VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

KVANTOVĚ INSPIROVANÉ OPTIMALIZAČNÍ ALGORITMY

QUANTUM-INSPIRED OPTIMISATION ALGORITHMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DOMINIK KOSÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL BIDLO, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Kosík Dominik, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Strojové učení
Název: **Kvantově inspirované optimalizační algoritmy**
Quantum-Inspired Optimisation Algorithms
Kategorie: Umělá inteligence
Zadání:

1. Nastudujte problematiku optimalizačních algoritmů využívajících principy kvantové fyziky.
2. Zvolte vhodný algoritmus z této kategorie a implementujte jej ve zvoleném prostředí.
3. Na vhodně zvolené množině úloh proveďte sadu experimentů demonstrujících schopnosti vybraného algoritmu, případně jeho modifikací.
4. Porovnejte výsledky z bodu 3 s výsledky získanými pomocí běžných optimalizačních metod, např. simulovaného žíhání nebo genetického algoritmu.
5. Vytvořte srovnávací studii s výsledky řešení vybraných úloh pomocí různých variant algoritmů z bodů 2 a 3.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího projektu.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Bidlo Michal, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 29. října 2021

Abstrakt

Tato práce se zabývá implementací zvoleného kvantově inspirovaného optimalizačního algoritmu a jeho rozšířeními, které budou porovnávány v závěru práce. Jako optimalizační algoritmus byl zvolen algoritmus pro simulované kvantové žíhání. V první části se nachází základní popis běžných optimalizačních metod použitých v této práci, teoretické základy fyziky, ze které vychází inspirace pro kvantově inspirované optimalizační algoritmy, a popis simulovaného kvantového žíhání. V druhé části práce je implementace algoritmů pro námi zvolené úlohy, kterými jsou problém obchodního cestujícího, hledání pravidel pro celulární automaty a problém MAX-SAT. Poslední část obsahuje modifikace simulovaného kvantového žíhání, srovnání se základní variantou a s běžnými optimalizačními algoritmy následované vyhodnocením tohoto srovnání.

Abstract

The focus of this work is an implementation of the chosen quantum-inspired optimisation algorithm and its modifications, which will be compared at the end of the work. As the optimisation algorithm was chosen simulated quantum annealing algorithm. The first part of the work will lay the theoretical groundwork of standard optimisation algorithms used in this work, physics from which the inspiration for the simulated quantum annealing originates, and a description of the chosen algorithm. The second part will focus on the implementation of the algorithms on the selected problems. The selected problems are travelling salesman problem, searching rules for cellular automaton and MAX-SAT problem. The last part will contain the proposed modifications of the simulated quantum annealing, a comparison to the basic variant and standard optimisations algorithms, and an evaluation of the results.

Klíčová slova

horolezecký algoritmus, simulované žíhání, simulované kvantové žíhání, klonální selekce, evoluční strategie, problém obchodního cestujícího, MAX-SAT, celulární automat, Monte Carlo založené na křivkovém integrálu, Isingův model spinových skel

Keywords

Hill Climbing, Simulated Annealing, Simulated Quantum Annealing, Clonal Selection, Evolution Strategy, Traveling Salesman Problem, MAX-SAT, Cellular Automaton, Path Integral Monte Carlo, Ising Spin Glass Model

Citace

KOSÍK, Dominik. *Kvantově inspirované optimalizační algoritmy.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Kvantově inspirované optimalizační algoritmy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Dominik Kosík
10. května 2022

Poděkování

Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy České republiky prostřednictvím e-INFRA CZ (ID:90140).

Obsah

1 Úvod **3**

2 Přírodou inspirované optimalizační algoritmy **4**
2.1 Evoluční algoritmy 4
2.1.1 Evoluční strategie 5
2.2 Algoritmy inspirované imunitním systémem 7
2.2.1 Klonální selekce 8
2.3 Klasické simulované žíhání 9
2.4 Kvantově inspirované algoritmy 10

3 Simulované kvantové žíhání **11**
3.1 Optimalizace založená na klasické fyzice 11
3.1.1 Energie, hybnost a hamiltonián 12
3.1.2 Termodynamika a statistická mechanika 13
3.1.3 Klasický Isingův model spinových skel 15
3.2 Optimalizace založená na kvantové fyzice 15
3.2.1 Kvantová mechanika 16
3.2.2 Pozorovaní kvantových jevů 16
3.2.3 Koherence a dekohorence 16
3.2.4 Kvantové provázání 17
3.2.5 Heisenbergův princip neurčitosti 18
3.2.6 Kvantové tunelování 18
3.2.7 Kvantový hamiltonián 18
3.2.8 Kvantový Isingův model spinových skel 19
3.3 Simulované klasické žíhání 19
3.3.1 Myšlenka klasického simulované žíhání 19
3.3.2 Horolezecký algoritmus 20
3.3.3 Popis simulovaného žíhání 20
3.4 Simulované kvantové žíhání 22
3.4.1 Simulované kvantové žíhání 22
3.4.2 Suzuki-Trotterova transformace 22
3.4.3 Monte Carlo založené na křivkovém integrálu 22
3.4.4 Aplikace teoretických poznatků do algoritmu 23

4 Řešené problémy **27**
4.1 Problém obchodního cestujícího 28
4.1.1 Řešení TSP za pomocí klasického simulovaného žíhání 28
4.1.2 Řešení TSP za pomocí simulovaného kvantového žíhání 29

4.2 Problém maximální splnitelnosti booleovské formule . . . 32
4.2.1 Splnitelnosti booleovské formule . . . 32
4.2.2 Maximální splnitelnosti booleovské formule . . . 32
4.2.3 Řešení MAX-SAT pomocí klonální selekce . . . 33
4.2.4 Řešení MAX-SAT pomocí simulovaného kvantového žíhání . . . 34
4.3 Hledání pravidel celulárního automatu . . . 36
4.3.1 Celulární automaty . . . 36
4.3.2 Pravidla automatu . . . 36
4.3.3 Účel celulárního automatu . . . 37
4.3.4 Účelová funkce pro pravidla celulárního automatu . . . 38
4.3.5 Hledání pravidel pomocí evoluční strategie . . . 39
4.3.6 Hledání pravidel pomocí simulovaného kvantového žíhání . . . 40

5 Nové varianty simulovaného kvantového žíhání 42
5.1 Ovlivnění nejlepším řešením . . . 42
5.2 Delší optimalizace jednotlivých replik . . . 43
5.3 Omezení nových řešení pomocí tabu meta-heuristicky . . . 43
5.4 Převzetí části řešení z nejlepšího řešení . . . 44

6 Výsledky problému obchodního cestujícího 45
6.1 100 měst . . . 47
6.2 1000 měst . . . 49
6.3 5000 měst . . . 51
6.4 Zhodnocení . . . 52

7 Výsledky maximální splnitelnosti booleovské formule 53
7.1 Jednodušší problémy . . . 54
7.1.1 MAX-2SAT, 160 proměnných a 2400 klauzulí . . . 54
7.1.2 MAX-2SAT, 200 proměnných a 1800 klauzulí . . . 56
7.1.3 MAX-3SAT, 110 proměnných a 1100 klauzulí . . . 57
7.2 Obtížnější problémy . . . 58
7.2.1 MAX-4SAT, 192 proměnných a 432 klauzulí . . . 59
7.2.2 MAX-4SAT, 448 proměnných a 1120 klauzulí . . . 60
7.2.3 MAX-4SAT, 1024 proměnných a 2816 klauzulí . . . 62
7.3 Zhodnocení . . . 63

8 Výsledky hledání pravidel celulárního automatu 64
8.1 Vzor francouzské vlajky . . . 65
8.2 Vzor loga VUT . . . 67
8.3 Zhodnocení . . . 69

9 Závěr 70

Literatura 71

Kapitola 1

Úvod

Přestože se výkon počítačů a techniky v dnešní době stále zvyšuje a jsme schopni řešit více úloh přesně a v přijatelném čase, tak pořád máme určité úlohy, které nejsme schopni efektivně vyřešit. Z tohoto důvodu existuje skupina postupů hledajících pouze přibližná řešení těchto úloh, ale v mnohem kratším čase. A s vývojem počítačů jsou i tyto postupy průběžně vylepšovány k dosažení lepších výsledků.

Současné metody výpočtu přibližných řešení úloh jsou vysoce efektivní, ale i tak existují problémy ve skutečném světě, kde výkon těchto postupů je nedostatečný. Zvýšení rychlosti a přesnosti hledání tohoto řešení by umožnilo efektivnější řešení problémů v mnoha oblastech plánování a výroby.

Jednou z oblastí, ze které výpočetních řada postupů čerpá inspiraci, je samotná příroda a chování organismů. Do této oblasti patří i inspirace ze zákonů fyziky a s rozvojem pochopení kvantové mechaniky se objevují nové postupy inspirované právě kvantovými jevy.

Tato kvantová oblast je z pohledu inspirace pro současné algoritmy zatím jen málo prozkoumaná a může skrývat myšlenky pro efektivnější návrhy a postupy i pro klasické počítače.

V této práci se budeme zabývat hledáním nových způsobů, které jsou inspirovány právě z jevů kvantové mechaniky a jejich aplikací na klasické postupy řešení úloh. Snahou je získat vylepšený postup dosahující lepších vlastností a umožnující ve vybraných oblastech efektivnější řešení problémů se kterými se potýkají.

Tato práce je členěna do několika kapitol. Nejprve popisuje přírodou inspirované algoritmy, které budou využity pro srovnání s kvantově inspirovaným algoritmem. Dále práce obsahuje základ klasické a kvantové mechaniky, ze které bude vycházet naše inspirace. Po těchto základech následuje popis řešených úloh a navržené modifikace pro kvantově inspirovaný algoritmus. Poslední kapitoly obsahují srovnávací studii algoritmů na zvolených úlohách.

Kapitola 2

Přírodou inspirované optimalizační algoritmy

Přírodou inspirované algoritmy získávají svoji inspiraci z velkého množství oborů zahrnujících mimo jiné matematiku, statistiku, informatiku, biologii, fyziku a chemii. Inspirace z těchto přírodních a často paralelních procesů řešících problémy přírody (jako například hledání zdrojů potravy nebo proces ochlazování kovů) nám dává nové nápady na návrh algoritmů pro řešení optimalizačních problémů. [7]

V průběhu času vzniklo velké množství algoritmů z této kategorie, které nejsou pouze ukázkou, že příroda navrhla svoje řešení na problémy, se kterýma se potýkala, ale z jejich inspirace se vytvořily algoritmy, které dokážou řešit reálné problémy. Dokonce některé algoritmy (například neuronové sítě) se staly dominantní pro určité problémy jako rozpoznávání obrazu a identifikace. [1]

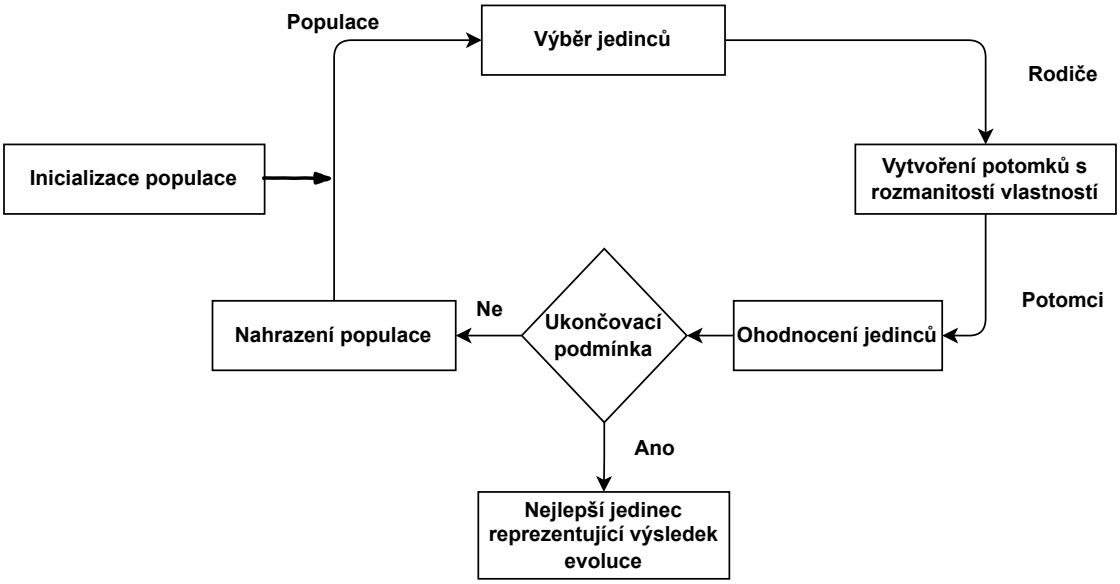
Pro experimenty v této práci budeme používat různé optimalizační techniky, které jsou právě inspirovány přírodou a podrobněji jsou popsány v následujících sekcích.

2.1 Evoluční algoritmy

Evoluční proces reprezentuje skupinu, která se povznesla nad její biologický základ. Tento proces může být rozlišen na základě těchto čtyřech složek: [7]

i. populace jedinců,
ii. mechanismus výběru,
iii. způsob udržení formy,
iv. vytvoření rozmanitosti.

V biologické evoluci je proces vytvoření nové generace řízen přežitím jedinců a vytvořením potomků. Tento proces dává vzniknout evolučnímu cyklu zobrazenému na obrázku 2.1.



Obrázek 2.1: Diagram znázorňující průběh výpočetního evolučního procesu.

Evoluční algoritmy se poté dělí do několika oblastí jako například genetické algoritmy, genetické programování, evoluční strategie a jiné. V této práci se budeme zabývat evoluční strategií, která bude použita pro řešení hledání pravidel pro celulární automat a bude popsána v následující sekci.

2.1.1 Evoluční strategie

Evoluční strategie (ES) byla vymyšlena Rechenbergem a Schwefelem [48]. V 60. letech získala oblibu výzkumníků, kde byla hojně využívána jako nástroj pro řešení optimalizačních problémů využívající desetinná čísla, optimalizaci kombinatorických úloh a jiných problémů. [7]

Evoluční strategie v této práci bude použita jako algoritmus pro srovnání s kvantově inspirovaným algoritmem pro úlohu hledání pravidel pro celulární automat, která bude podrobně popsána v kapitole 4.

Dvě význačné vlastnosti ES je použití na problémy mající reprezentaci řešení v desetinných číslech, a že hlavní proces optimalizace leží primárně v mutaci a selekci jedinců. Jedinci jsou obvykle reprezentováni jako vektor čísel popisující řešení dané úlohy. Vytvoření potomka poté znamená vytvořit kopii rodiče a provést mutaci. Tedy provést náhodnou modifikaci jednoho nebo více čísel v tomto vektoru, a tudíž vytvořit nové řešení úlohy. [7]

Základních algoritmů ES existuje několik, a ty se poté ještě dále specializují v závislosti na úloze pro kterou jsou použity. Jeden z obecný algoritmů ES pro optimalizaci funkce je znázorněn níže:

Algorithm 1: Základní algoritmus pro evoluční strategii $(\mu + \lambda)$ [7]

Vytvoření náhodné počáteční populace jedinců $\{x_1, ..., x_\mu\}$ s vektorem čísel $\{y_1, ..., y_m\}$ určující řešení problému.
Ohodnocení jedinců populace.

while *ukončovací podmínka* **do**
  while *počet vytvořených potomků* $< \lambda$ **do**
    Vyber náhodného rodiče z populace.
    Vytvoř potomka pomocí mutace vektoru rodiče.
  end
  Ohodnocení potomků.
  Seřazení potomků a jedinců původní populace na základě kvality řešení.
  Vyber μ nejlepších jedinců pro vytvoření populace pro další generaci.
end
return *nejlepší jedinec*

Různé modifikace ES poté spočívají v množství rodičů, potomků, způsobu mutace, modifikace potomků a výběru rodičů. Níže jsou uvedené některé modifikace z [7].

Evoluční strategie $(1 + 1)$

Tato modifikace redukuje celou populaci jedinců na pouze jednoho jedince a pouze jeden potomek je vytvořen v generaci. Tedy v každé nové generaci soupeří rodič s jeho potomkem o to, kdo má lepší řešení a zůstane pro nadcházející generaci.

Evoluční strategie $(\mu + \lambda)$ a (μ, λ)

Další rozšíření staví na modifikaci $(1 + 1)$, kde získáme varianty ES $(1 + \lambda)$ a $(1, \lambda)$. Tedy jeden jedinec populace vytvoří v jedné generaci λ potomků, kteří mezi sebou soupeří. V případě $(1 + \lambda)$ mezi sebou soupeří potomci a rodič, kde pro další generaci se vybere pouze jediný vítěz. Při modifikaci $(1, \lambda)$ spolu soupeří pouze potomci a jedinec z původní generace se nemůže dostat do nové generace.

Logicky další rozšíření zvětšuje velikost populace ze které se vytváří potomci. Poté se jedná o evoluční strategii $(\mu + \lambda)$ nebo (μ, λ) podle toho, zdali jedinci původní generace mohou být součástí nové generace nebo nemohou. V obou případech se vytvoří λ potomků z náhodných rodičů a vybere se μ vítězných pro nadcházející generaci. Algoritmus pro tuto evoluční strategii byl již ukázán výše: algoritmus 1.

Z pohledu optimalizace se pro $(\mu + \lambda)$ jedná o strategii výběru elity, jelikož nalezené řešení mohou být nahrazeny pouze lepšími řešeními (nebo stejně dobrými). Naproti tomu u (μ, λ) se může nejlepší řešení problému ztratit a být nahrazeno potomky, kteří mají horší řešení. Nicméně i přes možnost ztráty nejlepšího řešení je strategie (μ, λ) více používaná, jelikož má vyšší šanci vyprostit se z lokálního minima při optimalizaci [7].

Mutace

Každý jedinec z populace je popsán vektorem čísel, které jsou parametry řešení pro daný problém. Proces mutace modifikuje tento vektor čísel, a tedy vytváří nové možné řešení.

V závislosti na strategii řídící provádění této mutace probíhá proces hledání optimálního řešení.

Uvažujme nejjednodušší případ, kde vektor jedince je pouze jediné reálné číslo popisující parametr pro řešení problému. Poté vytvoření potomka z rodičce lze získat přidáním náhodného čísla z normálního rozložení $N(0, \sigma)$, kde σ je libovolně zvolený rozptyl normálního rozložení.

Tedy číslo reprezentující potomka je dáno vztahem: $x(t+1) = x(t) + r$, kde $x(t)$ je potomek v generaci t a $r = N(0, \sigma)$. Při použití normálního rozložení a volbě malého σ získáme malou mutaci (malý rozdíl vůči předkovi) s nižší pravděpodobností získání větší mutace.

Tento přístup je jednoduchý na implementaci, ale trpí na určité problémy jako problém při rozdílném škálování dimenzí, nebo že krok mutace je stejný v průběhu celého optimalizačního procesu. Řešením těchto problémů vyžaduje složitější postupy, které zde nebudou popsány s ohledem na zaměření této práce.

Rekombinace

Rekombinace (křížení) je proces vytváření jednoho nebo více potomků z kombinace genetické informace rodičů, která popisuje řešení úlohy.

Přestože původní implementace a hlavní myšlenka optimalizace je založená na principu mutace, tak může také zahrnovat rekombinaci. V tomto případě je původní algoritmus upraven tak, že μ potomků vznikne rekombinací z rodičů a poté je na tyto potomky použita mutace. Tato rekombinace typicky vytváří jediného potomka, ale lze jich vytvořit i více.

Evoluční strategie neurčuje množství z kolika rodičů musí potomci vzniknout, a tedy standardní notace pro ES udává tento parametr ρ udávající počet rodičů ke vzniku potomka. Celý zápis poté má tvar: $(\mu/\rho, +\lambda) - ES$, kde musí platit $\rho \leq \mu$.

Pro rekombinaci poté existují dva přístupy. Diskrétní, kde hodnoty pro potomka jsou náhodně zvoleny z některého rodiče, a aritmetická rekombinace, která vytváří průměrné hodnoty jednotlivých čísel z rodičů. Tyto přístupy jsou znázorněny v tabulce 2.1 uvažující dva rodiče pro vytvoření potomka.

Rodič 1	0,2	1,3	3,2	2,6
Rodič 2	2,1	1,2	2,4	4,0
Potomek diskrétní rekombinace	0,2	1,3	2,4	2,6
Potomek aritmetické rekombinace	1,15	1,25	2,8	3,3

Tabulka 2.1: Tabulka zobrazující diskrétní a aritmetickou rekombinaci dvou rodičů

2.2 Algoritmy inspirované imunitním systémem

Imunitní systém je tvořen spletitou spoluprací specializovaných tkání, orgánů, buněk a chemických procesů. Imunitní systém má schopnost rozpoznat vysoké množství různých patogenů, které mohou vstoupit do těla, a zneškodnit je. Tento systém také chrání tělo před vlastními buňkami, které se začnou chovat odlišně. [46]

Algoritmy založené na inspiraci imunitním systémem se mohou zhruba dělit na čtyři kategorie. Negativní a pozitivní selekce, která může být použita na detekci nebo klasifikaci

anomálií. Klonální selekce, která se dá využít na optimalizaci a klasifikaci. Teorie ohrožení, která dává vzniknout algoritmům pro detekci anomálií a algoritmy založené na teorii imunitní sítě. [7]

V této práci se budeme zabývat algoritmem z kategorie klonální selekce, kde inspiraci z tohoto odvětví použijeme jako srovnání se simulovaným kvantovým žíháním pro hledání maximální splnitelnosti booleovské formule.

2.2.1 Klonální selekce

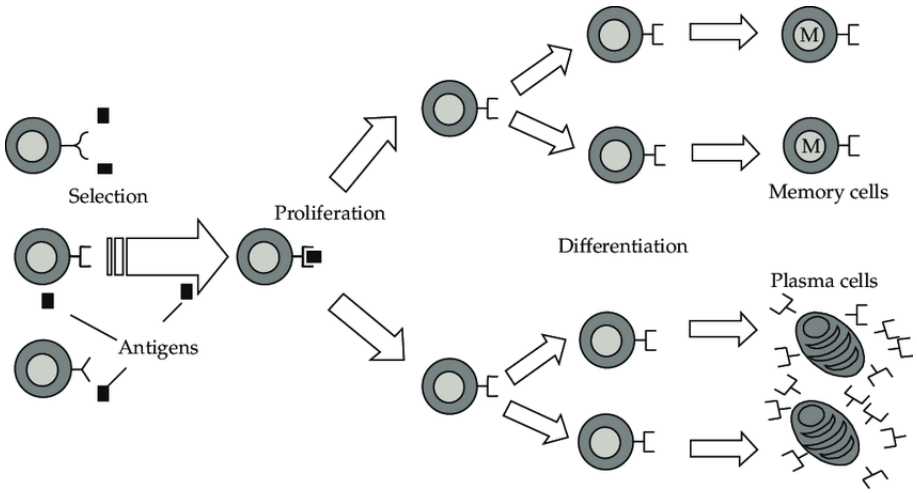
Pro řešení složitých optimalizačních problémů hledáme inspiraci ke tvorbě nových postupů nebo heuristik, kde jeden z velkých zdrojů je právě samotná příroda a funkce imunitního systému sem patří také. Imunitní systém je schopen tělo chránit díky analýze na několika úrovních: molekulární, buněčná a na úrovni orgánů. Tento systém se dá dělit na dvě hlavní části. Vrozená imunitní obrana, která tělo chrání od narození, a adaptivní imunitní systém, který reprezentuje všechny obranné prostředky získané za života jedince. [36]

Specifický druh buněk, známý jako B-lymfocyt (B buňky), je zodpovědný za ničení patogenů v těle. Tyto buňky produkují protilátky, které se vážou na patogeny a označují je pro zničení. Síla této vazby mezi protilátkou a antigenem se nazývá avidita. [3]

Funkce těchto B-lymfocytů je inspirací, která dala vzniknou konceptu algoritmu založeném na umělém imunitním systému. Tento systém se snaží zachytit principy imunitního systému. Především jeho schopnost zapamatování informací, rozpoznávání vzorů a adaptace. Jedna z vlastností imunitního systému je vytvořit specifické protilátky na nové antigeny (potenciálně škodlivé molekuly nebo pyly, které se mohou vázat na protilátky). Inspirace z této vlastnosti dává vznik algoritmu klonální selekce. [36]

Princip tohoto algoritmu je následující: když je spatřen neznámý antigen, tak B buňky začnou vytvářet protilátky. Tyto buňky s nejlepší schopností rozpoznávat tento nový antigen se začnou šířit za pomoci klonování. Klonované buňky mají vysokou míru mutace k podpoření jejich genetické variace. B-lymfocyt s vysokou aviditou se dělí na plazmatické buňky (plazmocyty) a paměťové T lymfocyty. B-lymfocyty s nízkou aviditou jsou buď zničeny nebo mutují. [29]

Princip tohoto procesu je znázorněn na obrázků 2.2.



Obrázek 2.2: Diagram znázorňující proces klonování B-lymfocytů a jejich dělení na plazmatické buňky a paměťové T lymfocyty. Obrázek převzat z [56]

Tento princip dal vzniknou klonální selekci (Clonal selection algorithm), která využívá umělého imunitního systému k řešení optimalizačních problémů. Pseudokód tohoto algoritmu pro optimalizaci je uveden níže (varianta pro hledání vzorů by obsahovala také paměťové buňky) [8]:

Algorithm 2: Základní algoritmus pro klonální selekci

Vstupy:

μ - množství protilátek
η - počet náhodných protilátek

Vytvoření náhodné počáteční populace protilátek $P = \{x_1, ..., x_\mu\}$ s vektorem čísel $\{y_1, ..., y_m\}$ určující řešení problému.

while *ukončovací podmínka* **do**
  foreach *protilátku p* z *P* **do**
    Spočítej aviditu pro p (ohodnocení řešení pomocí vektoru čísel dané protilátky).
  end
  Selekce n protilátek z P do pole P1.
  foreach *protilátku p* z *P1* **do**
    Vytvoř kopii p a vlož do pole C.
  end
  foreach *protilátku c* z *C* **do**
    Proveď hypermutaci protilátky c s množstvím mutací nepřímo úměrnou její aviditě.
    Spočítej novou aviditu pro c.
  end
  Do P se vloží μ nejlepších protilátek z C a P.
  η nejhorších protilátek je nahrazeno novými náhodnými protilátky.
end
return *nejlepší protilátka*

2.3 Klasické simulované žíhání

Jako algoritmus ke srovnání se simulovaným kvantovým žíháním pro problém obchodního cestujícího bylo zvoleno klasické simulované žíhání. Tento algoritmus je inspirován procesy probíhajícími při postupném ochlazování zahřátých kovů. Při pomalém ochlazování kovu se mění struktura jeho krystalické mřížky do více uspořádané [41]. Z pohledu optimalizace se při tomto procesu snižuje pnutí v kovu, a tedy se vytváří optimálnější struktura kovu. Popis tohoto jevu je vysvětlen klasickou fyzikou, termodynamikou a statistickou mechanikou.

