávané varianty přináší dobré výsledky i v případě kombinací s ruletovým kolem jsou výsledky příznivější.




*Obrázek 15 – Graf porovnání hybridního simulovaného žíhání na 3 zastávkách (autor)*

Některé varianty mají širší rozptyl hodnot, ale i přes velké množství extrémních hodnot, které nakonec ovlivňují nalezený průměr jsou výsledky velmi příznivé. Volba nejefektivnější varianty zde byla poměrně složitější.

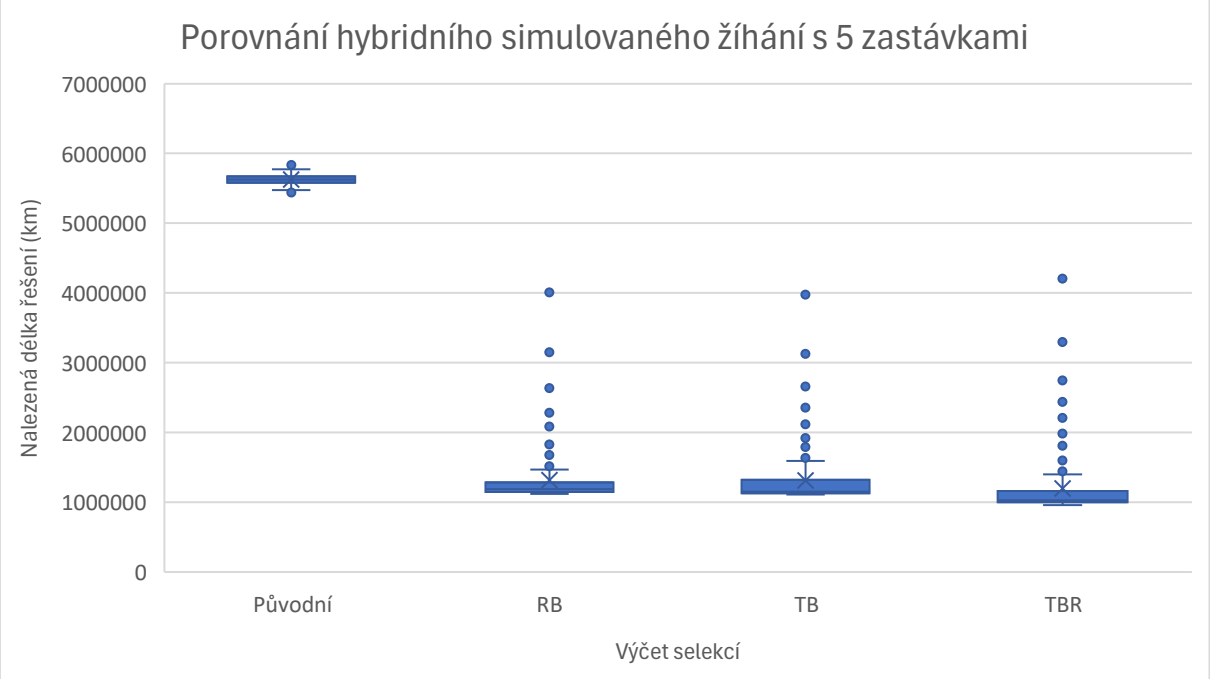
V tabulce 16 níže jsou popsané výsledky průměrných nalezených řešení. Veškeré selekce s výběrem nejlepšího řešení přináší velmi dobré průměrné výsledky. Jsou velmi blízké výkonnostně mezi sebou. Z časového hlediska jsou velmi úspěšné všechny, nicméně toto může být zapříčiněno malým množstvím testovaných vyhodnocení cost funkce. Pro další testování zastávek je vhodné tedy zvolit všechny tři selekce s výběrem nejlepšího řešení.

*Tabulka 16 – Porovnání výsledků hybridního simulovaného žíhání při 3 zastávkách*

| Typ selekce | Typ výběru | Zlepšení cesty (km) | Zlepšení řešení (%) | Časové trvání algoritmu (s) |
|---|---|---|---|---|
| Hodnostní výběr | Ruletové kolo | 1 667 370,039 | 70,385 | 25,629 |
| | Nejlepší řešení | 1 326 484,653 | 76,44 | 26,438 |
| Turnajový výběr se všemi souboji | Ruletové kolo | 1 534 930,083 | 72,74 | 26,368 |
| | Nejlepší řešení | 1 307 434,094 | 76,78 | 26,835 |
| Turnajový výběr s omezeným počtem soubojů | Ruletové kolo | 1 675 697,959 | 70,24 | 26,593 |
| | Nejlepší řešení | 1 221 527,002 | 78,30 | 25,669 |

**5 zastávek**

Z předchozího testování 3 zastávek byly pro další testování zvoleny všechny selekce, které následně provádí výběr nejlepšího řešení.




*Obrázek 16 – Graf porovnání hybridního simulovaného žíhání na 5 zastávkách (autor)*

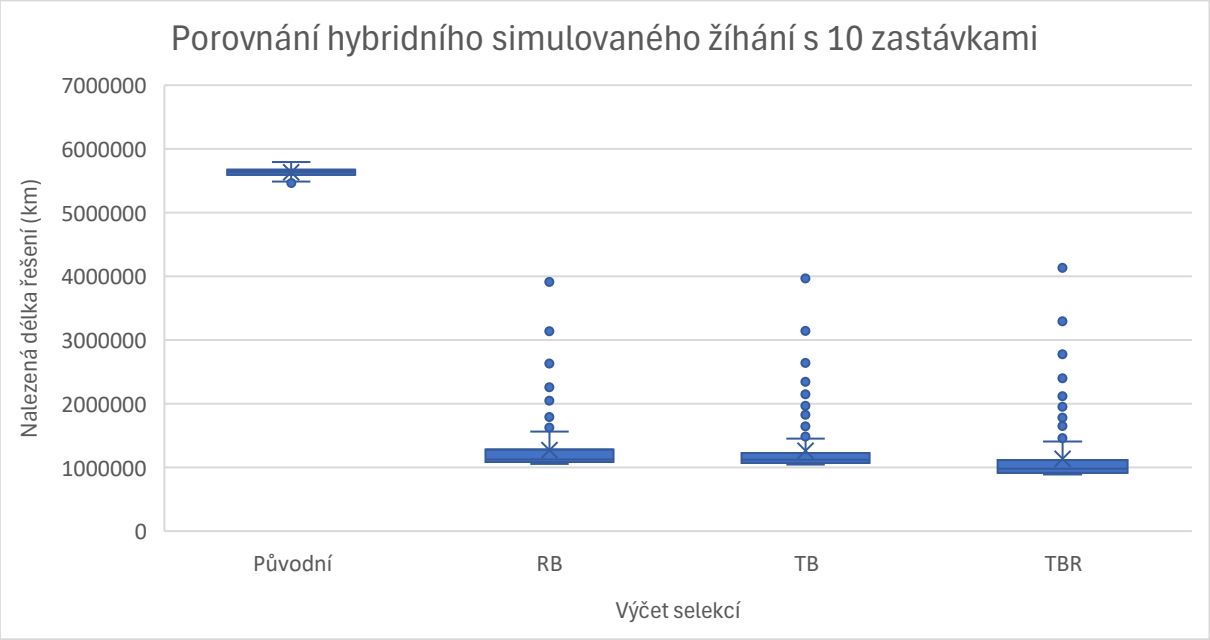
V grafu na obrázku 16 je dobře viditelné, že s vyšším počtem zastávek dochází k snížení nalezeného průměrného zlepšení. Je to způsobené tím, že v případě vyššího množství zastávek dochází k snížení počtu vyhodnocení cost funkce v jednotlivých zastávkách. Získané výsledky i přes tento fakt jsou velmi příznivě. Jednotlivé průměrné zlepšení jsou zobrazená v tabulce 17.

*Tabulka 17 –Porovnání výsledků hybridního simulovaného žíhání na 5 zastávkách*

| Typ selekce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|
| Hodnostní výběr | 1 318 428,318 | 76,57 | 26,707 |
| Turnajový výběr se všemi souboji | 1 310 540,441 | 76,71 | 26,453 |
| Turnajový výběr s omezeným počtem soubojů | 1 197 302,638 | 78,73 | 26,075 |

**10 zastávek**

Testování 10 zastávek slouží pouze k ověření tvrzení, že v případě vyššího počtu zastávek s menším počtem vyhodnocení cost funkce v jednotlivých zastávkách jsou nalezeny horší výsledky.




