VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYUŽITÍ EVOLUČNÍCH ALGORITMŮ V KVANTOVÉM POČÍTÁNÍ

APPLICATION OF EVOLUTIONARY ALGORITHMS IN QUANTUM COMPUTING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **PETR ŽUFAN**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Žufan Petr, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Využití evolučních algoritmů v kvantovém počítání**
Application of Evolutionary Algorithms in Quantum Computing
Kategorie: Umělá inteligence
Zadání:

1. Nastudujte problematiku evolučních algoritmů.
2. Seznamte se s principy a možnými aplikacemi kvantového počítání.
3. Navrhněte a implementujte evoluční systém pro účely návrhu či optimalizace kvantových výpočtů.
4. Zvolte alespoň dvě úlohy z oblasti kvantových výpočtů a realizujte jejich implementaci ve zvoleném prostředí.
5. Pro zvolené úlohy vytvořte vhodnou reprezentaci s cílem automatického návrhu jejich řešení pomocí evolučního systému z bodu 3.
6. Proveďte sadu experimentů a sestavte srovnávací studii řešení těchto úloh pomocí existujících kvantových algoritmů a evolučního přístupu.
7. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího projektu.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání, demonstrace alespoň jedné úlohy z bodu 4.

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 2019
Datum odevzdání: 3. června 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce implementuje evoluční systém pro nelezení kvantového operátoru ve formě unitární matice. Cílem je ověření různých přístupů reprezentace kandidátních řešení a nastavení evolučního algoritmu. V práci byly použity dva evoluční algoritmy: genetický algoritmus a evoluční strategie. Dále je zde představen způsob generovaní unitární matice založený na QR dekompozici, který je pro tuto úlohu použit poprvé. Ten je v některých směrech lepší než předešlé. Na závěr je na experimentech ukázáno srovnání všech použitých technik.

Abstract

In this thesis, an evolutionary system for searching quantum operators in the form of unitary matrices is implemented. The aim is to propose several representations of candidate solutions and settings of the evolutionary algorithm. Two evolutionary algorithms were applied: the genetic algorithm and evolutionary strategy. Furthermore, a method of generating a unitary matrix is presented which is used for the first time for this task. This method is in some aspects better than the previous ones. Finally, a comparison of all used techniques is shown in experiments.

Klíčová slova

evoluční algoritmus, genetický algoritmus, evoluční strategie, kvantové počítání, kvantový obvod, kvantový operátor, unitární matice

Keywords

evolutionary algorithm, genetic algorithm, evolutionary strategy, quantum computing, quantum circuit, quantum operator, unitary matrix

Citace

ŽUFAN, Petr. *Využití evolučních algoritmů v kvantovém počítání*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Bidlo, Ph.D.

Využití evolučních algoritmů v kvantovém počítání

Prohlášení

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

. . . . . .
Petr Žufan
2. června 2020

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Bidlovi Ph.D. za odborné vedení, cenné rady, trpělivost a vstřícnost, které mi v průběhu zpracování diplomové práce věnoval. Tato práce byla podpořena Ministerstvem školství, mládeže a tělovýchovy z podpory Velkých infrastruktur pro výzkum, experimentální vývoj a inovace v rámci projektu "IT4Innovations národní superpočítačové centrum - LM2015070".

Obsah

Kapitola 1

Úvod

V posledním desetiletí se kvantové počítání dostalo z čistě matematického a teoretického oboru i do praktické roviny. Začaly se objevovat první kvantové počítače a s nimi nastupuje i vývoj nových kvantových algoritmů a technologií. Proč tomu tak ale je? Výkonnost klasických počítačů stále roste. Nevystačíme si s nimi?

Americký vynálezce Gordon Moore v roce 1965 postuloval myšlenku dnes nazývanou Moorův zákon. Ten v přeneseném významu říká, že se každé dva roky výkon počítačových čipů zdvojnásobí. Tento odhad exponenciálního růstu se držel několik desetiletí. V dnešní době už jsme však dospěli do situace, kdy velikost tranzistorů dosahuje své spodní hranice [44]. Proto musejí vědci a inženýři, snažící se pokračovat v trendu, hledat jiná řešení a výpočetní principy. Jednou z těchto technologií jsou právě kvantové počítače s potenciálem až exponenciálního zrychlení některých algoritmů.

Přestože první kvantové počítače se objevují až v posledním desetiletí, teoretické základy byly položeny už mnohem dříve. Počátky teorie o kvantovém počítání se přisuzují práci R. Feynmana z roku 1982 [10]. Od té doby bylo objeveno mnoho kvantových algoritmů. Nejvýznamnější z nich jsou Groverův algoritmus pro vyhledání v databázi [17] a Shoreův algoritmus pro faktorizaci čísel [47], které nastartovaly zájem o kvantové počítání. K těm moderním patří algoritmus QAOA (z anglického Quantum Approximate Optimization Algorithm) [9], nebo hibridní algoritmus VQE (z anglického Variational Quantum Eigensolver) [38]. I přes značné pokroky se stále hledají nové a efektivnější kvantové algoritmy.

Cílem této práce je vytvořit systém, který bude pomocí evolučních technik hledat kvantové algoritmy. Navážu na výzkum S.R. Hutsella a G.W. Greenwooda [22] a navazující práci D. MacKinnona [32]. Ti využili fakt, že každý kvantový algoritmus lze matematicky reprezentovat jako unitární matici a obráceně. Tato práce postupuje podle jejich vzoru. Algoritmus evolučně vytvoří unitární matici a pomocí simulátoru kvantového počítače vypočítá přesnost výstupů kvantového algoritmu. V práci jsou experimentálně srovnány tři reprezentace kandidátních řešení. Jedna je převzatá z článku Hutsella a Greenwooda a druhá z práce MacKinnona. Poslední, nově navržená, je založená na QR dekompozici [12]. Ve vytvořeném evolučním systému jsou implementovány dva evoluční algoritmy: genetický algoritmus a evoluční strategie. Oba přístupy jsou pro tento problém vhodné, avšak evoluční strategie jsou v této práci použity poprvé. I tyto dvě techniky budou experimentálně srovnány. V prvních dvou experimentech se ověří funkčnost implementovaného systému na znovuobjevení dvou základních hradel. Další experimenty ověří škálovatelnost algoritmu a ukáží jeho další možné aplikace. V závěru práce je provedeno vyhodnocení experimentů a zamyšlení se nad výsledky.

Kapitoly jsou rozděleny následovně. Principy kvantového počítaní jsou nezbytné pro plné pochopení práce. Věnuje se jim druhá kapitola. Ve třetí kapitole je souhrn použitých evolučních technik a postupů. Čtvrtá kapitola pojednává o již provedených výzkumech a pracích na téma evoluč-

ního návrhu kvantových algoritmů. Na to navazuje kapitola pátá obsahující moji konkrétní práci. Zde je popsán vytvořený evoluční systém i všechny techniky v něm použité. Šestá kapitola popisuje a komentuje experimenty a jejich výstupy. Poslední kapitola shrne výsledky práce a nastíní, kudy ve výzkumu pokračovat.

Kapitola 2

Kvantové počítání

Předtím než přejdeme k jádru práce, zaměříme se na koncept kvantového počítání, který je nezbytný k jeho pochopení. Myšlenka kvantového počítání vznikla v roce 1982, kdy si R. Feynman všiml, že některé jevy kvantové mechaniky nemohou být na klasickém počítači simulovány efektivně [10]. Bylo by tedy výhodnější, kdyby počítač mohl těchto jevů využít [42]. Velký boom a zájem o kvantové počítání přinesl P. Shor v roce 1994 se svým kvantovým algoritmem faktorizace celých čísel v polynomiálním čase [47]. Tím předčil dosud známé klasické algoritmy se složitostí exponenciální. V řadě vědeckých prací pak bylo dokázáno, že kvantový počítač dokáže efektivně simulovat klasický [3, 4]. Avšak, zda je tomu i obráceně nebylo doposud ani dokázáno ani vyvráceno [37]. Stále zůstává otázkou, jestli stejně efektivní klasický algoritmus neexistuje, nebo jenom nebyl vymyšlen. Tato nejistota ovšem není odrazující, nýbrž motivující. Lidé se nebojí že je jejich celoživotní práce zbytečná, naopak s nadšením hledají každou cestu, která by mohla posunout lidstvo zas o kousek dál.

Dalším souvisejícím využitím kvantové mechaniky je kvantová komunikace nebo kvantové generování náhodných čísel, které je teoreticky absolutně náhodné [28].

2.1 Kvantová mechanika

Kvantová mechanika může být těžko pochopitelná, protože často odporuje intuici získané pozorováním reálného světa. V kvantové mechanice je stav fyzikálních objektů popsán vlnovou funkcí, která popisuje pravděpodobnost výskytu částice na určitém místě v určitém čase. Tím se dostáváme k rozdílům oproti klasické mechanice, významným v kvantové informatice [18].

- **Pravděpodobnostní popis:** všem stavům systému jsou přiděleny hodnoty hustoty pravděpodbnosti.
- **Princip superpozice stavů:** kvantový systém existuje ve stavu daném lineární kombinací jiných stavů.
- **Kvantová provázanost:** systém více částic může existovat ve stavu, kdy nelze hovořit o stavech jednotlivých částic odděleně.
- **Měření:** operace měření na objektu vede ke změně stavu tohoto objektu.
- **Nerozlišitelnost částic:** částice se stejnými vlastnostmi od sebe nelze rozlišit.

2.2 Hilbertův prostor

Hilbertův prostor je matematický koncept umožňující popis kvantové mechaniky. Jde o n-dimenzionální vektorový prostor nad komplexními čísly, který je definován následovně.

Definice 2.2.1. [18] **Unitární prostor** H je komplexní vektorový prostor s definovaným skalárním součinem $\langle\cdot|\cdot\rangle : H \times H \to \mathbb{C}$, splňujícím pro jakékoliv vektory $\phi, \psi, \phi_1, \phi_2 \in H$ a jakékoliv $c_1, c_2 \in \mathbb{C}$ následující axiomy[1]:

$$\langle\phi|\psi\rangle = \langle\psi|\phi\rangle^*, \tag{2.1}$$

$$\langle\psi|\psi\rangle \geq 0 \text{ a } \langle\psi|\psi\rangle = 0 \text{ právě tehdy, když } \psi = 0, \tag{2.2}$$

$$\langle\psi|c_1\phi_1 + c_2\phi_2\rangle = c_1\langle\psi|\phi_1\rangle + c_2\langle\psi|\phi_2\rangle. \tag{2.3}$$

Norma prvku $\psi \in H$ je

$$\|\psi\| = \sqrt{\langle\psi|\psi\rangle}. \tag{2.4}$$

Vzdálenost dvou prvků $\psi, \phi \in H$ je

$$\mathrm{d}(\phi, \psi) = \|\phi - \psi\|. \tag{2.5}$$

Definice 2.2.2. [18] Unitární prostor H se nazývá **úplný**, když pro každou sekvenci $\{\phi_i\}_{i=1}^{\infty}$, kde $\phi_i \in H$, takovou, že

$$\lim_{i,j\to\infty} \|\phi_i - \phi_j\| = 0, \tag{2.6}$$

existuje vektor $\phi \in H$, takový, že

$$\lim_{i\to\infty} \|\phi - \phi_i\| = 0. \tag{2.7}$$

Definice 2.2.3. [18] Úplný unitární prostor se nazývá **Hilbertův prostor**.

Dále v práci budeme uvažovat pouze Hilbertovy prostory s konečným počtem dimenzí. n-dimenzionální Hilbertův prostor budeme značit H^n.

Příklad 2.2.1. Mějme prostor $H = \mathbb{C}^n$.

- Když definujeme skalární součin vektorů $u = (u_0 \dots u_n) \in \mathbb{C}^n, v = (v_0 \dots v_n) \in \mathbb{C}^n$ jako $\langle u\,|v\,\rangle = u_0v_0 + \cdots + u_nv_n$, potom H je Unitární prostor.
- Když navíc řekneme, že n je konečné číslo například 2, pak H je Hilbertův prostor.

Poznámka. Stavy kvantového systému jsou reprezentovány vektory Hilbertova prostoru.

2.3 Bra-Ket notace

Pro přehlednější zápis složitějších operací s vektory Hilbertova prostoru byla zavedena takzvaná Diracova notace, někdy nazývaná Bra-Ket notace. V Diracově notaci se vektor ϕ Hilbertova prostoru H značí $|\phi\rangle$ a nazývá se **ket-vektor**. K němu příslušný **bra-vektor** se značí $\langle\phi|$ a odpovídá adjungovanému vektoru[2] k vektoru $|\phi\rangle$. V n-dimenzionálním Hilbertově prostoru odpovídá ket-vektor $|\phi\rangle$ n-dimenzionálnímu sloupcovému vektoru, bra-vektor n-dimenzionálnímu řádkovému vektoru. Potom $\langle\phi|\psi\rangle$ značí (vnitřní) skalární součin vektorů a $|\phi\rangle\langle\psi|$ značí (vnější) tenzorový součin vektorů [18].