Podrobné vysvětlení tohoto algoritmu a principů, ze kterých je tento algoritmus inspirován, bude uvedeno v následující kapitole, jelikož tento algoritmus přímo souvisí s vybraným kvantově inspirovaným algoritmem.

2.4 Kvantově inspirované algoritmy

Kvantový systém má určité vlastnosti, které jsou velice vhodné pro optimalizační algoritmy, umožňující vyřešit některé problémy v řádově nižším čase [7]. Nicméně k plnému využití těchto vlastností je nutný specializovaný hardware využívající principů kvantové mechaniky. Z tohoto důvodu se dále budeme zabývat pouze algoritmy, které jsou inspirované kvantovou mechanikou a fungují na klasickém hardwaru.

Hlavní algoritmus, kterým se v této práci budeme zabývat, je simulované kvantové žíhání. Tento algoritmus, jak již bylo nastíněno, vychází z algoritmu klasického simulovaném žíhání, který bude rozšířen o poznatky z kvantové fyziky dávající zrod simulovanému kvantovému žíhání.

Podrobný popis základů a inspirací z termodynamiky, statistické a kvantové mechaniky a algoritmů, na kterých simulované klasické a kvantové žíhání staví, bude uvedeno v následující kapitole.

Kapitola 3

Simulované kvantové žíhání

Tato práce se zabývá myšlenkou, zda za pomoci inspirace kvantovými jevy lze vylepšit již stávající algoritmus simulovaného žíhání, nebo dokonce vytvořit zcela nový algoritmus, který bude dosahovat lepších výsledků na optimalizačních problémech, přestože bude použit na klasických (nekvantových) počítačích.

Specifický algoritmus vybrán pro tuto práci je simulované kvantové žíhání, který rozšiřuje stávající algoritmus klasického simulované žíhaní na kvantově inspirovaný. Vzhledem tomu, že tento vybraný algoritmus je tedy již inspirován kvantovými jevy, tak se budeme v této práci snažit o jeho zlepšení.

Z tohoto důvodu bylo vymyšleno několik modifikací simulovaného kvantového žíhání, které vylepšují původní variantu tohoto algoritmu. Tyto modifikace budou podrobně popsány v kapitole 5.

Vysvětlení těchto modifikací bude předcházet úvod do klasické fyziky, kvantové fyziky, klasického žíhání (a simulovaného žíhání) a nakonec vysvětlení simulovaného kvantové žíhání.

3.1 Optimalizace založená na klasické fyzice

Předchozí kapitola popisovala algoritmy inspirované z oblastí živých prvků přírody. Nicméně se lze inspirovat z neživé přírody a to přesněji zákony fyziky. I ve fyzice lze nalézt určité principy, které se například snaží minimalizovat energii systému. Tuto vlastnost lze využít pro optimalizaci jiných problémů. Tedy pokud dokážeme převést náš problém na fyzikální systém, kde energie tohoto systému odpovídá našemu ohodnocení řešení, tak poté minimalizací energie tohoto systému hledáme řešení našeho problému [19].

Myšlenkou této práce je využít kvantové jevy k modifikaci nebo vytvoření optimalizačních algoritmů a dosáhnout tak lepších vlastností při optimalizaci. Z algoritmů, které využívají kvantových vlastností, to mohou být například: algoritmus optimalizace kvantovým hejnem částic[38] nebo algoritmus simulovaného kvantového žíhání[7].

Hlavní algoritmus, kterým se budeme zabývat v této práci bude kvantové simulované žíhání. K pochopení, jak a z čeho byl tento algoritmus inspirován, zde bude alespoň ve zkratce uvedena část fyziky, vysvětlující, na čem je algoritmus založen. Neboť k vysvětlení kvantových jevů, a jak souvisejí s energii systému, je nutné postupně projít vývoj fyziky až k tomuto bodu. Hlavní struktura tohoto popisu fyzikální principů a jejich vysvětlení je volně inspirováno z [7].

3.1.1 Energie, hybnost a hamiltonián

Energie je schopnost tělesa konat práci. Tato energie systému se skládá ze dvou částí: kinetická energie, označována K, kterou systém obsahuje na základě jeho pohybu, a potenciální energie V, která určuje množství energie uložené v tělesu na základe sil působících na dané těleso.

Energie je poté nezáporný skalár, kde celková energie tělesa je neměnná a rovna součtu kinetické a potenciální energie tělesa za předpokladu, že na těleso nepůsobí vnější síly. Tuto skutečnost popisuje zákon zachování energie: Celkové množství energie (všech druhů) izolované soustavy zůstává zachována [18].

Před pojmem hybnosti tělesa je nutné definovat jeho rychlost. Tato rychlost odpovídá první derivaci délky cesty podle času:

$$v = \frac{\partial s}{\partial t}. \tag{3.1}$$

Poté můžeme vyjádřit hybnost tělesa o hmotnosti m, pohybující se rychlostí v za pomoci vztahu:

$$p = mv. \tag{3.2}$$

Celková hybnost tělesa je poté dána součtem všech jednotlivých vektorů hybnosti tělesa a těleso setrvává v tomto pohybu, pokud na těleso nepůsobí vnější síly. Tato vlastnost je popsána 1. Newtonovým zákonem o setrvačnosti ve znění: Jestliže na těleso nepůsobí žádné vnější síly nebo výslednice sil je nulová, pak těleso setrvává v klidu nebo v rovnoměrném přímočarém pohybu [43].

Kinetická energie je poté definována pomocí hybnosti a hmotnosti tělesa za pomoci vztahu:

$$E_k = \frac{p^2}{2m}. \tag{3.3}$$

Dále budeme potřebovat 2. Newtonův zákon, říkající, že rychlost změny hybnosti tělesa v čase t je přímo úměrná síle působící na těleso a nastává ve stejném směru. Tento vztah je popsán rovnicí [43]:

$$F = \frac{\partial p}{\partial t}. \tag{3.4}$$

Jelikož působící síla na těleso mění jeho rychlost (a tedy polohu v čase), tak můžeme říct, že těleso koná práci.

Následně lze říci, že potenciální skalární pole udává v každém bodě rozdíl potenciálních energií tělesa. Ve dvou různých polohách tento rozdíl závisí pouze na vzájemné poloze tělesa a nikoliv na jeho dráze. Skalární pole ve třech dimenzích poté pouze závisí na jeho poloze. Tento vztah lze definovat jako vektorové pole $\mathbf{F}$ [24]:

$$\mathbf{F} = -\nabla P = -\left(\frac{\partial P}{\partial x}, \frac{\partial P}{\partial y}, \frac{\partial P}{\partial z}\right), \tag{3.5}$$

kde P je potenciální energie a ∇P je gradient P.

Toto pole udává síly mezi částicemi na základě potenciální energie E_V tělesa. Poté z Newtonových zákonů dostaneme, že pokud na těleso nepůsobí vnější síly, tak toto potenciální pole působí silou na těleso a snižuje jeho potenciální energii. Tento rozdíl energie je poté přeměněn na kinetickou energii tělesa, a tedy součet E_K *a* E_V zůstává konstantní a splňuje zákon zachování energie.

Nyní můžeme celkovou energii vyjádřit za pomoci hamiltoniánu, který bude podrobně vysvětlen v následující sekci. Tento hamiltonián je poté důležitý pro optimalizační algoritmy založené na principech fyziky, jelikož hamiltonán je tvořen potenciální energií, která odpovídá účelové funkci, a kinetickou energií, která odpovídá modifikacím účelové funkce. Tato kinetická energie je postupně snižována v průběhu algoritmu tak, že celková energie se rovná pouze potenciální energii, a tedy účelové funkci.

Hamiltonián

Hamiltonovská formulace mechaniky je jiný než popis Newtonovými zákony, který je v mnoha případech výhodnější pro naši práci se systémem.

Mějme tedy systém částic a těles. Poté můžeme uvažovat systém konfigurací $\mathcal{S}$ popisující tento systém. Každý bod z $\mathcal{S}$ reprezentuje určité uspořádání všech částic a těles v prostoru. Tento systém $\mathcal{S}$ bude mít vyšší dimenzi N, která například pro n částic v 3D prostoru bude mít dimenzi $\mathcal{S}$ rovnu $3n$, zachycující pozici každé z n částic ve všech 3 souřadných osách. Poté bod $q \in \mathcal{S}$ (reprezentující určité uspořádání systému) se bude pohybovat na základě úplně definovaných zákonů popisujících klasické chovaní tohoto systému.

Pokud se na $\mathcal{S}$ díváme jako na lagrangovský systém (systém ke studiu pohybu a změn energií ve klasické fyzice), tak tyto zákony tvoří skalární Lagrangeovu funkci $\mathcal{L}$, kde tato funkce je kinetická energie, od které je odečtena potenciální energie. Poté bod $q \in \mathcal{S}$ (reprezentující náš systém) se bude pohybovat podél křivky C, kde princip nejmenší akce [18] nám říká, že pohyb q po křivce C bude minimální. Tedy se můžeme na tuto křivku C dívat jako na nejkratší cestu mezi počátečním a koncovým místem bodu q.

Pokud se na tento systém díváme jako na hamiltoniánský systém, tak definovaná funkce $\mathcal{H}$ se nazývá Hamiltonova funkce nebo hamiltonián definovaný na fázovém prostoru $\mathcal{P}$ (prostor obsahující všechny možné stavy systému, kde každý stav má vlastní bod v tomto prostoru). Tato funkce je součet kinetické a potenciální energie systému, tedy celkové energie systému vyjádřené polohou a hybností.

Nechť souřadnice bodu $x \in \mathcal{S}$ jsou zobecněné souřadnice: $x_1, \ldots, x_N$ a těmto souřadnicím odpovídají zobecněné hybnosti: $p_1, \ldots, p_N$. Poté bod p ve fázovém prostoru $\mathcal{P}$ odpovídá $x \in \mathcal{S}$, kde $(x, p) = (x_1, \ldots, x_N, p_1, \ldots, p_N)$ a hodnota hamiltoniánu je $\mathcal{H}(x, p) = \mathcal{H}(x_1, \ldots, x_N, p_1, \ldots, p_N)$. Tento hamiltonán nám dává hamiltonovské kanonické rovnice:

$$\frac{\mathrm{d}p_i}{\mathrm{d}t} = -\frac{\partial \mathcal{H}}{\partial x_i}, \qquad \frac{\mathrm{d}x_i}{\mathrm{d}t} = \frac{\partial \mathcal{H}}{\partial p_i}, \qquad i = 1, \ldots, N, \tag{3.6}$$

popisující náš originální systém jako trajektorii bodu ve fázovém prostoru $\mathcal{P}$.

3.1.2 Termodynamika a statistická mechanika

Algoritmus simulovaného žíhání je inspirován procesem žíhání, který pochází z metalurgie. Při tomto procesu je kov zahřátý na žíhací teplotu a pomalu ochlazován. Díky tomuto procesu se kov překrystalizuje a sníží se počet atomů, které jsou špatně umístěny v krystalické mřížce kovu. Změny umístění atomů působí změny v kujnosti a tvrdosti kovu. Jelikož je simulované kvantové žíhání inspirováno kvantovou fyzikou, a to především klasickým simulovaným žíháním, tak zde bude uveden základ pro klasické žíhání.

Termodynamika

Hlavní myšlenka použití termodynamiky k optimalizaci vychází ze 2. termodynamického zákona. Tento zákon říká, že izolovaný systém, který je ponechán k samovolnému vývoji ne-

může snížit svoji entropii a vždy skončí ve stavu termodynamické rovnováhy, kde je entropie nejvyšší. Pokud jsou všechny procesy v systému vratné, tak je entropie konstantní [51].

K pochopení tohoto principu je nejprve nutné popsat termodynamický systém a vysvětlit, co jsou to makro-stavy tohoto systému.

Termodynamický systém je systém s velkým počtem rozlišitelných částic. Tyto částice se buď pohybují volně v systému, což odpovídá plynům a kapalinám, anebo jsou v pevném skupenství, kde částice mohou pouze vibrovat kolem své střední hodnoty umístění. Celý tento systém se poté skládá ze dvou energií: kinetická energie a potenciální energie. Kinetická energie systému je dána náhodným pohybem částic a potenciální energie je určena vzájemnou polohou částic. Uvažujeme o tomto systému v čase jako o stavu (mikro-stav) ve fázovém prostoru $\mathcal{P}$. Rozdělení do makro-stavů je podle rozlišitelnosti prvků v těchto mikro-stavech. Tedy pro všechny prvky z $\mathcal{P}$ z makro-stavu M platí, že nemůžeme rozlišit jejich teplotu, tlak a jiné veličiny z důvodu přesnosti měření.

Entropie, značená S, našeho systému, je definována za pomoci Boltzmannovy konstanty k_B a objemu makro-stavu v $\mathcal{P}$, značeného Ω, ve kterém se nachází mikro-stav $x \in \mathcal{P}$:

$$S = k_B ln(\Omega). \tag{3.7}$$

Na základě této definice entropie a makro-stavu můžeme definovat pojem termodynamická rovnováha jako nejvíce pravděpodobný makro-stav ve fázovém prostoru $\mathcal{P}$. Aby tento makro-stav byl nejvíce pravděpodobný v prostoru $\mathcal{P}$, tak z rovnice (3.7) vychází, že nejvyšší pravděpodobnost nastane, bude-li tento makro-stav mít největší objem, a tedy tento makro-stav bude mít i nejvyšší entropii.

Teplota T systému je měřena jako průměrná teplota na stupeň volnosti. Důležitá vlastnost termodynamického systému je z pohledu na naši optimalizaci, že pokfud systém není v termodynamické rovnováze, tak teplota v různých částech systému se bude lišit. Energie v nějakých místech bude vyšší/nižší než v ostatních. Tato nevyváženost dává teplotě potenciál a vytváří vektorové pole gradientů. Na základě působení tohoto pole vzniká proud tepelné energie, a tedy systém koná práci, jak je definováno v podkapitole 3.1.1 v odstavci o kinetické energie tělesa. Postupně, jak se tepelná energie pohybuje na základě potenciální energie, tak se snižuje velikost gradientu vektorového pole. Pohyb tepelné energie probíhá do doby, dokud systém není v termodynamické rovnováze. Poté je v tomto stavu tepelná energie vyvážena, a tak neexistují rozdíly teplot, gradienty jsou nulové a systém již nemůže konat práci.

Nyní můžeme vysvětlit myšlenku zmíněnou v prvním odstavci. Bylo řečeno, že entropie systému se musí vždy pouze zvyšovat a nelze ji snížit. Tedy systém po určitém čase vždy dosáhne maximální entropie. Makro-stav, který reprezentuje nejvyšší entropii systému, musí být nejvíce pravděpodobný (nejvyšší objem ve fázovém prostoru $\mathcal{P}$) makro-stav, jak vychází z rovnice entropie (3.7). Dále bylo uvedeno, že nejvíce pravděpodobný stav termodynamického systému je systém v termodynamické rovnováze nebo blízko této rovnováhy. Takže víme, že termodynamický rovnovážný systém má nejnižší potenciální energii. Tedy z 2. termodynamického zákona vyplývá, že v průběhu času se termodynamický systém musí blížit termodynamické rovnováze, která má nejnižší potenciální energii.

Statistická mechanika

Statistická mechanika uvažuje, že každému mikro-stavu x termodynamického systému odpovídá hamiltonián $\mathcal{H}_x$. Pravděpodobnost, že systém je v mikro-stavu x, je přímo úměrná

Boltzmannovu faktoru:

$$Boltzmann\ factor = e^{-\beta \mathcal{H}_x}, \tag{3.8}$$

kde β je inverzní teplota:

$$\beta = \frac{1}{k_B T}. \tag{3.9}$$

Poté pravděpodobnost $P(x)$ je rovna:

$$P(x) = \frac{1}{Z} e^{-\beta \mathcal{H}_x}, \tag{3.10}$$

kde Z je normalizační konstanta:

$$Z = \sum_x e^{-\beta \mathcal{H}_x}. \tag{3.11}$$

Tato pravděpodobnost je součástí simulovaného žíhání za použití metody Monte Carlo, kde se umožňuje přejít do stavu s vyšší potenciální energií s určitou pravděpodobností danou pravě tímto vztahem: (3.10). Podrobnější popis simulovaného žíhaní a metody Monte Carlo bude popsán v sekci 3.3 o simulovaném klasickém žíhání .

3.1.3 Klasický Isingův model spinových skel

Přestože tato kapitola je v části klasické fyziky, tak nám později pomůže pochopit simulované kvantové žíhaní, kde tento model je rozšířen o kvantové jevy a použit pro kvantové simulované žíhání. Nicméně, tento model lze použít i pro vysvětlení konstrukce klasického simulovaného žíhání.

Isingův model je diskrétní seskupení spinů částic, které mohou mít hodnotu spinu buď 1 nebo -1. Tyto spiny tvoří pravidelnou mřížku a spiny S_i na sebe vzájemně působí potenciální energií J_{ij}. Tato energie může být buď záporná nebo kladná, podle orientace spinů, tedy jestli jsou spiny souhlasné nebo ne. Celková energie je zapsána pomocí hamiltoniánu:

$$\mathcal{H} = -\sum_{i>j}^{N} J_{ij} S_i S_j, \tag{3.12}$$

kde N je počet spinů v modelu a i, j značí indexy v hamiltoniánu.

Tento hamiltonián bude později hrát roli naší účelové funkce, kterou se budeme snažit minimalizovat, a tedy najít optimum našeho problému.

Systém těchto spinů lze také reprezentovat neorientovaným váženým grafem, kde každý spin je vrchol grafu, a pokud je J_{ij} nenulové, tak mezi vrcholem i a vrcholem j existuje hrana. Tento model popisuje, jak se spiny orientují, aby celý model dosáhl nejnižší energie. Podrobný popis a pseudo algoritmus bude uveden v sekci 3.3 popisující klasické simulované žíhání.

3.2 Optimalizace založená na kvantové fyzice

V předchozích částech bylo uvedeno z čeho vychází inspirace pro klasické simulované žíhání, které vychází pouze z termodynamických zákonů a klasické statistické mechaniky. V následujících podkapitolách bude popsána velice obecně kvantová mechanika, která je rozšířením

klasické mechaniky o děje, které probíhají buď na velice krátké vzdálenosti nebo v krátkých časových úsecích.

Tato rozšíření klasické mechaniky vytvořila myšlenku také rozšířit klasické simulované žíhání o kvantové jevy a pokouší se o lepší vlastnosti tohoto optimalizačního algoritmu. Tento upravený algoritmus se nazývá simulované kvantové žíhání. Nebo pouze kvantové žíhaní, pokud výpočet algoritmu skutečně probíhá na kvantových počítačích, kde není zapotřebí tyto kvantové jevy simulovat.

3.2.1 Kvantová mechanika

Kvantová mechanika vychází z klasické fyziky za pomoci procesu zvaného kvantování. Tento proces umožňuje z fyzikálního pole vytvořit kvantové pole, které popisuje částice jako kvanta v těchto polích a vzájemné působení částic je popsáno za pomoci Lagrangiánu.

Tedy kvantování reprezentuje pozorovatelné fyzikální vlastnosti (poloha, hybnost, energie, spin, atd.) jako lineární operátor ve vektorovém prostoru, kterým je většinou Hilbertův prostor nad komplexními čísly. Prvek Ψ v Hilbertově prostoru popisuje kvantový stav systému, který se nazývá vlnová funkce.

3.2.2 Pozorovaní kvantových jevů

Klasická interpretace kvantových jevů, která se obvykle vyučuje je Kodaňská interpretace. Podle této interpretace fyzika (a celý svět) je statistický, tedy svět není úplně předvídatelný [17]. Z toho vyplývá, že na mikroskopické úrovni objekty nemají přesně definované vlastnosti, ale kvantová mechanika dokáže pouze předvídat pravděpodobnostní rozložení těchto vlastností. Hustotu pravděpodobnosti těchto vlastností pro stav systému Ψ popisuje Schrödingerova rovnice:

$$i\hbar\frac{\partial}{\partial t}\Psi(t) = \mathcal{H}\Psi(t), \tag{3.13}$$

kde i je imaginární jednotka, $\hbar$ je redukovaná Planckova konstanta[1] [9], Ψ je vektor kvantového stavu, t je čas a $\mathcal{H}$ je hamiltoniánský operátor. Takže tato rovnice popisuje časový průběh hustoty pravděpodobnosti vlnové funkce.

Nicméně, v momentu pozorování vlnové funkce, se toto pravděpodobnostní rozložení redukuje na jednu hodnotu, která odpovídá určitému stavu (tzv. vlastní hodnotě operátoru $\mathcal{H}$) z klasické fyziky. Tento děj se nazývá kolaps vlnové funkce. Před tímto pozorováním systém nazýváme, že je ve superpozici stavů, tedy lineární kombinaci všech možných klasických stavů. Potom se v době pozorování "vybere" jeden stav na základě hustoty pravděpodobnosti ze Schrödingerovy rovnice (3.13). Pravděpodobnost polohy částice je nejvyšší v absolutní hodnotě komplexní vlnové funkce $|\Psi|^2$. Jelikož by se částice měla nacházet někde v prostoru, tak používáme normalizovanou vlnovou funkci [49]:

$$\int_{-\infty}^{+\infty} |\Psi(x,t)|^2 dx = 1. \tag{3.14}$$

3.2.3 Koherence a dekohorence

Před vysvětlením kvantové koherence a dekohorence je nejprve nutné vysvětlit, co je myšleno měřením/pozorováním stavu systému. Z pohledu kvantové mechaniky není pozorování myšleno, že jej nějaký organismus musí vidět anebo někam zaznamenat. Pozorování je

[1] $h = 6.62607015 \times 10^{-34}$ J s, $\hbar = h/2\pi$

definováno na mnohem nižší úrovni. Tedy, aby částice byla pozorována okem, přístrojem nebo jiným pozorovatelem, tak vždy dochází k interakci s tímto systémem. Například, pro pozorování systému je nutné, aby se nějaký foton odrazil od částice v systému a potom měříme nějakou vlastnost na základě tohoto fotonu. Právě touto interakcí fotonu (nebo jiných subatomárních částic) vzniká pozorování a to způsobí kolaps vlnové funkce popisující tento systém.

Jak již bylo uvedeno, tak kvantové částice jsou popsány vlnovou funkcí, která popisuje kvantový stav systému. Dokud existuje fázová relace mezi různými stavy systému, tak říkáme, že tento systém je koherentní. Pokud je tento systém dokonale izolovaný, a tedy žádné vnější pozorování tohoto systému není možné, tak tento systém bude stále koherentní. [33]

V případě, že budeme uvažovat nějaký neizolovaný systém, jako například zrnko prachu, tak se od něho mohou odrážet fotony, a tedy se provede měření. Na základě tohoto měření se prostředí "dozví" o tomto systému a způsobí kolaps vlnové funkce tohoto systému. Tento jev se nazývá kvantová dekoherence, kde původní koherenci tohoto prostředí již nelze pozorovat. Nicméně koherence systému není "zničena", ale pouze přemístěna do většího systému, kde se fotony (nebo jiné částice, které provedly měření/pozorovaní tohoto původně koherentního systému) nachází. [33]

3.2.4 Kvantové provázání

Kvantové provázání je jeden z jevů kvantové mechaniky mající velký rozdíl oproti klasické mechanice a pravděpodobně nejvíce překvapující. Uvažujme vlnovou funkci systému tvořenou mnoha částicemi. Tato vlnová funkce popisuje hustotu pravděpodobnosti na základě výše uvedené Schrödingerovy rovnice (3.13), dokud nenastane pozorování. Zajímavá vlastnost kvantového provázání je, že tuto vlnovou funkci systému nelze rozložit na nezávislé vlnové funkce pro jednotlivé částice. Tedy měření i pouze jediné částice způsobí kolaps vlnové funkce celého systému. V tento moment už nejsou provázané částice popsány pravděpodobností, ale všechny tyto částice jsou ve vlastním specifickém stavu. Pozoruhodná vlastnost je, že kvantové provázání není závislé na vzdálenosti částic v systému.

Nicméně jev kvantového provázání systému přináší problémy pro výpočet, jelikož správná dimenze Hilbertova prostoru pro m částic v n různých polohách odpovídá m^n oproti $m \times n$, jak je tomu v klasické fyzice. Z tohoto důvodu je velice obtížné simulovat kvantové jevy (alespoň na klasickém počítači) pomocí analytických metod. Na druhou stranu tento jev dal vzniknout myšlence kvantového počítání, jelikož kvantový procesor implicitně zahrnuje kvantové provázání. Tento jev umožňuje počítat v dimenzi m^n oproti klasickým procesorům v dimenzi pouze $m \times n$. Tedy by mohlo jít o urychlení některých výpočtů, pokud existuje efektivnější algoritmus, který řeší danou úlohu ve kvantovém prostoru. [7]

3.2.5 Heisenbergův princip neurčitosti

Princip neurčitosti byl prvně popsán německým fyzikem Wernerem Heisenbergem v roce 1927 v článku [28]. Tento princip neurčitosti říká, že čím přesněji určíme polohu nějaké částice, tím méně přesně můžeme zjistit její hybnost z původních podmínek a naopak. Tuto přesnost nelze ani zlepšit lepšími měřicími přístroji. Později Earle Hesse Kennard [32] a Hermann Weyl [54] formálně vytvořili nerovnici, která udává vztah mezi směrodatnou odchylkou měření polohy σ_x a směrodatnou odchylkou měření hybnosti σ_p:

$$\sigma_x \times \sigma_p \geq \frac{\hbar}{2}, \tag{3.15}$$

kde $\hbar$ je redukovaná Planckova konstanta: $\frac{h}{2\pi}$.

3.2.6 Kvantové tunelování

Klasické simulované žíhání funguje na principu termodynamických zákonů, které byly uvedeny v předchozích podkapitolách. Konvergence tohoto přístupu k optimalizaci je zaručena na základě těchto termodynamických zákonů. Díky vlastnosti vycházející z těchto zákonů klasické žíhání překonává bariéry mezi lokálními extrémy za pomoci hybnosti vznikající ze sil tepelné nevyváženosti systému. Tento princip umožňuje konvergenci ke globálnímu extrému. Simulované kvantové žíhání nepoužívá tuto vlastnost k dosažení konvergence, ale místo toho využívá jevů, které vycházejí z kvantové mechaniky. Jeden z těchto jevů, kterým je inspirováno simulované kvantové žíhání, se nazývá kvantové tunelovaní.

Efekt kvantového tunelování umožňuje vlnové funkci částice překonat potenciálovou bariéru. Míra tohoto přenosu skrze bariéru závisí na její výšce a šířce. Tento kvantový jev může být vysvětlen za pomoci Heisenbergova principu neurčitosti. Kvantová částice může být vyjádřena jako její vlnová funkce. Z principu neurčitosti vyplývá, že poloha této částice v žádném bodě nemůže mít pravděpodobnost výskytu 100 %, ale nemůže být ani nulová. Vzniká tedy určitá pravděpodobnost, že se částice nachází za danou bariérou. Tato pravděpodobnost je dána šířkou a výškou bariéry, kterou se tento kvantový jev snaží překonat, na rozdíl od termodynamického jevu využitého pro klasické žíhání, kde tato pravděpodobnost závisí pouze na výšce této bariéry. Tento rozdíl je důvodem zkoumání simulovaného kvantového žíhání, jelikož za pomocí kvantového tunelování může částice projít velmi vysokou, ale úzkou bariérou snadněji než klasické simulované žíhání umožnující efektivnější prohledávání v určitých stavových prostorech.