*Obrázek 17 – Graf porovnání hybridního simulovaného žíhání na 10 zastávkách (autor)*

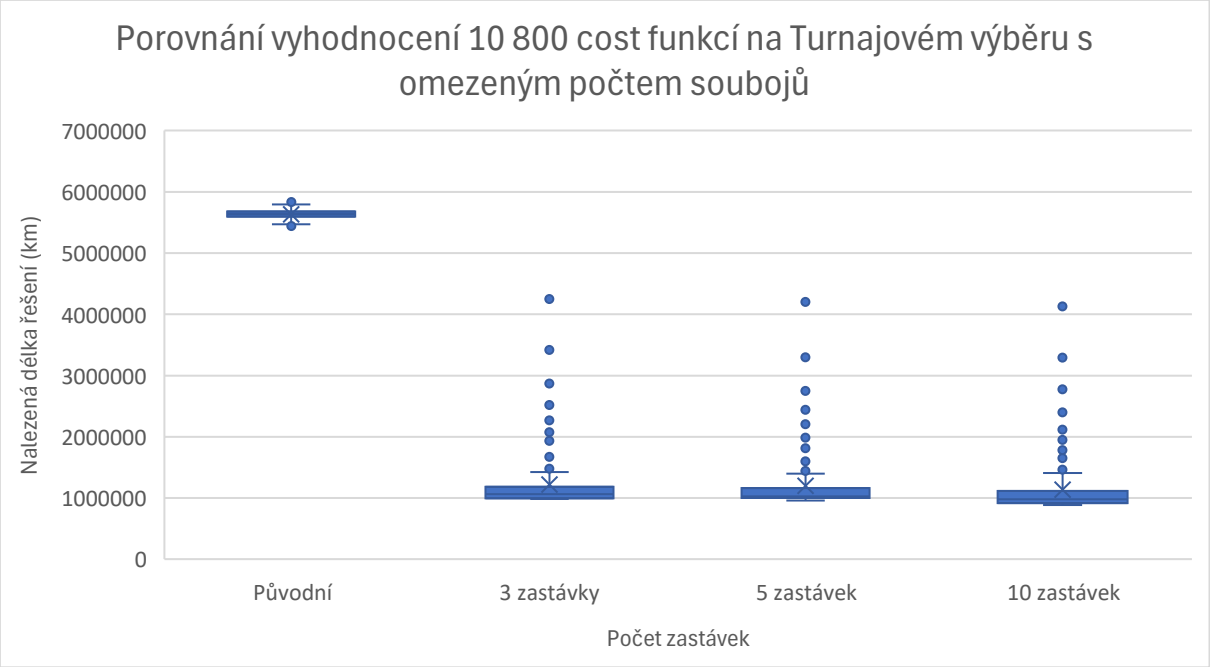
Z dostupného grafu na obrázku 17 je možné vidět, že toto tvrzení je potvrzující. S vyšším počtem zastávek v případě nižšího počtu vyhodnocení fitness funkcí dochází k nalézání horších řešení.

### 4.7.3 Testování vyhodnocení počtu cost funkcí

Pro otestování rozdílných počtů vyhodnocení cost funkcí byla zvolena selekce s omezeným počtem soubojů, protože přinášela velmi dobré výsledky jak z hlediska kvality řešení tak i časového trvání algoritmu. Testování bylo realizováno na vyhodnocení specifického počtu fitness funkcí 10 800 a 108 000. Porovnávaný počet vyhodnocení cost funkcí je poměrně vysoký díky provádění třech paralelních běhů a počtu generovaných sousedů. V následujících grafech je možné vidět, jakým trendem se výdavaly průměrné změny kvality řešení v počtech vyhodnocení cost funkce, s rozdílným počtem zastávek. Počet zastávek pro porovnání zde byl zvolen 3, 5 a 10.

**10800 vyhodnocení cost funkce**

V grafu na obrázku 18 je možné vidět srovnání testovaných možností počtů zastávek. Nejpřívětivější výsledky přináší použití menšího počtu zastávek s vyšším počtem kroků. V případě 3 zastávek je možné vidět, že došlo k nalezení několika extrému, které i přesto z porovnávaných variant přináší nejpřívětivější výsledky. Z hlediska časového to je ovšem o něco méně přívětivé. V tabulce 18 je možné vidět, že algoritmus je mnohonásobně horší z časového hlediska horší, ale z hlediska nalezené kvality řešení je velmi efektivní.




*Obrázek 18 – Graf porovnání vyhodnocení 10 800 cost funkcí (autor)*

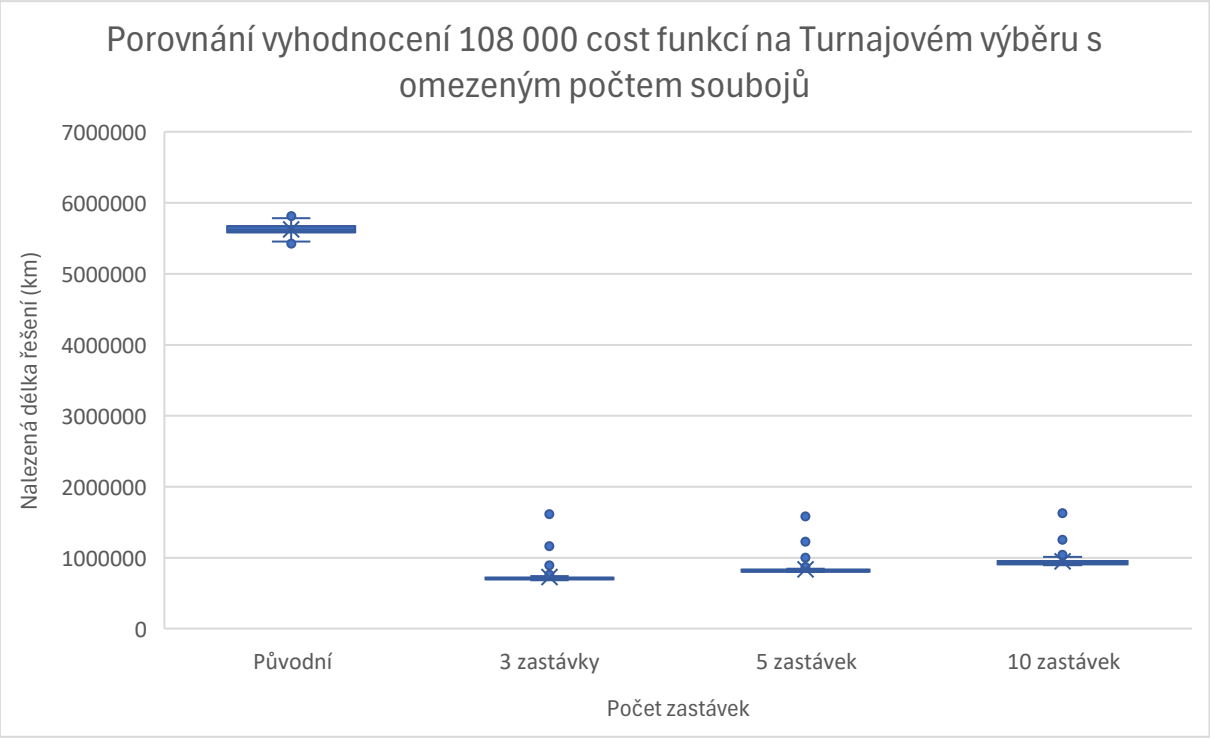
*Tabulka 18 – Porovnání získaných výsledků z vyhodnocení 10 800 cost funkcí*

| Počet zastávek | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|
| 3 | 1221527,002 | 78,30 | 25669.0 |
| 5 | 1197302,638 | 78,73 | 26075.0 |
| 10 | 1135956,432 | 79,83 | 26835.0 |

**108000 vyhodnocení cost funkce**

V grafu na obrázku 19 je dostupné srovnání testovaných možností počtů zastávek. Zde to není již tak jednoznačné rozhodnutí.

V grafu je možné vidět, že s vyšším počtem zastávek i přes menší počet kroků v jednotlivých zastávkách jsou výsledky velmi podobné. Jednoznačně lze potvrdit, že v případě veškerých variant je nacházené mnohem menší množství extrémů, které mohou nepřívětivě ovlivnit nalezené řešení. Ve všech případech dochází k velkému zúžení spektra nalezených hodnot. A z hlediska nejmenšího nalezeného řešení vyhrává použití menšího množství zastávek. Dále z hlediska časového je nejlepší mnohonásobně lepší použití většího množství zastávek před menším. Tyto výsledky je možné vidět v tabulce 19.




*Obrázek 19 – Graf porovnání vyhodnocení 108 000 cost funkcí (autor)*

*Tabulka 19 – Porovnání získaných výsledků z vyhodnocení 108 000 cost funkcí*

| Počet zastávek | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|
| 3 | 728 888,0145 | 87,03 | 25,730 |
| 5 | 836 475,4919 | 85,14 | 25,056 |
| 10 | 946 858,4633 | 83,18 | 26,014 |

### 4.7.4 Vyhodnocení výsledků konfigurací PSA

Při závěrečném vyhodnocení konfigurací PSA je možné říct, že při porovnání paralelizovaného tradičního simulovaného žíhání je vhodné použít menší počet instancí, ale s vyšším počtem vyhodnocení cost funkce.

Testování konfigurace hybridního simulovaného žíhání bylo složitější. Bylo zde nutné otestovat několik faktorů, které ovlivňují výsledky celého algoritmu. Prvním krokem bylo porovnání počtu paralelizovaných realizací, které dle předchozí znalosti z tradiční verze byly převzaty.

Dalším zásadním faktorem bylo zvolit vhodný počet realizovaných zastávek. Při testování došlo k otestování všech implementovaných selekcí na 3 zastávkách. Dle získaných výsledků byly zvolené varianty selekcí, které přináší nejpřívětivější výsledky. Z tohoto testování bylo zjištěno že výběr řešení pomocí poměrového ruletového kola je velmi neefektivní, přináší mnohem slabší výsledky než výběr nejlepšího řešení. Zvolené selekce pro další testování byly všechny z možného výběru. Výsledky byly velmi podobné ve všech případech, jak je možné vidět v grafu na obrázku 15. Při testování vyššího počtu zastávek na vybraných selekcích se pouze při 10 zastávkách ukázal Turnajový výběr s omezeným počtem realizovaných soubojů jako horší. Nicméně z časového hlediska byl nejefektivnější z testovaných variant.

Z tohoto důvodu byl zvolen dále pro testování vhodného počtu vyhodnocení cost funkcí. Toto testování proběhlo na 10 800 a 108 000 celkových vyhodnoceních. Zde se projevilo, že mnohem efektivnější je použít menší množství zastávek s vyšším počtem vyhodnocení sousedů v jednotlivých zastávkách než vyšší množství s malým počtem vyhodnocení cost funkce.