[1] z^* značí číslo komplexně sdružené číslu z

[2] Adjungovaný vektor je transponovaný a komplexně sdružený vektor.

Příklad 2.3.1. Mějme ket-vektory $|\phi\rangle = \begin{pmatrix} \frac{1+i}{2} \\ \frac{1-i}{2} \end{pmatrix}$ a $|\psi\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

1. bra-vektor[3] $\langle\phi| = |\phi\rangle^\dagger = \begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix}$
2. skalární součin $\langle\phi|\psi\rangle = \begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1-i}{2}$
3. tenzorový součin $|\phi\rangle\langle\psi| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} \frac{1-i}{2} & \frac{1+i}{2} \end{pmatrix} = \begin{pmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ 0 & 0 \end{pmatrix}$

2.4 Báze

Definice 2.4.1. [18] Dva vektory $\phi, \psi \in H$ jsou **ortogonální**, právě tehdy když $\langle\phi|\psi\rangle = 0$. Množina vektorů $B \subseteq H$ je **ortogonální**, když každé dva její vektory jsou ortogonální. Množina B je **ortonormální**, když je ortogonální a všechny její prvky mají normu 1.

Definice 2.4.2. [60] Množina vektorů B Hilbertova prostoru H tvoří (ortonormální) **bázi**, když

(a) B je ortonormální,

(b) každý vektor $\phi \in H$ může být zapsán jako lineární kombinace vektorů z B.

Pro každý vektorový prostor existuje mnoho bází. Nejlépe se pracuje s tzv. standardní bází definovanou následovně.

Definice 2.4.3. [60] Báze $B = \{b_0, b_1 \dots b_n\}$ se nazývá **standardní báze**, když pro každý vektor $b_x \in B$ platí:

$$b_x(i) = \begin{cases} 1, & i = x \\ 0, & \text{jinak} \end{cases} \tag{2.8}$$

Příklad 2.4.1. [60] Příklady bází v 3-dimenzionálním Hilbertově prostoru:

1. ortonormální báze: $\{(\frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}), (0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}), (\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})\}$
2. standardní báze: $\{(1,0,0), (0,1,0), (0,0,1)\}$

Poznámka. Jak uvidíme později, v kvantovém počítání jsou logická 0 a 1 reprezentovány jako standardní bázové vektory $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

2.5 Operátory

Definice 2.5.1. [20] Lineární zobrazení $A : H \to H$ se nazývá **lineární operátor** Hilbertova prostoru H a platí pro něj:

$$\forall|\psi\rangle, |\phi\rangle \in H \text{ a } \forall a, b \in \mathbb{C} \text{ platí, že } A(a|\psi\rangle + b|\phi\rangle) = aA|\psi\rangle + bA|\phi\rangle. \tag{2.9}$$

Aplikace operátoru A na vektor $|\psi\rangle$ značíme $A|\psi\rangle$. Aplikaci na vektor $\langle\phi|$ značíme $\langle\phi|A$. Každý lineární operátor A Hilbertova prostoru H s bází $B = \{|\theta_1\rangle, |\theta_2\rangle| \dots |\theta_n\rangle\}$ může být reprezentován maticí komplexních čísel $n \times n$ s hodnotami $\langle\theta_i|A|\theta_j\rangle$ na i-tém řádku a j-tém sloupci [18].

Operátor, který zobrazuje vektory na sebe samé, se nazývá jednotkový a značí se I.

[3] † značí sdružený operátor, který transformuje vektor na adjungovaný vektor.

Definice 2.5.2. [28] Nechť $B = \{b_1, b_2 \dots b_n\}$ je báze Hilbertova prostoru H. **Jednotkový operátor** I je definován:

$$I = \sum_{i=1}^{n} |b_i\rangle\langle b_i|. \tag{2.10}$$

Pro ortonormální bázi se jedná o jednotkovou matici. V kvantovém počítání jsou důležité dvě třídy operátorů: Hermitovské a unitární.

Definice 2.5.3. [28] Mějme Hilbertův prostor H, a operátor $A : H \to H$. Operátor $A^\dagger : H \to H$ se nazývá **sdružený (adjungovaný) operátor** , jestliže pro každé $\psi, \phi \in H$ platí:

$$\langle\psi|A\,|\phi\rangle^* = \langle\phi|A^\dagger|\psi\rangle \tag{2.11}$$

Definice 2.5.4. [28] Operátor A se nazývá **samosdružený (Hermitovský)**, jestliže pro všechny $|\phi\rangle, |\psi\rangle \in H$ platí, že

$$\langle\psi|A\,|\phi\rangle^* = \langle\phi|A\,|\psi\rangle \tag{2.12}$$

nebo-li

$$A = A^\dagger \tag{2.13}$$

Definice 2.5.5. [28] Lineární operátor A má **vlastní vektory** λ a **vlastní hodnoty** x, pro které platí:

$$A|\lambda\rangle = x|\lambda\rangle \tag{2.14}$$

Samosdruženému operátoru odpovídá Hermitovská matice. Hermitovské operátory používá kvantová mechanika pro popis fyzikálních pozorovatelných veličin. Mají všechny vlastní hodnoty reálné a odpovídají všem možným výsledkům měření. Vlastní vektory tvoří bázi Hilbertova prostoru nad kterým operátor pracuje [28].

Definice 2.5.6. [60] Lineární operátor U je **unitární**, když platí:

$$UU^\dagger = U^\dagger U = I. \tag{2.15}$$

Unitární operátor nemění skalární součin ani normu. Navíc jsou Unitární operátory reversibilní ve smyslu, že ke každému U existuje $U^\dagger$, který anuluje operaci U. Unitárním operátorům odpovídají unitární matice [60].

Příklad 2.5.1. [60] Uvedeme příklady některých operátorů.

1. Hermitovský operátor: $\begin{pmatrix} 5 & 4+5i & 6-16i \\ 4-5i & 13 & 7 \\ 6+16i & 7 & -2.1 \end{pmatrix}$
2. unitární operátor: $\begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Poznámka. V kvantovém počítání se používá jeden Hermitovský operátor, konkrétně operátor měření. Všechny ostatní operátory v kvantovém počítání jsou unitární [60].

2.6 Qubit

Kvantový bit, neboli qubit je v kvantovém počítání základní jednotkou informace. Standardní bit nabývá dvou stavů 0 nebo 1. Naproti tomu, Qubit může existovat i v nekonečném množství stavů mezi 1 a 0. Stavy kvantového systému, jakožto vektory Hilbertova prostoru, jsou vyjádřeny lineární kombinací bázových vektorů tohoto prostoru [37].

V kvantovém počítání volíme standardní bázi, nazývanou někdy **standardní výpočetní báze**:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.16}$$

kde intuitivně $|0\rangle$ reprezentuje logickou 0 a $|1\rangle$ logickou 1. Stav kvantového systému lze potom vyjádřit jako:

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{2.17}$$

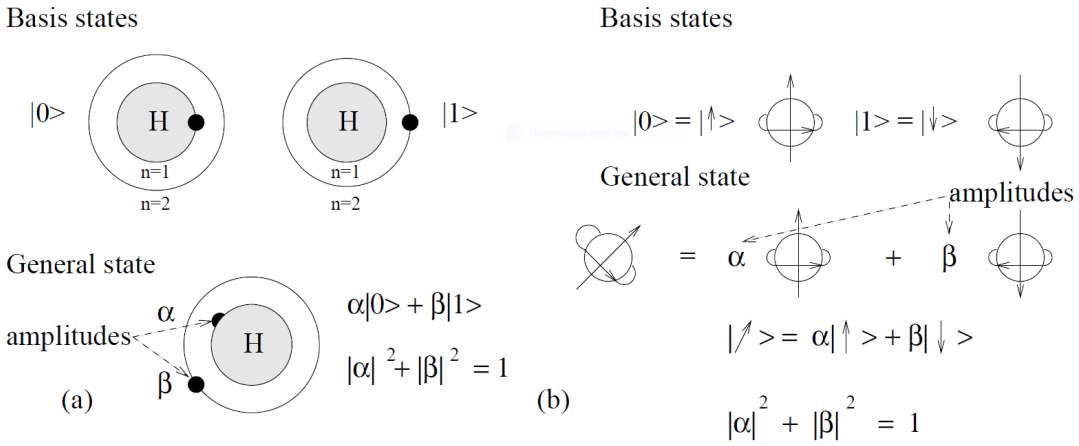
kde $\alpha, \beta \in \mathbb{C}$. Taková lineární kombinace bázových vektorů se nazývá **superpozice stavů** [37].

Definice 2.6.1. [18] Mějme dvourozměrný Hilbertův prostor H^2 s definovanou standardní bází $B = \{|0\rangle, |1\rangle\}$. **Qubit** je vektor $|\psi\rangle \in H^2$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.18}$$

kde $\alpha, \beta \in \mathbb{C}$ a $|\alpha|^2 + |\beta|^2 = 1$.

Konkrétní fyzická reprezentace qubitu může být různá. Může jít například o energetickou úroveň elektronu v atomu vodíku, kde $|0\rangle$ značí normální stav a $|1\rangle$ excitovaný stav, nebo o spin částice, kde $|0\rangle$ znamená spin nahoru $(+\frac{1}{2})$ a $|1\rangle$ spin dolů $(-\frac{1}{2})$ [18]. Tyto příklady jsou graficky znázorněny na obrázku 2.1.



Obrázek 2.1: Grafické reprezentace qubitu: vlevo energetická hladina elektronu, vpravo spin částice [18].

2.7 Blochova sféra

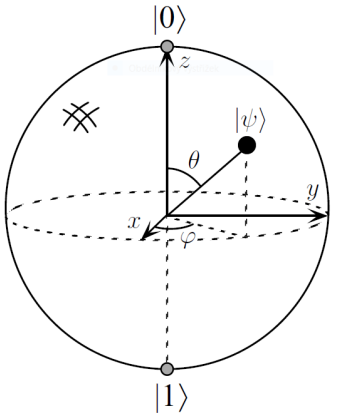
Kromě formálního popisu je možné qubit reprezentovat i jinými prostředky. Nejčastější je grafická reprezentace pomocí tzv. Blochovy sféry. Jde o zobrazení stavu qubitu jako bodu na povrchu koule

$x^2 + y^2 + z^2 = 1$. Vychází to z faktu, že se stav qubitu dá přepsat do tvaru

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right), \tag{2.19}$$

kde $\theta, \varphi, \gamma \in \mathbb{R}$. Faktor $e^{i\gamma}$ nemá geometrický význam, můžeme ho ignorovat. Parametry θ a φ pak definují bod na povrchu trojrozměrné sféry [37]. Blochova sféra je ukázána na obrázku 2.2.

Každý kvantový operátor lze pak chápat jako rotaci bodu ve sféře. Bohužel tato grafická reprezentace neumožňuje zobrazit systém více qubitů.



Obrázek 2.2: Blochova sféra [37].

2.8 Měření qubitu

Jak bylo zmíněno, qubit může nabývat i stavu mezi $|0\rangle$ a $|1\rangle$. Pozorováním nebo měřením qubit zkolabuje do jednoho ze stavů báze. Do $|0\rangle$ s pravděpodobností $|\alpha|^2$ nebo do $|1\rangle$ s pravděpodobností $|\beta|^2$.

Formálně měření znamená aplikaci Hermitovského operátoru A jehož vlastní hodnoty a_i odpovídají možným výsledkům měření [28].

Definice 2.8.1. [28] Operátor P Hilbertova prostoru H se nazývá **projekční operátor**, jestliže platí:

(a) $P = P^\dagger$

(b) $P = P^2$

Mějme $\psi \in H$, $\|\psi\| = 1$ potom je projekční operátor $P = |\psi\rangle\langle\psi|$ definován jako [18]:

$$|\psi\rangle\langle\psi|(\phi) = \langle\psi|\phi|\psi\rangle. \tag{2.20}$$

Po měření operátorem A s výsledkem a_i přechází systém do stavu $P_i|\psi\rangle$, kde P_i je projekční operátor generovaný všemi vlastními vektory $|\psi_{ij}\rangle$ operátoru A, které odpovídají a_i. tj. $P_i = \sum_j |\psi_{ij}\rangle\langle\psi_{ij}|$ [28].