3.2.7 Kvantový hamiltonián

Analogicky z klasické mechaniky je hamiltonián typicky vyjádřen jako operátor celkové energie systému odpovídající součtu kinetických a potenciálních energií systému. V našem případě budeme používat hamiltonián vyjadřující energii systému jako:

$$\mathcal{H} = \mathcal{H}_{pot} + \mathcal{H}_{kin}, \tag{3.16}$$

kde $\mathcal{H}_{pot}$ je potenciální energie systému odpovídající naší účelové funkci a $\mathcal{H}_{kin}$ je kinetická energie systému odpovídající rozrušení, které reprezentuje kvantové tunelování k překonání lokálních extrémů.

3.2.8 Kvantový Isingův model spinových skel

Pro výpočet simulace kvantového žíhání se tento problém obvykle mapuje na kvantový Isingův model spinových skel. Podrobnější důvod pro toto mapování bude uveden později v sekci 3.4 o simulovaném kvantovém žíhání. Základní myšlenka je, že při použití Suzuki-Trotterovy transformace přímo na náš problém vznikne rovnice, která je obtížně řešitelná na klasickém procesoru [7]. Nicméně, reprezentováním naší úlohy za pomoci Isingova modelu spinových skel, můžeme použít transformaci na tento model, který je výrazně jednodušší na výpočet.

Z klasického modelu spinových skel víme, že:

$$\mathcal{H} = -\sum_{i>j}^{N} J_{ij} S_i S_j, \tag{3.17}$$

kde N je počet spinů v modelu.

K tomuto hamiltoniánu je poté přidána kvantová složka. Tato složka je parametr určující sílu tunelovacího pole značený Γ a způsobuje rozrušení pro náš kvantový hamiltonián. Výsledný hamiltonián tedy bude mít podobu:

$$\mathcal{H} = -\sum_{i>j}^{N} J_{ij} S_i^z S_j^z - \Gamma \sum_{i=1}^{N} S_i^x = \mathcal{H}_{pot} + \mathcal{H}_{kin}, \tag{3.18}$$

kde pro potenciální energii jsou spiny zarovnány v ose z a kde pro kinetickou energii jsou spiny zarovnány v ose x a působí navzájem silovým polem určeným parametrem Γ.

Tento hamiltonián lze poté triviálně transformovat z d-dimenzního kvantového prostoru do (d+1)-dimenzního klasického prostoru za pomoci Suzuki-Trotterovy transformace.

3.3 Simulované klasické žíhání

V této kapitole bude popsán algoritmus a myšlenka klasického simulovaného žíhání. Důvod podrobnějšího popisu tohoto algoritmu, mimo to, že je základ pro simulované kvantové žíhání, je použití klasického simulovaného žíhání k porovnání s kvantově inspirovaným pro problém obchodního cestujícího.

Taktéž k lepšímu pochopení kvantové varianty je záhodno pochopit princip jednoduššího klasického nekvantového algoritmu vycházejícího z termodynamických zákonů a statistické mechaniky.

3.3.1 Myšlenka klasického simulované žíhání

Z inspirace z metalurgie a statistické mechaniky vznikla myšlenka pro vytvoření optimalizačního algoritmu simulovaného žíhání. Původní pojem žíhání z metalurgie označuje proces zahřátí kovu na vysokou teplotu a postupné ochlazování. Při tomto procesu se mění krystalická mřížka kovu a získává odlišné vlastnosti. Důvod, proč tento jev nastává, vysvětluje statistická mechanika, jak bylo popsáno v kapitole 3.1.2. Samotný algoritmus pro tuto optimalizaci vychází z horolezeckého algoritmu [4], jevů z termodynamiky a statistické mechaniky. Díky těmto zákonům systém, který je dostatečně pomalu ochlazován, postupně konverguje k minimální celkové energii systému, a tedy nalezení optima řešení.

3.3.2 Horolezecký algoritmus

Na horolezecký algoritmus (HA) by se mohlo nahlížet jako na předchůdce simulovaného žíhání, jelikož hlavní myšlenka tohoto algoritmu je stejná s algoritmem pro simulované žíhání a jeho i kvantovou variantou. Nicméně tento algoritmus zaručeně konverguje ke globálnímu extrému jen ve speciálních případech, kdy prostor problému je monotónně rostoucí nebo klesající, tedy bez lokálních extrémů. Simulované žíhání a kvantová varianta konverguje k optimu i přes výskyt lokálních extrémů, pokud jsou splněny určité předpoklady, které budou uvedeny u zmíněných algoritmů.

Samotný horolezecký algoritmus funguje velice jednoduše. Jde o techniku prohledávání lokálního prostoru, která směruje k lokálnímu minimu nebo maximu v závislosti na řešeném problému a v tomto extrému se zastaví. Z principu jde o zvolení pozice v prostoru a optimalizace se provádí zvolením nového náhodného řešení v okolí současného řešení. Pokud toto nové řešení má lepší hodnotu účelové funkce, tak se přesune do tohoto stavu, jinak zůstává v aktuálním stavu. V obou případech se pokračuje zvolením dalšího náhodného řešení a tímto způsobem se prochází stavový prostor dokud se nedosáhne ukončovací podmínky. Tato podmínka muže být maximální počet kroků/pokusů o přesun a po dosažení tohoto počtu algoritmus zastaví a vrátí nalezené řešení. Jiná varianta, pokud počet sousedů není extrémní počet, tak lze skončit, pokud žádný soused aktuálního stavu nemá lepší hodnotu účelové funkce, a tedy už se nelze nikam přesunout. Pseudo kód HA algoritmus je tedy následující:

Algorithm 3: Horolezecký algoritmus pro minimalizaci

```
Vstup  : Náhodné řešení problému
Výstup: Nejlepší nalezené řešení
aktuální_stav = počáteční_stav
Loop Ukončující podmínka     //Hlavní smyčka
    aktuální_fitness = ohodnocení řešení aktuální_stav
    nové_řešení = nové řešení z aktuální_stav
    foreach řešení v množine nové_řešení do
        nová_fitness = ohodnocení řešení řešení
        if sousedící_fitness < nová_fitness then
            aktuální_stav = řešení
            continue hlavní smyčka
        end
    end
    break    //žádné lepší řešení z aktuálního stavu neexistuje
EndLoop
return aktuální_stav
```

3.3.3 Popis simulovaného žíhání

Jelikož přímý výpočet simulovaného žíhání je výpočetně složitý, tak využíváme metody typu Monte Carlo. Monte Carlo je název pro skupinu algoritmů, které vyžívají opakovaného vzorkovaní k odhadu neznámé proměnné. Tuto metodu použijeme k výpočtu simulovaného žíhání. Ze zákona velkých čísel poté vyplývá, že při dostatečném počtu vzorků se bude průměr těchto vzorků blížit k reálnému průměru, odkud byly vzorky získávány [14].

Funkce algoritmu je podobná horolezeckému algoritmu, ale na rozdíl od této metody simulované žíhání má schopnost dostat se z lokálních extrémů a konvergovat ke globálnímu

(hledanému) extrému. Důvod, proč tato konvergence může existovat, je, že algoritmus simulovaného žíhání umožňuje přijmout i stav s horší hodnotou účelové funkce. Tato pravděpodobnost je odvozena ze statistické mechaniky a závisí na teplotě systému a rozdílu hodnot účelové funkce:

$$P(x) = e^{-\beta \mathcal{H}_x}, \tag{3.19}$$

kde β je inverzní teplota a $\mathcal{H}_x$ je potenciální energie hamiltoniánu, v našem případě účelové funkce problému.

Jelikož se jedná o pravděpodobnost a tedy náhodné přechody, tak lze využít Monte Carlo metodu k optimalizaci za pomoci simulovaného žíhání. Při této optimalizaci se mnohokrát zkouší přejít do sousedních stavů. Tento přechod se provede, pokud má nový testovaný stav lepší hodnotu účelové funkce nebo na základě pravděpodobnosti udávané z rovnice (3.19).

Systém začíná s určitou teplotou (inversní hodnota β) a při postupném zkoušení přechodů se snižuje teplota tohoto systému na základě žíhacího plánu systému.

Pro toto ochlazování systému platí teorém, že pokud budeme systém dostatečně pomalu ochlazovat, tak simulované žíhání bude konvergovat ke globálnímu minimum ne rychleji než $\frac{N}{log(i)}$. Kde N je velikost systému a i je uplynutý čas. [21]

Nicméně, toto ochlazování je příliš pomalé pro praktické použití a v praxi se používá výrazně rychlejší ochlazování, které ale nemusí vždy vést k nalezení globálního extrému. Pseudo kód algoritmu pro simulované žíhání je poté následující:

Algorithm 4: Simulované žíhání pro minimalizaci

Vstup : Náhodné řešení problému
Počáteční teplota
Konečná teplota
Žíhací plán
Výstup: Nejlepší nalezené řešení
aktuální_stav = *počáteční_stav*
aktuální_fitness = ohodnocení řešení *aktuální_stav*
aktuální_teplota = Počáteční teplota
while *aktuální_teplota* > *konečná_teplota* **do**
  soused = náhodné sousedící řešení z *aktuální_stav*
  sousedící_fitness = ohodnocení řešení *soused*
  if *sousedící_fitness* < *aktuální_fitness* **then**
    aktuální_stav = *soused*
    aktuální_fitness = *sousedící_fitness*
  end
  else if $e^{-((sousedici\_fitness - aktualni\_fitness)/aktualni\_teplota)} \geq$ *náhodně z* $U(0,1)$[1]
  then
    aktuální_stav = *soused*
    aktuální_fitness = *sousedící_fitness*
  end
  aktuální_teplota = žíhací_plán(*aktuální_teplota*)
end
return *aktuální_stav*

[1] funkce uniformního rozložení pseudonáhodných čísel z intervalu

3.4 Simulované kvantové žíhání

Pro tuto práci byl vybrán algoritmus simulovaného kvantového žíhání (Simulated Quantum Annealing - SQA) z kategorie optimalizačních kvantových algoritmů. Simulované kvantové žíhání je inspirováno z jevů kvantové mechaniky a staví na již existujícím algoritmu klasického simulovaného žíhání.

3.4.1 Simulované kvantové žíhání

Při hledání, jak urychlit optimalizační algoritmy prohledávající lokální stavový prostor, vznikla myšlenka upravit klasické simulované žíhání podle principů kvantové mechaniky. Touto myšlenkou je využít vlastnost superpozice, tedy že systém je ve všech možných stavech naráz, a kvantového tunelování k úniku z lokálních extrémů a lepší konvergence ke hledanému globálnímu extrému. Myšlenka tedy je vyjádřit úlohu ve kvantovém prostoru za pomoci časově závislé Schrödingerovy rovnice. Při tomto časovém vývoji systém realizuje kvantový paralelismus a působením kvantového pole umožňuje jev kvantového tunelování v tomto systému. Pokud rychlost změny tohoto pole je dostatečně pomalá, tak tento kvantový systém bude setrvávat blízko základního stavu, tedy stavu, kdy systém má nejnižší celkovou energii [16]. Tento kvantový systém při dostatečně pomalé změně kvantového pole bude konvergovat ke stavu s nejnižší energií. Proto je využit v kvantovém simulovaném žíhání namapováním naší účelové funkce na potenciální energii systému. Tudíž jak se bude kvantový systém blížit základnímu stavu, tak se bude snižovat (při snaze najít globální minimum) hodnota účelové funkce, a tedy bude docházet k optimalizaci.

Tato myšlenka postačuje na implementaci kvantového žíhání, které by mohlo být spuštěno na adiabatickém kvantovém počítači vycházejícím z adiabatického teorému [15]. Nicméně, v této práci se budeme zabývat pouze inspirací z těchto jevů a výsledný algoritmus bude popsán pro klasické, tedy nekvantové, počítače. Z tohoto důvodu je nutné nejprve uvést pojmy a postupy, kdy je možné tyto kvantové jevy simulovat na klasických počítačích.

3.4.2 Suzuki-Trotterova transformace

Jak již bylo uvedeno, tak výše popsaný postup je proveditelný pouze na kvantových počítačích, jelikož klasický počítač neumožňuje využívat výpočty s kvantovými vlastnostmi bez velmi výrazného zpoždění výpočtů. Z tohoto důvodu se provádí pouze simulace těchto jevů za pomoci numerických metod, jelikož analytické řešení by bylo vypočítat Schrödingerovu rovnici hamiltoniánu, který popisuje náš systém. Tento výpočet je příliš složitý pro použití na klasickém počítači v přijatelném čase. Na místo toho tyto rovnice transformujeme pomocí Suzuki-Trotterovy transformace a použijeme metody z rodiny Monte Carlo.

Suzuki-Trotterova transformace ze zabývá transformací Schrödingerovy rovnice hamiltoniánu na aproximaci, kterou lze jednodušeji vypočítat na klasickém (nekvantovém) počítači [30]. Přímé využití této transformace v této práci nebude, ale budeme používat rovnici, která vznikla pomocí této transformace.

3.4.3 Monte Carlo založené na křivkovém integrálu

Pro numerickou simulaci za pomoci vzorkování musíme použít jinou metodu Monte Carlo, než jsme použili u klasického simulovaného žíhání, jelikož tato metoda neumožňuje simulaci kvantových jevů. Z tohoto důvodu použijeme metodu Monte Carlo, která umožňuje simu-

lovat kvantové jevy, pokud se Planckova konstanta nerovná nule (při rovnosti Planckovy konstanty nule by se jednalo o klasickou fyziku).

Tato metoda počítá součet všech cest mezi počátečním a koncovým stavem. Každá tato cesta přispívá fází $exp(iS/\hbar)$, kde S je nekvantová akce vyhodnocená podél cesty částice. Tyto sečtené fáze tvoří superpozici tohoto systému. Jelikož součet přes všechny spojité cesty je typicky proveden integrací, tak se tato metoda nazývá Monte Carlo založené na křivkovém integrálu (Path Integral Monte Carlo - PIMC). Jednou z vlastností této metody je, že umožňuje kvantové tunelování, a tedy je to vhodná metoda pro simulaci kvantového žíhání, které je inspirováno právě kvantovým tunelováním pro překonávání lokálních extrémů. [52]

Již výše uvedená rovnice pro kvantový model Isingových spinových skel má hamiltonián:

$$\mathcal{H} = -\sum_{i>j}^{N} J_{ij} S_i S_j - \Gamma \sum_{i=1}^{N} S_i^x = \mathcal{H}_{pot} + \mathcal{H}_{kin}. \tag{3.20}$$

Aproximace tohoto hamiltoniánu za pomoci Suzuki-Trotterovy transformace na klasický simulační model je [13], [47]:

$$\mathcal{H} = \frac{\mathcal{H}_{pot}}{P} - J_\Gamma \Delta \mathcal{H}_{kin}. \tag{3.21}$$

Tato metoda pro systém o dimenzi D a malé konečné teplotě spočívá v namapování na (D+1) dimenzní systém s P kopiemi. Tyto kopie původního systému se nazývají Trotter repliky. Přidaná dimenze se nazývá imaginární čas. Každá replika má částice se spiny jako v původním systému a tyto spiny jsou provázány s odpovídajícími spiny v replikách. Tato provázanost je rovna [47]:

$$J_\Gamma = -\left(\frac{T}{2}\right) ln(tanh\left(\frac{\Gamma}{PT}\right)), \tag{3.22}$$

kde P je počet replik, T je teplota systému, Γ je síla kvantového pole. $P \times T$ se také nazývá efektivní kvantová teplota [5].

V průběhu výpočtu je Γ snižována k nule a ze vzorců vyplývá, že síla, kterou jsou jednotlivé repliky navzájem ovlivňovány, se zvyšuje a nakonec donutí všechny repliky, aby byly stejné. Tedy jejich spiny budou orientovány stejným směrem. Při dostatečně pomalém snižování Γ a dostatečném počtu replik by se v čase, kdy jsou již všechny repliky identické, systém měl nacházet v rovnovážném stavu. Tedy ve stavu, kdy systém má minimální celkovou energii a globální minimum/maximum z pohledu naší účelové funkce.

3.4.4 Aplikace teoretických poznatků do algoritmu

Z teoretických základů pro simulované kvantové žíhání a jeho aplikace na kvantový Isingův model spinových skel lze vyřešit problémy vyžadující optimalizační přístup k nalezení řešení. Při použití poznatků a inspirací z kvantového světa můžeme upravit klasické simulované žíhání na variantu, která bude využívat myšlenky z kvantové mechaniky.

Využití poznatků z kvantové fyziky nelze přímo použít k vytvoření algoritmu k řešení optimalizačních problému na klasickém počítači. Nejprve se musí vyřešit Suzuki-Trotterova transformace tohoto problému, aby se jednalo o simulaci kvantového jevu a nebyl potřeba kvantový počítač. K této transformaci je nutné vytvořit hamiltonián, který bude popisovat naše řešení a přechody mezi stavy problému. A na závěr bude použit již zmíněny algoritmus PIMC k výpočtu numerického řešení tohoto transformovaného hamiltoniánu. Tyto podúlohy budou postupně popsány v následujících sekcích s pseudo algoritmem pro řešení optimalizačních problémů.

Hamiltonián popisující problém

Simulované kvantové žíhání, které bylo popsáno výše, předpokládá, že systém, na který je tento princip aplikován, je popsán pomocí hamiltoniánu. Z tohoto důvodu je nejdříve nutné vytvořit funkci hamiltoniánu, který popisuje náš problém. Tento Hamiltonián pro optimalizační úlohu může být dán následujícím vztahem:

$$\mathcal{H} = \mathcal{H}_{pot} + \mathcal{H}_{kin}, \tag{3.23}$$

kde $\mathcal{H}_{pot}$ je naše účelová funkce systému a $\mathcal{H}_{kin}$ je rozrušení našeho systému, které reprezentuje kvantové tunelování k překonání lokálních extrémů.

Účelová funkce je závislá na řešeném problému, kde tato funkce popisuje ohodnocení řešení. Funkci se budeme snažit minimalizovat, a tedy by měla být klesající se zvyšující se kvalitou řešení.

Kinetická energie, která reprezentuje rozrušení z kvantového tunelovaní je závislá na způsobu, jakým se vybírají nová řešení. V praxi se jedná o rozdíl mezi dvěma řešeními. Pro ilustraci na řešení zadaném binárním číslem by se jednalo o počet rozdílných bitů mezi dvěma řešeními (Hammingova vzdálenost). Na příkladu níže by byla kinetická energie rovná číslu 3.

	Binární řešení úlohy						
Řešení 1	0	1	0	1	1	1	0
Řešení 2	1	0	0	0	1	1	0

Tabulka 3.1: Tabulka zobrazující kinetickou energii mezi dvěma řešeními, která se rovná číslu 3.

V původní myšlence inspirace kvantovými jevy by potencionální energie systému měla být vyjádřena formou, kterou je možné převést na Isingův model spinových skel. Tedy funkce, která počítá účelovou funkci z matice se spiny o hodnotě 1 nebo -1 popisující řešení problému.

Později u řešení problémů si ukážeme, že jelikož algoritmus se pouze inspiruje kvantovou mechanikou, tak tento požadavek není přímo nutný k implementaci algoritmu. Nicméně prozatím budeme uvažovat, že naše řešení problému je popsáno pomocí matice spinů S o velikosti $N \times N$.

Poté lze definovat potenciální energii definující naši účelovou funkci jako [7]:

$$\mathcal{H}_{pot} = \sum_{i,j=1}^{N} \left(d_{ij} \times \left(\frac{S_{ij} + 1}{2} \right) \right), \tag{3.24}$$

kde d_{ij} odpovídá změně ohodnocení řešení v závislosti na nastavení spinu s indexem i a j v matici S.

Z této rovnice vychází, že přičtení hodnoty d_{ij} k výsledné potenciální energii dojde pouze pokud je spin S_{ij} roven hodnotě 1 a né -1.

Suzuki-Trotterova transformace

K vypočítání tohoto hamiltoniánu, který popisuje náš kvantový systém za pomoci Monte Carlo metody na klasickém nekvantovém počítači, je nutné na tento hamiltonián použít

Suzuki-Trotterovu transformaci. Přestože je možné provést tuto transformaci, tak výsledná funkce by byla příliš složitá k výpočtu, a namísto toho se využívá jiná myšlenka pro mapování našeho problému na kvantový hamiltonián [7].

Nejprve zvolíme nějaký kvantový hamiltonián, který umíme jednoduše transformovat za pomoci Suzuki-Trotterovy transformace na jednodušeji vypočitatelný tvar. Model, který lze takto transformovat, je právě kvantový Isingův model spinových skel. Tento model má hamiltonovskou funkci:

$$\mathcal{H} = \mathcal{H}_{pot} + \mathcal{H}_{kin} = -\sum_{i>j}^{N} J_{ij} S_i S_j - \Gamma \sum_{i=1}^{N} S_i^x. \tag{3.25}$$

Aproximace tohoto hamiltoniánu již byla ukázána v podkapitole 3.4.3 o PIMC metodě a dostáváme aproximovanou rovnici:

$$\mathcal{H} = \frac{\mathcal{H}_{pot}}{P} - J_\Gamma \Delta \mathcal{H}_{kin}. \tag{3.26}$$

$$J_\Gamma = -\left(\frac{T}{2}\right) ln(tanh\left(\frac{\Gamma}{PT}\right)), \tag{3.27}$$

kde $\mathcal{H}_{pot}$ odpovídá naší účelové funkci, kterou se snažíme minimalizovat. $\mathcal{H}_{kin}$ odpovídá kinetické energii vzniklé působením kvantového pole na jednotlivé spiny. Při použití Isingova modelu ale nastává problém, že změny ve spinové matici nemusí vést na platné řešení problému. Tedy výsledné uspořádání spinu popisující řešení neodpovídá požadavkům na platné řešení dané úlohy. Tento problém může být vyřešen ve fázi Monte Carlo, kde budeme uvažovat pouze vzorky, které odpovídají správnému tvaru řešení. Příklad tohoto omezení na vzorky, které jsou platné pro danou úlohu, bude ukázán na řešení problému obchodního cestujícího dále v práci.

Využití metody Monte Carlo založené na křivkovém integrálu

Poslední krok ve výpočtu simulovaného kvantového žíhání je využít a sestrojit Monte Carlo metodu, která umožňuje využití kvantových jevů v numerickém výpočtu. K řešení našeho problému využijeme právě výše popisovanou metodu PIMC, která umožňuje numericky spočítat transformovaný hamiltonián pomocí Suzuki-Trotterovy transformace, a hlavně umožňuje využít jev kvantového tunelování.

Při použití takto transformované funkce je potřeba nezapomínat na důsledek transformace z kvantového prostoru do klasického. Tento převod nám způsobí přidání jedné dimenze prostoru, která se také nazývá imaginární čas a velikost tohoto rozměru je závislá na počtu Trotterových replik, které použijeme při našem výpočtu. Při použití pouze jediné repliky nebude možné využít kvantových jevů a tento systém by se choval jako klasický systém, tedy pouhý horolezecký algoritmus. Na druhou stranu, čím více replik využijeme tím blíže budeme simulovanému kvantovému prostoru a výpočet bude více odpovídat kvantovému systému. V praxi se jedná o kompromis mezi těmito extrémy, jelikož chceme využít kvantových vlastností k získání základního stavu, ale s počtem replik roste náročnost provedení jednoho kroku Monte Carlo metody. Spiny těchto replik se navzájem ovlivňují v závislosti na velikosti síly kvantového pole a v našem případě jsou ovlivněny pouze sousedními replikami. Výsledný pseudo kód algoritmu je uveden na následující straně.

Algorithm 5: Simulované kvantové žíhání pro minimalizaci (upraveno z [12])

Vstup : Zadání řešení problému
Teplota systému
Počáteční síla kvantového pole
Konečná síla kvantového pole
Plán změny kvantového pole
Počet Trotterových replik
Výstup: Nejlepší nalezené řešení
repliky = cyklická fronta pro repliky
for *i* od 0 to počtu replik **do**
  Přidej repliku s náhodným řešením problému do cyklické fronty *repliky*.
end
nejlepší = replika s nejlepší účelovou funkcí
replika_teplota = teplota × počet replik
současné_kvantové_pole = Počáteční síla kvantového pole
while *současné_kvantové_pole* > *ukončovací podmínka* **do**
  kvantová_teplota =
  $-replika\_teplota \times 0.5 \times ln(tanh(soucasne\_kvantove\_pole/replika\_teplota))$
  foreach *replika* v *repliky* **do**
    soused = náhodné sousedící řešení z *replika*
    levá_replika = replika nalevo od *replika* ve frontě *repliky*
    pravá_replika = replika napravo od *replika* ve frontě *repliky*
    potenciální_delta = rozdíl učelové funkce mezi *replika* a *soused*
    spin_delta_původní = rozdíl spinů *replika* and *levá_replika*
      + rozdíl spinů *replika* and *pravá_replika*
    spin_delta_soused = rozdíl spinů *soused* and *levá_replika*
      + rozdíl spinů *soused* and *pravá_replika*
    kinetická_delta = (*spin_delta_soused* − *spin_delta_původní*)
      ×*kvantová_teplota*
    energie_delta = *potenciální_delta* + *kinetická_delta*
    if *potenciální_delta* < *0 nebo energie_delta* < *0* **then**
      replika = *soused*
    end
    else if $e^{-((energie\_delta/replika\_teplota)} \geq$ *náhodně z* $(0, 1)$ **then**
      replika = *soused*
    end
    if účelová funkce *replika* je lepší než účelová funkce *nejlepší* **then**
      nejlepší = *replika*
    end
  end
  současné_kvantové_pole = změna_kvantového_pole(*současné_kvantové_pole*)
end
return *nejlepší*

Kapitola 4

Řešené problémy

V této kapitole bude představeno několik problémů, kterými se budeme zabývat v této práci ke srovnání existujících algoritmů se simulovaným kvantovým žíháním. Tyto problémy a algoritmy jsou následující.

Problém obchodního cestujícího, kde se budeme snažit nalézt nejkratší možnou cestu navštěvující každé město pravě jednou a zakončit tuto cestu v počátečním městě. Ke srovnání implementovaného simulovaného kvantového žíhání zde využijeme nekvantovou verzi stejného algoritmu, tedy klasické simulované žíhání.

Další vybraná úloha se týká rozšíření rozhodovacího problému řešeného v mnoha oblastech jako například v teoretické informatice [42] nebo kryptografii [40]. Jedná se o maximální splnitelnost booleovské formule. Tato úloha je rozšíření problému, zdali existuje ohodnocení proměnných formule takové, že je formule splněna. V této nerozšířené podobě se však nejedná o optimalizační problém a z toho důvodu budeme řešit maximální splnitelnost této formule, kde se snažíme najít takové ohodnocení formule, které bude splňovat maximální počet splnitelných klauzulí této formule.