Při testování vyššího celkového počtu vyhodnocení cost funkce, byly výsledky velmi rozdílné. Problém zde byl výskyt vyššího množství extrémních hodnot, které výsledky zkreslily. Zde se ukázal tedy opak, že v případě vyššího celkového počtu kroků je efektivnější použít vyšší množství zastávek. Výsledky jsou sice více rozptýlené, jak je možné vidět v grafu na obrázku 19. v podkapitole 4.7.3, nicméně je pravděpodobnější, že tato varianta může lépe vyváznout z lokálního optima.

# 5 VYHODNOCENÍ VÝSLEDKŮ

Po otestování všech konfigurací bylo zvoleno, jaké výsledné varianty algoritmů se budou dále porovnávat vzájemně mezi sebou. Pro porovnání bude samozřejmě testováno tradiční simulované žíhání, které bude sloužit jako pomyslný mezník od kterého se budou moci porovnat zvolené varianty. Z testování byl počet generovaných sousedů zvolený na tři. Veškeré algoritmy budou používat mutaci partial reverse, která přinesla nejpřívětivější výsledky.

Vybraná varianta za modifikované simulované žíhání je selekce pomocí Turnajového výběru s výběrem nejlepšího řešení. Pro hybridní paralelní simulované žíhání byla zvolena selekce pomocí Hodnostního výběru s výběrem nejlepšího řešení a Turnajového výběru s omezeným počtem realizovaných soubojů.

Samotné testování bylo realizováno s několika různými konfiguracemi vstupních parametrů.

Testování výkonnosti jednotlivých algoritmů bylo realizování na stejném počátečním datovém souboru.

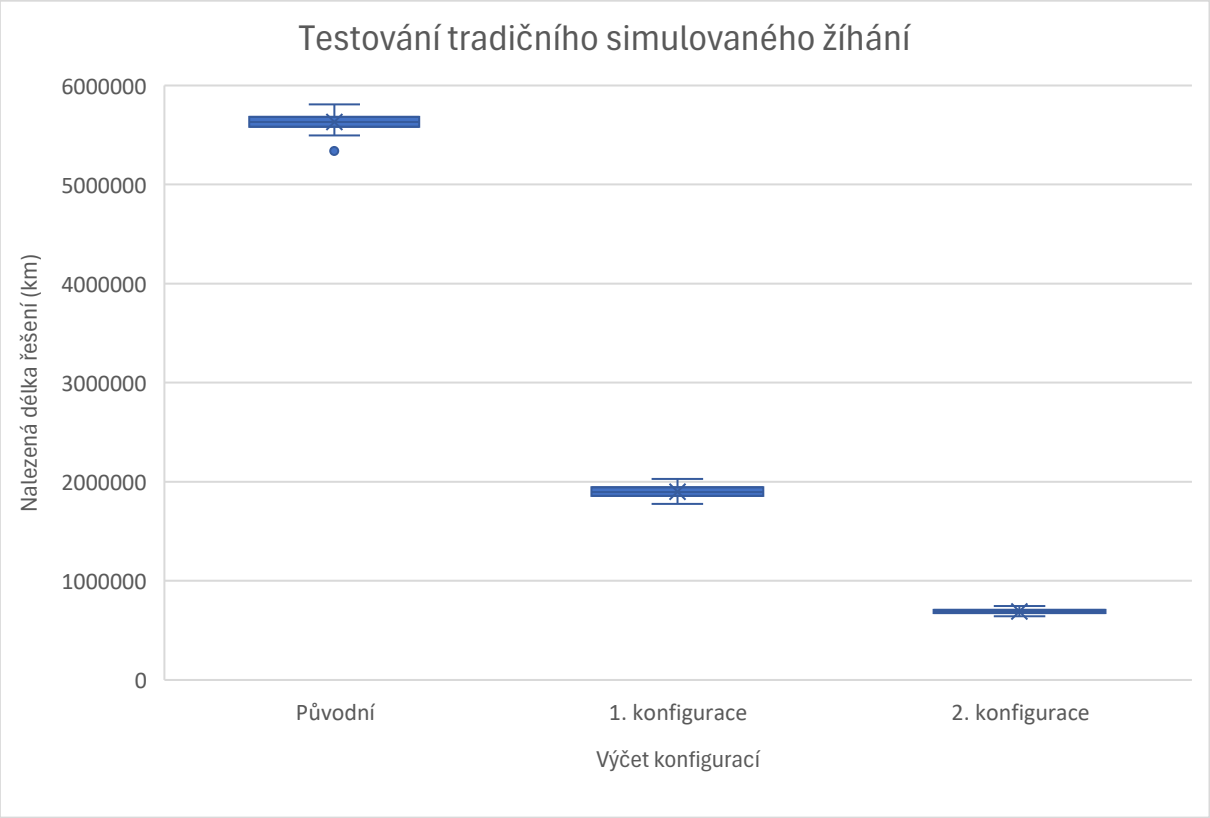
*Tabulka 20 – Konfigurace pro společné porovnávání algoritmů*

| Parametr | Konfigurace 1 | Konfigurace 2 |
|---|---|---|
| Počáteční teplota | 1000 | |
| Ochlazovací hodnota | 0.999 | |
| Počet vyhodnocení cost funkce | 20 250 | 101 250 |
| Počet testujících běhů | 100 | |
| Velikost datového souboru | 1979 | |
| Počet zastávek v PSA | 3 | |
| Počet generovaných sousedů | 3 | |
| Počet generovaných sousedů v případě Turnajového výběru s omezeným počtem soubojů | 5 | |
| Počet soubojů v případě Turnajového výběru s omezeným počtem soubojů | 3 | |

## 5.1 Vyhodnocení tradičního simulovaného žíhání

Testování simulovaného žíhání bylo provedené na dvou konfiguracích. Počáteční parametry byly stejné rozlišoval se pouze celkový počet kroků.

V rámci testování tradičního simulovaného žíhání byly porovnány dvě různé konfigurace. Porovnání konfigurací je dostupné v grafu na obrázku 20. Zde jsou znázorněné nalezené délky řešení se zlepšením proti počátečnímu řešení. Průměrné počáteční řešení bylo 5 630 525 km. V grafu je možné vidět, že obě varianty vedly k výraznému zlepšení nalezených řešení proti počátečnímu řešení. Obě konfigurace generují velmi ustálená řešení bez extrémních hodnot.




*Obrázek 20 – Graf porovnaných výsledků tradičního simulovaného žíhání (autor)*

Průměrné zlepšení v 1. konfiguraci testující 20 250 vyhodnocení cost funkce přináší průměrné zlepšení o 66,25 % s časovým trváním 82 sekund. 2. konfigurace přináší při 101 250 vyhodnoceních cost funkce průměrné zlepšení o 87,74 % s časovým trváním 375 sekund.

*Tabulka 21 – Porovnání výsledků konfigurací tradičního simulovaného žíhání*

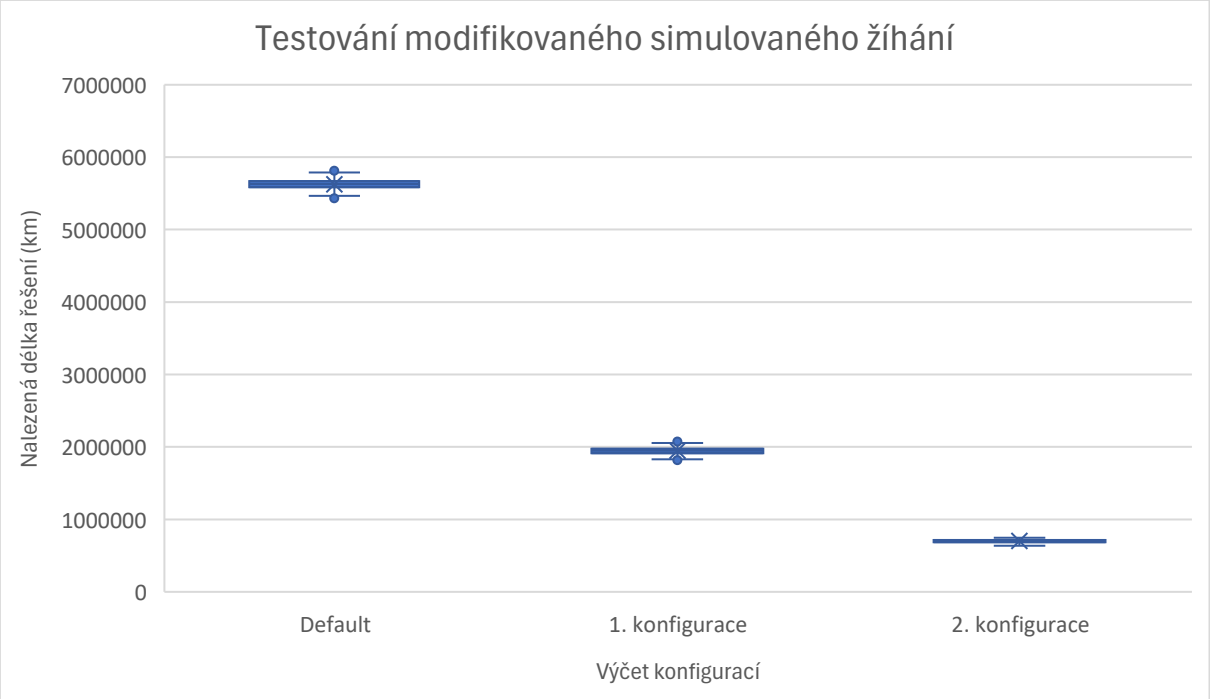
| Konfigurace | Celkový počet vyhodnocení cost funkce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|---|
| 1 | 20 250 | 1 897 102,29 | 66,25 | 82,355 |
| 2 | 101 250 | 691 228,7564 | 87,74 | 375,081 |

Při zhodnocení těchto dvou konfigurací je z hlediska výkonnosti lepší zvolit vyšší počet vyhodnocení cost funkcí. Tato varianta sebou ale může nést problém vyššího časového trvání, které je možné vidět v tabulce 21. Nicméně i v případě nižšího počtu kroků je algoritmus schopný nalézt výrazně lepší řešení oproti původnímu v krátkém časovém trvání.