2.9 Kvantový registr

V předchozí podkapitole jsme uvažovali systém s pouze jedním qubitem. Stejně jako klasické počítače pracují s několika-bitovými registry, i kvantové počítače potřebují více qubitů. Model, který popisuje systém více qubitů se nazývá tenzorový součin a značí se $\otimes$ [28].

Poznámka. Pro jednoduchost budeme dále pracovat pouze s dvoudimenzionálním Hilbertovým prostorem, který se v kvantovém počítání používá. Obecně tato pravidla platí pro libovolný konečný n-dimenzionální Hilbertův prostor. Také v této kapitole definujeme tenzorový součin pouze pro systém s dvěma qubity. Definice platí analogicky i pro systémy s více qubity.

Definice 2.9.1. [60] Mějme $|\phi_a\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \in H_a^2$ a $|\phi_b\rangle = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \in H_b^2$, pak $|\phi_a\rangle \otimes |\phi_b\rangle \in H_a^2 \otimes H_b^2$ je **tenzorový součin**, pro který platí:

$$|\phi_a\rangle \otimes |\phi_b\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} a_0 \cdot \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \\ a_1 \cdot \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix} \tag{2.21}$$

Vektor $|\phi_a\rangle \otimes |\phi_b\rangle$ se zkráceně značí $|\phi_a \phi_b\rangle$. Tenzorovým součinem n-rozměrného a m-rozměrného Hilbertova prostoru dostáváme nový Hilbertův prostor s dimenzí nm. V našem případě 4-rozměrný prostor má nové 4 standardní bázové vektory:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.22}$$

Z výše uvedeného si můžeme všimnout, že vektor $|00\rangle = |0\rangle \otimes |0\rangle$, analogicky ostatní. Tyto 4 stavy významem odpovídají 4 stavům dvoubitového klasického systému: 00, 01, 10, 11.

Obdobně jako pro jeden qubit, pár qubitů také existuje v superpozici bázových stavů [37].

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.23}$$

Výsledek měření bude jeden ze stavů $x \in \{00, 01, 10, 11\}$ s pravděpodobností $|\alpha_x|^2$ a se stavem qubitů $|x\rangle$. Měření stavu vícequbitového systému odpovídá měření jednotlivých qubitů. Například při změření prvního qubitu naměříme 0 s pravděpodobností $|\alpha_{00}|^2 + |\alpha_{01}|^2$ [37]. Celý systém zkolabuje do stavu:

$$\begin{aligned} |\psi\rangle &= \frac{\alpha_{00}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|00\rangle + \frac{\alpha_{01}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|01\rangle + 0|10\rangle + 0|11\rangle = \\ &= \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} = \\ &= \frac{1}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle) = \\ &= |0\rangle \otimes \frac{\alpha_{00}|0\rangle + \alpha_{01}|1\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \end{aligned} \tag{2.24}$$

Všimněme si, že vektor je normalizován hodnotou $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$. Je to z důvodu, aby pořád platilo pravidlo, že $\sum_{\alpha \in |\psi\rangle} |\alpha_x|^2 = 1$ [37].

V souvislosti s tenzorovým součinem a kvantovými registry si můžeme všimnout, že kvantový registr v superpozici lze chápat jako exponenciálně paralelizovanou verzi klasického registru, který je schopen na n qubitech pojmout 2^n hodnot současně. Kvantovou operací nad takovým registrem manipulujeme s 2^n amplitudami zároveň. Proto by mohl kvantový počítač řešit některé exponenciální úlohy efektivněji než klasický počítač.

2.10 Kvantové provázání

Velmi zajímavým stavem dvojice qubitů je takzvaný **Bellův stav** nebo EPR pár:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.25}$$

Ten má takovou vlastnost, že po změření jednoho qubitu s pravděpodobností 50% naměříme 1 a systém zkolabuje do stavu $|11\rangle$ a s pravděpodobností 50% naměříme 0 a systém bude ve stavu $|00\rangle$. Každopádně potom při měření druhého qubitu získáme stejnou hodnotu jako u prvního [37].

Výstupy měření jsou tedy mezi sebou korelovány. Experimentálně bylo zjištěno, že i po aplikaci některých operátorů na jeden z qubitů tato korelace zůstává. EPR stav je názván podle výzkumu, ve kterém Einstein, Podolsky a Rosen poprvé ukázali tuto jeho vlastnost a podle J. Bella, který na tomto stavu předvedl, že kvantová mechanika umožňuje věci, které překračují hranice klasického světa [37].

Stavům, které jsou mezi sebou korelovány, říkáme, že jsou **provázány**. Provázané stavy nelze vyjádřit tenzorovým součinem [28]. Existují 4 maximálně provázané Bellovy stavy, které navíc tvoří Bázi 4-rozměrného Hilbertova prostoru nazývanou **Bellova báze** [18].

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.26}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.27}$$

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.28}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.29}$$

2.11 Kvantová hradla

Konečně se dostáváme k principům kvantového programování a kvantových algoritmů. Podobně, jako je klasický počítač tvořen elektrickými obvody z logických hradel, je kvantový počítač tvořen kvantovými obvody z kvantových hradel [37]. Kvantové hradlo chápeme jako operátor v Hilbertově prostoru a jak bylo ukázáno v předchozích podkapitolách, reprezentujeme ho unitární maticí a jeho aplikaci na kvantový stav chápeme jako násobení matic.

Začneme s nejjednodušším klasickým hradlem NOT. To transformuje 0 na 1 a obráceně. Pro kvantové počítání je myšlenka stejná. Hradlo NOT transformuje $|0\rangle$ na $|1\rangle$ a obráceně. To ovšem nestačí. Je li stav v superpozici $\alpha|0\rangle + \beta|1\rangle$, hradlo NOT ho transformuje na stav $\beta|0\rangle + \alpha|1\rangle$ [37]. Takové hradlo samozřejmě existuje. Definujeme ho pomocí unitární matice:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.30}$$

Jeho aplikace na stav $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ pak vypadá následovně:

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta|0\rangle + \alpha|1\rangle. \tag{2.31}$$

Označení hradla X není náhodné. Jeho význam lze ukázat na Blochově sféře, kde překlopí vektor ψ kolem osy x. Podobně existují hradla Y a Z, mající význam překlopení kolem osy y respektive z. Tato hradla se někdy označují σ_x, σ_y a σ_z a říká se jim **Pauliho operátory** [37].

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{2.32}$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.33}$$

Pro úplnost zmíníme ještě operátor identity I, který má podobu jednotkové matice. Jeho aplikace na qubit neprovede žádnou změnu.

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.34}$$

Další a nejdůležitější hradlo v kvantovém počítání je Hadamardův operátor

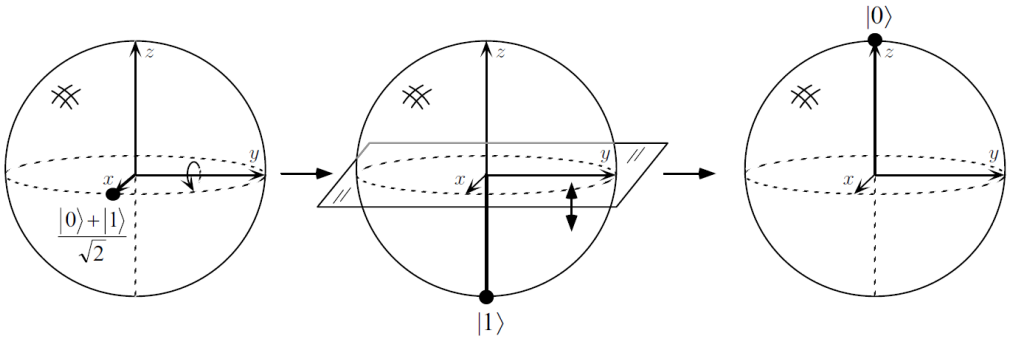
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.35}$$

Hadamard změní bázový vektor na maximálně smíšený stav.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \tag{2.36}$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{2.37}$$

Tyto stavy se označují $|+\rangle$ a $|-\rangle$ a také tvoří bázi. Druhá aplikace Hadamarda vrátí stav qubitu do původního. V Blochově sféře si ho můžeme představit jako rotaci kolem osy y o $90°$ následovanou překlopením přes rovinu xy [18], jak ukazuje obrázek 2.3.



Obrázek 2.3: Vizualizace aplikace Hadamardova operátoru na stav $|+\rangle$ v Blochově sféře [37].

Teoreticky existuje nekonečně mnoho unitárních 2×2 matic, takže i nekonečně mnoho hradel. Bylo dokázáno [37], že jakéhokoliv hradlo lze rozložit na aplikaci tří hradel rotace. Hradla R_x, R_y a R_z představují v Blochově sféře rotaci o libovolný uhel kolem příslušné osy [37].

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_y(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{2.38}$$

Poznámka. Jelikož hradlo chápeme jako unitární matici a aplikaci hradla na stav jako násobení matic, aplikace několika hradel za sebou je chápána jako postupné násobení několika matic. Násobení matic je asociativní operace, proto můžeme nejprve vynásobit matice hradel mezi sebou. Získáme tak novou unitární matici a hradlo.

Rozklad pak odpovídá rotaci kolem osy z o úhel ϕ, kolem osy y o úhel ψ a opět kolem osy z o úhel χ, to celé násobené globálním fázovým faktorem $e^{i\lambda}$. Libovolné hradlo lze tedy definovat pomocí 4 reálných čísel a univerzálního hradla U [18]:

$$U(\lambda,\phi,\psi,\chi) = e^{i\lambda}\begin{pmatrix} e^{-i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}\begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}\begin{pmatrix} e^{-i\chi} & 0 \\ 0 & e^{i\chi} \end{pmatrix} = \\ = \begin{pmatrix} \cos(\psi)e^{i\phi} & \sin(\psi)e^{i\chi} \\ -\sin(\psi)e^{-i\chi} & \cos(\psi)e^{-i\phi} \end{pmatrix} \tag{2.39}$$

Do teď jsme se zabývali pouze jedno-qubitovými hradly, které jsou reprezentovány maticí 2×2. Nyní se podíváme na hradla aplikovaná na více qubitů. Obecně platí, že hradlo pro n qubitů má velikost $2^n \times 2^n$. Hlavním hradlem pro dva qubity je $controlled - NOT$ neboli $CNOT$ [37].

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.40}$$

První qubit se nazývá řídicí a druhý cílový. Proč se jim tak říká vychází z toho, jak hradlo funguje. Pokud je první qubit roven $|0\rangle$ na druhý qubit se aplikuje I, když je první qubit $|1\rangle$ na druhý se aplikuje NOT. První qubit řídí co se bude dít s druhým. Zajímavé je, co se stane když je první qubit ve stavu $|+\rangle$, zatímco druhý v $|0\rangle$. Celkový stav systému je tedy $1/\sqrt{2}(|00\rangle + |10\rangle)$. Aplikace $CNOT$ pak probíhá jak bylo popsáno výše. Část $1/\sqrt{2}|00\rangle$ se nezmění protože je první qubit v $|0\rangle$. Část $1/\sqrt{2}|10\rangle$ se změní na $1/\sqrt{2}|11\rangle$. Konečný stav je potom $1/\sqrt{2}(|00\rangle + |11\rangle)$, což jak víme odpovídá kvantovému provázání [37].

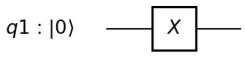
Přidat řídicí qubit je možné ke kterémukoliv jedno-qubitovému hradlu. Je to možné i k jakémukoliv více-qubitovému hradlu. Dokonce i k hradlu, které už řídicí qubit má. Přidáním řídicího qubitu k $CNOT$ dostaneme Toffoliho hradlo. To funguje tak, že na cílový qubit aplikuje NOT, když jsou oba dva řídicí qubity v $|1\rangle$ [37]. Matice vypadá následovně:

$$Toffoli = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.41}$$

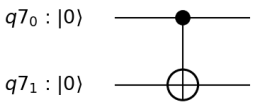
Toto hradlo představil T. Toffoli v roce 1980 a je zajímavé tím, že je univerzální. Jakoukoliv klasickou logickou funkci lze vytvořit použitím pouze Toffoliho hradel. Tím bylo ukázáno, že všechny logické operace jsou spočitatelné na kvantovém počítači [37].

2.12 Kvantové obvody

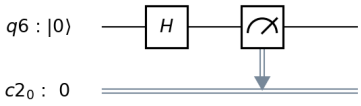
Kvantová hradla aplikujeme na qubity a spojujeme do kvantových obvodů. Tato forma programu má výhodu jednoduché grafické reprezentace. Aplikace hradla X na qubit v počátečním stavu $|0\rangle$ je ukázána na obrázku 2.4. Pokud má hradlo řídicí qubity, zaznačí se to do obvodu puntíkem. Příklad hradla $CNOT$ vidíme na obrázku 2.5. Dále aplikace měření na obrázku 2.6. Ta nám vrací jeden z klasických stavů 0 nebo 1. Proto se v kvantových obvodech objevují i "dráty"reprezentující klasické registry. Další zajímavostí je operátor podmíněný hodnotou klasického registru na obrázku 2.7. Na závěr této podkapitoly ukážeme na obrázku 2.8 obvod pro kvantové provázání, který se v algoritmech často využívá.