Pro porovnání s kvantově inspirovaným algoritmem, v úloze hledání maximální splnitelnosti formule, bude využit algoritmus z oblasti algoritmů inspirovaných imunitním systémem: klonální selekce. Tento algoritmus a jeho inspirace již byla popsána v sekci 2.2 v kapitole o algoritmech inspirovaných přírodou.

Poslední úloha se týká hledání pravidel pro celulární automaty. V našem případě se bude jednat o celulární automat pracující na 2D mřížce. Tento automat mění buňky na základě stavu okolních buněk. Tyto změny jsou definovány pravidly celulárního automatu, kde každé pravidlo uvádí, na jaký stav buňku změnit pokud je splněn předpoklad pravidla definovaný stavy okolních buněk. Náš záměr při řešení této úlohy bude nalézt pravidla celulárního automatu, který z definovaného výchozího stavu mřížky dosáhne určeného cílového stavu a v něm setrvá beze změny.

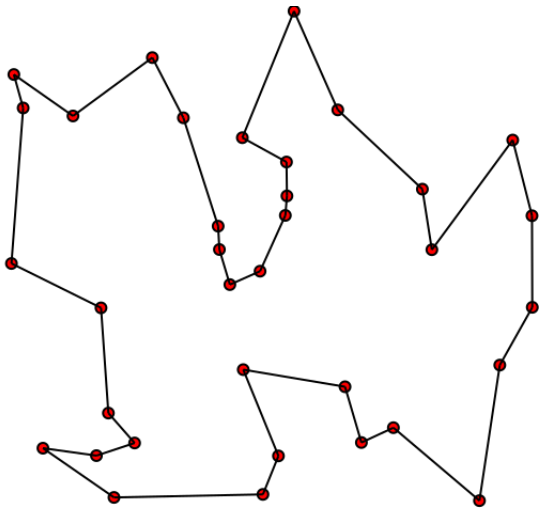
Jako referenční algoritmus pro hledání pravidel celulárního automatu bude využita evoluční strategie uvedená v sekci 2.1.1.

Později v následující kapitole bude hlavní část této práce obsahující modifikace a varianty převzatého simulovaného kvantového žíhání. Tyto modifikace se budou snažit dosáhnout lepších výsledků na zde zvolených úlohách než dosáhne originální simulované kvantové žíhání a referenční algoritmy.

4.1 Problém obchodního cestujícího

První problém pro experimentování s kvantově inspirovaným optimalizačním algoritmem je široce známý NP-těžký problém obchodního cestujícího (TSP). Tato úloha byla vybrána, jelikož je mu věnována vysoká pozornost, a stala se oblíbeným problémem na porovnávání různých optimalizačních algoritmů [2].

Tato úloha je dána množinou měst nebo míst, Poté úloha zní následovně: nalézt cestu, která prochází každým městem právě jedenkrát a výchozí město je stejné jako konečné, a tedy uzavírá cestu. Množina měst může být zadána souřadnicemi míst, ale postačuje znát pouze vzdálenosti mezi městy, tedy vzdálenost mezi každou dvojicí měst.



Obrázek 4.1: Graf zobrazující optimální cestu skrz města, která jsou označena červenými body. Nalezení této cesty je problém obchodního cestujícího. Obrázek převzat z [55].

Experimentování právě na této úloze má své uplatnění i v praxi. Kromě očividného využití k plánování cest a logistiky jej lze s určitými úpravami použít např. při výrobě mikročipů anebo např. i sekvencování DNA, či pohyby teleskopu [34].

Pro řešení problému obchodního cestujícího využijeme již zmíněný algoritmus pro simulované kvantové žíhání a ke srovnání použijeme klasické simulované žíhání.

4.1.1 Řešení TSP za pomocí klasického simulovaného žíhání

Nejdříve zde uvedeme, jak je navrženo referenční řešení problému obchodního cestující pomocí klasického simulovaného žíhání. Pro implementaci tohoto algoritmu, představeného v sekci 3.3.3, musíme nejprve vyřešit jeden hlavní problém. Tento problém se týká kódování a uložení cesty do efektivní podoby pro vytváření nových řešení problému.

Kódování řešení

Pro potřeby této práce byla zvolena následující reprezentace kódování řešení. Pro uložení cesty byla vybrána jistá obdoba cyklické fronty o velikosti shodné s počtem měst. Každé město má poté svoji pozici v této frontě. Důvodem zvolení této struktury je její vlastnost, která přesně popisuje platné řešení. Při předpokladu, že bude každé město ve frontě právě jednou, dostáváme popis cesty pořadím měst ve struktuře. Pokud například začneme ve městě na prvním místě fronty, tak následující město na cestě je město na následující pozici ve frontě. Cyklická vlastnost pole nám poté zaručuje, že cestu zakončíme ve městě, ve kterém jsme začali.

Při využití této struktury máme zaručeno, že každé uspořádání měst bude platné řešení, pokud budeme pouze měnit pozice měst v cyklické frontě.

4.1.2 Řešení TSP za pomocí simulovaného kvantového žíhání

Pro implementaci algoritmu simulovaného kvantového žíhání ukázaném dříve je nutné vyřešit několik záležitostí. První záležitost je vytvořit hamiltonián popisující TSP. Dále je nutné vyřešit kódování do matice spinů a transformaci této matice pro vytváření nových řešení.

Vytvoření hamiltoniánu popisujícího TSP

Vytvořený algoritmus v čisté podobě vyžaduje hamiltonián popisující daný problém. Tento hamiltonián může vypadat následovně:

$$\mathcal{H} = \mathcal{H}_{pot} + \mathcal{H}_{kin}, \tag{4.1}$$

kde $\mathcal{H}_{pot}$ je naše účelová funkce systému a $\mathcal{H}_{kin}$ je rozrušení našeho systému, které reprezentuje kvantové tunelování k překonání lokálních extrémů.

Naše účelová funkce pro TSP je celková délka cesty mezi městy, tedy délka, kterou se snažíme minimalizovat. Kinetická energie, která reprezentuje rozrušení z kvantového tunelovaní je závislá na způsobu, jakým se vybírají sousední řešení. Obecně ale zatím můžeme uvažovat, že tato energie systému odpovídá počtu rozdílných cest mezi dvěma různými řešeními.

Potenciální energii systému lze tedy vyjádřit rovnicí [7]:

$$\mathcal{H}_{pot} = \frac{1}{2}\sum_{i>j}^{n}(d_{ij} \times c_{ij}), \tag{4.2}$$

kde d_{ij} odpovídá vzdálenosti mezi městy s indexem i a j. Pro c_{ij} poté platí, že je rovno 1 právě tehdy, když existuje cesta z města s indexem i do města j v aktuálním řešení. V ostatních případech je c_{ij} rovno 0.

Tento součet je poté vydělen 2, jelikož každá cesta je započítána 2krát, tedy z města do i do j a z j do i.

Jak bylo zmíněno v sekci o implementaci simulovaného kvantového žíhání, tak rovnice popisující naše řešení by měla být upravena na Isingův model spinových skel, kde spiny mají hodnoty 1 nebo -1. Tyto spiny lze zapsat do matice spinů o rozměrech $N \times N$, kde N je počet měst v řešeném TSP. Tato matice musí být symetrická, jelikož všechny cesty mezi městy jsou neorientované. Také tato matice musí mít na diagonále samé 0, jelikož nemůže existovat cesta z města hned zpátky do tohoto města. Spiny v Isingově modelu v této matici odpovídají cestě mezi městy i a j a lze je definovat jako:

$$S_{ij} = 2c_{ij} - 1. \tag{4.3}$$

Tedy S_{ij} pro cestu c_{ij} bude buď -1 nebo 1 v závislosti na existenci této cesty v řešení TSP.

Poté lze upravit předchozí potenciální energii do Isingova modelu následovně [7]:

$$\mathcal{H}_{pot} = \sum_{i>j}^{n}\left(d_{ij} \times \left(\frac{S_{ij}+1}{2}\right)\right). \tag{4.4}$$

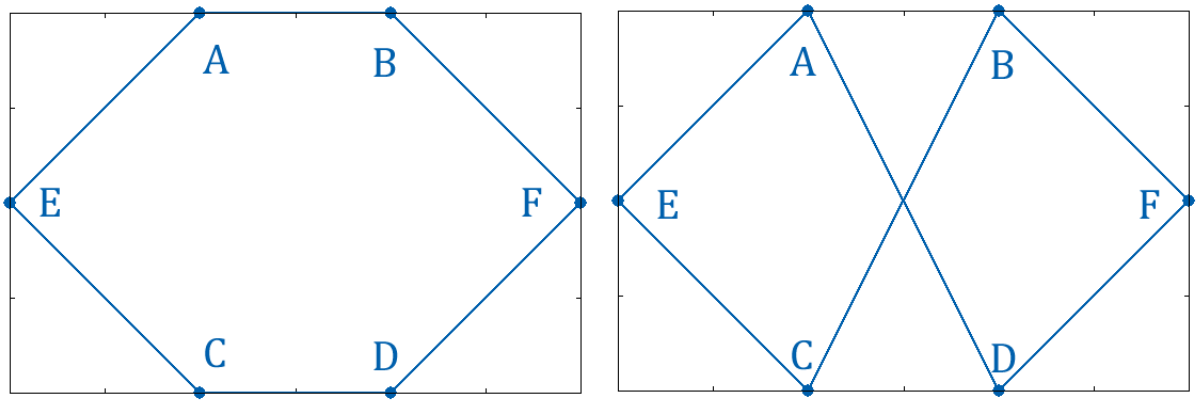
Pro podrobnější popis kinetické energie systému je nejprve nutné popsat, jak se budeme pohybovat ve stavovém prostoru. Tedy jak vypadají sousední řešení pro každé řešení.

Nalezení sousedících řešení pro problém obchodního cestujícího

Popis, jaká jsou sousedící řešení anebo jak je získat z aktuálního stavu, definuje, jakým způsobem se bude procházet stavový prostor problému. Tedy toto procházení definuje lokální extrémy tohoto prostoru, kde je nutné použít více než jeden přechod k nalezení lepšího řešení. Z tohoto důvodu je nutné zvolit dobré způsoby výpočtu/určení sousedních řešení.

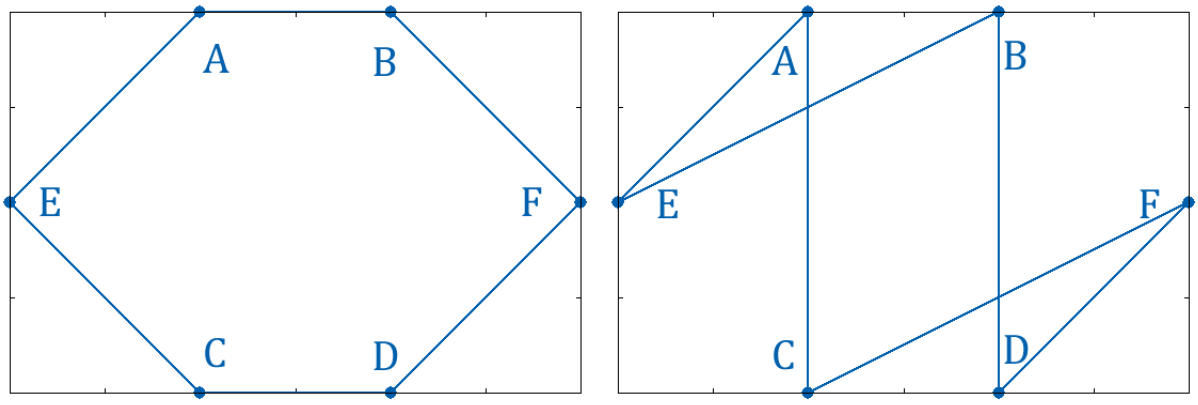
Pro náš problém obchodního cestujícího jsou zvoleny 2 různé transformace cest ve stavovém prostoru [12][39].

Jedna transformace vezme dvě cesty, které nemají společná města, a prohodí je. Pro jednodušší vysvětlení uvažujme, že první cesta je mezi městy A a B a druhá cesta je mezi městy C a D. Poté se při této transformaci tyto cesty zruší a nahradí se cestami z A do C a z B do D. Tato transformace je ilustrován na obrázku 4.2. Z hlediska Isingova modelu se jedná o změnu 4 spinů. 2 spiny nad diagonálou a jejich symetrické spiny v matici spinů.



Obrázek 4.2: Ukázka jednoho způsobu získáni souseda řešení TSP prohozením cest mezi městy A a B a městy C a D.

Druhá transformace je o něco komplikovanější. Pro tuto transformaci jsou vybrána 2 města (města B a C pro ilustraci). Poté jsou vybrány 4 cesty, 2 pro každé město. Tedy teoreticky odchozí a příchozí cesty z/do těchto dvou měst. Poté cesty, které původně vedly do a z města B budou vést do a z města C. Obdobná změna se provede s městem C. Tato transformace je ilustrován na obrázku 4.3. Z hlediska matice spinů se celkem změní 8 spinů. 4 spiny nad diagonálou matice spinů a 4 odpovídající symetrické spiny.



Obrázek 4.3: Ukázka druhého způsobu získáni souseda řešení TSP prohozením cest jdoucích do/z B a jdoucích do/z C.

Pokud již máme definované možné přechody ve stavovém prostoru, tak můžeme zapsat výraz odpovídající kinetické části hamiltoniánu. Pro tento výraz budeme pouze uvažovat první definovanou transformaci cesty, jelikož je jednoduší. Nicméně druhá transformace by se vytvořila obdobným způsobem.

Spiny, které se tedy změní při první transformaci cesty pro cesty mezi městy A a B a městy C a D jsou:

- změna na $+1$ pro S_{AC} a S_{BD},
- změna na -1 pro S_{AB} a S_{CD}.

Poté lze zapsat časově závislý hamiltonián popisující celý systém, který teoreticky lze vypočítat za pomocí Schrödingerovy rovnice:

$$\mathcal{H} = \sum_{i>j}^{n} \left(d_{ij} \times \left(\frac{S_{ij}+1}{2} \right) \right) - \frac{1}{2} \sum_{i>j}^{n} \sum_{i'>j'}^{n} \Gamma(i,j,i',j';t) S^{+}_{\langle i,i' \rangle} S^{+}_{\langle j,j' \rangle} S^{-}_{\langle i,j' \rangle} S^{-}_{\langle j,i' \rangle}, \quad [7] \qquad (4.5)$$

kde $S^{\pm}_{\langle a,b \rangle}$ je spin definován jako $S_{\langle a,b \rangle} = 2c_{ab} - 1 = \pm 1$ na pozici (a,b) a (b,a) v matici spinů. Γ je síla kvantového pole.

Implementace popisu řešení pro simulované kvantové žíhání

Jak již bylo zmíněno v sekci o implementaci simulovaného kvantového žíhání, tak výslednou Schrödingerovu rovnici 4.5 lze efektivně spočítat na kvantovém počítači, ale nehodí se pro výpočet na klasickém počítači.

Z toho důvodu pro výslednou implementaci použijeme rovnici získanou ze Suzuki-Trotterovy transformace popsané v sekci 3.4.3:

$$\mathcal{H} = \frac{\mathcal{H}_{pot}}{P} - J_{\Gamma} \Delta \mathcal{H}_{kin}. \qquad (4.6)$$

$$J_{\Gamma} = -\left(\frac{T}{2} \right) ln(tanh\left(\frac{\Gamma}{PT} \right)), \qquad (4.7)$$

kde $\mathcal{H}_{pot}$ odpovídá naší účelové funkci, kterou se snažíme minimalizovat. $\mathcal{H}_{pot}$ odpovídá kinetické energií vzniklé působením kvantového pole na jednotlivé spiny.

$\mathcal{H}_{pot}$ jsme pro problém TSP již dříve určili jako:

$$\mathcal{H}_{pot} = \frac{1}{2} \sum_{i,j=1}^{n} \left(d_{ij} \times \left(\frac{S_{ij} + 1}{2} \right) \right). \tag{4.8}$$

$\mathcal{H}_{kin}$ lze na základně rovnice 4.5 definovat jako rozdíl spinů sousedních replik.

Odchýlení od matice spinů

Výše zmíněná implementace v praxi funguje, nicméně práce se spiny pro popsání cesty je složitá a není příliš intuitivní. Z toho důvodu můžeme převzít pouze myšlenku této implementace a převést výpočet $\mathcal{H}_{pot}$ a $\mathcal{H}_{kin}$ do obecnější podoby, kde nebude potřeba matice spinů.

Obecně lze $\mathcal{H}_{pot}$ vyjádřit jako hodnotu účelové funkce. V našem případě se jedná o délku cesty řešení.

$\mathcal{H}_{kin}$ lze vyjádřit jako rozdíl mezi dvěma řešeními replik. V případě TSP se jedná o to, kolik neshodujících cest se mezi řešeními replik nachází.

Při těchto nových definicích potenciální a kinetické energie řešení můžeme použít namísto matice spinů cyklickou frontu jako byla využita u klasického simulovaného žíhání.

4.2 Problém maximální splnitelnosti booleovské formule

Další problém, který se budeme snažit vyřešit pomocí kvantově inspirovaného algoritmu je problém maximální splnitelnosti booleovské formule (maximum satisfiability problem: MAX-SAT).

Před vysvětlením problému MAX-SAT je zapotřebí vysvětlit rozhodovací problém splnitelnosti booleovské formule a jeho význam.

4.2.1 Splnitelnosti booleovské formule

Booleovská formule F je logický výraz definovaný nad proměnnými, které mohou nabývat hodnot pravda nebo nepravda (1 nebo 0). Pravdivostní přiřazení do množiny V booleovských proměnných je zobrazení $\sigma : V \to \{0, 1\}$. Splnitelné přiřazení pro F je pravdivostní přiřazení σ takové, že F je pravdivá pro σ. Formule, kterými se budeme zabývat, budou ve specifickém tvaru: F je v konjunktivní normální formě (CNF), pokud je konjunkcí klauzulí, kde klauzule jsou disjunkce literálů a každý literál je buď proměnná nebo její negace. Na příklad $F = (a \vee \bar{b}) \wedge (a \vee c \vee d) \wedge (\bar{a} \vee c)$ je formule o čtyřech proměnných a třech klauzulích. [25]

Rozhodovací problém splnitelnosti booleovské formule (satisfiability problem: SAT) je zadán následovně. Mějme formuli v konjunktivní normální formě a máme rozhodnout, zda existuje splnitelné přiřazení. Tento rozhodovací problém je NP-úplný [10]. V praxi se nejen snažíme rozhodnout, jestli je daná instance problému rozhodnutelná, ale snažíme se i najít ohodnocení proměnných, při kterých je formule splněná. [25]

4.2.2 Maximální splnitelnosti booleovské formule

Problém SAT je rozhodovací problém, ale existuje i rozšíření tohoto problému na optimalizační. Klasicky se při řešení SAT nesnažíme rozhodnout, zdali je formule splnitelná, ale

často chceme i ohodnocení proměnných, které ji splní. Nalézt takové ohodnocení není vždy snadné, jelikož se jedná o NP-úplný problém, anebo tato formule dokonce není splnitelná.

Takový problém lze řešit i jiným způsobem, kde se nesnažíme pouze určit, jestli je problém splnitelný a nalézt jeho ohodnocení, ale snažíme se najít takové ohodnocení, které splní co nejvíce klauzulí ve formuli. Takto specifikovaný problém se nazývá problém maximální splnitelnosti booleovské formule, který budeme v této práci řešit.

K vyřešení použijeme kvantově inspirovaný algoritmus pro simulované kvantové žíhání a také použijeme algoritmus z oblasti inspirované imunitními systémy, tedy algoritmus klonální selekce. Tento algoritmus je zvolen k porovnání se simulovaným kvantovým žíháním a také pochází z kategorie přírodou inspirovaných algoritmů.

4.2.3 Řešení MAX-SAT pomocí klonální selekce

Klonální selekce je algoritmus inspirovaný přírodou, přesněji imunitním systémem. Podrobný popis této inspirace byl již uveden v sekci 2.2 o algoritmech inspirovaných imunitním systémem.

Pro vyřešení optimalizačního problému MAX-SAT byl poté vybrán algoritmus klonální selekce založené na lokálním tabu prohledávání [36].

Kódování řešení

K vyřešení MAX-SAT problému pomocí klonální selekce je nejprve zapotřebí zakódovat řešení do formy pro B buňky, kterou lze zmutovat na jiná potencionální řešení. Pro zakódování existuje několik možných přístupů jako například zakódovat ohodnocení proměnných pomocí reálných čísel [45], ale pro naše řešení nám postačí intuitivní binární kódování.

Binární kódování je pole o velikosti n, kde n je počet proměnných v booleovské formuli. Tedy proměnná x_i má hodnotu v poli na místě i. Hodnota 1 v tomto poli reprezentuje pravdu a hodnota 0 reprezentuje nepravdu.

Hypermutace

Cílem hypermutace je získání diverzity mezi protilátkami. Proces, který tuto diverzitu zajišťuje, je hypermutace, kde se na protilátce provede několik náhodných mutací za cílem zvýšení avidity. Existuje několik způsobů mutací. Mutace použitá v publikaci [36], ze které vychází naše implementace, je nepřímo úměrná mutace. To znamená, že protilátky s vysokou aviditou mají jen malou šanci ke zmutování, ale protilátky s nízkou aviditou projdou hypermutací se snahou zvýšit jejich aviditu.

Proces selekce

Způsob selekce z populace ma veliký vliv na výslednou funkci algoritmu [50]. V naší implementaci je vytvořena nová populace z původní, kde šance výběru předka je přímo úměrná aviditě B buňky. Po provedení mutací popsaných výše a použitím lokálního prohledávání, které bude popsáno v nadcházející sekci, je určena avidita B buněk. Pokud má nejlepší buňka aviditu vyšší než paměťová buňka, tak se stává novou paměťovou buňkou. Výsledná populace je poté tvořena n nejlepšími B buňkami a paměťovou buňkou a nejhorších m buněk v populaci je nahrazeno náhodnými, kde platí $m \ll n$.

Výpočet avidity

Výpočet avidity mezi antigenem a protilátkou nám určuje ohodnocení řešení. Protilátky s vyšší aviditou by měly být lepším řešením problému a výpočet této avidity má důležitý význam na proces optimalizace. Tento výpočet je přímo závislý na typu úlohy. V našem případě se jedná o MAX-SAT, kde se snažíme splnit maximální počet klauzulí daným ohodnocením proměnných. Pro náš algoritmus je zvolena jednoduchá metrika, kde avidita je rovna počtu klauzulí, které ohodnocení splňuje. Nicméně existují i jiné způsoby ohodnocování avidity pro MAX-SAT problém jako je například postupná adaptace vah klauzulí [26].

Lokální prohledávání

Studie konvergence klonální selekce ukázala, že při používání hypermutace je konvergence pro MAX-SAT pomalá [37]. Z tohoto důvodu tento algoritmus kombinuje klonální selekci s lokálním tabu prohledáváním. Toto lokální prohledávání je uplatněno, když se paměťová buňka nezlepšila v daném počtu generací.

Samotná metoda lokálního prohledávání náhodně prohazuje proměnné. Pokud se nezlepšila avidita, tak prohození není přijato a proměnná se dá do FIFO struktury o velikosti 10 % proměnných formule. Tato velikost fronty byla empiricky určena na základě experimentů [22]. Pokud se zlepšila avidita, tak se ukončuje lokální prohledávání a pokračuje se v klonální selekci. Lokální prohledávání je omezeno na maximální počet pokusů, než se pokračuje klonální selekcí i bez zlepšení.

4.2.4 Řešení MAX-SAT pomocí simulovaného kvantového žíhání

Stejně jako při řešení problému obchodního cestujícího musíme splnit několik záležitostí, než můžeme aplikovat algoritmus simulovaného kvantového žíhání (SQA). Nicméně jak bylo ukázáno na konci sekce u implementace algoritmu pro TSP, tak není nutné definovat řešení problému do matice spinů, i když v případě problému SAT a následně MAX-SAT není složitá.

Vytvoření hamiltoniánu popisujícího problém MAX-SAT

Hamiltonián pro rozhodovací problém splnitelnosti booleovské formule pro SQA může být ve tvaru, který jsme použili pro problém TSP, tedy:

$$\mathcal{H} = \mathcal{H}_{pot} + \mathcal{H}_{kin}, \tag{4.9}$$

kde $\mathcal{H}_{pot}$ je naše účelová funkce systému a $\mathcal{H}_{kin}$ je rozrušení našeho systému, které reprezentuje kvantové tunelování k překonání lokálních extrémů.

Pro srovnatelné porovnání s klonální selekcí budeme uvažovat stejnou účelovou funkci, tedy počet splněných klauzulí dané formule. V případě vytváření hamiltoniánu chceme, aby platilo, že nižší energie systému znamenala lepší řešení. Tedy v našem případě budeme počítat počet nesplněných klauzulí. Tuto potenciální energii systému lze vyjádřit následovně:

$$\mathcal{H}_{pot} = n - \sum_{i=1}^{n} c_i(A), \tag{4.10}$$

kde n je počet klauzulí, c_i je funkce ohodnocující splnitelnost i-té klauzule při ohodnocení proměnných A.

Při upravení na Isingův model spinových skel můžeme uvažovat následující konfiguraci matice ohodnocení A s velikostí $1 \times 2v$, kde v je počet proměnných ve formuli. Každý sloupec matice odpovídá jedné proměnné nebo její negaci. Spiny u proměnných, které jsou pravdivé mají spin 1, jinak -1.

Matice pro zakódování booleovské formule F je velikosti $n \times 2v$, kde n je počet klauzulí a sloupce jsou stejné jako v matici popisující řešení. Každý řádek odpovídá jedné konjunkci ve formuli, kde spin je hodnoty 1 pokud se proměnná nebo její negace vyskytuje v konjunkci, jinak je spin roven -1.

Spiny odpovídající 1 a -1 můžeme nahradit za 1 a 0 pomocí výrazu:

$$c_{ij} = \frac{S_{ij} + 1}{2}. \tag{4.11}$$

Za pomoci těchto matic poté můžeme vyjádřit funkci c_i následovně:

$$c_i(A) = min(1, \sum_{j=1}^{2v} A_j F_{ij}), \tag{4.12}$$

kde v je počet proměnných, A je ohodnocení proměnných. Indexy i a j jsou indexy v matici spinů.

Prohledávání stavového prostoru MAX-SAT

Nalezení sousedních řešení problému MAX-SAT je jednodušší, než bylo u problému obchodního cestujícího. Žádná pravidla, která by definovala formu nebo platnost řešení, neexistují, a tedy můžeme libovolně měnit proměnné formule z pravdy na nepravdu a opačně.

Z pohledu matice spinů je při každé změně proměnné nutné změnit dva spiny, jelikož každé proměnné náleží dva sloupce v matici spinů.

Při definovaných přechodech ve stavovém prostoru našeho problému můžeme vypočítat kinetickou energii jako rozdíl matic spinů mezi replikami.

Implementace simulovaného kvantového žíhání

Stejně jako bylo řečeno u řešení problému obchodního cestujícího, tak lze uplatnit Suzuki-Trotterovu transformaci na vytvořený hamiltonián popisující energii systému. Nicméně v této práci uvažujeme pouze inspiraci z kvantových jevů.