## 5.2 Vyhodnocení modifikovaného simulovaného žíhání

Testování modifikovaného simulovaného žíhání bylo provedené na dvou konfiguracích. Počáteční parametry byly stejné rozlišoval se pouze počet celkového vyhodnocení cost funkce. Zde došlo propočítání nutného snížení počtu kroků u jednotlivých konfigurací. Z důvodu generování vícero sousední řešeních v tomto případě 3. 1. konfigurace testuje 6 750 kroků v každém sousedovi, celkový počet vyhodnocení cost funkce je 20 250. 2. konfigurace testuje 33 750 kroků v každém sousedovi, celkový počet vyhodnocení cost funkce je 101 250.

Porovnání konfigurací je dostupné v grafu na obrázku 21. Zde jsou znázorněné nalezené délky řešení se zlepšením proti počátečnímu řešení. Průměrné počáteční řešení bylo 5 625 964 km. Z grafu je možné vidět, že obě varianty vedly k výraznému zlepšení nalezených řešení proti počátečnímu řešení. Obě konfigurace generují velmi ustálená řešení. V 1. konfiguraci došlo k nalezení dvou extrému, které by však nemělo výsledky příliš zkreslit, v 2. konfiguraci není výskyt žádných extrémních hodnot.




*Obrázek 21 – Graf porovnaných výsledků modifikovaného simulovaného žíhání (autor)*

*Tabulka 22 – Porovnání výsledků konfigurací modifikovaného simulovaného žíhání*

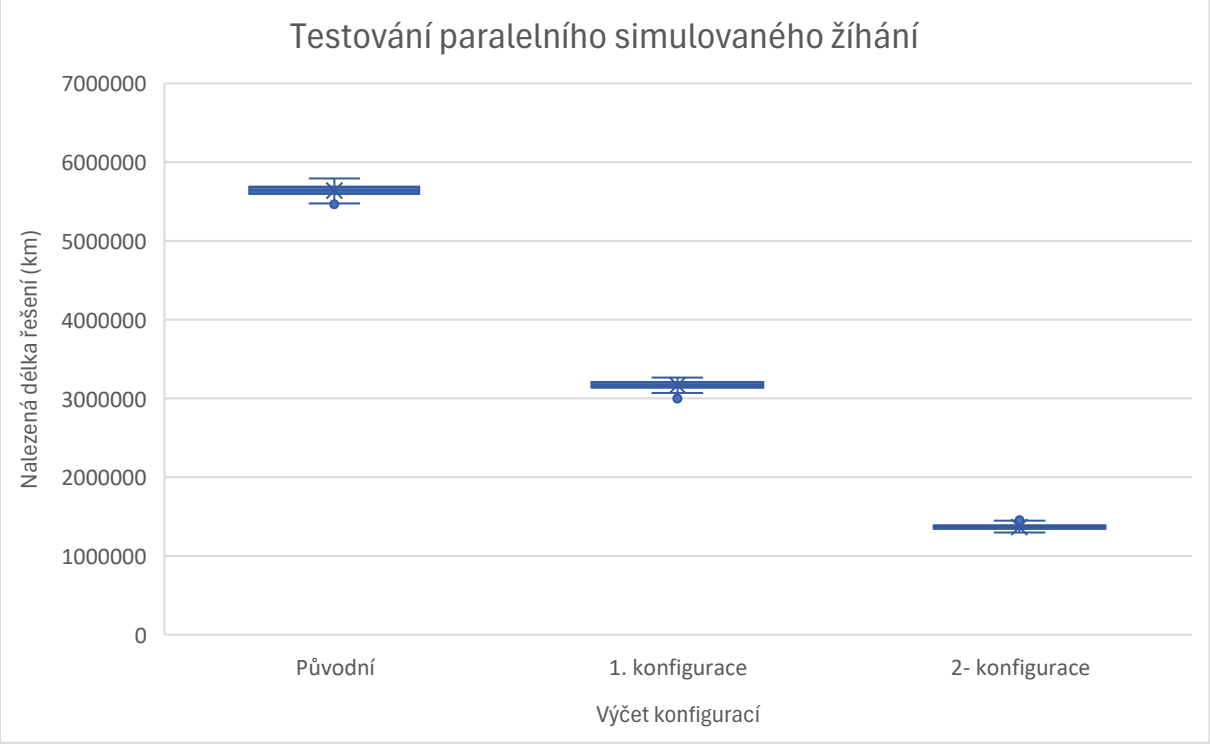
| Konfigurace | Celkový počet vyhodnocení cost funkce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|---|
| 1 | 20 250 | 1 946 565,541 | 65,39 | 57,496 |
| 2 | 101 250 | 701 029,0622 | 87,54 | 258,006 |

Při zhodnocení těchto dvou konfigurací je z výkonnostního hlediska lepší zvolit vyšší počet vyhodnocení cost funkcí. Tato varianta sebou ale může nést problém vyššího časového trvání, které je možné vidět v tabulce 22. Nicméně i v případě nižšího počtu je algoritmus schopný nalézt výrazně lepší řešení v krátkém časovém trvání.

## 5.3 Vyhodnocení paralelního simulovaného žíhání

Testování paralelního simulovaného žíhání bylo provedené na dvou konfiguracích. Počáteční parametry byly stejné rozlišoval se pouze počet celkového vyhodnocení cost funkce. Zde došlo propočítání nutného snížení vyhodnocení cost funkce u jednotlivých konfigurací. Z důvodu paralelizace běhu řešení, v tomto případě 3 instancí. 1. konfigurace 1 testuje 6 750 kroků v každé paralelní instanci, celkové vyhodnocení cost funkce je 20 250. 2. konfigurace testuje 33 750 kroků v každé paralelní instanci, celkový počet vyhodnocení cost funkce je 101 250.

Porovnání konfigurací je dostupné v grafu na obrázku 22. Zde jsou znázorněné nalezené délky řešení se zlepšením proti počátečnímu řešení. Průměrné počáteční řešení bylo 5 638 579 km. Z grafu je možné vidět, že obě varianty vedly k výraznému zlepšení nalezených řešení proti počátečnímu řešení. Obě konfigurace generují velmi ustálená řešení. S minimálním výskytem počtem extrémních hodnot.




*Obrázek 22 – Graf porovnaných výsledků paralelního simulovaného žíhání (autor)*

*Tabulka 23 – Porovnání výsledků konfigurací paralelního simulovaného žíhání*

| Konfigurace | Celkový počet vyhodnocení cost funkce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|---|
| 1 | 20 250 | 3 167 906,682 | 43,82 | 74,363 |
| 2 | 101 250 | 1 367 951,799 | 75,74 | 345,819 |

Při zhodnocení těchto dvou konfigurací je z výkonnostního hlediska lepší zvolit vyšší počet vyhodnocení cost funkcí. Tato varianta sebou ale může nést problém vyššího časového trvání, které je možné vidět v tabulce 23. Nicméně i v případě nižšího počtu je algoritmus schopný nalézt výrazně lepší řešení v krátkém časovém trvání.

## 5.4 Vyhodnocení hybridního paralelního simulovaného žíhání

Testování hybridního paralelního simulovaného žíhání bylo provedené ve dvou variantách. Se selekcí pomocí Ranking selection s výběrem nejlepšího řešení a Tournament selection s omezeným počtem realizovaných soubojů s výběrem nejlepšího.

Obě varianty byly testované na dvou konfiguracích.

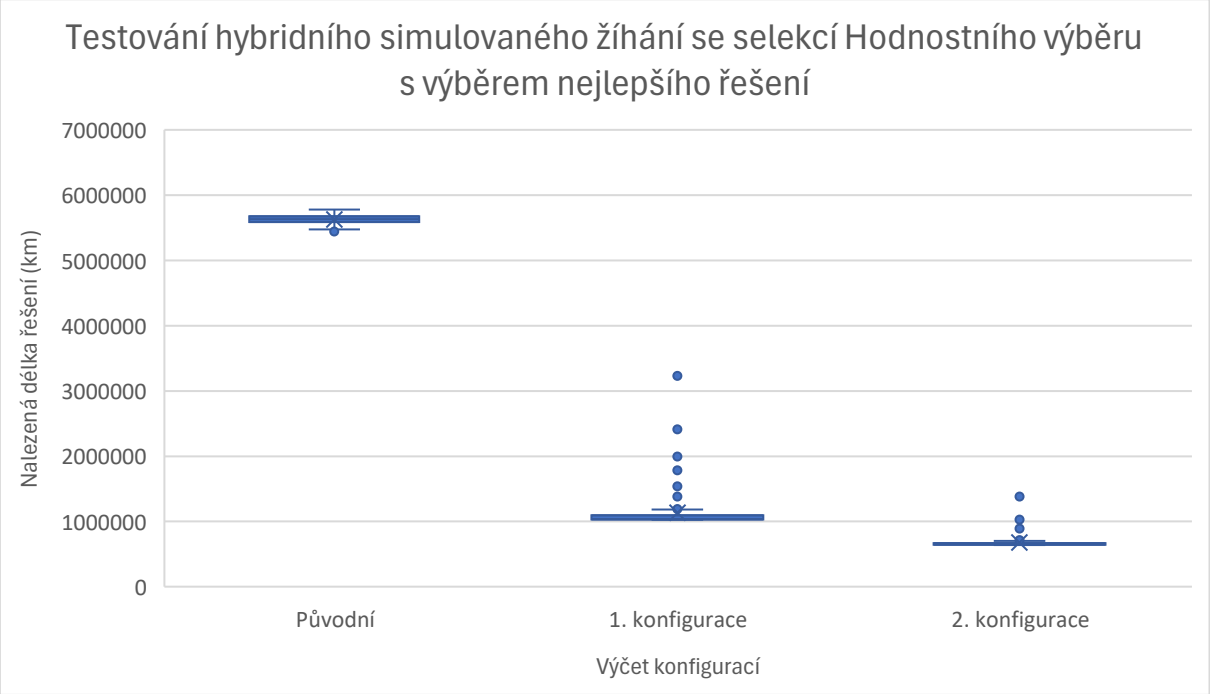
V počátečních parametrech došlo k velké shodě, nicméně bylo zde rozdílné generování počtu sousedů. Ve variantě Ranking selection byli generovaní 3 sousedé, v Tournament selection bylo generováno 5 sousedů. Tyto rozdílné počty generovaných sousedů ovlivnily počet vyhodnocení cost funkce v rámci jedné zastávky.