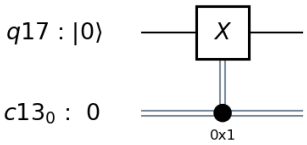
Obrázek 2.4: Obvod s hradlem X



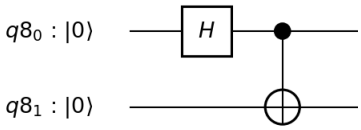
Obrázek 2.5: Obvod s hradlem $CNOT$



Obrázek 2.6: Obvod s měřením



Obrázek 2.7: Hradlo podmíněné klasickým registrem



Obrázek 2.8: Obvod kvantového provázání

2.13 Kvantové algoritmy

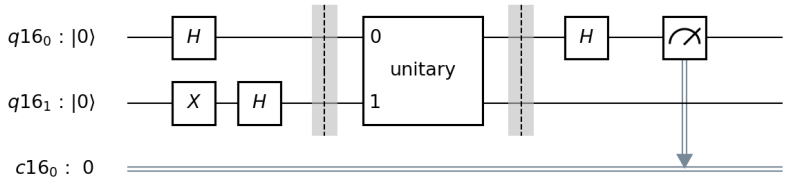
Nyní si ukážeme a popíšeme některé kvantové algoritmy. Seznam rozhodně není úplný ani neobsahuje nejnovější důležité algoritmy. Jde spíše o ukázání principů a potenciální síly kvantového počítání.

2.13.1 Deutschův algoritmus

Nejjednodušší kvantový algoritmus je Deutschův algoritmus. Nemá sice praktické využití, ale je to jeden z prvních kvantových algoritmů lepších než klasický [54]. Algoritmus se zabývá funkcí $f : 0, 1 \rightarrow 0, 1$. O této funkci řekneme, že je konstantní, když $f(0) = f(1)$ a vyvážená, když $f(0) \neq f(1)$. Zadání problému zní následovně. Mějme funkci f ve formě černé skříňky, u které můžeme vyhodnotit vstup na výstup, ale nevíme, jak je definována. Cílem je najít algoritmus, který určí, zda je funkce f konstantní nebo vyvážená. Klasický algoritmus by prvně vyhodnotil jeden vstup, poté druhý a nakonec by porovnal výstupy. Klasický počítač musí vyhodnotit funkci f dvakrát [60].

Zatímco kvantový počítač může být v superpozici stavů 0 a 1 a získat jejich výstupy paralelně pouze jedním vyhodnocením funkce. Obvod algoritmu je na obázku 2.9. Hradlo "unitary" (běžně označované U_f) znázorňuje kvantovou reprezentaci funkce f. Každé kvantové hradlo musí být

reverzibilní, proto se "black-box" hradla většinou znázorňují operací $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$, kde $\oplus$ značí XOR [60].



Obrázek 2.9: Deutschův algoritmus [37].

Poznámka. Svislá čára v obvodu je pouze grafický oddělovač funkčních bloků. Nemá žádný funkční význam.

Algoritmus začíná ve stavu $|01\rangle$, proto počáteční hradlo X. Potom se oba qubity dají do superpozice užitím hradla H. Stav systému je $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$. Po aplikaci hradla U_f se stav změní na $\frac{1}{2}(|0(0 \oplus f(0))\rangle - |0(1 \oplus f(0))\rangle + |1(0 \oplus f(1))\rangle - |1(1 \oplus f(1))\rangle)$. A máme 4 možnosti:

1. $f(x) = 0$:

$$\frac{1}{2}(|0(0 \oplus 0)\rangle - |0(1 \oplus 0)\rangle + |1(0 \oplus 0)\rangle - |1(1 \oplus 0)\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.42}$$

2. $f(x) = 1$:

$$\frac{1}{2}(|0(0 \oplus 1)\rangle - |0(1 \oplus 1)\rangle + |1(0 \oplus 1)\rangle - |1(1 \oplus 1)\rangle) = \frac{1}{2}(|01\rangle - |00\rangle + |11\rangle - |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) \tag{2.43}$$

3. $f(x) = x$:

$$\frac{1}{2}(|0(0 \oplus 0)\rangle - |0(1 \oplus 0)\rangle + |1(0 \oplus 1)\rangle - |1(1 \oplus 1)\rangle) = \frac{1}{2}(|00\rangle - |01\rangle + |11\rangle - |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.44}$$

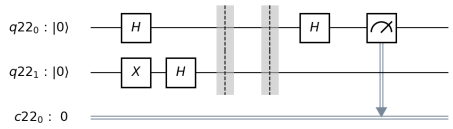
4. $f(x) = 1 - x$:

$$\frac{1}{2}(|0(0 \oplus 1)\rangle - |0(1 \oplus 1)\rangle + |1(0 \oplus 0)\rangle - |1(1 \oplus 0)\rangle) = \frac{1}{2}(|01\rangle - |00\rangle + |10\rangle - |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) \tag{2.45}$$

Nyní už můžeme vidět, že po aplikaci posledního H bude první qubit ve stavu $|0\rangle$ pro konstantní funkci f a ve stavu $|1\rangle$ pro vyváženou f. Měřením prvního qubitu se zjistí řešení.

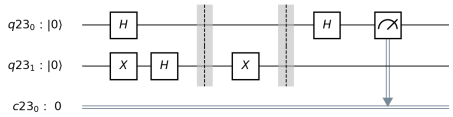
Pro zajímavost uvedeme ještě, jak vytvořit hradlo U_f znázorňující funkci f. Může nabývat 4 podob:

- (a) $f(x) = 0$



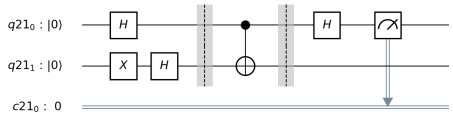
Obrázek 2.10: Deutschův algoritmus s hradlem U_f pro $f(x) = 0$

- (a) $f(x) = 1$



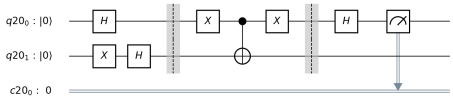
Obrázek 2.11: Deutschův algoritmus s hradlem U_f pro $f(x) = 1$

- (a) $f(x) = x$



Obrázek 2.12: Deutschův algoritmus s hradlem U_f pro $f(x) = x$

- (a) $f(x) = 1 - x$



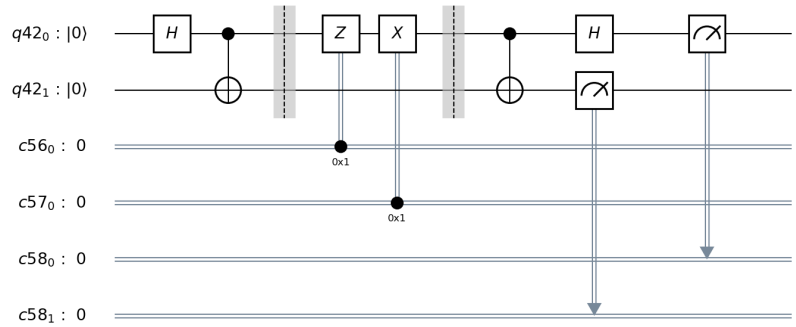
Obrázek 2.13: Deutschův algoritmus s hradlem U_f pro $f(x) = 1 - x$

Deutschův algritmus je zjednodušená verze Deutch-Jozsova algoritmu. Ten předpokládá funkci $f : 0, 1^n \to 0, 1$, kde f je vyvážená, když přesně pro polovinu vstupů je výstup 0 a pro polovinu 1, a konstantní, když všechny vstupy mají výstup 0, nebo všechny 1. Klasický algoritmus musí v nejhorším případě vyhodnotit funkci f $\frac{2^n}{2} + 1$-krát. Kvantový algoritmus to zvládne s jediným vyhodnocením funkce f. Dostáváme tak exponenciální zrychlení. Princip je obdobný výše uvedenému [60].

2.13.2 Superhusté kódování (Superdense coding)

Tento algoritmus je spíše komunikační protokol, který umožňuje do jednoho qubitu zakódovat dva bity informace a po pomyslném odeslání je opět dekódovat. Algoritmus nejprve na dvou qubitech vytvoří kvantové provázání. V druhém kroku do jednoho z qubitů zakóduje dva bity. Tyto dva

qubity se odešlou a příjemce nakonec algoritmu rozplete provázání a qubity změří. Tím je dekóduje. Příslušný obvod je na obrázku 2.14.



Obrázek 2.14: Obvod superhustého kódování [37].

Princip je ten, že počáteční stav $|00\rangle$ chceme změnit podle hodnot odesílaných bitů tak, aby příjemce při měření qubitů naměřil požadovanou hodnotu. Tedy na $|01\rangle$ při 01, na $|10\rangle$ při 10 nebo na $|11\rangle$ při 11. Díky tomu, že v prvním kroku vytvoříme provázání qubitů, stačí potom aplikovat hradla na jeden qubit a budou se měnit hodnoty obou. Pro pochopení, jak toto funguje, si musíme nejprve uvědomit spojitost standardní báze a Bellovy báze užitím kvantového provázání. V následujících rovnicích je ukázáno, který standardní bázový vektor se změní na kterou Bellovu bázi. Při rozvázání kvantového provázání je proces opačný (Bellův vektor na příslušný standardní vektor).

$$|00\rangle \xrightarrow{\text{provázání}} |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.46}$$

$$|01\rangle \xrightarrow{\text{provázání}} |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.47}$$

$$|10\rangle \xrightarrow{\text{provázání}} |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.48}$$

$$|11\rangle \xrightarrow{\text{provázání}} |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.49}$$

Následně je potřeba si všimnout, jaký efekt mají hradla X a Z aplikované na první qubit vektorů Bellovy báze. Protože jsme algoritmus začínali ve stavu $|00\rangle$, ukážeme si to pouze na vektoru $\Psi^+\rangle$. Začínat bychom mohli klidně v jiném stavu a postup by byl obdobný. Efekt je následující:

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \xrightarrow{\text{X}} \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) = |\Phi^+\rangle \tag{2.50}$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \xrightarrow{\text{Z}} \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = |\Psi^-\rangle \tag{2.51}$$

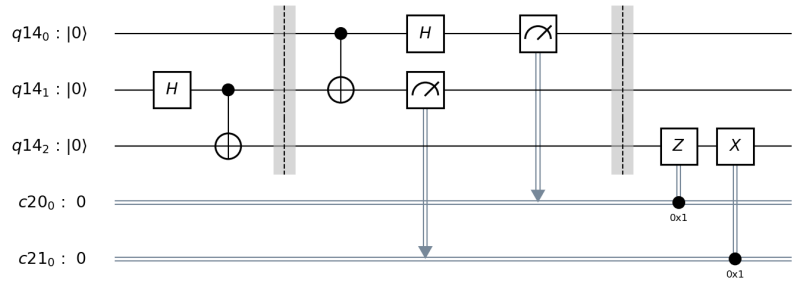
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \xrightarrow{\text{XZ}} \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) = |\Phi^-\rangle \tag{2.52}$$

V posledním kroku se rozváže Bellův vektor na standardní vektor a změří se příslušná hodnota. Nyní je jasné, proč se aplikuje Z, když je první odesílaný bit 1 a X, když je druhý bit 1.

Poznámka. Tvorba kvantově provázaného páru qubitů a distribuce každého z nich jednomu z účastníků komunikace je stěžejní úkon v každém kvantovém komunikačním protokolu.

2.13.3 Kvantová teleportace

Opět jde o algoritmus, který má využití v komunikaci. Tentokrát jde o přenos stavu qubitu pomocí dvou klasických bitů. Nejprve se opět vytvoří EPR pár, který se rozdělí mezi účastníky komunikace. Poté se vezme qubit, který chceme teleportovat a jeden qubit z EPR páru. Na nich se provede měření v Bellově bázi a změřené hodnoty se odešlou. Nakonec se na druhý qubit EPR páru aplikují hradla X a Z podle přijatých hodnot [37].



Obrázek 2.15: Obvod Teleportace [37].