Jako strukturu pro uchování řešení můžeme použít stejnou formu, jako byla použita pro uchování řešení v B buňkách. Tedy pole binárních hodnot, o velikosti shodné s počtem proměnných ve formuli, popisující stav každé proměnné (0 odpovídající nepravdě a 1 odpovídající pravdě).

Pro výpočet potenciální energie použijeme účelovou funkci, kterou jsme použili ve klonální selekci, která odpovídá počtu nesplněných konjunkcí ve formuli. Snaha algoritmu je poté minimalizovat hodnotu této funkce.

Pro výpočet kinetické energie v Isingově modelu určujeme počet spinů odlišných od spinů okolních replik. V naší implementaci ale nemáme spiny, ale máme ohodnocení proměnných. Kinetická energie bude odpovídat rozdílům ohodnocení těchto proměnných v okolních replikách. Tedy při neshodě ohodnocení proměnných bude růst kinetická energie tohoto řešení.

Po definování výpočtu potenciální a kinetické energie a způsobu vytváření nových řešení ze stávajících můžeme implementovat algoritmus pro simulované kvantové žíhání jako byl uveden na konci sekce 3.4.4.

4.3 Hledání pravidel celulárního automatu

Poslední problém řešený v této práci se týká celulárních automatů. Přesněji se bude jednat o hledání pravidel pro celulární automat k řešení zadaného problému. K hledání těchto pravidel využijeme přírodou inspirované algoritmy, které jsou: evoluční strategie a simulované kvantové žíhání. Celulární automaty nacházejí uplatnění v mnoha oblastech, kde se dají využít například jako simulační modely pro simulace evoluce nemocí [44] nebo vytváření modelů v biologii [27]. V našem případě se budeme zabývat jednodušším cílem, který bude popsán v sekci později, ale nejprve zde bude popsán základní princip celulárního automatu.

4.3.1 Celulární automaty

Celulární automat (CA) je jednoduchý matematický model založený na přírodních systémech. Tyto automaty sestávají z matice identických buněk, kde každá buňka může nabývat předem definované sady konečných stavů. Tyto stavy se deterministicky mění v průběhu času v závislosti na hodnotách okolních buněk. Tato diskrétní povaha systému může vést k porovnání s paralelním digitálním počítači, kde každá buňka je jeden jednoduchý uzel. [11]

Nejjednodušší celulární automat je dimenze 1 a buňky mohou nabývat pouze dvou hodnot: 1 nebo 0. Každá buňka sousedí se dvěma okolními, kromě okrajových buněk, u kterých je sousední buňka definovaná hodnotou 0. Takový automat obsahuje 8 pravidel popisující stav buňky na základě svého stavu a svých dvou sousedů. Při 8 pravidlech na jeden automat dostaneme maximálních 256 odlišných automatů [20].

I tyto jednoduché automaty mohou vytvářet složité vzory, pokud je vykreslíme na dvou dimenzí plochu, kde druhá dimenze je čas. Příklad takového vzoru generovaného jednodimenzionálním automatem s neomezenou šířkou, kde počáteční stav buněk je náhodně určen, je zobrazen na obrázku níže.

Obrázek 4.4: Vzor generovaný jedno-dimenzionálním celulárním automatem s označením pravidel: Wolfram's Rule 22[1]. Tento vzor je vytvořen z náhodné konfigurace buněk a zastaven po 33 krocích.

4.3.2 Pravidla automatu

Při automatu, který byl ukázán v předchozí sekci, není velice složité zapsat pravidla v podobě porovnání svojí hodnoty, hodnoty okolních buněk a nové hodnoty stavu. Nicméně velikost takového zápisu velice rychle roste se zvyšujícím počtem možným hodnot buněk. Například při buňkách majících hodnoty celého bajtu, tedy 256 hodnot, by počet pravidel pro jednorozměrný celulární automat byl roven 256^3.

[1] https://conwaylife.com/wiki/OCA:Rule_22

Ještě horší je případ pro dvourozměrný automat, kterým se budeme zabývat v této práci. V případě, kdy budeme uvažovat čtyři sousedy pro jednu buňku, dostaneme počet pravidel rovných číslu 256^5. Z tohoto důvodu budeme naše pravidla pro celulární automat uvažovat ve tvaru představeném ve článku [6].

Tato pravidla nemají klasickou podobu přiřazení hodnoty při splnění specifických hodnotách okolních buněk, ale jsou více obecné. Navrhované řešení snižuje nutný počet pravidel pro celulární automat třemi způsoby. Tento popis celulárního automatu má pouze zadaný počet pravidel, který je výrazně nižší, než by bylo nutné pro popsání každého pravidla celulární automatu, ale vzniká závislost na pořadí pravidel.

První způsob redukce pravidel je výchozí "pravidlo", které je aplikováno, pokud hodnoty buněk nevyhovují žádnému pravidlu v popisu automatu. Pokud tato situace nastane, tak hodnota buňky zůstane nezměněná. V principu jde pouze o zjednodušení zápisu, jelikož použití tohoto principu nerozšiřuje schopnosti celulárního automatu. Tyto situace, kde buňka není změněná, by šly zapsat klasickými pravidly ve tvaru porovnání pěti hodnot v případě našeho celulárního automatu a přiřazení hodnoty, která se rovná nové hodnotě vyhodnocované buňky.

Druhý způsob redukce množství pravidel je úprava způsobu zápisu na podmínková pravidla. Namísto porovnání, zda okolní buňky jsou rovny předepsané hodnotě v pravidle, se použijí i jiné operace k popsání podmínky, kdy bude pravidlo uplatněno. V našem případě se bude jednat o rovnost, nerovnost, větší než, menší než a kontrola, zda číslo leží v daném intervalu. I v případě této redukce pravidel se nejedná o rozšíření schopnosti celulárního automatu, jelikož tato podmínková pravidla pouze seskupují klasická pravidla do jednoho.

Poslední změna, která je navržena v této práci, je úprava výsledné hodnoty pravidel. Klasické pravidla celulárního automatu mají pevně stanovenou výslednou hodnotu, pokud je podmínka splněna. V případě klasických pravidel má toto přiřazení smysl, jelikož hodnota středové buňky je v pravidle porovnána se specifickou hodnotou, a tedy je pro pravidlo pevně dána. Nicméně v podmínkových pravidlech popsaných výše může hodnota středové buňky nabývat různých hodnot při splnění podmínek pravidla (například pokud má operaci *nižší než* nějaká konstanta). V tomto případě jsou v této práci navrženy další operace kromě přímého přiřazení výsledné hodnoty, kde se jedná o zvýšení nebo snížení hodnoty středové buňky o konstantní hodnotu. Tato výsledná hodnota je omezena na interval možných stavů buněk a opět nerozšiřuje schopnosti automatu, jelikož při převodu na klasické pravidla se dá výsledná hodnota spočítat z hodnoty středové buňky a operace určující výslednou hodnotu.

Podmínkové pravidla pro 2D celulární automaty, neuvažující diagonální buňky jako sousedící, mohou být například zobrazena pomocí 5ti podmínek pro porovnání hodnot buněk a výrazem k výpočtu nové hodnoty:

```
!= 21 | ==  0 | == 27 | <= 44 | == 27 -> -- 12
== 34 | == 14 | <=  4 | != 13 | != 40 -> == 29
<= 17 | != 20 | >= 21 | != 48 | <= 21 -> ==  6
```

4.3.3 Účel celulárního automatu

Jak již bylo uvedeno dříve, tak celulární automaty se dají využít pro mnoho úloh. V této práci se nebudeme zabývat celulárním automatem představující model, ale automatem, který bude generovat specifické vzory.

Definice problému, který budeme řešit je tedy následující. Na vstupu úlohy bude dán počáteční stav buněk v matici s předem definovanými rozměry a předem danou množinou

stavů, kterou mohou buňky nabývat. Tento počáteční stav bude v našem případě celý roven hodnotě 0, kromě jediné buňky někde ve středu matice s odlišnou hodnotou. Tato buňka bude považována za "semínko" pro vznik vzoru. Tento vzor bude matice se stejnou velikostí jako je počáteční matice a s buňkami o hodnotách, kterých se bude celulární automat snažit dosáhnout.

Cíl optimalizace je tedy najít podmínková pravidla pro celulární automat, který ze zadaného počátečního stavu automatu vytvoří cílový vzor. Na tato pravidla jsou kladeny ještě další dva nároky.

Celulární automat má pouze omezenou velikost matice s buňkami. Pro buňky, které jsou na okraji této matice, uvažujeme, že hodnoty sousedících buněk, které by byly mimo matici, mají hodnotu 0. Nicméně jeden z požadavků na pravidla celulárního automatu a posléze na algoritmy hledající tato pravidla bude zákaz používat tento okraj k vytvoření cílového vzoru.

Druhý nárok se týká výsledného vzoru. V mnoha případech při hledání pravidel pro celulární automat se podaří nalézt sadu pravidel, která z počátečního zadaní opravdu vytvoří cílový vzor, ale tento vzor existuje pouze po dobu jednoho kroku automatu. V této práci nebudeme taková pravidla považovat za správná řešení a budeme požadovat, že cílový vzor, který celulární automat vytvoří, musí být statický. Tedy po vytvoření vzoru z počátečního semínka se tento vzor již neztratí v následujících krocích automatu.

4.3.4 Účelová funkce pro pravidla celulárního automatu

Výběr účelové funkce vysoce ovlivňuje schopnost algoritmů naleznout řešení a jeho výslednou kvalitu. Před popisem vybrané účelové funkce k ohodnocení je nutné uvést jednu další podmínku, která byla na algoritmy kladena. V obecné implementaci zmíněné výše se často stane, že výsledná pravidla automatu nemají stabilní stav (nebo se automat ustálí pouze za vysoký počet kroků). Z tohoto důvodu má celulární automat dáno k kroků, ve kterých musí vytvořit daný vzor.

Při uvažování této podmínky můžeme tedy předpokládat, že nejvýše po k krocích dostaneme výslednou matici buněk. Poté jasná volba účelové funkce je porovnání této výsledné matice s maticí popisující zadaný cílový vzor. V průběhu experimentování na této úloze byly uvažovány dvě podobné metriky.

První účelová funkce je střední absolutní chyba (Mean absolute error - MAE) jednotlivých buněk výsledné a cílové matice celulárního automatu. Tato funkce je daná vzorcem:

$$\mathrm{MAE} = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} |T_{ij} - R_{ij}|, \tag{4.13}$$

kde N a M je šířka a výška matice buněk. T je matice popisující cílový vzor a R je výsledná matice z automatu. Indexy i a j jsou souřadnice v matici buněk.

Druhá uvažovaná účelová funkce je více přísná na ohodnocení podmínkových pravidel. V této účelové funkci považujeme za správné pouze ty buňky, které přesně odpovídají cílové matici. Tato funkce tedy počítá počet správných buněk ve výsledné matici celulárního automatu. Z důvodu sjednocení jednotlivých úloh v této práci budeme počítat počet chybných buněk, aby se jednalo o minimalizaci účelové funkce. Tato účelová funkce je poté popsána vzorcem:

$$\text{Počet chybných buněk} = \sum_{i=1}^{N}\sum_{j=1}^{M} C(T_{ij}, R_{ij}), \tag{4.14}$$

$$C(x,y) = \begin{cases} 1, & \text{pokud } x \neq y \\ 0, & \text{jinak} \end{cases} \tag{4.15}$$

kde N a M je šířka a výška matice buněk. T je matice popisující cílový vzor a R je výsledná matice z automatu. Funkce $C(x, y)$ porovnává hodnoty buněk.

Pro kapitolu s výsledky porovnání algoritmů byla vybrána funkce uvažující počet chybných buněk. Nicméně v experimentech s těmito funkcemi nebyl pozorován výrazný rozdíl v rychlosti optimalizace.

Zvolená funkce byla použita jako hlavní účelová funkce, ale při optimalizačním procesu bylo využito další kritérium. Toto kritérium určuje za kolik kroků dosáhne celulární automat jakéhokoliv stabilního stavu. Tato hodnota byla poté použita jako druhotná účelová funkce k porovnání řešení, která mají stejnou hlavní účelovou funkci. Důvod k použití druhotné funkce je snaha dosáhnout kvalitnějších řešení, kde kvalita je v našem případě rychlost vytvoření cílového vzoru ze semínka. Další možné takové kritérium by mohlo být počet použitých pravidel nebo počet využitých stavů z množiny možných stavů.

Poslední část ohodnocující funkce jsou "pokuty". V současné implementaci existují dvě pokuty, které se mohou přičíst k výsledku ohodnocující funkce. Tyto pokuty slouží k odrazení algoritmů od určitých řešení. První pokuta se týká stability vzoru. Pokud automat nedosáhne v daném počtu kroků jakýkoliv stabilní vzor, tak dostane tuto první pokutu. Tato pokuta udržuje algoritmus u stabilních výsledných vzorů. Druhá pokuta se týká hranicí matice buněk. Pokud automat při vytváření vzoru využije buněk na okrajích matice, tak dostane tuto pokutu, jelikož nechceme, aby vytvoření výsledného vzoru bylo závislé na okrajových podmínkách matice. Pokud by automat využil těchto okrajových podmínek, tak by tato pravidla selhala při vytváření vzoru v neomezené matici buněk.

4.3.5 Hledání pravidel pomocí evoluční strategie

Jako referenční algoritmus pro porovnání se simulovaným kvantovým žíháním byla zvolena evoluční strategie. Evoluční strategie již byla úspěšně použita při hledání podmínkových pravidel celulárního automatu například pro filtrování obrazu [6].

Evoluční strategie (μ, λ)

Tato evoluční strategie udržuje populaci rodičů o velikosti μ, kde v každé generaci se vytvoří λ potomků, kde platí $(\mu < \lambda)$. Prvotní generace rodičů je tvořena náhodným vygenerováním pravidel. Pro následnou sadu potomků se vybírají rodiče principem Round-robin. Rodiče jsou tedy postupně vybíráni ze seřazeného pole a vytváří potomka s počtem mutací odpovídající indexu jeho rodiče ve seřazené populaci podle jejich kvality. Každá mutace ovlivňuje pouze jedno pravidlo a může změnit porovnávající podmínku, hodnotu u této podmínky nebo výsledné přiřazení hodnoty při uplatnění pravidla. Jelikož počet potomků je vetší než je populace rodičů $(\mu < \lambda)$, tak při dojití na posledního rodiče v populaci se pokračuje generování znova od začátku populace dokud nebude vytvořeno λ potomků. Tito potomci jsou poté ohodnoceni na základě účelové funkce, která byla uvedena v předchozí sekci. Po ohodnocení potomků se tito potomci a rodiče seřadí na základě jejich ohodnocení a nová

generace rodičů odpovídá μ nejlepším jedincům. V průběhu této evoluční strategie si pamatujeme nejlepší doposud nalezení řešení, které bude výsledkem po dosažení maximálního počtu generací nebo časového limitu. [6]

4.3.6 Hledání pravidel pomocí simulovaného kvantového žíhání

Další použitý algoritmus, který je cílem této práce, pro hledání podmínkových pravidel pro celulární automaty je simulované kvantové žíhání. Jako v předchozích úlohách je zapotřebí popsat tuto úlohu za pomocí hamiltoniánu mající potenciální a kinetickou energii. Rovnice pro výpočet těchto energií jsou potřeba pro implementaci tohoto algoritmu.

Vytvoření hamiltoniánu popisujícího problém hledání pravidel CA

Pro teoretickou aplikaci kvantového žíhání by bylo užitečné popsat řešení úlohy transformací do Isingova modelu spinových skel. Nicméně při ohodnocování řešení u celulární automatů se jedná o několikanásobnou aplikaci podmínkových pravidel a transformaci vstupní matice buněk. Tento postup by byl velice složitý popsat funkcí a pravděpodobně by tato úloha nebyla vhodná na řešení pomocí kvantového hardwaru řešícího problém kvantových spinových skel.

Avšak v našem případě se jedná pouze o algoritmus inspirovaný procesem kvantového žíhání. Tedy, i když se trochu vzdalujeme od původní myšlenky kvantového žíhání, tak k vyřešení úlohy hledání pravidel pro celulární automat nepotřebujeme tuto úlohu popsat pomocí modelu spinových skel. Ale i tak k optimalizaci pomocí simulovaného kvantového žíhání stále potřebujeme hamiltonián popisující tuto úlohu. Rozdíl je v tom, že nemusíme mít popis jednotlivých energií pomocí matematických vzorců, ale vystačíme si i s definicí algoritmu, který bude provádět výpočet potenciální a kinetické energie.

Potenciální energie tedy odpovídá účelové funkci. Pro tuto účelovou funkci použijeme výpočet účelové funkce, který by již popsán v sekci o účelových funkcích pro pravidla celulárního automatu shodující se s účelovou funkcí použitou v evoluční strategii (rovnice 4.14).

Vytvoření výpočtu kinetické energie

Pro úlohy popsané pomocí Isingova modelem spinových skel je výpočet kinetické energie mezi replikami jednoduchá úloha, kde se jedná pouze o součet nesouhlasných spinů sousedních replik. Nicméně v našem případě musíme navrhnou složitější výpočet této energie.

Jádro myšlenky kinetické energie v Isingově modelu je porovnat podobnost dvou matic spinů a vyčíslit tento rozdíl pomocí kinetické energie. Tento předpoklad využijeme pro výpočet kinetické energie pro řešení hledání pravidel pro CA.

Pravidla automatu jsou uspořádaná v seznamu, jelikož při používání podmínkových pravidel závisí na jejich pořadí, které musí být deterministické. Tato struktura nám dává možnost porovnání pravidel dvou replik. Při porovnání seznamů pravidel dvou replik bude výsledná kinetická energie součet rozdílů pravidel na stejném indexu v těchto seznamech, tedy rovnicí:

$$\mathcal{H}_{kin} = \sum_{i=1}^{n} \mathrm{dif}(A_i, B_i), \tag{4.16}$$

kde $\mathrm{dif}(A, B)$ je funkce určující kinetickou energii mezi dvěma pravidly, která bude popsána dále. A a B jsou seznamy podmínkových pravidel o délce n.

Funkce pro porovnání dvou pravidel ($dif(A_i, B_i)$) taktéž není jednoduchá, jelikož v této práci byla snaha vytvořit funkci, která nebude pouze binární porovnání dávají shodu (0) nebo neshodu (2), ale hodnotu v intervalu $\langle 0, 2\rangle$. Každé pravidlo je dáno 5ti podmínkovými výrazy (jelikož neuvažujeme diagonální sousední buňky) a jedním výrazem určujícím výslednou hodnotu buňky při aplikaci pravidla. Výsledná kinetická energie mezi dvěma pravidly je součet rozdílu těchto 6ti výrazů popsaný obdobnou funkcí, jako byla uvedena výše:

$$\mathcal{H}_{kin} = \sum_{i=1}^{5} \mathrm{dif}_p(A_i, B_i) + \mathrm{dif}_v(A, B), \tag{4.17}$$

kde funkce dif_p a dif_v jsou funkce porovnávající jednotlivé výrazy v podmínkových pravidlech A a B. dif_p porovnává 5 podmínkových výrazů a $\mathrm{dif}_v(A, B)$ porovnává výraz pro výstupní hodnotu pravidla.

Tyto porovnávající funkce jsou definovány následovně. Pokud se liší typ výrazu (například výraz *rovnosti* oproti výrazu *menší než*), tak výsledkem je hodnota 2. Číslo 2 bylo zvolené, protože hodnoty od 0 do 1 jsou použity při menší neshodě popsané dále. Nicméně přesné hodnoty pro tato čísla mohou být libovolné, dokud bude platit, že větší neshoda mezi pravidly bude mít vyšší číslo. Pro výrazy, které se shodují v operaci, je výsledná hodnota mezi 0 a 1 v závislosti na tom, jak blízko jsou od sebe daná čísla v operacích. Při shodě těchto čísel je výsledkem 0 a při maximálním rozdílu je výsledkem hodnota 1.

Vytváření nových řešení

K funkci optimalizace je nutné nějakým způsobem modifikovat podmínková pravidla jednotlivých replik k vytváření nových řešení. Pro tento účel byla zvolena podobná strategie jako je u evoluční strategie, tedy mutace pravidel. Počet těchto mutací je dán v rozmezí mezi jednou změnou nebo více změnami v závislosti na nastavené pravděpodobnosti. Některá rozšíření tohoto algoritmu, popsané v následující kapitole, také přidávají nové způsoby modifikace těchto pravidel v závislosti na pravidlech okolních replik.

Kapitola 5

Nové varianty simulovaného kvantového žíhání

Cílem této práce není pouze aplikace algoritmu inspirovaného kvantovými jevy na nové úlohy, ale i snaha navrhnout nové modifikace tohoto algoritmu. Tyto modifikace byly navrženy za účelem vylepšení procesu optimalizace pomocí simulovaného kvantového žíhání. Při návrhu byla hlavní snaha vytvořit varianty algoritmu k lepší optimalizaci, a tedy tyto modifikace nevychází čistě z vlastností kvantové mechaniky. Nicméně některé varianty využívají vlastností tohoto algoritmu pracujícího na základě paralelní optimalizace replik.

Při práci na vylepšení stávajícího simulovaného kvantového žíhání bylo navrženo několik variant tohoto algoritmu, které budou popsány v následujících sekcích. Následně v kapitolách o vyhodnocení budou tyto modifikace porovnány jak s vybraným nekvantovým algoritmem, tak s původním kvantovým algoritmem na vybraných úlohách představených v předchozí kapitole.

5.1 Ovlivnění nejlepším řešením

První modifikace je pořád poměrně blízko inspirace kvantovou fyzikou. Pro výpočet kvantového žíhání na klasickém počítači musíme využít Suzuki-Trotterovu transformaci a metodu Monte Carlo založenou na křivkovém integrálu. Tento převod z kvantového světa do klasického přidává imaginární čas s určitým počtem Trotterových replik. V perfektním případě by se tento počet replik blížil velikosti stanového prostoru, jelikož tyto repliky nahrazují kvantový jev superpozice. V případě naší implementace simulovaného žíhání, z časové náročnosti běhu, používáme pouze poměrně nízký počet replik a jejich interakce je omezena pouze na sousední repliky (v kvantovém světě by repliky interagovaly se všemi replikami).

Nová varianta algoritmu se snaží přiblížit simulaci kvantového žíhání ke kvantové verzi přidáním další repliky, která ovlivňuje výpočet kinetické energie všech replik. Tato nová replika je tvořena doposud nejlepším nalezeným řešením dané úlohy. Nicméně tato kopie nejlepšího řešení se odlišuje od ostatních replik ve dvou oblastech.

První odlišnost se týká optimalizace replik. Normální repliky v každé generaci algoritmu vytvoří nové kandidátní řešení, které je buď přijato nebo odmítnuto. Na rozdíl od normálních replik je nejlepší replika vyjmuta z této optimalizace a pouze je nahrazena novým lepším (nebo stejně kvalitním) řešením, pokud je takové řešení nalezeno v průběhu optimalizace.

Druhá odlišnost se týká normálních replik. Tyto repliky v původní variantě počítají svoji kinetickou energii z rozdílů spinů mezi sousedními replikami, kde repliky jsou uspořádány v cyklické frontě. Tedy každá replika sousedí právě se dvěma dalšími replikami. Tento výpočet kinetické energie je právě místo, kde replika s nejlepším řešením pomáhá v optimalizaci ostatním replikám. Při výpočtu kinetické energie v této variantě každá replika sousedí se dvěma replikami jako v původní variantě algoritmu, ale také vždy "sousedí" s replikou reprezentující nejlepší řešení.

Předpokládaný přínos na optimalizaci je směrování replik k oblasti stavového prostoru, kde se nachází aktuální nejlepší nalezené řešení. Toto směrování by mohlo pozitivně působit na nalezení lepšího řešení na cestě k oblasti stavového prostoru obsahující nejlepší řešení podobně jako optimalizační algoritmus inspirovaný černými dírami [35].

5.2 Delší optimalizace jednotlivých replik

V klasické variantě simulovaného kvantového žíhání se postupně střídá výpočet jednotlivých replik, kde každá replika projde modifikací. Tato modifikace může být přijata nebo odmítnuta na základě procesu, který je popsán v sekci 3.4.4 popisující implementaci simulovaného kvantového žíhání. Nicméně bez ohledu na výsledek tohoto procesu se pokračuje další replikou v pořadí. Ovlivnění tohoto procesu je záměr této varianty, kde namísto jednoho pokusu o nalezení lepšího řešení pro danou repliku je uskutečněno několik pokusů, než se přistoupí k optimalizaci další repliky.

V aktuální implementaci varianty se vždy provádí předem definovaný počet pokusů k nalezení lepšího řešení bez ohledu na to, zda bylo nalezeno lepší řešení při tomto kroku optimalizace. Při pokračování v této práci by se mohlo vytvořit srovnání, zda zastavení tohoto procesu při nalezení lepšího řešení nemá lepší optimalizační vlastnosti než aktuální implementace.

Zlepšení optimalizace se předpokládá díky tomu, že replika, která následuje po aktuálně optimalizované replice, bude s vyšší pravděpodobností sousedit s replikou, která zlepšila svoje řešení. Sousedění s lepšími replikami má přínos na právě optimalizovanou repliku jelikož je ovlivněna okolními replikami při výpočtu kinetické energie.

5.3 Omezení nových řešení pomocí tabu meta-heuristicky

Další varianta simulovaného kvantového žíhání je inspirována lokálním tabu prohledáváním [23]. Při prohledávání stavového prostoru u některých úloh, jako bude v našem případě problém obchodního cestujícího a úloha MAX-SAT, může být výhodné dočasně redukovat část prohledávaného prostoru. Pro pokus o tuto redukci je v našem případě použita meta-heuristická metoda tabu prohledávání.

Optimalizace jednotlivých replik je prováděna modifikací části současného řešení popsané danou replikou, kde tato modifikace může být přijata nebo odmítnuta. Přidáním tabu meta-heuristiky každá replika dostane omezenou paměť předešlých modifikací, které byly odmítnuty. Poté v kroku vytváření nových řešení se vyhýbá modifikacím, které jsou obsažené v této paměti odmítnutých modifikacích, jelikož v minulosti tyto modifikace vedly k zamítnutí nového řešení. Zakázáním změnit určité části řešení se význačně snižuje stavový prostor o část, která s vysokou pravděpodobností obsahuje nekvalitní řešení. Proces zapomínání je realizován omezenou velikostí paměti, kde při plné paměti nově přidaná položka nahrazuje nejstarší uloženou položku v paměti.

5.4 Převzetí části řešení z nejlepšího řešení

Poslední modifikace navržená v této práci do jisté míry souvisí s první modifikací, která přidává ovlivnění nejlepším řešením. Tato modifikace ovlivňuje způsob, jakým se vytváří nové řešení pro repliky.