V 1. konfiguraci byl počet vyhodnocení počtu kroků v rámci jedné zastávky ve variantě Ranking selection 750 kroků, ve variantě Tournament selection 450 kroků, celkový počet vyhodnocení cost funkce bylo 20 250. V 2. konfiguraci bylo v Ranking selection došlo k vyhodnocení 3 750 kroků a v Tournament selection došlo k vyhodnocení 2 250 kroků v rámci jedné zastávky, celkový počet vyhodnocení cost funkce bylo 101 250.

### 5.4.1 Ranking selection s výběrem nejlepšího řešení

Porovnání konfigurací je dostupné v grafu na obrázku 23. Zde jsou dostupné nalezené délky řešení se zlepšením proti počátečnímu řešení. Průměrná délka počátečního řešení byla 5 630 126 km a sloužila jako základ pro hodnocení efektivity jednotlivých konfigurací.

Z grafu je možné vidět, že obě varianty vedly k výraznému zlepšení nalezených řešení proti počátečnímu řešení. V 1. konfiguraci je možné vidět větší rozptyl hodnot z důvodu výskytu většího množství extrémních hodnot. V 2. konfiguraci je výskyt extrémních hodnot menší především z důvodu většího množství vyhodnocení cost funkce a tím tedy i většímu ustálení hodnot.




*Obrázek 23 – Graf porovnaných výsledků hybridního paralelního simulovaného žíhání s Ranking selection s výběrem nejlepšího řešení (autor)*

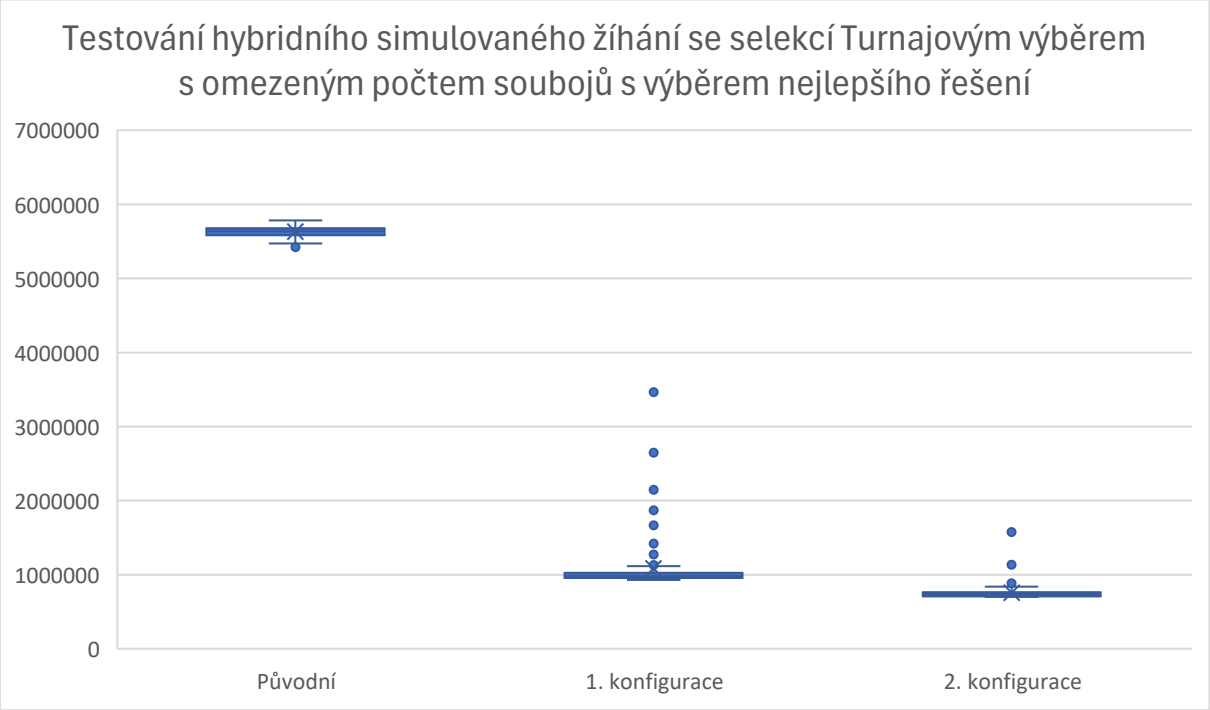
*Tabulka 24 – Porovnání výsledků hybridního paralelního simulovaného žíhání s Ranking selection s výběrem nejlepšího řešení*

| Konfigurace | Celkový počet vyhodnocení cost funkce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|---|
| 1 | 20 250 | 1 133 761,838 | 79,87 | 56,352 |
| 2 | 101 250 | 677 583,979 | 87,96 | 251,436 |

Při zhodnocení těchto dvou konfigurací je z hlediska výkonnosti lepší zvolit vyšší počet vyhodnocení cost funkcí. Jak je možné vidět v grafu na obrázku 23 tak počet extrémních hodnot se velmi ustálil při generování velkého množství cost funkcí Tato varianta sebou, ale může nést problém vyššího časového trvání, které je možné vidět v tabulce 24. Nicméně i v případě nižšího počtu je algoritmus schopný nalézt výrazně lepší řešení v krátkém časovém trvání.

### 5.4.2 Tournament selection s omezeným počtem soubojů s výběrem nejlepšího řešení

Testování je dostupné v grafu na obrázku 24 a tabulce 25, ukazuje výrazné zlepšení oproti původnímu řešení. Průměrné výchozí řešení bylo o délce 5 630 126 km. V grafu je možné vidět, že obě varianty vedly k výraznému zlepšení nalezených řešení proti počátečnímu řešení. V 1. konfiguraci je možné vidět velké množství extrémních hodnot, které způsobily vyšší rozptyl. V 2. konfiguraci je ten výskyt extrémních hodnot mnohem menší a díky tomu je více stabilní.




*Obrázek 24 – Graf porovnaných výsledků hybridního paralelního simulovaného žíhání s Tournament selection s omezeným počtem soubojů s výběrem nejlepšího řešení (autor)*

*Tabulka 25 – Porovnání výsledků hybridního paralelního simulovaného žíhání s Tournament selection s omezeným počtem soubojů s výběrem nejlepšího řešení*

| Konfigurace | Celkový počet vyhodnocení cost funkce | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|---|
| 1 | 20 250 | 1 082 848,504 | 80,75 | 48,211 |
| 2 | 101 250 | 754 782,0676 | 86,60 | 228,944 |

Při zhodnocení těchto dvou konfigurací jsou z hlediska výkonnostni si obě konfigurace jsou vzájemně více podobné. Při pětinásobném zvýšení počtu vyhodnocení cost funkce došlo ke zlepšení pouze o necelých 6 %. Dalším ovlivňujícím faktorem bylo nalezení vyššího počtu extrémních hodnot. Výhoda vyššího počtu vyhodnocení cost funkce je zde v získání vyšší stability výsledku, za cenu delšího časového trvání, které je dostupné v tabulce 25. Použití menší počtu vyhodnocení cost funkce je velmi užitečné, protože přináší skvělé výsledky za velmi krátký čas.