Postupně si projdeme stavy, kterými systém prochází. Na začátek máme libovolný qubit a dvojici provázaných qubitů. Stav systému je

$$a|0\rangle + b|1\rangle\big(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\big) = \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle) \tag{2.53}$$

Měření v Bellově bázi se dá nahradit měřením ve standardní bázi tak, že před měřením provedeme "kvantové rozvázání". Aplikujeme tedy hradla $CNOT$ na první dva qubity a H na první qubit.

$$\begin{aligned}
&\frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle) \xrightarrow{CNOT} \frac{1}{\sqrt{2}}(a|000\rangle + a|011\rangle + b|110\rangle + b|101\rangle) \xrightarrow{H} \\
&\frac{1}{\sqrt{2}}\big(\frac{1}{\sqrt{2}}(a|000\rangle + a|100\rangle) + \frac{1}{\sqrt{2}}(a|011\rangle + a|111\rangle) + \frac{1}{\sqrt{2}}(b|010\rangle + b|110\rangle) + \frac{1}{\sqrt{2}}(b|001\rangle + b|101\rangle)\big) = \\
&\frac{1}{2}|00\rangle(a|0\rangle + b|1\rangle) + \frac{1}{2}|01\rangle(a|1\rangle + b|0\rangle) + \frac{1}{2}|10\rangle(a|0\rangle - b|1\rangle) + \frac{1}{2}|11\rangle(a|1\rangle - b|0\rangle)
\end{aligned} \tag{2.54}$$

Nyní se změří první dva qubity. Podle výsledku je jisté, ve kterém stavu je třetí qubit. Naměřené hodnoty se pošlou po klasické lince. Nakonec je potřeba opravit stav třetího qubitu. Když je první bit roven 1 aplikujeme Z hradlo, když je druhý roven 1 aplikujeme hradlo X.

$$00 : a|0\rangle + b|1\rangle \tag{2.55}$$

$$01 : a|1\rangle + b|0\rangle \xrightarrow{X} a|0\rangle + b|1\rangle \tag{2.56}$$

$$10 : a|0\rangle - b|1\rangle \xrightarrow{Z} a|0\rangle + b|1\rangle \tag{2.57}$$

$$11 : a|1\rangle - b|0\rangle \xrightarrow{ZX} a|0\rangle + b|1\rangle \tag{2.58}$$

Vidíme, že se stav prvního qubitu přenesl na třetí qubit aniž by byly hodnoty a a b známy.

Kapitola 3

Evoluční algoritmy

Pod pojmem evoluční algoritmus se rozumí celá rodina algoritmů vycházejících z Darwinovy teorie evoluce. Proces biologické evoluce pracuje s celou populací organismů, které se vzájemně kříží a mutují za vzniku nových jedinců. Tímto způsobem se hledá řešení problému přežití organismu v prostředí. Evoluční algoritmy přejímají myšlenku i názvosloví z biologické evoluce. Populace je zde množina kandidátních řešení hledaného problému. Každý jedinec populace je reprezentován chromozomem, na který jsou aplikovány genetické operátory. Vlivem selekčního tlaku je populace aktualizována kvalitnějšími jedinci. Kvalita jedince vzhledem k danému problému je určena takzvanou fitness funkcí. Obecně lze evoluční algoritmy popsat následujícím algoritmem [6].

Algoritmus 3.0.1: Obecné schéma evolučního algoritmu [5]

```
1  Inicializuj počáteční populaci P(0);
2  Vyhodnoť fitness jedinců v P(0);
3  g = 0 ;                                   // zde g je počitadlo generací.
4  while není splněna ukončující podmínka do
5      Vyber jedince (rodiče) z populace P(g);
6      Vytvoř nové jedince (potomky) z rodičů aplikováním genetických operátorů;
7      Vytvoř populaci P(g+1) nahrazením některých nebo všech jedinců z P(g) potomky;
8      Vyhodnoť fitness jedinců v P(g+1);
9      g = g + 1;
10 end
```

Evoluční algoritmy se dělí do několika skupin. Uvedu zde čtyři hlavní: evoluční programování, evoluční strategie, genetické algoritmy a genetické programování. Rozdělení je dáno historicky, kdy tyto směry vznikaly nezávisle na sobě i přesto, že si jsou velice podobné [5]. V následujících podkapitolách se zaměřím na dva z nich, které byly v práci implementovány a na techniky v nich použité.

3.1 Evoluční strategie

Evoluční strategie (dále ES) byly vyvinuty začátkem 60. let v Berlíně [41, 45]. Jedinec zde má dvě reprezentace: fenotyp a genotyp. Fenotyp je samotné řešení problému, genotyp je jeho zakódování v chromozomu pro potřeby evoluce. Jednu z možností zakódování uvádí příklad 3.1.1. Pro evolučních strategie je typický genotyp ve formátu vektoru reálných čísel [6].

Příklad 3.1.1. Mějme úlohu, ve které evolučně hledáme nějakou matematickou funkci popsanou polynomem daného stupně. Jedinec má v této úloze podobu rovnice. Této reprezentaci se říká fenotyp. Během evoluce se však pracuje s kódovanou reprezentací - genotypem. To by v tomto případě mohl být vektor reálných čísel reprezentující hodnoty koeficientů jednotlivých členů polynomu. Příklad jedince reprezentujícího polynomem třetího stupně:
Fenotyp = $f(x) = 2 + \frac{1}{3}x^2 - 6x^3$
Genotyp = $(2, 0, \frac{1}{3}, -6)$

Dále je pro evoluční strategie typické použití pouze mutace jakožto genetického operátoru. Mějme genotyp $X \in \mathbf{R}^n$, kde $X = (x_1, \ldots, x_n)$. Mutací $x_i \in X$ se zde rozumí aktualizace hodnoty x_i podle vztahu

$$\overline{x}_i = x_i + \sigma N(0, 1), \tag{3.1}$$

kde σ je koeficient mutace. σ může být pro všechna $x_i \in X$ stejná, nebo pro každé jiná [6].

Algoritmus evoluční strategie je parametrizován dvěma čísly μ a λ, kde μ je počet jedinců v populaci a λ počet vygenerovaných potomků v každé generaci.

Algoritmus pak vypadá následovně.

Algoritmus 3.1.1: Algoritmus evoluční strategie [6]

```
1  Inicializuj počáteční populaci P(0) velikosti μ;
2  Vyhodnoť fitness jedinců v P(0);
3  g = 0;
4  while není splněna ukončující podmínka do
5      repeat
6          Náhodně vyber jednoho jedince (rodiče) z populace;
7          Vytvoř potomka mutací tohoto rodiče;
8      until vygenerováno λ potomků;
9      Vyhodnoť fitness potomků;
10     Vytvoř populaci P(g+1) výběrem μ jedinců s nejlepší fitness ;   // viz strategie
       níže
11     g = g + 1;
12 end
```

V tomto algoritmu rozlišujeme 2 strategie:

- (μ, λ) strategie
- $(\mu + \lambda)$ strategie

Tyto strategie pak ovlivňují řádek 10 algoritmu 3.1.1. Při strategii (μ, λ) je nová populace vybrána pouze z potomků. Při $(\mu + \lambda)$ strategii je nová populace vybrána ze sjednocené množina potomků a předchozí populace [6]. Konkrétní příklady evolučních strategií jsou znázorněny v příkladu 3.1.2.

Příklad 3.1.2. Příklady evolučních strategií: $(1 + 1)$, $(4, 8)$, $(1 + 10)$, $(8 + 16)$

Dále byly úspěšně vyvinuty pokročilejší techniky jako samoadaptace parametru σ nebo strategie s genetickým operátorem rekombinace. Tyto techniky nebyly v práci použity, více se o nich můžete dočíst třeba v knize Natural Computing Algorithms [6].

3.2 Genetické algoritmy

V polovině 70. let v USA vznikly genetické algoritmy [21] (Dále GA). Jejich kanonickou podobu popisuje algoritmus 3.2.1. Mezi hlavní genetické operátory zde patří operátory selekce, rekombinace, mutace a nahrazení, které budou popsány v dalších odstavcích [6].

Algoritmus 3.2.1: Genetický algoritmus [29]

```
1  Inicializuj počáteční populaci P(0);
2  Vyhodnoť fitness jedinců v P(0);
3  g = 0;
4  while není splněna ukončující podmínka do
5      while není vygenerováno μ potomků do
6          Selekce: Vyber 2 rodiče z P(g).;
7          Rekombinace: S pravděpodobností p_cross aplikuj křížení na rodiče.;
8          Mutace: S pravděpodobností p_mut aplikuj mutaci na každý gen potomků vzniklých
           křížením;
9          Nahrazení: Vlož potomky do nové populace P(g+1);
10     end
11     Vyhodnoť fitness potomků v P(g);
12     g = g + 1;
13 end
```

3.2.1 Techniky selekce

Selekce představuje v GA mechanismus výběru rodičů, z nichž budou generováni potomci. Selekce typicky upřednostňuje jedince s lepší fitness, čímž vzniká během evoluce tzv. selekční tlak vedoucí k utváření kvalitnějších potomků. Uvedeme příklady několika základních technik, které byly v práci implementovány [6].

Turnaje

Typickým způsobem realizace selekce je tzv. turnaj. Ten funguje tak, že náhodně vybere z aktuální populace dvojici jedinců. Za rodiče je potom zvolen ten kvalitnější. Pro výběr dalšího rodiče se proces opakuje.

Ruleta

Jinou možností, jak provést selekci rodičů s ohledem na jejich fitness je tzv. vážená ruleta. V ní je pravděpodobnost výběru jedince rovna podílu

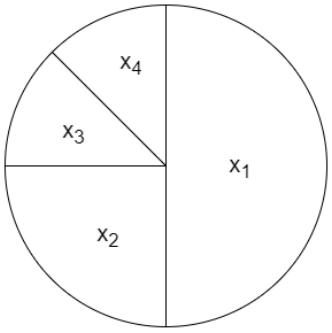
$$\text{fitness jedince} / \text{součet fitness všech jedinců v populaci.} \tag{3.2}$$

Zde platí, že jedinec s vyšší fitness je kvalitnější, než jedinec s nižší. Můžeme si to představit jako roztočení rulety, na jejímž kole jsou výseče. Pro každého jedince je zde právě jedna výseč, jejíž velikost odpovídá vztahu 3.2.

Příklad 3.2.1. Mějme populaci čtyř jedinců $P = (x_1, x_2, x_3, x_4)$ a jejich fitness $f(x_1) = 8$, $f(x_2) = 4$, $f(x_3) = 2$, $f(x_4) = 2$, pak podle vzorce 3.3

$$p(x) = \frac{f(x)}{\sum_{i=1}^{4} f(x_i)} \tag{3.3}$$

vytvoříme ruletu s pravděpodobnostmi výběru $p(x_1) = \frac{1}{2}$, $p(x_2) = \frac{1}{4}$, $p(x_3) = \frac{1}{8}$, $p(x_4) = \frac{1}{8}$. Znázornění jejich výsečí je zobrazeno na obrázku 3.1.



Obrázek 3.1: Grafická ilustrace realizace selekce váženou ruletou

Rank

Poslední metoda, kterou uvedeme, je selekce na základě pořadí (tzv. rank). Rank jedince je číslo udávající kolik jedinců v populaci má horší fitness. Toto číslo se poté použije na výpočet pravděpodobnosti výběru jedince. Rovnice pro přepočet mohou mít více podob [8]. Jednou z variant je lineární mapování, pro zvýšení selekčního tlaku lze použít exponenciální funkci. V rovnici 3.4 [8] je uvedeno lineární mapování, které je v práci použité.

$$p(r) = \frac{1-s}{\mu} + \frac{2rs}{\mu(\mu-1)}, \tag{3.4}$$

kde r je rank jedince, $s \in \langle 0, 1\rangle$ je selekční parametr a μ je počet jedinců v populaci. V práci bylo použito $s = 1$. Potom lze výpočet zjednodušit na rovnici 3.5.

$$p(r) = \frac{2r}{\mu(\mu-1)} \tag{3.5}$$

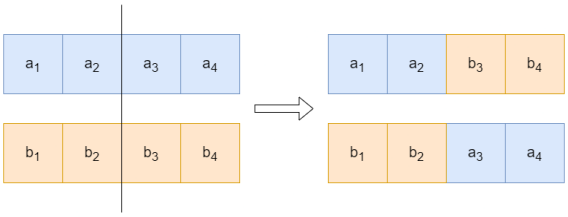
Příklad 3.2.2. Mějme populaci čtyř jedinců $P = (x_1, x_2, x_3, x_4)$ a jejich fitness $f(x_1) = 8$, $f(x_2) = 4$, $f(x_3) = 3$, $f(x_4) = 1$.
Potom rank jedinců je $r(x_1) = 3$, $r(x_2) = 2$, $r(x_3) = 1$, $r(x_4) = 0$
a výsledné pravděpodobnosti výběru podle rovnice 3.5 jsou $p(x_1) = \frac{1}{2}$, $p(x_2) = \frac{1}{3}$, $p(x_3) = \frac{1}{6}$, $p(x_4) = 0$.