V normálním případě se vytváří nové řešení bez vlivu ostatních replik a vliv okolních replik je přidán až ve fázi výpočtu kinetické energie. Výpočet této kinetické energie je rozšířen v případě první navržené varianty, kde kromě okolních replik je i nejlepší řešení součástí výpočtu kinetické energie. Toto nejlepší řešení pomáhá směrovat repliku k nejlepšímu řešení, ale pouze pokud se náhodnou mutací řešení přiblížíme k nejlepšímu řešení. Tento nedostatek se snaží vylepšit tato varianta algoritmu.

Při vytváření nového řešení je řešení změněno pomocí náhodné mutace, kde šance provést změnu, která se přibližuje k nejlepšímu řešení, se snižuje s růstem velikosti stavového prostoru. Z tohoto důvodu zavádíme určitou malou pravděpodobnost, že namísto náhodné mutace se provede změna, která přiblíží řešení k nejlepšímu nalezenému řešení.

Kapitola 6

Výsledky problému obchodního cestujícího

Na úloze řešení problému obchodního cestujícího je porovnáván algoritmus simulovaného kvantového žíhání (SQA), klasického simulovaného žíhání (SA) a tři modifikace simulovaného kvantového žíhání popsané výše. Tyto modifikace jsou:

- Ovlivnění nejlepším řešením s názvem *Best* ve vyhodnocení.
- Delší optimalizace replik s názvem *Longer* ve vyhodnocení.
- Využití tabu meta-heuristiky s názvem *Tabu* ve vyhodnocení.

Pro vyhodnocení byl využit superpočítač Barbora[1]. Výpočetní uzly obsahují 2 osmnáctijádrové procesory Intel Cascade Lake s výkonem 2.6 GHz. K vytvoření srovnání bylo provedeno 108 nezávislých běhů každého algoritmu nebo varianty na 3 uzlech superpočítače.

Srovnání na problému bylo provedeno na třech obtížnostech této úlohy. Úloha o 100 městech, 1000 městech a 5000 městech, které byly vytvořen pomocí softwaru na převedení rastrových obrázků na zadaní pro TSP zvané TSP Art [2].

Obrázek 6.1: Ukázka zadání úlohy pro TSP s 5000 “městy” použita pro porovnání algoritmů.

[1] https://www.it4i.cz/infrastruktura/barbora
[2] https://www.drububu.com/illustration/tsp/index.html

Algoritmus SA ve vyhodnocení odpovídá klasickému simulovanému žíhání. SQA odpovídá simulovanému kvantovému žíhání bez modifikací. Pro modifikace jsou ve vyhodnocení použity upravené varianty SQA s jednou nebo více modifikacemi popsané tabulkou 6.1 níže:

	Modifikace		
Algoritmus	*Best*	*Loop*	*Tabu*
SQA	ne	ne	ne
SQA Be	ano	ne	ne
SQA BeLo	ano	ano	ne
SQA BeLoTa	ano	ano	ano

Tabulka 6.1: Tabulka znázorňující použité varianty algoritmu SQA při srovnání algoritmů v následujících sekcích.

6.1 100 měst

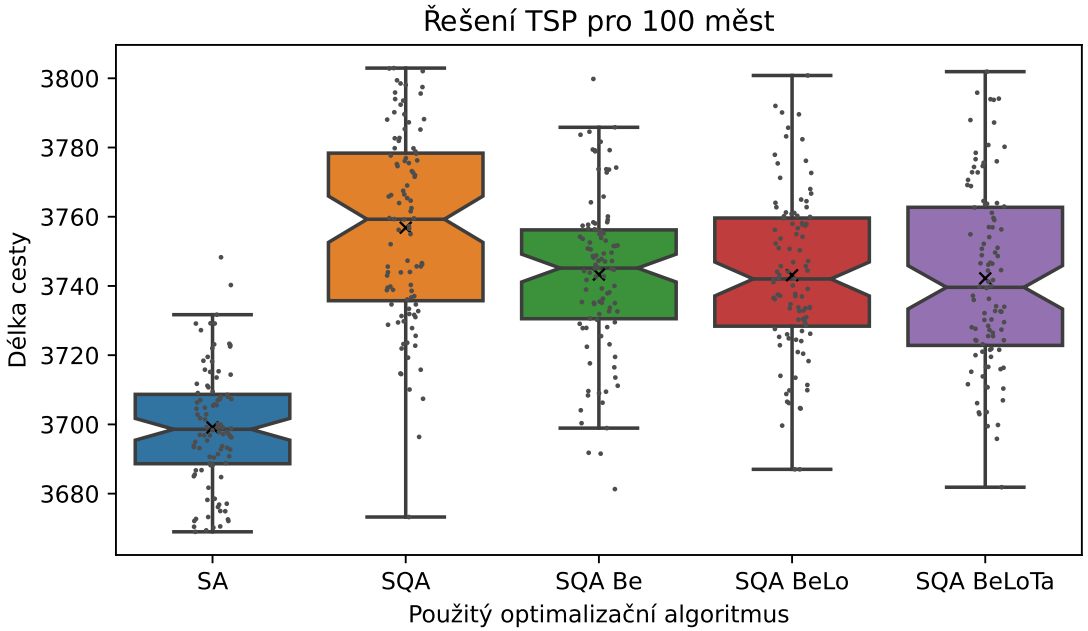
První srovnání algoritmů uvedených na začátku této kapitoly pro problém obchodního cestujícího je s velikostí problému 100 měst. Každý algoritmus běžel po dobu 10 milionů iterací. Tabulka 6.2 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počáteční teplota ($1/\beta$)	Počet iterací před změnou teploty	Rychlost ochlazování
SA	250	100	0.9995

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.015	1.5	0.99975	15
SQA Be	0.015	1.5	0.99975	15
SQA BeLo	0.015	1.5	0.99975	15
SQA BeLoTa	0.015	1.5	0.99975	15

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*
1	500	10 %

Tabulka 6.2: Parametry pro porovnávané algoritmy.

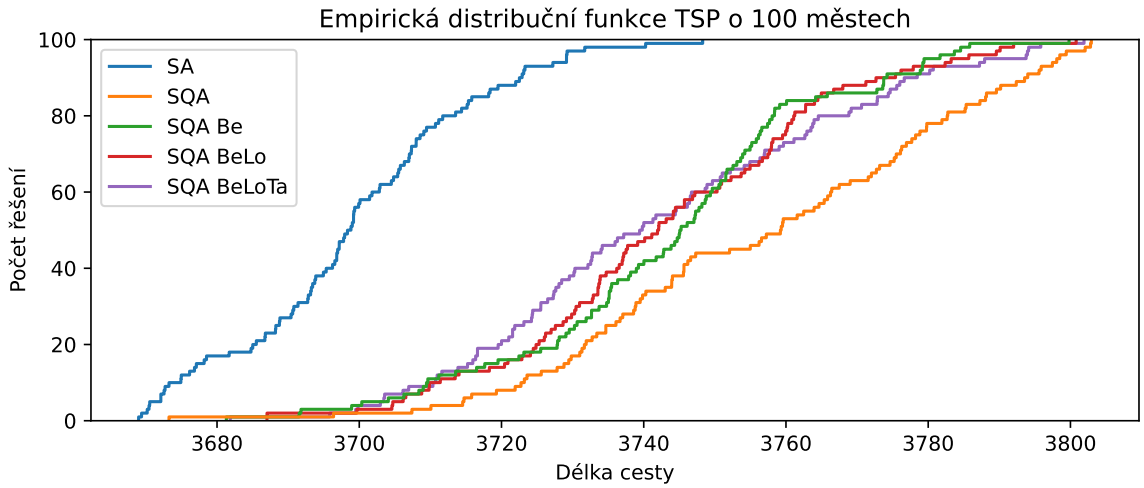


Obrázek 6.2: Statistické vyhodnocení nejkratších cest nalezeních za pomocí simulovaného žíhání, simulovaného kvantového žíhání a modifikací SQA pro problém obchodního cestujícího o 100 městech.

Snahou optimalizace je dosáhnout minimální možné cesty mezi městy. Z grafu 6.2 lze pozorovat, že klasické simulované žíhání dosahuje lepších výsledků než SQA a jeho varianty, ale rozdíl mediánů není větší než 2 % celkové délky cesty. Přestože není překonán klasický

algoritmus, tak lze pozorovat zlepšení optimalizace ve variantách SQA. Nejlepší skokové zlepšení přináší modifikace *Best* za cenu menšího počtu velice dobrých řešení. Ostatní modifikace poté opět rozšiřují variaci kvality jednotlivých řešení, ale pořád zachovávají zlepšení oproti základní verzi SQA i malým vylepšením oproti předchozím variantám.

Na grafu 6.3 můžeme pozorovat, že pokud budeme chtít více kvalitních řešení z SQA, tak je varianta SQA BeLoTa nejlepší z variant SQA. Nicméně pokud nám budou stačit horší řešení (25. percentil), tak SQA Be a BeLo nám dává lepší výsledky než BeLoTa.



Obrázek 6.3: Empirická distribuční funkce pro jednotlivé srovnávané algoritmy.

Horší výsledky SQA oproti SA jsou pravděpodobně způsobeny tím, že klasické simulované žíhání má dobré vlastnosti vedoucí k vysoké šanci dostat se z lokálních minim. Režie počítání paralelních řešení replik je tedy nadbytečně náročný výpočet k uniknutí z lokálních extrémů.

Varianta SQA *Best* přináší lepší výsledky směrováním všech replik k jedinému řešení, které by mělo být dobré. Z tohoto důvodu pravděpodobně můžeme pozorovat lepší průměr a medián za cenu menšího počtu velice kvalitních řešení. Ostatní varianty mají jen mírný vliv na výsledek, kde *Tabu* se snaží omezit stavový prostor a *Loop* se snaží najít lepší řešení pro ovlivnění ostatních replik.

6.2 1000 měst

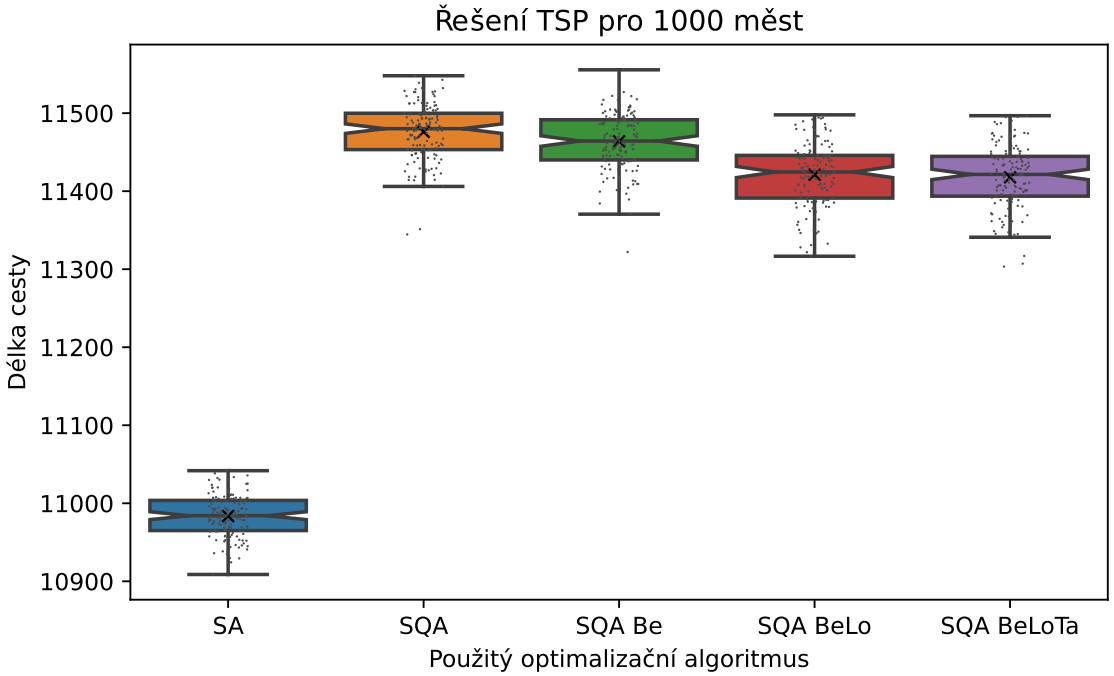
Další srovnání algoritmů je na 1000 městech, kde každý algoritmus běžel po dobu 500 milionů iterací. Tabulka 6.3 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počáteční teplota ($1/\beta$)	Počet iterací před změnou teploty	Rychlost ochlazování
SA	250	100	0.9999

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.015	1.5	0.9999	15
SQA Be	0.015	1.5	0.9999	15
SQA BeLo	0.015	1.5	0.9999	15
SQA BeLoTa	0.015	1.5	0.9999	15

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*
1	500	5%

Tabulka 6.3: Parametry pro porovnávané algoritmy.

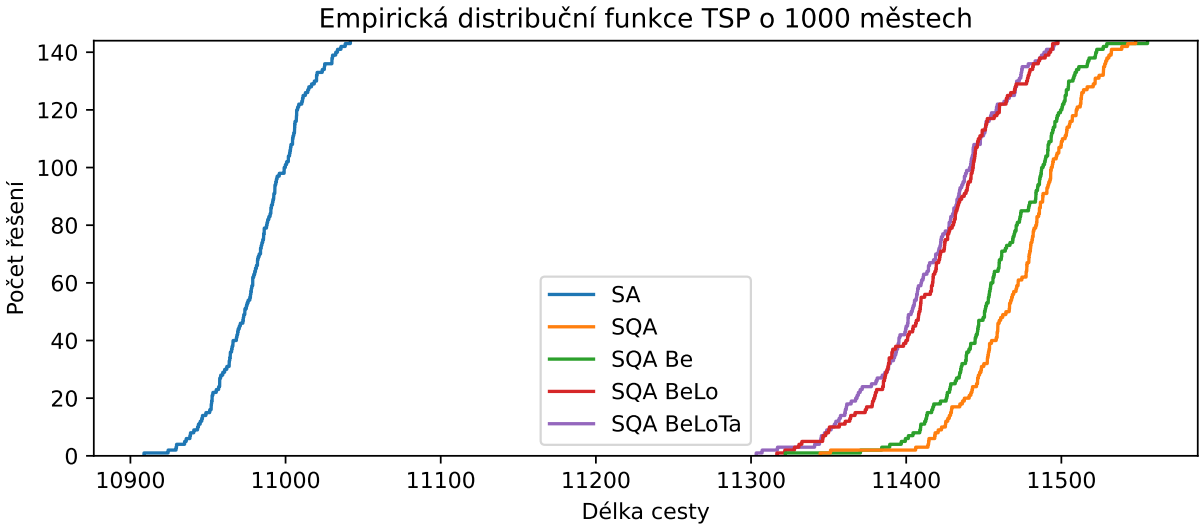


Obrázek 6.4: Statistické vyhodnocení nejkratších cest nalezeních za pomocí simulovaného žíhání, simulovaného kvantového žíhání a modifikací SQA pro problém obchodního cestujícího o 1000 městech.

Oproti výsledkům z optimalizace 100 měst můžeme na grafu 6.4 pozorovat, že SQA se výrazně zhoršuje s rostoucí složitostí úlohy, kde rozdíl mediánů je již téměř 5 %. Nicméně i tomto případě vytvořené varianty SQA umožňují lepší optimalizaci než základní varianta.

Při vyšší obtížnosti také můžeme pozorovat že varianta *Loop* přináší dobré zlepšení a *Tabu* zaostává.

V případě složitější úlohy nám graf 6.5 ukazuje, že varianta SQA Be již není lepší pro horší řešení a je vždy výhodnější využít SQA BeLo nebo BeLoTa. Také můžeme pozorovat že přidáním *Tabu* modifikace není výsledná optimalizace příliš ovlivněna a mírné zlepšení může být způsobeno statistickou odchylkou.



Obrázek 6.5: Empirická distribuční funkce pro jednotlivé srovnávané algoritmy.

Při větší obtížnosti úlohy se zvyšuje zlepšení použitím varianty *Loop*, kde tento přínos je pravděpodobně způsoben větším lokálním prohledáváním a nalezením lepšího řešení pro okolní repliky. Varianta se 100 městy měla pravděpodobně příliš malý stavový prostor k projevu této modifikace. Oproti tomuto *Tabu* má jen nepatrné zlepšení, které může být i pouze statistický odchylka. Toto je asi způsobeno velikostí stavového prostoru, kde redukce prostoru je malá vzhledem k velikosti stavového prostoru.

6.3 5000 měst

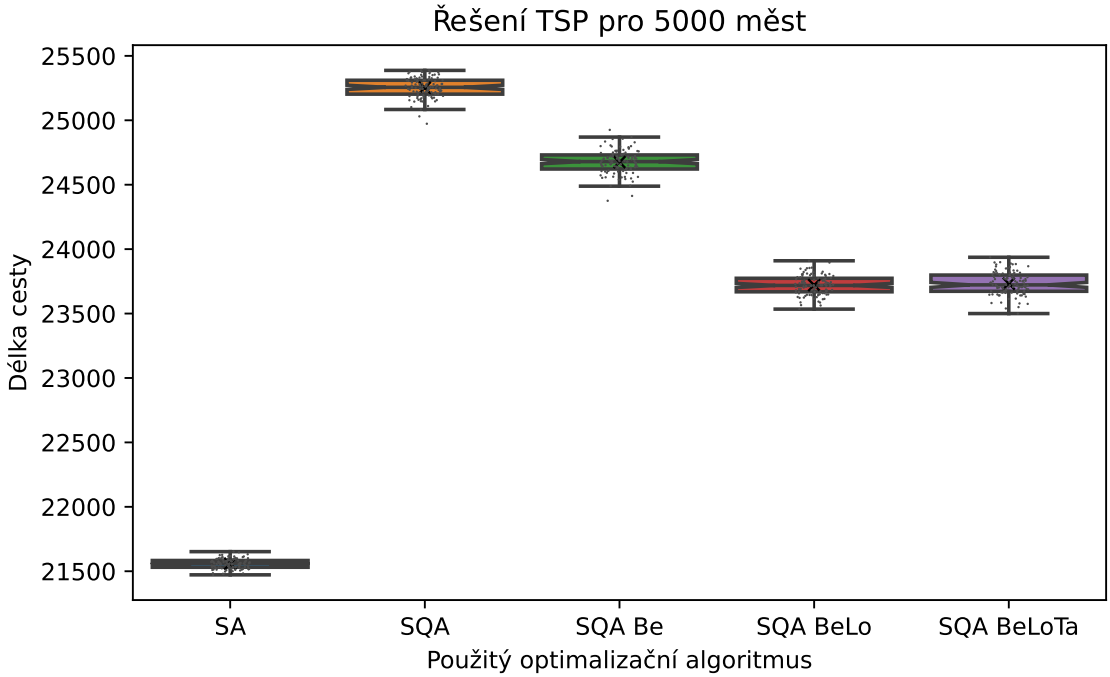
Poslední srovnání na úloze obchodního cestujícího je o velikosti 5000 měst, kde algoritmům bylo poskytnuto 5 miliard iterací k optimalizaci. Tabulka 6.4 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počáteční teplota ($1/\beta$)	Počet iterací před změnou teploty	Rychlost ochlazování
SA	250	100	0.99995

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.015	1.5	0.999975	15
SQA Be	0.015	1.5	0.999975	15
SQA BeLo	0.015	1.5	0.999975	15
SQA BeLoTa	0.015	1.5	0.999975	15

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*
1	500	2.5%

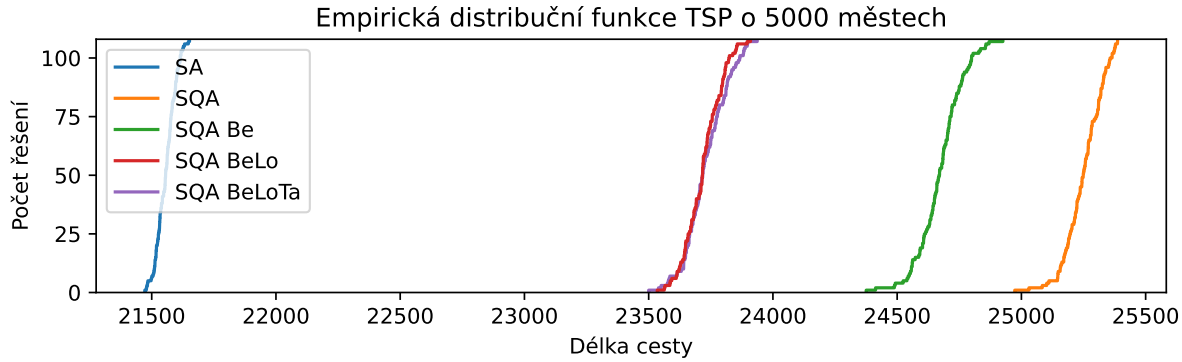
Tabulka 6.4: Parametry pro porovnávané algoritmy.



Obrázek 6.6: Statistické vyhodnocení nejkratších cest nalezeních za pomocí simulovaného žíhání, simulovaného kvantového žíhání a modifikací SQA pro problém obchodního cestujícího o 5000 městech.

Z optimalizace 5000 měst na grafu 6.6 již můžeme jasně vidět, že SQA se výrazně zhoršuje oproti SA. Nicméně zde již také jde jasně pozorovat přínos modifikací *Best* a *Loop*. Medián *Tabu* ukazuje nepatrné zhoršení optimalizace.

Stejně jako v případě 1000 měst, tak tady můžeme vidět na grafu 6.7, že varianta SQA BeLo je nejlepší z kategorie SQA variant, ale znatelně zaostává za SA.



Obrázek 6.7: Empirická distribuční funkce pro jednotlivé srovnávané algoritmy.

Výsledky z optimalizace na 5000 městech odpovídají úvahám v předchozích sekcích pro TSP se 100 městy a 1000 městy.

6.4 Zhodnocení

V optimalizaci délky cesty u problému obchodního cestujícího simulované kvantové žíhání sice nedosahuje výsledků jako klasické žíhání, ale můžeme pozorovat zlepšení algoritmu SQA po přidání rozšířeních. Rozšíření *Best* dosahující lepších výsledků než originální SQA i na malém počtu měst a rozšíření *Loop* dosahuje výrazného zlepšení při větším počtu měst. Bohužel varianta *Tabu* se ukázala dobrá pouze pro zadání s malým počtem měst a pro těžší úlohy má dokonce negativní vliv.

Ohledně časové náročnosti je jedna iterace klasického simulovaného žíhání kratší. Toto je způsobeno vyšší režií SQA při porovnávání dvou cest k výpočtu kinetické energie. Výpočty trvaly od několika vteřin pro jednodušší variantu zadání k půl hodině pro 5000 měst pro SQA.

Kapitola 7

Výsledky maximální splnitelnosti booleovské formule

Na úloze hledání maximální splnitelnost booleovské formule je porovnáván algoritmus simulovaného kvantového žíhání (SQA), algoritmus klonální selekce (CS) a čtyři modifikace SQA:

- Ovlivnění nejlepším řešením s názvem *Best* ve vyhodnocení.
- Delší optimalizace replik s názvem *Longer* ve vyhodnocení.
- Využití tabu meta-heuristiky s názvem *Tabu* ve vyhodnocení.
- Převzatí části řešení z nejlepšího řešení s názvem *From* ve vyhodnocení.

Pro vyhodnocení byl využit superpočítač Barbora[1]. Výpočetní uzly obsahují 2 osmnáctijádrové procesory Intel Cascade Lake s výkonem 2.6 GHz. K vytvoření srovnání bylo provedeno 500 nezávislých běhů každého algoritmu nebo varianty na 3 uzlech superpočítače.

Srovnání na úloze MAX-SAT je rozděleno na dvě části. První část srovnává algoritmy na jednodušších úlohách, kde je poměrně snadné nalézt nejlepší řešení. Tedy ohodnocení splňující maximální počet klauzulí. Druhá část bude obsahovat zadání, kde je obtížnější nalézt nejlepší řešení, a tedy se bude jednat hlavně o snahu dosáhnout co nejblíže maximální splnitelnosti. Tato zadání jsou získána z benchmarků pro MAX-SAT z [31].

Algoritmus CS ve vyhodnocení odpovídá klonální selekci. SQA odpovídá simulovanému kvantovému žíhání bez modifikací. Pro modifikace jsou ve vyhodnocení použity upravené varianty SQA s jednou nebo více modifikacemi popsané tabulkou 7.1 níže:

[1] https://www.it4i.cz/infrastruktura/barbora

	Modifikace			
Algoritmus	*Best*	*Loop*	*Tabu*	*From*
SQA	ne	ne	ne	ne
SQA Be	ano	ne	ne	ne
SQA BeLo	ano	ano	ne	ne
SQA BeLoTa	ano	ano	ano	ne
SQA BeLoTaF	ano	ano	ano	ano

Tabulka 7.1: Tabulka znázorňující použité varianty algoritmu SQA při srovnání algoritmů v následující sekcích.

7.1 Jednodušší problémy

Jednodušší problémy jsou typu MAX-2SAT a MAX-3SAT s počtem proměnných od 110 do 200 a počtem klauzulí od 1100 do 2400. Přesnější specifikace zadání je v jednotlivých podsekcích zabývajících se vyhodnocením těchto porovnání. U těchto jednodušších problémů budeme porovnávat počet iterací nutný k nalezení maximální splnitelnosti formule.

7.1.1 MAX-2SAT, 160 proměnných a 2400 klauzulí

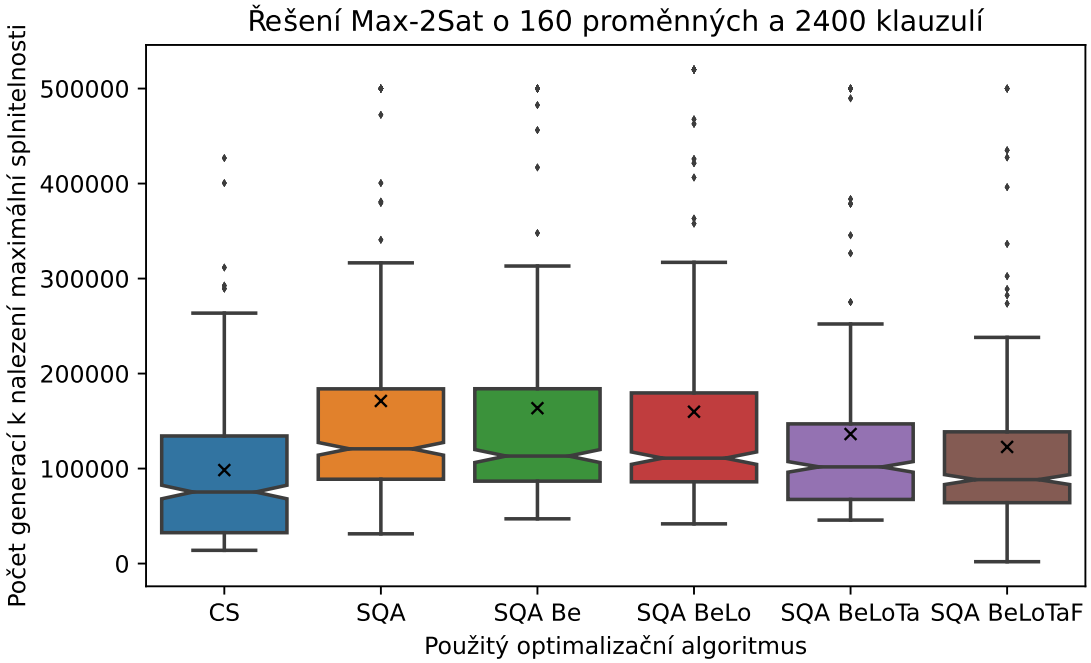
První zadání je problém MAX-2SAT se 160 proměnnými a 2400 klauzulemi. Tabulka 7.2 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	12	20	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.06	2.5	1.0	15
SQA Be	0.06	2.5	1.0	15
SQA BeLo	0.06	2.5	1.0	15
SQA BeLoTa	0.06	2.5	1.0	15
SQA BeLoTaF	0.06	2.5	1.0	15

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	80	10 %	1 %

Tabulka 7.2: Parametry pro porovnávané algoritmy.