## 5.5 Celkové vyhodnocení

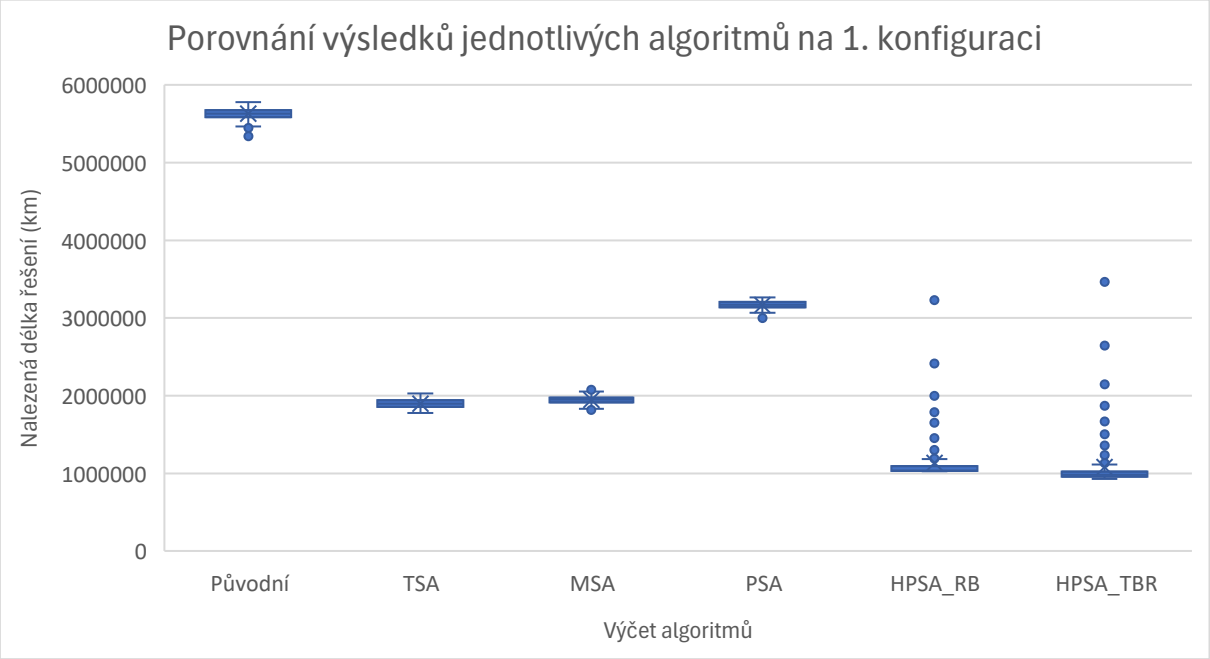
Celkové vyhodnocení bylo realizováno na 2 konfiguracích. Konfigurace měly stejné počáteční nastavení jejich rozdíl byl v množství celkového počtu vyhodnocení cost funkce. Rozdíl mezi těmito počty byl pětinásobný tedy srovnání bylo provedeno v rámci porovnání rychlosti a schopnosti nalézat vhodná řešení s malým počtem celkového vyhodnocení cost funkce. Pozorovaným faktorem tedy byly nalezené výsledky z hlediska kvality zlepšení vůči počáteční náhodné vytvořené trase a druhým výpočetní náročnost.

Druhá konfigurace následně testovala velké množství vyhodnocení cost funkce s následným porovnáním, jak se budou jednotlivé výsledky lišit v jednotlivých variantách, a jak se budou lišit v rámci všeobecného porovnání mezi všemi testovanými.

Porovnání bylo realizováno na pěti variantách simulovaného žíhání konkrétně

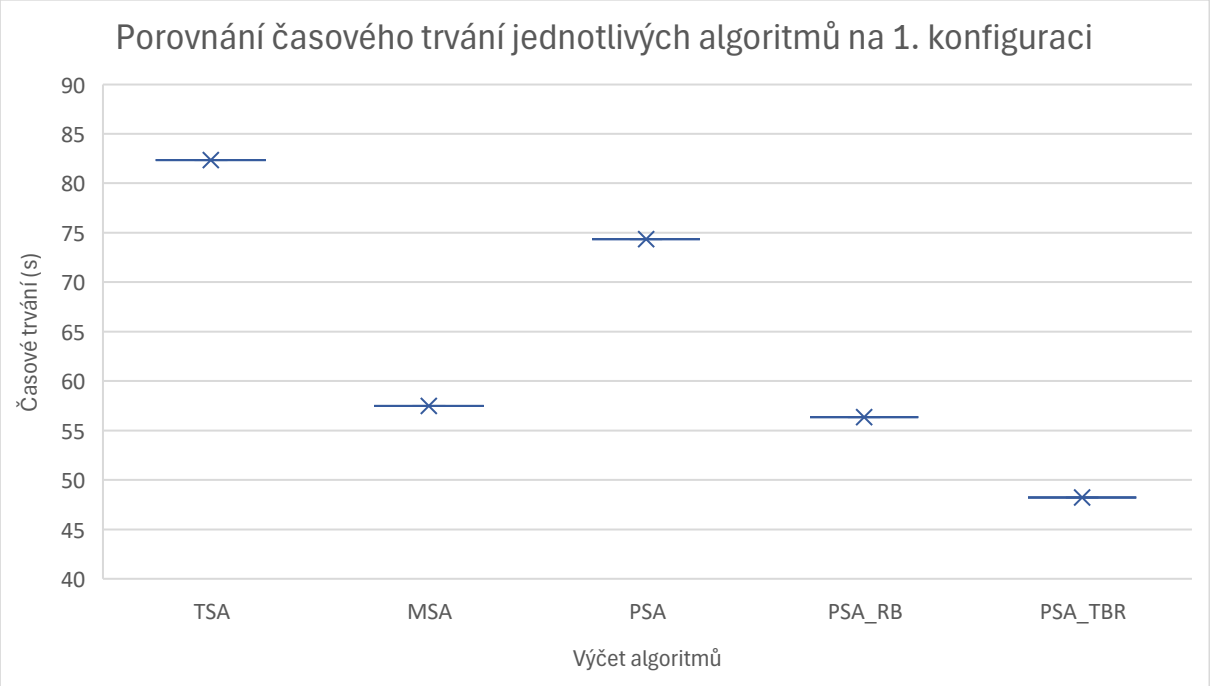
1. Tradiční simulované žíhání (TSA)
2. Modifikované simulované žíhání (MSA)
3. Paralelní simulované žíhání (PSA)
4. Hybridní paralelní simulované žíhání
    a. S Ranking selection s výběrem nejlepšího řešení (HPSA_RB)
    b. S Tournament selection s omezeným počtem soubojů s výběrem nejlepšího řešení (HPSA_TBR)

V rámci celkového vyhodnocení první konfigurace bylo vyhodnocováno 20 250 cost funkcí. Graf na obrázku 25 ukazuje, že všechny varianty výrazně překonávají počáteční řešení. Nicméně každá varianta je s rozdílnou efektivitou, stabilitou i výpočetní náročností. Nejvýkonnější z efektivity nalezeného řešení jsou jednoznačně obě varianty Hybridního paralelního simulovaného žíhání. V případě Tournament selection dochází k nalezení průměrného zlepšení o 80,75 % a s malým rozdílem je Ranking selection, která nachází průměrné zlepšení o 79,87 %. Paralelní simulované žíhání naléza průměrné zlepšení o 43,81 %. Zbylé dvě varianty tedy modifikované simulované žíhání a tradiční simulovaného žíhání přináší hodnoty ve velmi podobném rozmezí, liší se pouze v rámci jednotek %. Nalezená průměrná zlepšení těchto variant jsou v rozmezí 65,39 – 66,25 %.




*Obrázek 25 – Graf nalezených výsledků algoritmů u 1. konfigurace (autor)*

Graf na obrázku 26 popisuje jednotlivé trvání porovnávaných variant simulovaného žíhání. Zde většina variant přináší zhruba stejnou výpočetní náročnost. Nicméně varianta paralelního simulovaného žíhání je pomalejší. Zrychlení je možné v případě zavedení vláken, které by potenciálně měly zvýšit výpočetní rychlost celého běhu algoritmu. Při porovnání jednotlivých variant, ale opět nejrychlejší jsou hybridní varianty, obzvlášť pak varianta s Tournament selection, která má časové trvání 48 sekund. Druhá varianta Ranking selection již není tolik výkonná z časového hlediska, je velmi podobná variantě modifikovaného simulovaného žíhání. Časové trvání Ranking selection je 56 sekund a modifikovaného simulovaného žíhání 57 sekund rozdíl je tedy minimální. Zbylé dvě varianty jsou už pomalejší tradiční simulované žíhání mělo trvání 82 sekund a paralelní simulované žíhání 74 sekund.




*Obrázek 26 – Graf časového trvání variant u 1. konfigurace (autor)*

Při celkovém zhodnocení 1. konfigurace je nejvýkonnější Hybridní paralelní simulované žíhání s Tournament selection variantou. Z důvodu kvality nalezeného řešení, tak i z důvodu časového trvání, které je nejvíce vyvážené v této variantě.

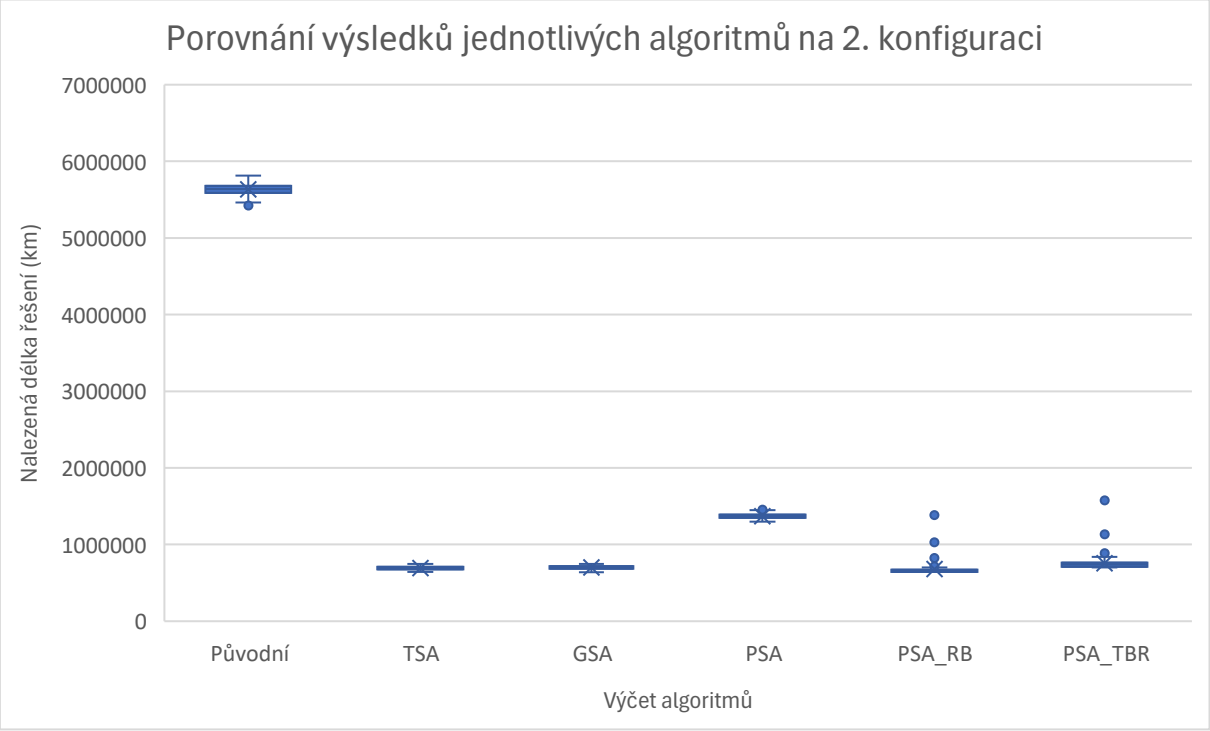
*Tabulka 26 – Shrnutí získaných výsledků algoritmů z 1. konfigurace*

| Algoritmus | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|
| TSA | 1 897 102,29 | 66,25 | 82,355 |
| MSA | 1 946 565,541 | 65,39 | 57,496 |
| PSA | 3 167 905,682 | 43,81 | 74,363 |
| HPSA_RB | 1 133 761,838 | 79,87 | 56,352 |
| HPSA_TBR | 1 082 848,504 | 80,75 | 48,211 |

V rámci porovnání druhé konfigurace bylo celkově vyhodnoceno 101 250 cost funkcí. V grafu na obrázku 27. je možné vidět, že veškeré varianty zásadně optimalizovaly nalezené výsledky proti počátečnímu řešení.