3.2.2 Techniky křížení

Křížení je v GA hlavní genetický operátor. Rekombinací dvou rodičů vytvoří dva potomky. Křížení je důležité při prohledávání prostoru řešení, protože umožňuje generovat nové jedince kombinací již existujících. Je zde tedy šance, že nový jedinec zdědí dobré vlastnosti obou rodičů. Rekombinace je v GA podmíněná pravděpodobností p_{cross}. Je tedy šance, že se rodiče nezkříží. V takovém případě jsou potomky samotní rodiče.

Jednobodové křížení

Jediná technika křížení implementovaná v práci je jednobodové křížení. To na genotypu ve formě vektoru čísel probíhá následovně. Nejprve se náhodně vybere index do tohoto vektoru. Tento index rozdělí oba rodiče na dvě části. První částí je vektor těch čísel, jejichž pozice je menší než zvolený index. Druhou část tvoří vektor čísel s pozicí větší nebo rovnu zvolenému indexu. Potomci jsou potom vytvoření konkatenací první části prvního rodiče a druhou částí druhého rodiče respektive první částí druhého rodiče a druhou částí prvního rodiče [5]. Grafické znázornění křížení je ukázáno na obrázku 3.2.



Obrázek 3.2: Grafické znázornění jednobodového křížení

3.2.3 Mutace

Mutace je v genetických algoritmech podmíněna pravděpodobností p_{mut}. Každý gen z chromozomu se s touto pravděpodobností zmutuje. Pravděpodobnost mutace bývá v genetických algoritmech nízká narozdíl od pravděpodobnosti křížení. Mutace reálných čísel probíhá stejně jako v evolučních strategiích [6]. Díky mutaci mohou vznikat v genotypu hodnoty, které se nevyskytují u žádného jedince.

3.2.4 Nahrazení

Při nahrazení je nová populace obvykle tvořena pouze nově vytvořenými potomky. V některých případech se zavádí takzvaný **elitismus**. V takovém případě vždy do nové populace přežívá beze změny nejlepší jedinec nebo i několik jedinců z předchozí generace [6]. Ve své práci jsem ještě vyzkoušel techniku, kterou MacKinnon nazval **nová krev** [32], při které je část nové populace nově náhodně vygenerována. V biologické evoluci můžeme najít podobnost třeba v příchodu evropana do indiánského kmene v Americe [32].

I v genetických algoritmech vznikly pokročilejší techniky a adaptace pro genotypy nad reálnými čísly. V tomto souhrnu jsem uvedl pouze stručný přehled technik, na které se v následujících kapitolách budu odkazovat.

Kapitola 4

Evoluční přístup ke kvantovým výpočtům

Návrh kvantových algoritmů představuje komplexní problém. Ačkoliv existuje řada algoritmů pro různé problémy, smysl mají jen ty, které mají přínos oproti klasické variantě. Aby člověk uspěl, musí být velmi znalý kvantové mechaniky a principů kvantového počítání. Principy kvantové fyziky se často odlišují od zákonů, kterými se řídí makrosvět, proto je návrh kvantových algoritmů náročný a nelze použít postupů známých z klasické informatiky. Z tohoto důvodu se začaly hledat nové cesty návrhu kvantových algoritmů. Jednou z metod je evoluční návrh. Použití evolučních algoritmů pro návrh nebo optimalizaci kvantových algoritmů v minulosti zaznamenalo úspěchy a v současné době na toto téma vzniká více a více prací. Na evoluční návrh lze nahlížet třemi směry:

1. Hledání kvantového obvodu pro zadanou unitární matici.
2. Hledání kvantového obvodu pro zadané vstupní a výstupní stavy.
3. Hledání unitární matice pro zadané vstupní a výstupní stavy.

Poznámka. Z předchozích kapitol víme, že kvantový obvod je sekvence hradel, jejichž aplikaci chápeme jako násobení matic. Množina unitárních matic je uzavřená nad operací násobení [37]. (Násobením unitární matice s unitární maticí získáme opět unitární matici.) Můžeme tedy celý kvantový obvod reprezentovat jedinou unitární maticí.

V následujících podkapitolách shrnu články a díla, které pokrývají vybrané podstatné rysy tohoto oboru. Velmi pěkný přehled stavu tohoto odvětví do roku 2008 poskytli ve svém článku Gepp a Stocks [13].

4.1 Návrh kvantového obvodu pro zadanou unitární matici

V této sekci uvedu dosavadní práci v oblasti evolučního návrhu kvantových obvodů. Tato úloha je specifikována zadanou unitární maticí. Cílem návrhu je nalézt takový obvod sestávající z kvantových hradel, který koresponduje s touto maticí. Dále musí být zadána množina hradel, které mohou obvod tvořit. Nakonec je podstatná také fitness funkce, která rozhodne kvalitu řešení.

4.1.1 Williams a Gray

Williams a Gray ve svém článku v roce 1999 [59] jako první zkombinovali evoluční algoritmy a kvantové obvody. Jejich cílem bylo najít lepší řešení známých algoritmů. Pro hledání řešení použili

pouze dvě blíže nespecifikovaná jedno-qubitová hradla a hradlo $CNOT$. Pro evoluci zvolili lineární reprezentaci obvodu, která se skládala ze sekvence trojic (hradlo, paramtery hradla, qubity, na které je hradlo aplikované). Fitness funkce $f(S,U) = \sum_{i=1}^{2^N} \sum_{j=1}^{2^N} |U_{ij} - S_{ij}|$ porovnávala referenční matici U s maticí S odpovídající kandidátnímu obvodu. S populací o velikosti 100 jedinců a v průměrném čase 26,4 generací se jim podařilo úspěšně najít obvod pro kvantovou teleportaci, který použil o 2 méně hradel než obvody dosud známé [59].

4.1.2 Lukac a Perkowski

S novou reprezentací kvantového obvodu přišli Lukac a Perkowski v roce 2002 [31]. Místo posloupnosti trojic (hradlo, parametry, qubity), aplikovaných postupně po sobě použili posloupnost n-tic hradel, kde každá n-tice obsahovala hradla aplikovaná v jeden okamžik na všechny qubity. Například obvod provázání $\{\{H, I\}, \{CNOT\}\}$ vyjadřuje současnou aplikaci H na první qubit a I na druhý qubit a poté aplikaci $CNOT$ na oba qubity. Fitness funkci počítali jako součet rozdílů jednotlivých prvků matice referenční a matice odvozené od hledaného obvodu. V prvním experimentu hledali pouze jedno-hradlové obvody, aby ověřili konvergenci algoritmu. Cílem druhého experimentu bylo evolučně navrhnout algoritmus kvantové teleportace a dva obvody kvantového provázání pro 3 a 4 qubity, dříve vyvinuté Rubinsteinem (viz 4.2.2) [31]. Všechny tyto algoritmy se jim úspěšně podařilo znovuobjevit v kratším čase a s menším počtem generací, než v dříve publikovaných pracích [43, 59]. Navíc zjistili, že operátor mutace má lepší vliv na výsledek než operace křížení [13].

4.2 Návrh kvantového obvodu pro dané stavy systému

Tato sekce pojednává o druhém způsobu návrhu kvantového obvodu, uvažujícím zadané vstupní a výstupní stavy. Fitness funkce pak typicky porovnává požadovaný konečný stav se stavem kandidátního obvodu po jeho aplikaci na počáteční stav.

4.2.1 Spector

Spector se začal zabývat kvantovými algoritmy v roce 1998, kdy na toto téma vydal první článek [50]. Později svoji práci více propracoval a postupně vydal několik dalších článků a knih [51, 52, 49, 53, 54]. Narozdíl od ostatních, Spector použil metodu zvanou Genetické programování, kde gen jedince je přímo spustitelným programem. Tento program spustil na svém simulátoru QGAME. Pro výpočet fitness funkce použil simulací získané hodnoty. Problém byl opět definován množinou dvojic vstup-výstup, které můžeme nazvat případy. Fitness funkce měla 3 aspekty. *Zásahy* - počet případů, při kterých byla pravděpodobnost správného výsledku větší než 0.52. *Správnost* - součet pravděpodobností správného výsledku přes všechny případy. *Efektivita* - počet použitých hradel. Navíc *správnost* se vyhodnocovala jen v případě shodných *zásahů* a *efektivita* pouze v případě shodnosti obou předchozích složek [13].

Ve svých pracích použil postupně 3 modely kvantového programu. Prvním modelem byla klasická stromová struktura, kterou aplikoval na Deutschův problém [7] . S velikostí populace 10000 se mu podařilo nalézt řešení lepší než klasické [52]. Druhý model používal lineární strukturu programu, která ukládala argumenty do externí paměti v podobě zásobníků. S tímto modelem úspěšně vyvinul algoritmus lepší než klasický pro vyhledání prvku ve čtyřmístném neseřazeném poli [13]. Poslední model odstranil externí paměť se zásobníky. S tímto modelem vyřešil problém "and-or"logických sítí, který řeší, zda logická funkce $(f(0) \vee f(1)) \wedge (f(2) \vee f(3))$ vrací $true$ nebo

$false$. Z výsledků své práce Spector usoudil, že poslední reprezentace je pro genetické programování nejlepší [13].

4.2.2 Rubinstein

Další v řadě, kdo pracoval na evoluci kvantových algoritmů, byl Rubinstein [43]. V roce 2001 úspěšně vyvinul algoritmy pro maximální provázání dvou až pěti qubitů. Tedy vytvoření stavu $\frac{1}{\sqrt{2}}(|0\ldots0\rangle|1\ldots1\rangle)$. Použil opět lineární reprezentaci obvodu tvořenou sekvencí trojic hradlo, parametry hradla a qubity, na které se hradlo aplikuje. Tuto sekvenci zakódoval do bitového řetězce. S populací velikosti 5000 jedinců vyřešil problém lépe než klasické algoritmy. Přestože našel obvod pro maximálně 5 qubitů, řešení bylo rozšířeno na získání provázání pro libovolně velké kvantové registry. Tento článek ukázal potenciál evolučních algoritmů vyvíjet malé ale škálovatelné algoritmy [13].

4.2.3 Leier a Banzhaf

Leier ve své disertační práci vedené Banzhafem v roce 2004 [30] navázal na Spectora a použil dva jeho modely Genetického programování. Oba modely byly aplikovány na problém Deutsh-Jozsa a na problém 1-SAT (více v [30]). V práci se jim povedlo vytvořit pouze algoritmy stejně dobré jako již známé kvantové algoritmy. I přesto z ní plynou zajímavé poznatky. Použití hradla měření nezlepšilo výsledky. Evoluční operace křížení má menší vliv na výsledek než operace mutace. V pozdějších experimentech také zjistili, že operace výběru rodičů přináší lepší výsledky, když se provádí náhodně [13].

4.2.4 Stadelhofer a spol

Další disertační práce v oboru byla napsaná v roce 2006 [55] a rozšířená článkem napsaným spolu s Banzhafem a Suterem v roce 2008 [56]. Genetickým programováním objevili dva nové algoritmy. Jeden pro problém parity a druhý pro specifický případ problému skrytých podmnožin (hidden subgroup problem, více v [56]). První problém vyřešili s menším počtem hradel, než dosud známé kvantové obvody. Druhý problém dokonce s menším počtem volání "black-box" funkce (někdy zvané orákulum)[1]. Použili lineární reprezantaci programu, sadu pěti hradel (Hadamard, $CNOT$, dvě hradla rotace a hradlo orákulum), velikost populace 500 jedinců a fitness funkci skládající se ze tří komponent: *střety* - počet případů, pro které nešlo najít řešení, *chyba* - nejhorší a později průměrná pravděpodobnost nesprávného výsledku a *délka* - počet hradel orákula a později počet všech hradel v obvodu [13].

4.2.5 Massey, Clark a Stepney

Podobně jako Spector, Massey a kolektiv vytvořili univerzální evoluční systém pro návrh libovolného kvantového algoritmu. Postupně od roku 2004 do roku 2006 vytvořili několik verzí svého softwaru [33, 34, 35]. Všechny verze používaly stromovou reprezentaci programu a fitness funkce představené Rubinsteinem (viz 4.2.2) a Spectorem (viz 4.2.1). Sada hradel se skládala z mnoha jedno-qubitových hradel a jejich dvou-qubitových "controlled-"ekvivalentů. V pozdějších verzích přidali hradlo Toffoli a Swap. Svým evolučním přístupem vytvořili prvně algoritmus úplné sčítačky, později pravděpodobnostní poloviční sčítačku a pravděpodobnostní algoritmus nalezení maxima.