Obrázek 7.1: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-2SAT o 160 proměnných a 2400 klauzulí.

Z optimalizace tohoto problému lze pozorovat na grafu 7.1, že klonální selekce dosahuje lepších výsledků než SQA. Dopad na zlepšení má pouze přidání *Tabu* a varianty *From*, kde ostatní varianty jsou téměř beze změny. Největší přínos na algoritmus má varianta SQA BeLoTaF, která přidává schopnost přebírat část řešení z nejlepší repliky, kde tato varianta oproti CS už není o tolik horší.

Pro toto specifické zadání je CS lepší než SQA a jeho varianty. Při bližším prozkoumání tohoto zadání bylo zjištěno, že SQA má problém s uváznutím v lokálních minimech tohoto zadání, kde CS v průběhu optimalizace přidává zcela nové B buňky umožnující nalézt jiné řešení. Tento jev jde pozorovat vysokým rozdílem mezi mediánem a průměrem u box plotů SQA. Z variant SQA se daří *Tabu* nejvíce, ale i varianta *From* dosahuje znatelného zlepšení přebíráním části výsledků nejlepší repliky, což pravděpodobně vede k rychlejší optimalizaci a nižší pravděpodobnosti uváznutí v lokálním minimu.

7.1.2 MAX-2SAT, 200 proměnných a 1800 klauzulí

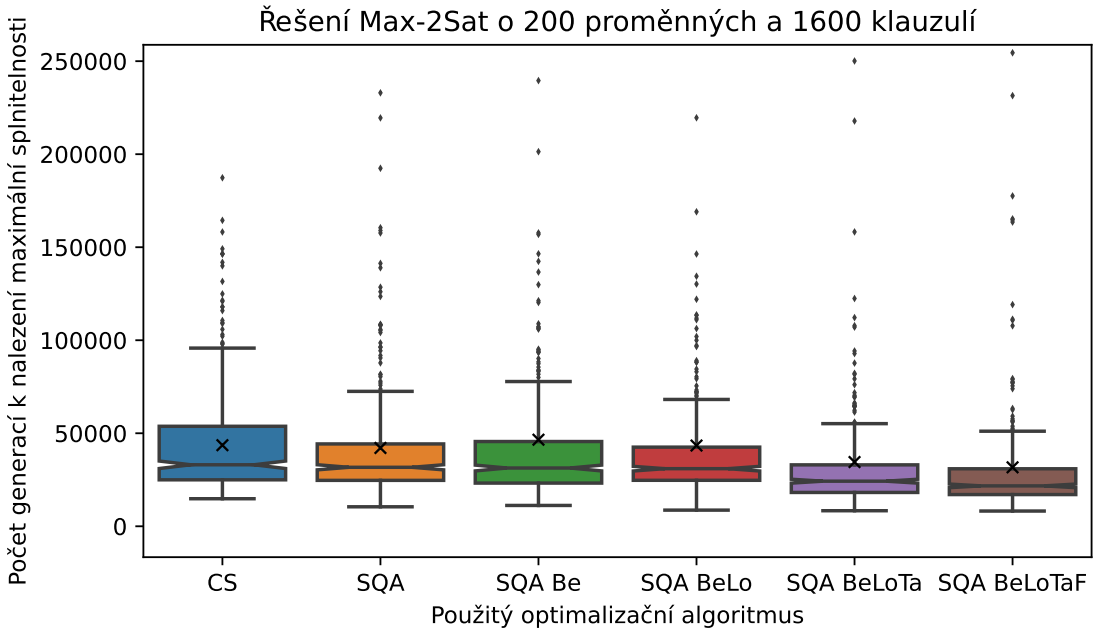
Další zadání je problém MAX-2SAT se 200 proměnnými a 1800 klauzulemi. Tabulka 7.3 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	10	18	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.18	2.5	1.0	5
SQA Be	0.18	2.5	1.0	5
SQA BeLo	0.18	2.5	1.0	5
SQA BeLoTa	0.18	2.5	1.0	5
SQA BeLoTaF	0.18	2.5	1.0	5

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	40	10 %	1 %

Tabulka 7.3: Parametry pro porovnávané algoritmy.



Obrázek 7.2: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-2SAT o 200 proměnných a 1600 klauzulí.

V tomto zadání můžeme na grafu 7.2 pozorovat, že základní varianta SQA je mírně lepší než CS. Modifikace SQA je mírně horší, ale ostatní varianty jsou zlepšením nebo neutrální

změna. Výrazné zlepšení je opět varianta přidávající *Tabu* a varianta *From* také přináší zlepšení.

Při porovnání v tomto zadání úlohy je i základní SQA lepší než CS z důvodu menšího počtu lokálních minim umožnující snížit nutný počet replik, a tedy snížit počet iterací nutný k nalezení řešení. U variant SQA je přidání *Tabu* opět výrazné zlepšení optimalizace kvůli snížení velikosti prohledávaného prostoru. Samotná varianta *Best* přináší mírné zhoršení, ale při experimentu s odděláním této varianty z BeLoTa a BeLoTaF výsledná optimalizace měla horší výsledky.

7.1.3 MAX-3SAT, 110 proměnných a 1100 klauzulí

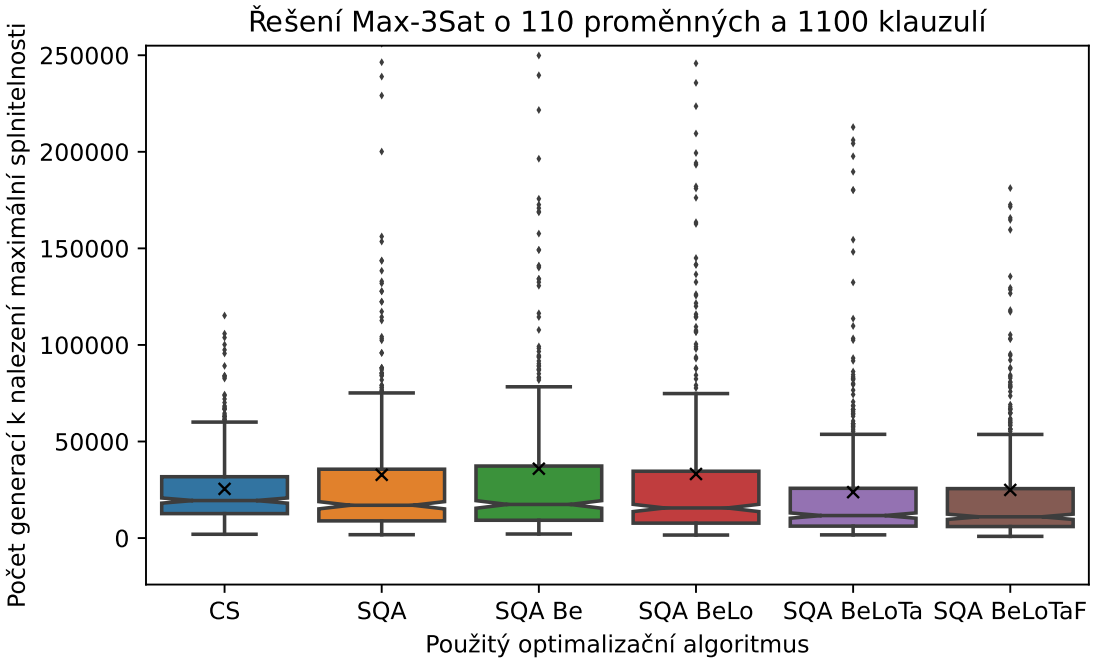
Poslední zadání ze sady jednodušších zadání je tentokrát problém MAX-3SAT se 110 proměnnými a 1100 klauzulemi. Tabulka 7.4 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	11	19	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.11	2.5	1.0	6
SQA Be	0.11	2.5	1.0	6
SQA BeLo	0.11	2.5	1.0	6
SQA BeLoTa	0.11	2.5	1.0	6
SQA BeLoTaF	0.11	2.5	1.0	6

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	50	10 %	1 %

Tabulka 7.4: Parametry pro porovnávané algoritmy.



Obrázek 7.3: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-3SAT o 110 proměnných a 1100 klauzulí.

V poslední jednoduché úloze lze na grafu 7.3 pozorovat stejné výsledky jako v předchozích dvou zadáních. SQA má problémy s lokálními minimy jak lze vidět z rozdílu průměru a mediánu. I přesto ale dosahuje SQA lepších výsledků než CS a především varianta *Tabu* přináší další zlepšení. Varianta SQA BeLo také přináší mírné zlepšení.

Výsledky porovnání na úloze MAX-3SAT jsou téměř shodné s předešlým zadáním, kde SQA má lepší medián počtu iterací nutný k nalezení řešení, ale průměr je horší kromě varianty SQA BeLoTa, který redukuje prohledávaný prostor a dosahuje průměru téměř stejné optimalizace jako CS. Ostatní varianty dosahují zanedbatelné změny v optimalizaci pravděpodobně z důvodu velkého množství lokálních minim, kde může SQA uváznout.

7.2 Obtížnější problémy

Pro obtížnější zadání jsou zvoleny zadání typu MAX-4SAT s rostoucím počtem proměnných a klauzulí. Počet proměnných je od 192 proměnných do 1024 a počet klauzulí od 432 do 2816. Na rozdíl od jednodušších problému, kde se dalo poměrně snadno najít nejlepší řešení, tak zde budeme porovnávat algoritmy na základě nalezených řešení jednotlivých běhů, jelikož pro tyto úlohy není možné nalézt ohodnocení splňující maximální počet klauzulí v rozumném čase.

7.2.1 MAX-4SAT, 192 proměnných a 432 klauzulí

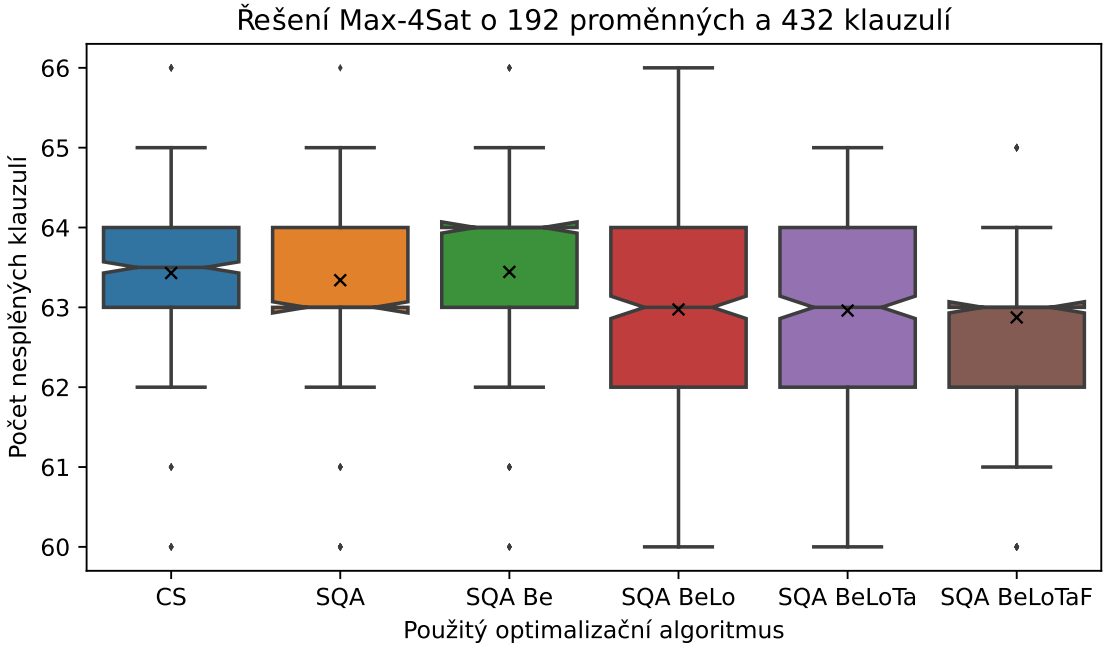
První zadání má 192 proměnných a 432 klauzulí. Algoritmy byly spuštěny po dobu 25 tisíce iterací. Tabulka 7.5 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	12	20	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.09	2.5	1.0	10
SQA Be	0.09	2.5	1.0	10
SQA BeLo	0.09	2.5	1.0	10
SQA BeLoTa	0.09	2.5	1.0	10
SQA BeLoTaF	0.09	2.5	1.0	10

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	80	10 %	1 %

Tabulka 7.5: Parametry pro porovnávané algoritmy.



Obrázek 7.4: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-4SAT o 192 proměnných a 432 klauzulí.

Z grafu 7.4 lze vidět, že základní varianta SQA dosahuje lepších výsledků než CS. Rozšíření *Best* ukazuje mírné zhoršení. Algoritmus SQA BeLo sice zvětšuje rozpětí kvality řešení, ale dosahuje výrazně lepšího průměrného řešení než základní varianta SQA a CS. *Tabu* přináší mírné zlepšení omezením počtu velice špatných řešení. Varianta *From* způsobuje nižší rozpětí jednotlivých řešení, ale za cenu menšího počtu velice dobrých řešení.

V těžších MAX-SAT úlohách už i základní SQA dosahuje lepších výsledků než CS. Varianta *Best* způsobuje horší řešení, což je pravděpodobně způsobeno nízkým počtem iterací, kde repliky nemají tolik času ke konvergenci k nejlepšímu řešení. Nicméně stejně jak bylo u jednodušších zadání, tak ostatní varianty algoritmů dosahovaly horších výsledků bez této varianty. Varianta *Loop* přináší výrazné zlepšení oproti ostatním variantám, kde umožňuje replikám více lokálního prohledávání. *Tabu* omezuje stavový prostor a z tohoto důvodu je pravděpodobně dosaženo nižšího počtu špatných řešení. Varianta *From* snižuje rozptyl řešení kvůli přebírání části řešení z nejlepší repliky, a tedy způsobuje konvergenci k tomuto řešení, které se bude blížit průměru box plotu.

7.2.2 MAX-4SAT, 448 proměnných a 1120 klauzulí

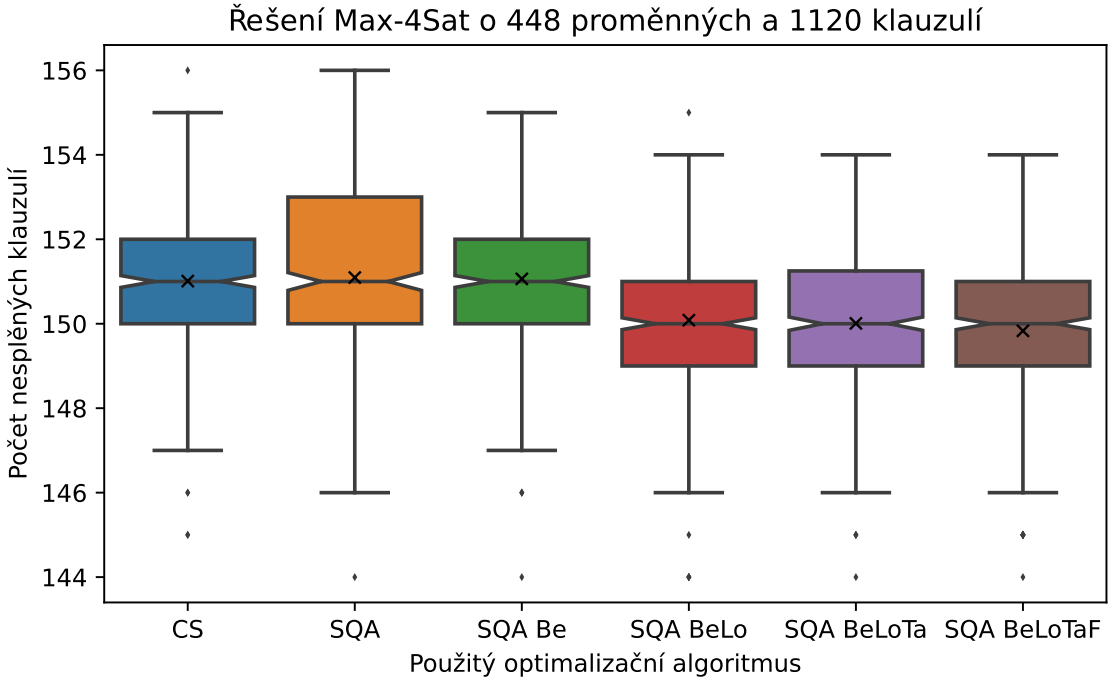
Další zadání má 448 proměnných a 1120 klauzulí. Algoritmy byly spuštěny po dobu 300 tisíc iterací. Tabulka 7.6 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	12	20	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.09	2.5	1.0	10
SQA Be	0.09	2.5	1.0	10
SQA BeLo	0.09	2.5	1.0	10
SQA BeLoTa	0.09	2.5	1.0	10
SQA BeLoTaF	0.09	2.5	1.0	10

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	90	10 %	1 %

Tabulka 7.6: Parametry pro porovnávané algoritmy.



Obrázek 7.5: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-4SAT o 448 proměnných a 1120 klauzulí.

Výsledky na tomto zadání zobrazené na grafu 7.5 jsou podobné jako v předchozím zadání, ale tentokrát je CS mírně lepší než základní SQA, ale varianta algoritmu SQA Be již není horší než základní. *Loop* opět vede k největšímu zlepšení z variant SQA. *Tabu* má lepší průměrné řešení stejně jako modifikace *From*.

V tomto zadání již můžeme vidět, že varianta *Best* zlepšuje optimalizaci až při větším počtu iterací. Ostatní výsledky jsou stejné jako v předchozím zadání, kromě většího počtu horších řešení u algoritmu SQA BeLoTa, kde se ale pravděpodobně jedná pouze o statistickou chybu.

7.2.3 MAX-4SAT, 1024 proměnných a 2816 klauzulí

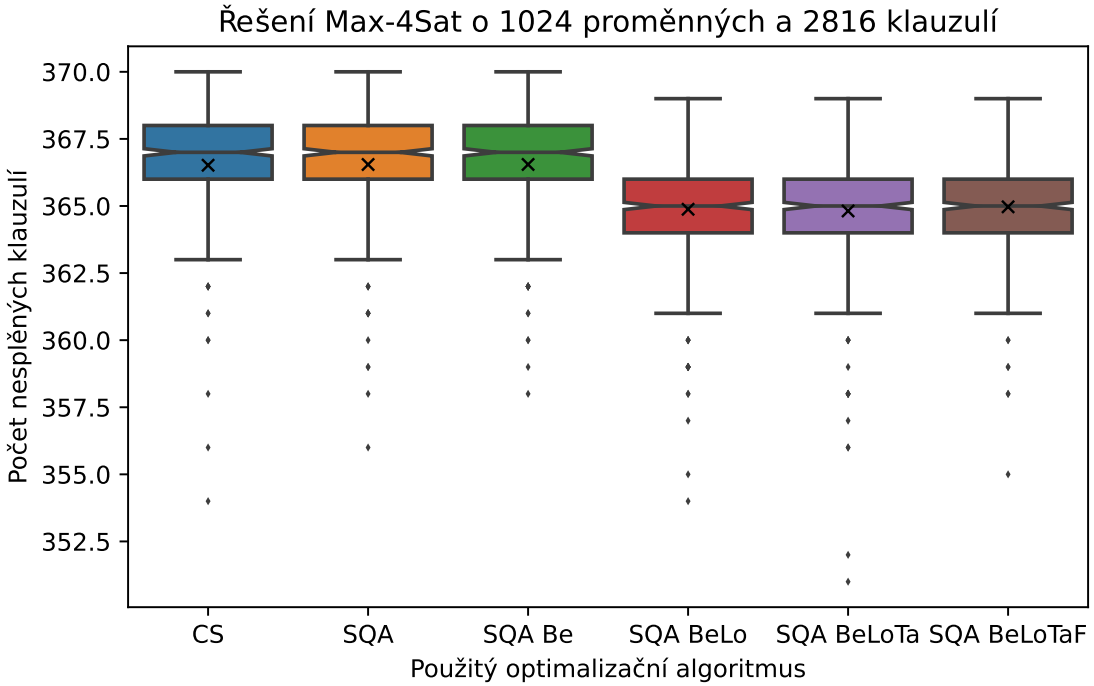
Poslední srovnání algoritmů na úloze MAX-4SAT má 1024 proměnných a 2816 klauzulí. Algoritmy byly spuštěny po dobu 1 milionů iterací. Tabulka 7.7 obsahuje nastavení parametrů pro různé optimalizační algoritmy použité v experimentech.

Algoritmus	Počet rodičů	Počet potomků	Min. šance mutace	Max. šance mutace	Počet náhodných buněk v generaci	Práh lokálního prohledávání
CS	12	20	1	3	5	10

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.09	2.5	1.0	10
SQA Be	0.09	2.5	1.0	10
SQA BeLo	0.09	2.5	1.0	10
SQA BeLoTa	0.09	2.5	1.0	10
SQA BeLoTaF	0.09	2.5	1.0	10

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Velikost *Tabu*	Pravděpodobnost *From*
5	100	10 %	1 %

Tabulka 7.7: Parametry pro porovnávané algoritmy.



Obrázek 7.6: Statistické vyhodnocení klonální selekce, simulovaného kvantového žíhání a modifikací SQA pro problém MAX-4SAT o 1024 proměnných a 2816 klauzulí.

Výsledky posledního srovnání zobrazené na grafu 7.6 vypadají téměř stejně jako v předchozím zadání. Hlavní zlepšení přináší varianta *Loop* s menším zlepšením u modifikaci *Tabu*. Algoritmus SQA BeLoTaF je mírně horší než bez přidané modifikace.

Výsledky posledního srovnání znova ukazují že varianta *Loop* přináší hlavní zlepšení a ostatní varianty dosahují jen mírné změny, která může být i v rámci statistické odchylky. Nicméně varianta *Tabu* našla malý počet velice dobrých řešení v porovnání s ostatními variantami, což je pravděpodobně způsobeno redukcí stavového prostoru.

7.3 Zhodnocení

V úloze hledání maximální splnitelnosti booleovské formule dosahovalo simulované kvantové žíhání poměrně podobných výsledků jako klonální selekce. Přidáním varianty *Loop* a varianty *Tabu* se zlepšila optimalizace pomocí SQA, kde tato optimalizace byla ve většině zadáních MAX-SAT lepší než CS. Ostatní varianty měly většinou neutrální dopad na optimalizaci, kromě varianty *From*, která na některých úlohách vedla k menšímu zlepšení.

Časová náročnost je v řádu několika vteřin nebo méně pro jednodušší zadání a několik desítek vteřin pro těžší zadání. Jedna iterace SQA na superpočítači trvala okolo déle než jedna iterace ES.

Kapitola 8

Výsledky hledání pravidel celulárního automatu

Na úloze hledání pravidel pro celulární automat jsou porovnávány algoritmy simulovaného kvantového žíhání (SQA), evoluční strategie (ES) a tři modifikace SQA:

- Ovlivnění nejlepším řešením s názvem *Best* ve vyhodnocení.
- Delší optimalizace replik s názvem *Longer* ve vyhodnocení.
- Převzatí části řešení z nejlepšího řešení s názvem *From* ve vyhodnocení.

Pro vyhodnocení byl využit superpočítač Barbora[1]. Výpočetní uzly obsahují 2 osmnáctijádrové procesory Intel Cascade Lake s výkonem 2.6 GHz. K vytvoření srovnání bylo provedeno 108 nezávislých běhů každého algoritmu nebo varianty na 3 uzlech superpočítače.

Srovnání na úloze hledání pravidel pro CA bylo provedeno na dvou vzorech, které budou popsány v jednotlivých sekcích. Z důvodu omezení výpočetního času na superpočítači byly provedeny běhy po maximální dobu jedné hodiny a pro srovnání byly použity výsledky algoritmů do specifického počtu iterací kterých dosáhly všechny běhy. Tedy běhy jsou ohraničeny počtem iterací dosaženým "nejpomalejším" během pro daný vzor.

Algoritmus ES ve vyhodnocení odpovídá evoluční strategii [6]. SQA odpovídá simulovanému kvantovému žíhání bez modifikací. Pro modifikace jsou ve vyhodnocení použity upravené varianty SQA s jednou nebo více modifikace popsané tabulkou 8.1 níže:

	Modifikace		
Algoritmus	*Best*	*Loop*	*From*
SQA	ne	ne	ne
SQA Be	ano	ne	ne
SQA BeLo	ano	ano	ne
SQA BeLoF	ano	ano	ano

Tabulka 8.1: Tabulka znázorňující použité varianty algoritmu SQA při srovnání algoritmů v následující sekcích.

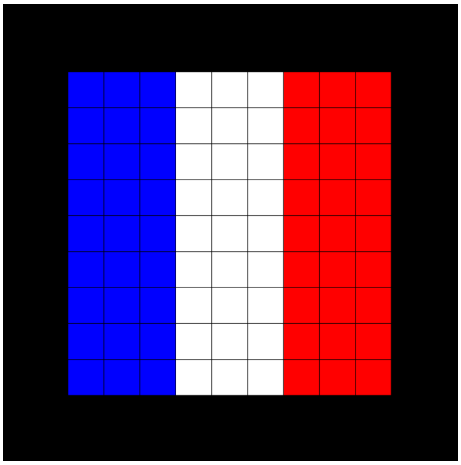
[1] https://www.it4i.cz/infrastruktura/barbora

8.1 Vzor francouzské vlajky

První vzor odpovídá francouzské vlajce. Tento vzor je velikosti 9×9 buněk s 2 buňkami na okrajích vzoru a je zobrazen na obrázku 8.1. Doba hledání pravidel byla na základě nejpomalejšího běhu stanovena na 1.384 milionů iterací. Parametry pro celulární automat jsou popsány tabulkou 8.2.

Počet pravidel	Počet stavů	Maximální počet kroků
100	32	80

Tabulka 8.2: Tabulka znázorňující nastavení CA pro vzor vlajka.



Obrázek 8.1: Cílový vzor pro toto zadání hledání pravidel. Vzor je francouzská vlajka 9×9 s okrajem 2 buněk.