Z grafu je viditelné, že paralelní simulované žíhání přináší horší výsledky. Zbylé varianty nachází podobné výsledky pouze s menšími odchylkami. Hybridní varianty také jako v předchozí porovnávané konfiguraci nachází větší množství extrémních hodnot vůči ostatním. Z hlediska efektivity jsou dá se říct, že varianty kromě paralelního simulovaného žíhání nachází velmi podobné výsledky. Paralelní simulované žíhání, nalézá průměrné zlepšení o 75,74 %.

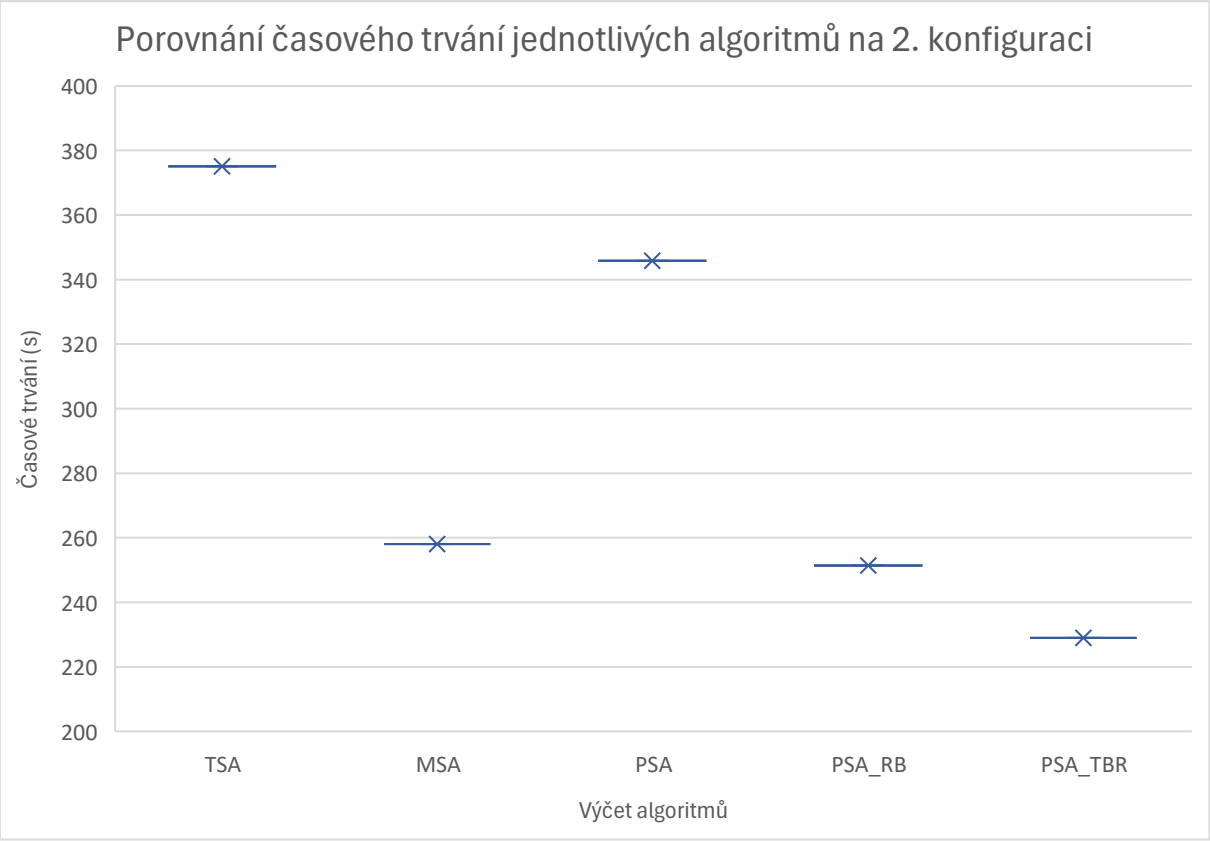
Nejlepší průměrné nalezené řešení je v případě hybridního paralelního simulovaného žíhání s Ranking selection, které nalézá průměrné zlepšení o 87,96 %, varianty tradiční simulovaného žíhání, modifikované simulované žíhání a hybridní paralelní simulované žíhání s Tournament selection přináší velmi podobné výsledky s minimálními rozdíly. Nalezená průměrná zlepšení těchto variant jsou v rozmezí 86,60 – 87,74 % tedy velmi zanedbatelné. Díky tomu lze konstatovat, že s vyšším počet cost funkcí se ztrácí vzájemné rozdíly a HPSA varianty nemají možnost vyniknout.




*Obrázek 27 – Graf nalezených výsledků algoritmů u 2. konfigurace (autor)*

Graf na obrázku 28 popisuje jednotlivé trvání porovnávaných variant simulovaného žíhání v 2. konfiguraci. Zde varianty opakují stejný vzorec jako u 1. konfigurace. Většina variant přináší stejnou výpočetní náročnost. Varianta paralelního simulovaného žíhání je velmi pomalá, jak bylo zmíněno v 1. konfiguraci zavedení vláken může potenciálně zkrátit výpočetní náročnost. Při porovnání jednotlivých variant, ale jsou nejrychlejší hybridní varianty.

Varianta s Tournament selection, má časové trvání 228 sekund. Druhá varianta Ranking selection se liší pouze o malý časový rozdíl a je opět podobná velmi variantě modifikovaného simulovaného žíhání. Časové trvání Ranking selection je 251 sekund a modifikovaného simulovaného žíhání 258 sekund rozdíl je tedy minimální. Zbylé dvě varianty jsou už pomalejší tradiční simulované žíhání mělo trvání 375 sekund a paralelní simulované žíhání 345 sekund.




*Obrázek 28 – Graf časového trvání algoritmů u 2. konfigurace (autor)*

Při celkovém zhodnocení 2. konfigurace je nejefektivnější řešení nachází varianta paralelního simulovaného žíhání, která má velmi náročné časové trvání. Dá se říct, že z hlediska nejvhodnější varianty je v případě vysokého počtu vyhodnocení cost funkce dobré použít libovolnou jinou variantu, kromě paralelního simulovaného žíhání, protože přináší přibližně podobné zlepšení.

*Tabulka 27 – Shrnutí získaných výsledků algoritmů z 2. konfigurace*

| Algoritmus | Průměrné zlepšení cesty (km) | Průměrné zlepšení řešení (%) | Časové trvání algoritmus (s) |
|---|---|---|---|
| TSA | 691 228,7564 | 87,74 | 375,081 |
| MSA | 701 029,0622 | 87,54 | 258,006 |
| PSA | 1 367 951,799 | 75,74 | 345,819 |
| HPSA_RB | 677 583,979 | 87,96 | 251,436 |
| HPSA_TBR | 754 782,0676 | 86,60 | 228,944 |

# ZÁVĚR

Cílem diplomové práce bylo porovnat vybrané varianty simulovaného žíhání na problému obchodního cestujícího.

V teoretické části byla popsána Problematika obchodního cestujícího, který má za úkol navštívit veškerá města v jeho seznamu pouze jednou, a přitom se snaží minimalizovat své náklady na cestu. Dále v této kapitole byly popsány tradiční algoritmy pro řešení tohoto problému jako například brute force, algoritmus metropolis nebo algoritmy lokálního prohledávání. Další kapitolou v teoretické části byl popis simulovaného žíhání a jeho dostupných alternativních variant. V této kapitole je rovněž popis použitých genetických operátorů (selekce, mutace), které byly využívány v modifikované variantě simulovaného žíhání a také v nově navržené variantě simulovaného žíhání vycházející z kolaborativních paralelních SA.

V praktické části byly postupně navrženy experimenty pro jednotlivé varianty, které byly následně provedeny. Pro porovnání byly vybrány čtyři varianty, které následně mezi sebou byly porovnány na základě provedených experimentů. Zvolené varianty čerpaly z tradičního simulovaného žíhání. První variantou bylo tradiční simulované žíhání. Druhá vznikla jako kombinace základního simulovaného žíhání a lokálního prohledávání s operacemi inspirovanými z genetických algoritmů (mutace, selekce), kde v každém kroku vzniká několik sousedních řešení a jen jedno se vybírá pro případné pokračování. Třetí varianta je základní paralelizované simulované žíhání s nezávislými instancemi. Čtvrtá varianta byla inspirována technikou Paralllel tempering, kde jednotlivé souběžné instance si mezi sebou mohou kolaborativní vyměňovat průběžná dílčí řešení, ale znovu s operacemi převzatých z genetických algoritmů (hlavně selekce). Na rozdíl od parallel tempering navržená varianta několikrát restartuje simulované žíhání z průběžného řešení vybraného selekcí.