[1] Orákulum je předem určené hradlo tvořící funkci černé skříňky. Tedy nevíme a ani se nestaráme o to, jak funguje. Můžeme si to představit například jako volání nějaké externí funkce, kterou nemůžeme ovlivnit.

Všechna řešení byla stejně dobrá nebo lepší než dosud existující. Nejvýznamnější prací bylo vyvinutí kvantové furierovy transformace (QFT). Na pozdějších verzích jejich genetického systému se jim povedlo vytvořit tento algoritmus pro libovolný počet qubitů. Řešení bylo shodné s nejlepším dosud známým kvantovým algoritmem QFT [13].

4.3 Návrh kvantového obvodu v podobě unitární matice

Poslední varianta se snaží najít unitární matici, která transformuje zadaný vstupní stav na požadovaný výstupní. Jde o universálnější řešení, protože není omezené použitými hradly. V případě potřeby se dá libovolná matice na hrdla rozložit. Rozmachu tohoto přístupu dochází až v posledních 15 letech. Tímto přístupem se zabývá i tato práce, proto v popisu zajdeme do větších detailů, než v předchozích sekcích.

4.3.1 Hutsell a Greenwood

Hutsell a Greenwood byli první, kdo se ve své práci v roce 2007 zabýval evolučním návrhem unitárních matic [22]. Pro generování matic použili metodu, kterou vytvořili Zyzkowski a Kus [61]. Ta využívá dekompozice unitární matice U o velikosti $N \times N$ na součin $N(N-1)/2$ matic velikosti 2×2, které aplikují Eulerovu rotaci na dvourozměrný podprostor prostoru matice U. Každá taková matice je parametrizovaná třemi reálnými čísly a je definovaná následovně [22].

$$E_{mn}^{(j,k)}(\phi,\psi,\chi) = \begin{cases} cos(\phi)e^{i\psi} & \text{pro } m=n=j \\ sin(\phi)e^{i\chi} & \text{pro } m=j, n=k \\ -sin(\phi)e^{-i\chi} & \text{pro } m=k, n=j \\ cos(\phi)e^{-i\psi} & \text{pro } m=n=k \\ 1 & \text{jinak} \end{cases} \tag{4.1}$$

kde $\psi \in \langle 0, \pi/2\rangle$ a $\phi, \chi \in \langle 0, 2\pi)$. Celkový počet $N(N-1)/2$ matic je potom sdruženo do $N-1$ matic kompozitních rotací $E_1 \ldots E_{N-1}$ [22].

$$\begin{aligned} E_1 &= E^{(1,2)}(\phi_{1,2},\psi_{1,2},\chi_{1,2}) \\ E_2 &= E^{(1,3)}(\phi_{1,3},\psi_{1,3},\chi_{1,3})E^{(2,3)}(\phi_{2,3},\psi_{2,3},0) \\ &\vdots \\ E_{N-1} &= E^{(1,N)}(\phi_{1,N},\psi_{1,N},\chi_{1,N})E^{(2,N)}(\phi_{2,N},\psi_{2,N},0)\ldots E^{(N-1,N)}(\phi_{N-1,N},\psi_{N-1,N},0) \end{aligned} \tag{4.2}$$

Celková matice U je potom součin těchto kompozitních rotací spolu s fázovým faktorem [22]:

$$U = e^{i\chi_0}E_1E_2\ldots E_{N-1}. \tag{4.3}$$

Jakákoliv unitární matice $N \times N$ tak může být popsána celkem N^2 reálnými parametry: $(N-1)N/2$ hodnotami ϕ, $(N-1)N/2$ hodnotami ψ a N hodnotami χ [22].

Pro ohodnocení jedinců použili Hutsell a Greenwood fitness funkci ve tvaru:

$$fitness(U) = \sum_{i=0}^{N-1} \frac{1}{(|c_i|^2 - |c_i^*|^2)^2 + \epsilon}, \tag{4.4}$$

kde c_i a c_i^* jsou i-té položky výstupního stavu a cílového výstupního stavu a ϵ značí tolerovanou chybu.

Zaměřili se na tři úlohy. V první úloze bylo cílem z maximálně provázaného stavu $\frac{1}{\sqrt{N}}\begin{pmatrix}1\\ \dots \\ 1\end{pmatrix}$ získat stav, který naměří $|00\dots0\rangle$ s pravděpodobností alespoň 70%. Úspěch zaznamenali pro registry o velikosti 1, 2 a 3 qubity. Pro více qubitů časová složitost (kvůli častému násobení matic) exponenciálně rostla. Druhým experimentem bylo znovuobjevení Hadamardova hradla. V tomto úkolu neuspěli. Se vstupní množinou stavů $\{\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\}$ a k ní příslušnou výstupní množinou $\{\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}\}$ nalezly pouze hradlo $\frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\-1&1\end{pmatrix}$ [22]. Přestože byl v jejich práci neúspěch vysvětlen jinak, stal se proto, že v rovnici 4.3 zapomněli na globální faktor $e^{i\chi_0}$ [32].

Nakonec se pokusili vytvořit hradlo reprezentující orákulum v Deutschově algoritmu. Zvolili jeden případ, kdy $f(x) = NOT(x)$ a nalezli hradlo

$$U = \begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&-1\\0&0&-1&0\end{pmatrix}, \tag{4.5}$$

které je sice jiné, než originál, ale požadovanou funkci splňuje. Ve všech experimentech použili populaci o velikosti 100 jedinců [22]. Použité genetické operátory neuvedli. I přesto, že nepřinesli nic nového do kvantového počítání, ukázali potenciál nového směru evolučního vývoje kvantových algoritmů.

4.3.2 Krawec

Na práci Hutsella a Greenwooda navázal v roce 2014 Krawec [27]. Použil stejnou metodu generování unitárních matic podle Zyczkowskeho [61]. Pro specifikaci problému však použil rozsáhlejší přístup. Umožnil specifikovat úlohu jako hledání všech matic U v sekvenci $U_1, V_1, U_2, V_2, \dots U_n$, kde V jsou předem definované matice (například orákulum). Tímto způsobem definoval několik případů a ke každému z nich i očekávaný výstup. Výstup specifikuje pravděpodobnost s jakou se dvojice (qubit, hodnota) po aplikaci algoritmu naměří. Celý obecný tvar zadání problému vypadal následovně [27]:

$$((\text{počáteční stav}) : (\text{posloupnost matic}) : (\text{pravděpodobnost} : [\text{qubit} : \text{hodnota}])) \tag{4.6}$$

Konkrétní zadání pro první experiment vypadalo takto [27]:

$$((\Psi_0) : (U_0) : (1 : [1 : 1])) \tag{4.7}$$

Slovy přepsáno: Najdi matici U_0, která maximálně smíšený stav Ψ_0 transformuje do takového stavu, že měřením prvního qubitu získáme hodnotu 1 s pravděpodobností 1. Jde tedy o zopakování prvního experimentu Hutsella a Greenwooda. Dosáhl v něm výsledků ve 32. generaci, zatímco předchozí výzkum v 450 generaci [27]. Neuvedl však velikost populace.

Druhý experiment měl za cíl vytvořit Deutsch-Jozsův algoritmus. Tento problém už v sobě obsahuje volání orákula a tedy potenciál plného využití nového přístupu. Zadání vypadalo takto [27]:

$$
\begin{aligned}
&((\Psi_0) :(U_0, V_0, U_1) : (1 : [2 : 1])) \\
&((\Psi_0) :(U_0, V_1, U_1) : (1 : [2 : 1])) \\
&((\Psi_0) :(U_0, V_2, U_1) : (0 : [2 : 1])) \\
&((\Psi_0) :(U_0, V_3, U_1) : (0 : [2 : 1])) \\
&\quad\vdots
\end{aligned}
\tag{4.8}
$$

Vidíme, že tentokrát se generují dvě matice, mezi které se vloží hradlo orákula. I tento algoritmus úspěšně vytvořil hledané řešení. V obou úlohách použil operátor křížení v jednom bodě a operátor mutace, který vybral náhodnou konstantní hodnotu z intervalu $(0, 2\pi)$, a tou upravoval parametry matice. Pro menší hodnoty mutace algoritmus lépe konvertoval [27].

Ve své práci Krawec vymyslel obecnější evoluční přístup. Na rozdíl od předchozí práce popsal i použité evoluční operátory. Posunul tento obor zase o kousek dál.

4.3.3 Bang a Yoo

Další článek publikovali v roce 2014 Bang a Yoo [2]. Stejně jako Krawec umožnili generovat více unitárních matic s orákulem mezi nimi. Zadáním úlohy však byla pouze množina dvojic vstup-výstup. Jejich gen byl sestaven z posloupnosti bitových řetězců. Každý takový řetězec byl indexem do pole parametrů $\bar{p_k} = \{-\pi, -\pi + k, -\pi + 2k \ldots \pi - 2k, \pi - k, \pi\}$, kde krok k rozdělil interval $\langle -\pi, \pi \rangle$, na $(N^2 - 1)$ částí. N zde značí rozměr Hilbertova prostoru. Výsledná matice o velikosti $N \times N$ byla definována jako $U_j(\bar{p_j}) = e^{-i\bar{p_j}\bar{\sigma}}$, kde $\bar{\sigma} = (\sigma_1 \ldots \sigma_{N^2-1})$ je vektor, jehož prvky jsou generátory množiny $SU(N)$ [2].[2] Jelikož byl gen sestaven pouze z bitových hodnot, evoluční operace byly implementovány jako křížení v jednom bodě a mutace jednoho bitu. Svůj evoluční systém aplikovali na Deutsch-Jozsův problém. S populací o velikosti pouze 10 jedinců dosáhli výsledku v 50. generaci pro systém se třemi qubity, ve 180. generaci pro čtyři qubity a ve 400. generaci s pěti qubity [2]. Tato metoda je tedy ve srovnání s předchozími výpočetně náročnější. To však může být způsobeno pouze zvolenou velikostí populace.

4.3.4 MacKinnon

Ve své diplomové práci z roku 2017 MacKinnon rozšířil a vylepšil metodu Hutsella a Greenwooda [32]. V první řadě chtěl redukovat množství násobení matic. Místo toho, aby spojoval $N(N-1)$ matic, které aplikují rotaci na dvourozměrný podprostor, do jedné N-rozměrné matice, vytvořil pouze dvourozměrné matice a aplikoval je postupně na příslušné dvourozměrné podprostory kvantového stavu. Nakonec celý stav vynásobil globálním fázovým faktorem. Druhá modifikace zobecnila vztah $U = e^{i\chi} E_1 E_2 \ldots E_{N-1}$. Ve své práci dokázal, že jednotlivé matice, pracující nad podprostorem jk, mohou být mezi sebou násobeny v libovolném pořadí. Parametry j a k pak mohl přidat do chromozomu a evolučně je hledat. Tato flexibilnější reprezentace stále zajišťuje úplnost. Navíc pro každou unitární matici existuje více reprezentací. To by mělo zaručit lepší prohledávání prostoru řešení a snížit shlukování okolo lokálních extrémů [32]. V neposlední řadě jeho reprezentace umožňuje experimentovat s délkou genu. S menším počtem matic není zaručena úplnost a konvergence, ale může vést ke zrychlení. Nakonec, poslední úpravou je reprezentace a tvorba elementární unitární

[2] Více o množinách $SU(N)$ je pěkně sepsáno v učebních materiálech od H. Serodia [46] nebo více formálně v knize H. F. Jonese [24]. O jejich využití v kvantovém počítání pak v článku F. T. Hioe [19].

matice. Místo

$$U\ (\phi,\psi,\chi) = \begin{pmatrix} cos(\phi)e^{i\psi} & sin(\phi)e^{i\chi} \\ -sin(\phi)e^{-i\chi} & cos(\phi)e^{-i\psi} \end{pmatrix} \tag{4.9}$$

odvodil pomocí Pauliho operátorů X, Y, Z a čísel $a_0, a_1, a_2, a_3 \in \mathbb{R}$ vztah

$$U\ = a_0 I\ + i(a_1 X + a_2 Y + a_3 Z) \tag{4.10}$$

následovně [32]:

$$\begin{aligned} U\ &= \begin{pmatrix} \alpha & \beta \\ -\beta^* & \alpha^* \end{pmatrix}, \text{ kde } \alpha, \beta \in \mathbb{C} \text{ a } |\alpha|^2 + |\beta|^2 = 1 \\ &= \begin{pmatrix} \alpha_r + i\alpha_i & \beta_r + i\beta_i \\ -\beta_r + i\beta_i & \alpha_r - i\alpha_i \end{pmatrix}, \text{ kde } \alpha = \alpha_r + i\alpha_i \text{ a } \beta = \beta_r + i\beta_i \\ &= \alpha_r \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + i\alpha_i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + i\beta_r \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + i\beta_i \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \alpha_r I\ + i(\beta_i X + \beta_r Y + \alpha_i Z) \end{aligned} \tag{4.11}$$

kde $a_0 = \alpha_r, a_1 = \beta_i, a_2 = \beta_r, a_3 = \alpha_i$ a vektor (a_0, a_1, a_2, a_3) má normu 1. Tímto způsobem se dá generovat jakákoliv unitární matice pomocí čtyř reálných hodnot. Oproti původním 2-3 parametrům na jednu matici je to zhoršení. Na druhou stranu je zde značná výhoda jednoduché rozložitelnosti na 4 základní kvantové operátory [32].