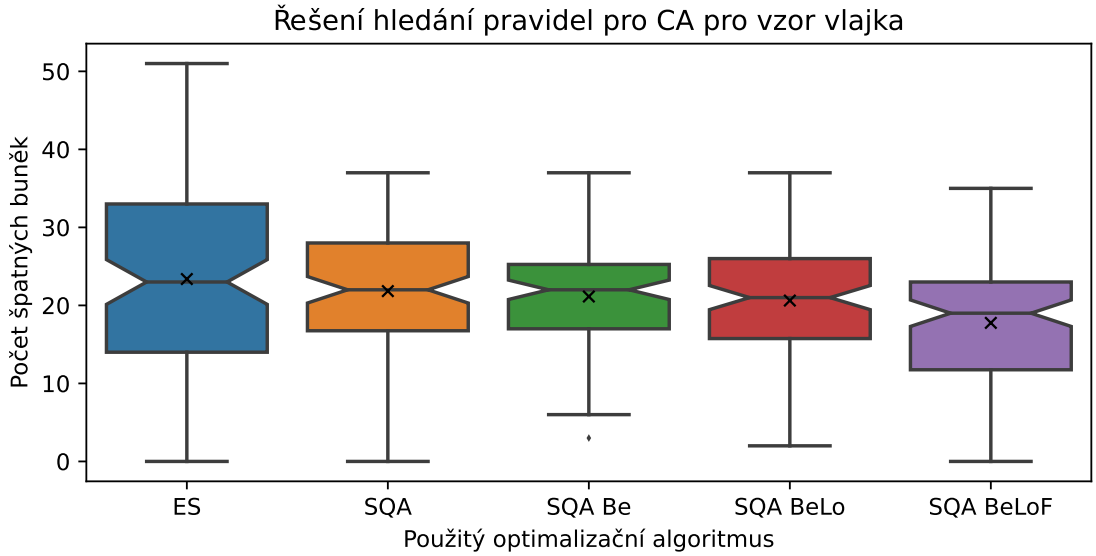
Následující tabulky 8.3 popisují parametry ES, SQA a modifikací:

Algoritmus	Počet rodičů	Počet potomků
ES	4	12

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.08	4.5	0.9998	10
SQA Be	0.08	4.5	0.9998	10
SQA BeLo	0.08	4.5	0.9998	10
SQA BeLoF	0.08	4.5	0.9998	10

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Pravděpodobnost *From*
5	5	20 %

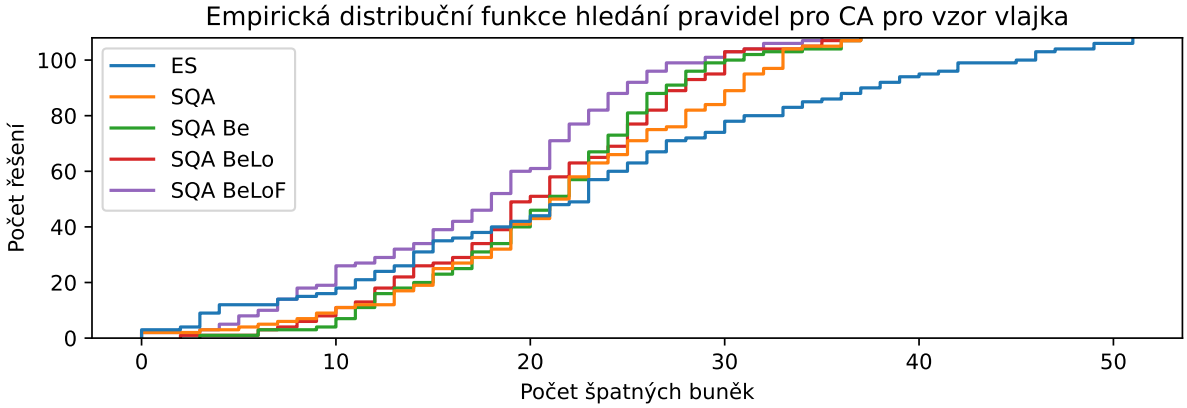
Tabulka 8.3: Parametry pro porovnávané algoritmy.



Obrázek 8.2: Statistické vyhodnocení evoluční strategie, simulovaného kvantového žíhání a modifikací SQA pro problém hledání pravidel pro CA pro vzor vlajka.

Na grafu vyhodnocení 8.2 už i základní varianta SQA dosahuje mírně lepších výsledků než ES. Vylepšení *Best* snižuje počet perfektních řešení, ale i přesto dosahuje o trochu lepšího průměru než SQA. Varianty *Loop* a *From* opět dosahují perfektních řešení, kde SQA BeLoF vychází statistky nejlépe ze všech vyhodnocovaných algoritmů.

Na grafu distribuční funkce 8.3 můžeme vidět, že pravděpodobnost nalézt velice dobré řešení pro ES je lepší nebo srovnatelná s SQA a jeho variantami, ale tento algoritmus má také vysokou pravděpodobnost nalézt velice špatné řešení, která jsou i z velké části horší něž nejhorší řešení nalezená za pomocí SQA.



Obrázek 8.3: Empirická distribuční funkce pro jednotlivé srovnávané algoritmy.

Oproti základní variantě SQA je ES mírně horší v průměru a značný počet řešení je velice špatný, ale má i vetší množství velice dobrých řešení oproti SQA. Toto je pravděpodobně způsobeno vysokou mutací potomků vytvořených z horších rodičů v ES. Varianta *Best* poté přináší mírné zlepšení za cenu menšího počtu velice dobrých řešení. Tento jev je shodný s výsledky v ostatních řešených úlohách, kde tato variant snižuje rozptyl kvality řešení. Delší čas strávený na vylepšení jedné repliky před přechodem na další poté způsobuje nelezení

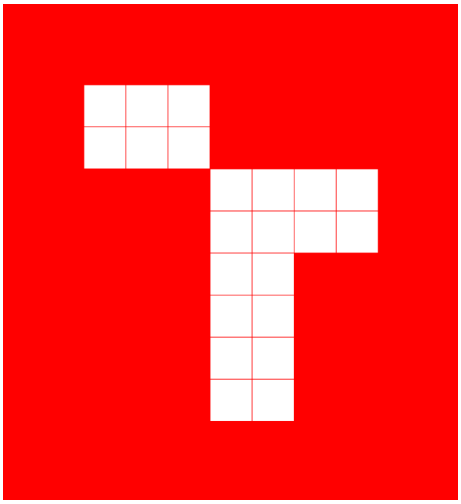
lepších řešení ve spojení s *Best*. Varianta *From* přebírá části pravidel z nejlepší repliky a tak pravděpodobně umožňuje převzít dobře fungující pravidla do ostatních replik.

8.2 Vzor loga VUT

Druhý vzor je logo univerzity Vysokého učení technického v Brně. Doba trvání hledání pravidel byla na základě nejpomalejšího běhu stanovena na 10.7 milionů iterací. Tento vzor je velikosti 7×8 buněk s 2 buňkami na okrajích. Vzoru zobrazen na obrázku 8.4.
Parametry pro celulární automat jsou popsány tabulkou 8.2.

Počet pravidel	Počet stavů	Maximální počet kroků
100	32	80

Tabulka 8.4: Tabulka znázorňující nastavení CS pro vzor logo.



Obrázek 8.4: Cílový vzor pro toto zadání hledání pravidel. Vzor je logo univerzity VUT s velikostí 7×8 a okrajem 2 buněk.

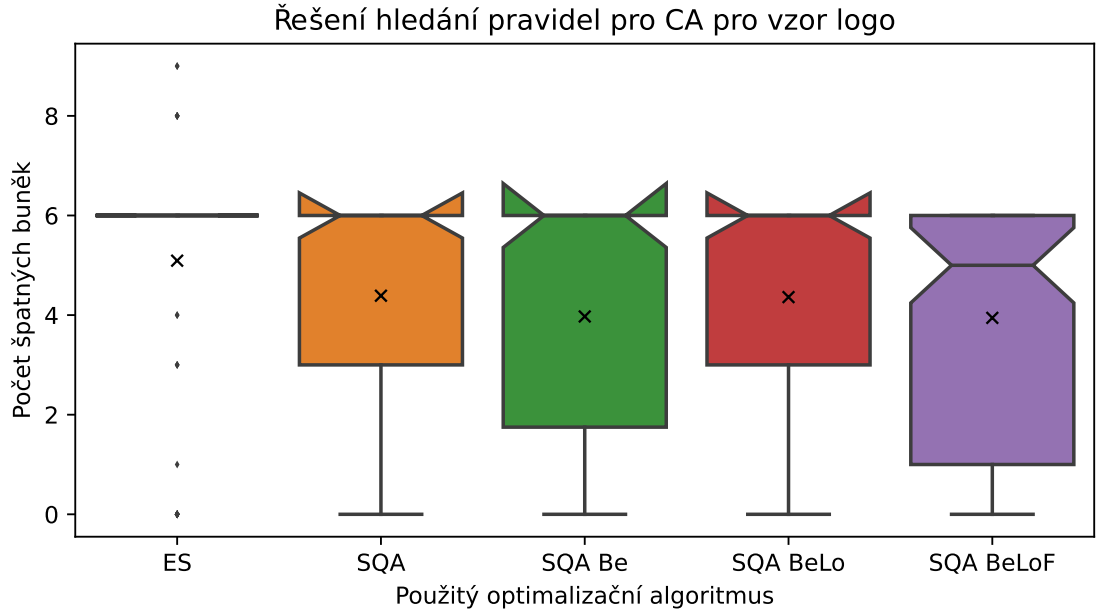
Následující tabulky 8.5 popisují parametry ES, SQA a modifikací:

Algoritmus	Počet rodičů	Počet potomků
ES	4	12

Algoritmus	Teplota (T)	Síla pole (Γ)	Rychlost změny pole	Počet replik (P)
SQA	0.08	4.5	0.9998	10
SQA Be	0.08	4.5	0.9998	10
SQA BeLo	0.08	4.5	0.9998	10
SQA BeLoF	0.08	4.5	0.9998	10

Kinetická energie pro *Best* repliku	Počet iterací pro repliku u *Loop*	Pravděpodobnost *From*
5	5	20 %

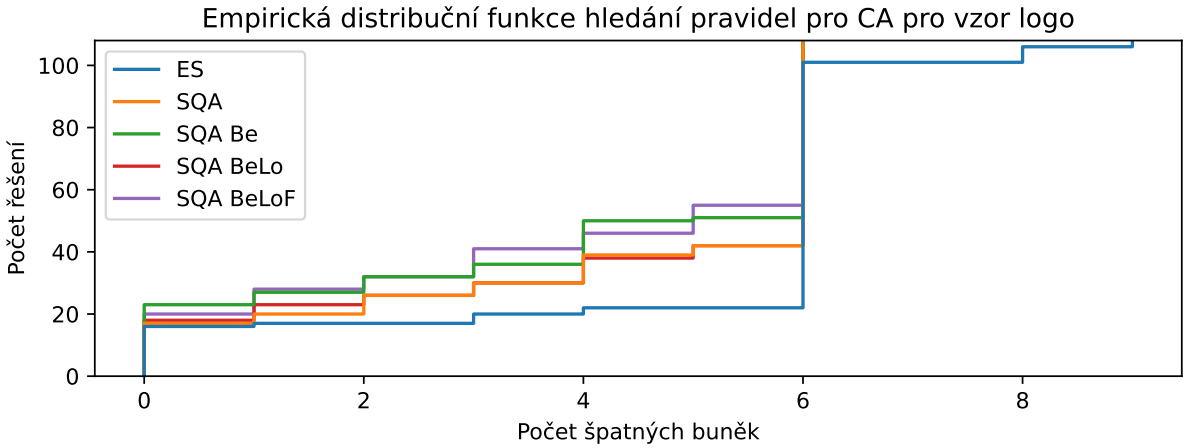
Tabulka 8.5: Parametry pro porovnávané algoritmy.



Obrázek 8.5: Statistické vyhodnocení evoluční strategie, simulovaného kvantového žíhání a modifikací SQA pro problém hledání pravidel pro CA pro vzor logo.

Graf vyhodnocení 8.5 ukazuje, že všechny varianty SQA jsou lepší než ES při srovnání pro tento vzor. Algoritmy SQA Be a BeLoF dosahují lepších výsledků kromě varianty *Loop*, která má negativní vliv na optimalizaci při porovnání s algoritmem bez této modifikace.

Na grafu distribuční funkce 8.6 můžeme vidět, že SQA je striktně lepší než ES. Také můžeme pozorovat výrazné lokální minimum se 6 špatnými buňkami.



Obrázek 8.6: Empirická distribuční funkce pro jednotlivé srovnávané algoritmy.

Pro tento vzor je simulované kvantové žíhání lepší než evoluční strategie. Varianty *Best* a *From* značně zlepšují optimalizaci pravděpodobně z důvodu dostání řešení z lokálního minima. Toto minimum je logo VUT bez levé horní části loga, kde tento úsek loga není spojený s hlavní částí, jelikož pravidla CA neporovnávají diagonální buňky. Pravděpodobně z tohoto důvodu varianta *Loop* dosahuje horších výsledků, jelikož zvyšuje lokální prohledávání, které nejspíše uvázne v lokálním minimu.

8.3 Zhodnocení

Základní verze simulovaného kvantového žíhání a evoluční strategie má srovnatelnou kvalitu hledání řešení. Varianta *Best* již dosahuje lepších výsledků než evoluční strategie. Varianta *Loop* pravděpodobně dosahuje lepších řešení na úlohách se spojitým vzorem bez velkých lokálních minim. Nakonec varianta *From* dosahuje výrazně lepších výsledků při hledání pravidel.

Jedna zmínka ke srovnání ES a SQA. Srovnání v této práci je provedeno na základě stejného počtu iterací pro všechny algoritmy a varianty. Nicméně délka jedné iterace pro ES je výrazně delší než u SQA z důvodu kopírování při tvorbě potomků v evoluční strategii, kde SQA pracuje in situ.

Kapitola 9

Závěr

Cílem této práce bylo zvolit a implementovat optimalizační algoritmus inspirovaný z kvantové fyziky. Poté navrhnout modifikace zvoleného algoritmu a vytvořit srovnávací studii na sadě zvolených úloh oproti běžným optimalizačním metodám.

Jako kvantově inspirovaný algoritmus pro tuto práci byl vybrán algoritmus simulovaného kvantového žíhání mající svoji inspiraci z jevu kvantového tunelování a klasického simulovaného žíhání. K tomuto algoritmu byly navrženy čtyři modifikace za účelem vylepšit optimalizační vlastnosti tohoto algoritmu. Pro vytvoření srovnávací studie s vyhodnocením simulovaného kvantového žíhání a navržených modifikací byly zvoleny tři odlišné úlohy s běžnými optimalizačními metodami pro srovnání. Tyto úlohy jsou: problém obchodního cestujícího s klasickým simulovaným žíháním, problém maximální splnitelnosti booleovské formule s klonální selekcí a hledání pravidel pro celulární automat s evoluční strategií.

Původní implementace simulovaného kvantového žíhání vykazuje horší optimalizační vlastnosti než běžný optimalizační algoritmus na problému obchodního cestujícího, srovnatelné výsledky na problému maximální splnitelnosti booleovské formule a lepší výsledky při hledání pravidel pro celulární automat. Navržené modifikace simulovaného kvantového žíhání byly při srovnání s původní implementací vždy efektivnější. Pro problém obchodního cestujícího ale i modifikovaný algoritmus dosahuje horší optimalizace než klasické simulované žíhání. Na ostatních úlohách modifikovaný algoritmus dosahuje lepších výsledků než vybrané běžné optimalizační metody.

Tato práce se zabývala pouze jedním algoritmem z oblasti kvantově inspirovaných algoritmů, kde se můžou nacházet i další algoritmy nebo principy, které by mohly umožnit lepší optimalizaci než dosavadní běžné optimalizační metody. Další možné pokračování v této práci je vytvořit další modifikace založené na heuristice ke konkrétním problémům využívající specifické vlastnosti, které kvantově inspirované optimalizační algoritmy přináší díky jejich pseudo-paralelní povaze vycházející z kvantového jevu superpozice.

Literatura

[1] Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. et al. State-of-the-art in artificial neural network applications: A survey. *Heliyon*. 2018, sv. 4, č. 11, s. e00938. ISSN 2405-8440. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2405844018332067.

[2] AL Madi, N. a Khader, A. T. The Traveling Salesman Problem as a Benchmark Test for a Social-Based Genetic Algorithm. *Journal of Computer Science*. Říjen 2008, sv. 4.

[3] Ali Roghanian, R. N. *B Cells* [online]. 2021 [cit. 2022-04-10]. Dostupné z: https://www.immunology.org/public-information/bitesized-immunology/cells/b-cells.

[4] Anderson, B. *Simulated Annealing* [online]. Carnegie Mellon University, School of Computer Science [cit. 2022-04-08]. Dostupné z: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html.

[5] Battaglia, D. A., Santoro, G. E. a Tosatti, E. Optimization by quantum annealing: Lessons from hard satisfiability problems. *Phys. Rev. E*. American Physical Society. Jun 2005, sv. 71, s. 066707.

[6] Bidlo, M. Evolution of Cellular Automata with Conditionally Matching Rules for Image Filtering. červenec 2020, s. 1–8.

[7] Brabazon, A., O'Neill, M. a McGarraghy, S. *Natural Computing Algorithms*. Springer Publishing Company, Incorporated, 2015. ISBN 978-3-662-43630-1.

[8] Brownlee, J. Clonal selection algorithms. 2007.

[9] Bunker, P. a Jensen, P. The Planck constant of action hA. *Journal of Quantitative Spectroscopy and Radiative Transfer*. 2020, sv. 243, s. 106835. ISSN 0022-4073. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0022407319309148.

[10] Cook, S. A. The Complexity of Theorem-Proving Procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 1971, s. 151–158. STOC '71. ISBN 9781450374644.

[11] Cooper, N. G. Los Alamos Science, Fall 1983 No. 9. Říjen 1983. Dostupné z: https://www.osti.gov/biblio/5419252.

[12] Crispin, A. a Syrichas, A. Quantum Annealing Algorithm for Vehicle Scheduling. říjen 2013, s. 3523–3528.

[13] Das, A. a Chakrabarti, B. K. Colloquium: Quantum annealing and analog quantum computation. *Rev. Mod. Phys.* American Physical Society. Sep 2008, sv. 80, s. 1061–1081.

[14] Dekking, F., Kraaikamp, C., Lopuhaä, H. a Meester, L. *A Modern Introduction to Probability and Statistics*. Springer-Verlag London, 2005. ISBN 978-1-85233-896-1.

[15] Dommelen, L. van. *D.34 The adiabatic theorem* [online]. FAMU-FSU College of Engineering [cit. 2021-01-12]. Dostupné z: http://eng-web1.eng.famu.fsu.edu/~dommelen/quantum/style_a/contents.html.

[16] Farhi, E., Goldstone, J., Gutmann, S., Lapan, J., Lundgren, A. et al. A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem. *Science*. American Association for the Advancement of Science. 2001, sv. 292, č. 5516, s. 472–475. ISSN 0036-8075. Dostupné z: https://science.sciencemag.org/content/292/5516/472.

[17] Faye, J. Copenhagen Interpretation of Quantum Mechanics. Zalta, E. N., ed. *The Stanford Encyclopedia of Philosophy*. Winter 2019. Metaphysics Research Lab, Stanford University, 2019. Dostupné z: https://plato.stanford.edu/archives/win2019/entries/qm-copenhagen.

[18] Feynman, R. *Feynman Lectures on Physics*. Addison-Wesley, 1970. ISBN 978-0-201-02115-8.

[19] Fox, G. Approaches to Physical Optimization. Leden 1991, s. 153–162.

[20] Francesco Berto, J. T. *The 256 Rules* [online]. Stanford Encyclopedia of Philosophy [cit. 2022-04-15]. Dostupné z: https://plato.stanford.edu/entries/cellular-automata/supplement.html.

[21] Geman, S. a Geman, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1984, PAMI-6, č. 6, s. 721–741.

[22] Glover, F. Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms). *Discrete Applied Mathematics*. Leden 1994, sv. 49, s. 231–255.

[23] Glover, F. a Laguna, M. Tabu Search Principles. Leden 1997, s. 125–151. ISBN 978-0-7923-8187-7.

[24] Goldstein, H. *Classical Mechanics*. Addison-Wesley, 1980. ISBN 978-0-201-02918-5.

[25] Gomes, C. P., Kautz, H. A., Sabharwal, A. a Selman, B. Satisfiability Solvers. *Handbook of Knowledge Representation*. 2008.

[26] Gottlieb, J., Marchiori, E. a Rossi, C. Evolutionary Algorithms for the Satisfiability Problem. *Evolutionary computation*. Únor 2002, sv. 10, s. 35–50.

[27] Green, D. Cellular automata models in biology. *Mathematical and Computer Modelling*. 1990, sv. 13, č. 6, s. 69–74. ISSN 0895-7177. Dostupné z: https://www.sciencedirect.com/science/article/pii/089571779090010K.

[28] Heisenberg, W. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *Zeitschrift für Physik*. 1927, sv. 43, s. 172–198. ISSN 0044-3328. Dostupné z: https://link.springer.com/article/10.1007%2FBF01397280.

[29] Hořejší, V. a Bartůňková, J. *Základy imunologie*. Triton, 2005. ISBN 80-7254-686-4.

[30] Jones, B. D. M., White, D. R., O'Brien, G. O., Clark, J. A. a Campbell, E. T. Optimising Trotter-Suzuki Decompositions for Quantum Simulation Using Evolutionary Strategies. New York, NY, USA: Association for Computing Machinery, 2019, s. 1223–1231. GECCO '19. ISBN 9781450361118.

[31] Josep Argelich, F. M. J. P. *Eleventh Max-SAT Evaluation* [online]. [cit. 2022-04-22]. Dostupné z: http://maxsat.ia.udl.cat/introduction/.

[32] Kennard, E. H. Zur Quantenmechanik einfacher Bewegungstypen. *Zeitschrift für Physik*. 1927, sv. 44, s. 326–352. ISSN 0044-3328. Dostupné z: https://link.springer.com/article/10.1007%2FBF01391200.

[33] Kiefer, C. a Joos, E. *Decoherence: Concepts and Examples*. quant-ph/9803052. FREIBURG-THEP-98-4. Freiburg im Breisgau: Freiburg Univ. Inst. Theor. Phys., Mar 1998. Dostupné z: https://cds.cern.ch/record/349557.

[34] Kolemen, E. a Kasdin, N. J. Optimal Configuration of a Planet-Finding Mission Consisting of a Telescope and a Constellation of Occulters. 2007. Dostupné z: https://www.princeton.edu/~hcil/papers/kolemen_aas_jan_2007_sedona.

[35] Kumar, S., Datta, D. a Singh, S. Black Hole Algorithm and Its Applications. *Studies in Computational Intelligence*. Prosinec 2015, sv. 575, s. 147–170.

[36] Layeb, A. A Clonal Selection Algorithm Based Tabu Search for Satisfiability Problems. *Journal of Advances in Information Technology*. Květen 2012, sv. 3.

[37] Layeb, A., Deneche, A. H. a Meshoul, S. A New Artificial Immune System for Solving the Maximum Satisfiability Problem. García Pedrajas, N., Herrera, F., Fyfe, C., Benítez, J. M. a Ali, M., ed. *Trends in Applied Intelligent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 136–142.

[38] Liu, G., Chen, W., Chen, H. a Xie, J. A Quantum Particle Swarm Optimization Algorithm with Teamwork Evolutionary Strategy. *Mathematical Problems in Engineering*. Duben 2019, sv. 2019, s. 1–12.

[39] Martonák, R., Santoro, G. a Tosatti, E. Quantum annealing of the Traveling Salesman Problem. *Physical review. E, Statistical, nonlinear, and soft matter physics*. Prosinec 2004, sv. 70, s. 057701.

[40] Massacci, F. a Marraro, L. Logical cryptanalysis as a SAT problem: Encoding and analysis of the U.S. data encryption standard. *Journal of Automated Reasoning*. Leden 2000, sv. 24, s. 165–203.

[41] Metal Supermarkets. *What Is Annealing?* [online]. 2018 [cit. 2022-04-11]. Dostupné z: https://www.metalsupermarkets.co.uk/what-is-annealing/.

[42] Miller, R. *Complexity of computer computations; proceedings.* New York: Plenum Press, 1972. ISBN 0-306-30707-3.

[43] Newton, I. *Newton's Principia : the mathematical principles of natural philosophy.* Daniel Adee, 45 Libery Street, 1846.

[44] Puente, R., Betancourt, P. a Mufeti. Cellular Automata And Its Applications In Modeling And Simulating The Evolution Of Diseases. Září 2015.

[45] Ramalho, M., Cordeiro, L. a Nicole, D. Encoding Floating-Point Numbers Using the SMT Theory in ESBMC: An Empirical Evaluation over the SV-COMP Benchmarks. Listopad 2017, s. 91–106. ISBN 978-3-319-70847-8.

[46] Riera Romo, M., Perez, D. a Ferrer, C. Innate immunity in vertebrates: An overview. *Immunology.* Únor 2016, sv. 148.

[47] Santoro, G. E., Martoňák, R., Tosatti, E. a Car, R. Theory of Quantum Annealing of an Ising Spin Glass. *Science.* American Association for the Advancement of Science. 2002, sv. 295, č. 5564, s. 2427–2430. ISSN 0036-8075. Dostupné z: https://science.sciencemag.org/content/295/5564/2427.

[48] Schwefel, H.-P. An interview with Hans-Paul Schwefel: with an introduction by Günter Rudolph. *ACM SIGEVOlution.* Prosinec 2008, sv. 3.

[49] Sil, A. *Quantum Mechanics-Schrodinger Equation* [online]. [cit. 2021-01-11]. Dostupné z: http://www.iitg.ac.in/asil/QM-02.pdf.

[50] Silva, E., Xavier, A. a Faria, M. Impact of Genomic Prediction Model, Selection Intensity, and Breeding Strategy on the Long-Term Genetic Gain and Genetic Erosion in Soybean Breeding. *Frontiers in Genetics.* Září 2021, sv. 12.

[51] Spakovszky, Z. S. *16.Unified: Thermodynamics and Propulsion* [online]. 2006 [cit. 2021-01-10]. Dostupné z: http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node38.html.

[52] Westbroek, M. J. E., King, P. R., Vvedensky, D. D. a Dürr, S. User's guide to Monte Carlo methods for evaluating path integrals. *American Journal of Physics.* 2018, sv. 86, č. 4, s. 293–304.

[53] Westerdiep, A. *Travelling Salesman Art* [online]. Arjan Westerdiep [cit. 2022-04-19]. Dostupné z: https://www.drububu.com/illustration/tsp/index.html.

[54] Weyl, H. Quantenmechanik und Gruppentheorie. *Zeitschrift für Physik.* 1927, sv. 46, s. 1–46. ISSN 0044-3328. Dostupné z: https://link.springer.com/article/10.1007%2FBF02055756.

[55] Xypron. *Solution of a travelling salesman problem* [online]. Červen 2010 [cit. 2021-01-09]. Dostupné z: http://en.wikipedia.org/wiki/Travelling_salesman_problem.

[56] Zhong, Y. a Zhang, L. A New Fuzzy Clustering Algorithm Based on Clonal Selection for Land Cover Classification. *Mathematical Problems in Engineering.* Leden 2011, sv. 21.