Práce porovnávala různé konfigurace jednotlivých variant, a nakonec porovnala úspěšné konfigurace jednotlivých variant mezi sebou. Výsledky ukázaly, že navržená varianta hybridního paralelního SA má ve zvolených konfiguracích na vybraném datasetu nadějné výsledky a je schopna v jistých situacích překonat základní varianty algoritmu simulovaného žíhání, a i některých jeho vylepšených variant. Pro obecnější výsledky by však bylo nutné provést hlubší analýzu a více konfigurací algoritmu na větším množství různorodých datasetů.

# POUŽITÁ LITERATURA

[1] HYNEK, Josef. Genetické algoritmy a genetické programování. Praha: Grada, 2008. Průvodce (Grada). ISBN 978-80-247-2695-3.

[2] HYNEK, Josef. Definice problému. In: Genetické algoritmy a genetické programování. Praha: Grada, 2008, s. 112. Průvodce (Grada). ISBN 978-80-247-2695-3

[3] DAVENDRA, Donald. *Traveling Salesman Problem, Theory and Applications*. InTech, 2010. ISBN 978-953-307-426-9.

[4] JOHANNS, Patrick, Tim LOWE a Robert PLANTE. Selection and sequencing heuristics to reduce variance in gas turbine engine nozzle assemblies. *European Journal of Operational Research* [online]. 2001, **132**(3), 490-504 [cit. 2025-05-16]. ISSN 0377-2217. Dostupné z: doi:10.1016/S0377-2217(00)00139-9.

[5] LAPORTE, Gilbert. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* [online]. 1992, **59**(2), 231-247 [cit. 2025-05-16]. ISSN 0377-2217. Dostupné z: doi:10.1016/0377-2217(92)90138-Y

[6] BAIDOO, Evans a Stephen OPPONG. Solving the TSP using Traditional Computing Approach. *International Journal of Computer Applications* [online]. 2016, **152**(1), 13-19 [cit. 2025-05-16]. Dostupné z: doi:10.5120/ijca2016911906

[7] CHI, Yuejie, Yue M. LU a Yuxin CHEN. Nonconvex Optimization Meets Low-Rank Matrix Factorization: An Overview. *IEEE Transactions on Signal Processing* [online]. 2019, **67**(20), 5239–5269 [cit. 2025-05-16]. Dostupné z: doi:10.48550/arXiv.1809.09573

[8] DELAHAYE, Daniel, Supatcha CHAIMATANAN a Marcel MONGEAU. Simulated annealing: From basics to applications. In: *Handbook of Metaheuristics* [online]. 3. Springer International, 2019, s. 1-35 [cit. 2025-05-16]. ISBN 978-3-319-91086-4. Dostupné z: https://doi.org/10.1007/978-3-319-91086-4_1

[9] KIRKPATRICK, Scott, C. Daniel GELATT a Mario P. VECCHI. Optimization by Simulated Annealing. *Science (New York, N.Y.)* [online]. 1983, **220**(1), 671-680 [cit. 2025-05-16]. Dostupné z: doi:10.1126/science.220.4598.671

[10] GENDREAU, Michel a Jean-Yves POTVIN. *Handbook of Metaheuristics* [online]. 3. Springer Cham, 2019 [cit. 2025-05-16]. ISBN 978-3-319-91086-4. Dostupné z: doi:10.1007/978-3-319-91086-4

[11] HACHTEL, Gary D. a Fabio SOMENZI. *Logic Synthesis and Verification Algorithms* [online]. Springer New York, NY, 2005 [cit. 2025-05-16]. ISBN 978-0-306-47592-4. Dostupné z: doi:10.1007/b117060

[12] CHIBANTE, Rui. *Simulated Annealing, Theory with Applications* [online]. IN TECH d.o.o., 2010 [cit. 2025-05-16]. ISBN 978-953-51-5931-5. Dostupné z: doi:10.5772/252

[13] AMINE, Khalil. Multiobjective Simulated Annealing: Principles and Algorithm Variants. *Advances in Operations Research* [online]. 2019, **2019**(1), 13 [cit. 2025-05-16]. Dostupné z: doi:10.1155/2019/8134674

[14] PEREIRA, Ana I a Edite M.G.P. FERNANDES. A study of simulated annealing variants. *XXVIII Congreso de Estadística e Investigación Operativa* [online]. 2004, **2004**(1), 16 [cit. 2025-05-16]. ISSN 84-689-0438-4. Dostupné z: http://hdl.handle.net/10198/1615

[15] FURQAN, Ahmed a Olav TIRKKONEN. Simulated annealing variants for self-organized resource allocation in small cell networks. *Applied Soft Computing* [online]. 2016, **38**(1), 762-770 [cit. 2025-05-16]. Dostupné z: doi:10.1016/j.asoc.2015.10.028

[16] DUECK, Gunter a Tobias SCHEUER. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics* [online]. 1990, **90**(1), 161-175 [cit. 2025-05-16]. Dostupné z: doi:10.1016/0021-9991(90)90201-B

[17] KONTOGHIORGHES, Erricos John a Cristian GATU. *Optimisation, Econometric and Financial Analysis* [online]. Springer Berlin, Heidelberg, 2007 [cit. 2025-05-16]. ISBN 978-3-540-36626-3. Dostupné z: doi:10.1007/3-540-36626-1

[18] GALLANT, Stephen I. a Stephen I. GALLANT. Simulated Annealing and Boltzmann Machines. In: *Neural Network Learning and Expert Systems* [online]. MIT Press, 1993, s. 245-251 [cit. 2025-05-16]. Dostupné z: https://ieeexplore.ieee.org/document/6285478

[19] LOU, Zhihao a John REINITZ. Parallel simulated annealing using an adaptive resampling interval. *Parallel Computing* [online]. 2016, **53**(1), 23-31 [cit. 2025-05-16]. ISSN 0167-8191. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167819116000430

[20] BERTSIMAS, Dimitris a John TSITSIKLIS. Simulated Annealing. *Statistical Science* [online]. 1993, **1993**(8), 10-15 [cit. 2025-05-17]. Dostupné z: http://www.jstor.org/stable/2246034

[21] MITCHELL, Melanie. *An introduction to genetic algorithms*. 2nd print. A Bradford Book. Cambridge, Mass.: MIT Press, 1996. ISBN 0-262-13316-4.

[22] RAZALI, Noraini a GERAGHTY, John. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. In: AO, SIO-IONG; GELMAN, Len; HUKINS, David; HUNTER, Andrew a KORSUNSKY, Alexander. *Proceedings of the World Congress on Engineering 2011, Vol II*. Londýn: Newswood Limited, 2011, s. 1134-1139. ISBN 978-988-19251-4-5. ISSN 2078-0966.

[23] GOLDBERG, David E. *Genetic Algorithms in Search, Optimization and Machine Learning* [online]. Addison-Wesley Longman Publishing Co., 1989. ISBN 978-0-201-15767-3.

[24] HAUPT, Sue Ellen, Sue Ellen HAUPT, Antonello PASINI a Caren MARZBAN. Introduction to Genetic Algorithms. In: *Artificial Intelligence Methods in the Environmental Sciences* [online]. Springer, Dordrecht, s. 103-125 [cit. 2025-05-17]. Dostupné z: https://doi.org/10.1007/978-1-4020-9119-3_5

[25] SONI, Nitasha a Tapas KUMAR. Study of Various Mutation Operators in Genetic Algorithms. *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3)*. India, 2014, **5**(3), 4519-4521. ISSN 0975-9646.

[26] ULLAH, Sami, Abdus SALAM a Mohsin MASOOD. Analysis and comparison of a proposed mutation operator and its effects on the performance of genetic algorithm. *Indonesian Journal of Electrical Engineering and Computer Science* [online]. 2022, **25**(1), 1208-1216 [cit. 2025-05-17]. Dostupné z: doi:10.11591/ijeecs.v25.i2.pp1208-1216

[27] Ali Jahed, Misagh Rahbari. Comparison of Three Neighbor Generation Structures by Simulated Annealing Method to Solve Quadratic Assignment Problem. 10th International Conference of Iranian Operations Research Society (ICIORS 2017), University of Mazandaran, May 2017, Babolsar, Iran. ffhal-01962309

[28] WANG, Chiaming, Jeffrey HYMAN, Allon PERCUS a Russel CAFLISCH. Parallel Tempering for the Traveling Salesman Problem. *International Journal of Modern Physics C - IJMPC*. 2009, **20**(4), 539-556.

[29] University of Waterloo. (n.d.). World TSP problem instances by country. University of Waterloo. Retrieved from https://www.math.uwaterloo.ca/tsp/world/countries.html#WI

[30] COOK, William. Po stopách obchodního cestujícího: matematika na hranicích možností. Zip. Praha: Argo, 2012. ISBN 978-80-7363-412-4.

[31] FERNÁNDEZ DE VEGA, Francisco; HIDALGO PÉREZ, José Ignacio a LANCHARES, Juan. *Parallel Architectures and Bioinspired Algorithms*. Online. Springer Berlin, Heidelberg, 2012. ISBN 978-3-642-28789-3. Dostupné z: https://doi.org/10.1007/978-3-642-28789-3. [cit. 2025-05-17].

[32] ARNOLD, Ken; GOSLING, James a HOLMES, David. *The Java programming language*. 3rd ed. The Java series. Boston: Addison-Wesley, 2000. ISBN 0-201-70433-1.