Po všech těchto úpravách se každý gen skládá z $2N(N-1)$ reálných čísel z intervalu $\langle 0, 1\rangle$ pro definici matic, z $N(N-1)$ celých čísel z množiny $\{1 \dots N\}$ pro indexy matic a z jednoho reálného čísla z intervalu $\langle 0, 2\pi)$ pro globální fázový faktor. To je celkem $3N^2 - 3N + 1$ parametrů. V porovnání s N^2 parametrů potřebných v práci Hutsella a Greenwooda se jedná pouze o lineární zvětšení. Proto MacKinnon věřil, že přinesené výhody budou mít větší účinek než s tím spojené nevýhody a dohromady to bude ku prospěchu [32].

Vytvořený evoluční systém otestoval na stejném problému jako Hutsell a Greenwood i jako Krawec. Snažil se najít operátor, který z maximálně provázaného stavu získá stav $|0 \dots 0\rangle$. Fitness funkci definoval tak, že požadovaný stav se musí naměřit s pravděpodobností 0.9. Složitost svého systému popsal počtem vyhodnocení fitness funkce. Parametr počet generací z předchozích prací lze převést vztahem

$$\text{počet vyhodnocení fitness} = \text{počet generací} \times \text{počet jedinců v generaci.} \tag{4.12}$$

V experimentu generoval matice pro 4, 5 a 6 qubitové registry. Ve většině případů byly jeho výsledky lepší než v předchozích pracích [32].

Práce MacKinnona navazuje na článek Hutsella a Greenwooda a přivádí mnoho nových myšlenek. Nakonec i experiment se úspěšně povedl pro více qubitů než v předchozích pracích.

4.3.5 Gregor

Poslední v řadě je diplomová práce C. A. G. Gregora z roku 2019 [16]. Ten převzal MacKinnonovu reprezentaci a provedl s ní sérii experimentů na registrech o velikosti dvou až šesti qubitů. První experiment sloužil k ověření funkčnosti systému. V problému identity jde o generování matice, která nezmění stav systému. Taková matice se povedla vytvořit opakovaně s přesností 99%. V druhém experimentu byl na vstupu náhodný stav. Cílem bylo vyvinout operátor, který maximalizuje prvek s největší amplitudou. Překvapivě čím více qubitů systém měl, tím rychleji byl získán výsledek.

Cílem třetí úlohy bylo najít matici, která náhodný stav transformuje na stav s největší Shanonovou entropií. Shanonova entropie H vektoru v je

$$H(v) = -\sum_{i=1}^{n} v_i log_2(v_i) \tag{4.13}$$

Ku příkladu kvantový stav s jedním qubitem má nejnižší entropii ve stavech $|0\rangle$ a $|1\rangle$, nejvyšší entropii ve stavu $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Nalezení matice trvalo nejdéle pro 6 a pro 2 qubity [16].

Poslední experiment byl inspirovaný "no-cloning"teorémem [58]. Ten říká, že neexistuje žádné hradlo U takové, že $U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle$ [16]. Gregor na začátku experimentu vygeneroval náhodný stav $|\psi\rangle$ a poté hledal matici, která transformuje stav $|0\rangle$ na $|\psi\rangle$. Experiment proběhl opět bez problémů [16].

Gregor ve své práci přišel s jednou specialitou. Popsal vybrané modely kvantových kanálů. Kvantový kanál je komunikační kanál umožňující přenos qubitu. Na kvantových kanálech se potom sleduje jejich přesnost porovnáním stavu před odesláním a po odeslání. Gregor při výpočtu fitness funkce pohlížel na hledaný stav jako na stav před odesláním po kvantovém kanálu a na získaný stav jako na stav přijatý. Fitness funkce pak odpovídala výpočtu přesnosti kvantového kanálu [16].

Kapitola 5

Evoluční návrh unitárních matic

Cílem této práce je implementovat evoluční systém pro návrh kvantových operátorů ve formě unitárních matic. Evoluční přístup byl zvolen z důvodů komplikovaného ručního návrhu kvantových algoritmů. Člověk musí být velice znalý principů kvantového počítání, aby zvládal vymýšlet nové algoritmy a i pak je to časově náročné. Pro reprezentaci kvantového obvodu byla zvolena unitární matice, protože je to univerzální řešení nevztahující se na konkrétní sadu hradel nebo kvantový počítač.

Tato práce navazuje na již zmíněný článek Hutsella a Greenwooda [22] a na práci MacKinnona [32] s cílem implementace jejich způsobů generování unitárních matic a dále jejich porovnání s další technikou použitou v této diplomové práci poprvé pro tuto úlohu. Další srovnání přináší použití různých typů evolučních algoritmů a experimentování s jejich parametry. Snahou je nalezení takové reprezentace, a nastavení evolučního algoritmu, které dokáže v nejkratším čase ale s dostatečnou přesností nalézt požadovaný operátor. V experimentech budu také sledovat schopnost vymanění se z lokálních extrémů.

Součástí práce je implementace evolučního systému. Pro kompletní popis jeho fungování je potřeba popsat následující komponenty:

- Evoluční algoritmus: Jaký typ a jaké nastavení algoritmu bylo použito.
- Struktura genu: Jak vypadá genotyp a jak ho převést na fenotyp - unitární matici.
- Definice úlohy: Jak je pro evoluční systém požadovaná úloha zadána.
- Fitness funkce: Jak kvantitativně ohodnotit kvalitu kandidátních řešení.

5.1 Evoluční algoritmus

Ve vytvořeném evolučním systému jsou implementovány 2 typy evolučních algoritmů. Genetický algoritmus (GA) a evoluční strategie (ES). Genetický algoritmus je použit, protože i předchozí práce [32, 16, 22] zvolily tuto techniku. Je tu tedy záruka úspěchu a možnost porovnání. Pro evoluční strategie je typická práce s reálnými čísly. Jsou schopny nalézt řešení s velmi malou populací (v porovnání s GA). Teoreticky by mohly přinést zajímavé výsledky.

Z obvyklých aplikací ES na optimalizace reálných chromozomů je možné očekávat, že bude ES schopna dané úlohy řešit efektivněji než GA. Jelikož byl ve výše citovaných pracích použit pouze GA, bude v rámci této diplomové práce ověřena hypotéza, že ES je pro řešení problému návrhu unitárních matic vhodnější.

5.1.1 Genetický algoritmus

Implementovaný GA má podobu uvedenou v algoritmu 3.2.1. Implementovány jsou 3 metody selekce: turnaje, ruleta a rank. Křížení je jednobodové s pravděpodobností p_{cross}. Mutace každého genu v chromozomu je podmíněna pravděpodobností p_{mut} a provádí se podle rovnice 3.1. Parametr σ je předem daný, neměnný a pro všechny geny stejný. Použité techniky nahrazení umožňují specifikovat, kolik jedinců v nové populaci je potomků, kolik je elita z předchozí generace a kolik je nové krve (nově náhodně vygenerovaných jedinců).

Experimentováním a laděním parametrů jsem našel vhodné nastavení GA uvedené v tabulce 5.1.

Parametr	Hodnota
Metoda selekce	Rank
p_{cross}	0.8
p_{mut}	0.2
σ	0.2

Tabulka 5.1: Použité nastavení genetického algoritmu.

V následujících experimentech jsem porovnal 4 varianty GA, všechny s populací o velikosti 100 jedinců:

1. elita + křížení + mutace
   - elita: 10
   - potomci: 90
2. elita + mutace
   - elita: 10
   - potomci: 90
   - p_{cross}: 0 $\rightarrow$ pouze mutace
3. elita + mutace + nová krev
   - elita: 10
   - nová krev: 10
   - potomci: 80
   - p_{cross}: 0 $\rightarrow$ pouze mutace
4. elita + křížení + mutace + nová krev
   - elita: 10
   - nová krev: 10
   - potomci: 80

Parametr	Hodnota
Metoda selekce	Rank
p_{mut}	0.2
σ	0.8

Tabulka 5.2: Použité nastavení evoluční strategie.

5.1.2 Evoluční strategie

Evoluční strategie jsou implementovány podle algoritmu 3.1.1. I když byla v ES úspěšně ukázána adaptace parametru σ, v implementovaném algoritmu je konstantní a předem daný. Experimentováním jsem vyladil hodnoty parametrů a jejich nastavení je ukázáno v tabulce 5.2.

V experimentech srovnávám 4 strategie: $(4+8)$, $(8+16)$, $(1+10)$, $(1+50)$. Použil jsem pouze plus-strategie, které svým způsobem aplikují elitismus a dávají lepší výsledky.

5.2 Struktura chromozomu

Pro porovnání jsem v systému implementoval 3 genotypy. Genotyp, který použili Hutsell a Greenwood [22], genotyp MacKinnona [32] a mnou navržený genotyp založený na QR dekompozici [12]. Fenotyp je vždy unitární matice.

5.2.1 Genotyp Hutsell a Greenwood

První implementovaný genotyp je převzatý od Hutsella a Greenwooda [22]. Pro připomenutí uvedu základní vztah. Pro systém s m qubity má matice velikost $N = 2^m$. A spočítá se následovně:

$$U = e^{i\chi_0} E_1 E_2 \dots E_{N-1}. \tag{5.1}$$

Matice E_i kompozitních rotací se spočítají podle následujícího vztahu:

$$\begin{aligned} E_1 &= E^{(1,2)}(\phi_{1,2}, \psi_{1,2}, \chi_{1,2}) \\ E_2 &= E^{(1,3)}(\phi_{1,3}, \psi_{1,3}, \chi_{1,3}) E^{(2,3)}(\phi_{2,3}, \psi_{2,3}, 0) \\ &\vdots \\ E_{N-1} &= E^{(1,N)}(\phi_{1,N}, \psi_{1,N}, \chi_{1,N}) E^{(2,N)}(\phi_{2,N}, \psi_{2,N}, 0) \dots E^{(N-1,N)}(\phi_{N-1,N}, \psi_{N-1,N}, 0). \end{aligned} \tag{5.2}$$

A matice elementárních rotací dvojrozměrného podprostoru mají tvar následující:

$$E^{(j,k)}_{mn}(\phi, \psi, \chi) = \begin{cases} cos(\phi)e^{i\psi} & \text{pro } m = n = j \\ sin(\phi)e^{i\chi} & \text{pro } m = j, n = k \\ -sin(\phi)e^{-i\chi} & \text{pro } m = k, n = j \\ cos(\phi)e^{-i\psi} & \text{pro } m = n = k \\ 1 & \text{jinak} \end{cases} \tag{5.3}$$

kde m a n značí m-tý řádek a n-tý sloupec matice velikosti N a $\psi \in \langle 0, \pi/2 \rangle$ a $\phi, \chi \in \langle 0, 2\pi \rangle$. Struktura genu je potom sekvence N^2 parametrů ψ, ϕ a χ:

$$(\phi_{1,2}, \psi_{1,2}, \chi_{1,2}, \phi_{1,3}, \psi_{1,3}, \chi_{1,3}, \phi_{2,3}, \psi_{2,3} \dots \phi_{N-1,N}, \psi_{N-1,N}) \tag{5.4}$$

5.2.2 Genotyp MacKinnon

Druhý implementovaný genotyp byl navržen MacKinnonem. Ze všech jeho modifikací byla použita pouze tvorba 2×2 unitární matice podle vztahu

$$U(a_0, a_1, a_2, a_3) = \begin{pmatrix} a_0 + ia_3 & a_2 + ia_1 \\ -a_2 + ia_1 & a_0 - ia_3 \end{pmatrix} = a_0 I \ + i(a_1 X + a_2 Y + a_3 Z), \tag{5.5}$$

kde vektor (a_0, a_1, a_2, a_3) má normu 1.

Kvůli objektivnímu porovnání s předchozím genotypem nebyly další úpravy uvedené v [32] použity.

Konečný genotyp má strukturu vektoru $1 + 2N(N-1)$ reálných čísel. První číslo je z rozsahu $\{0, 2\pi\}$ a je reprezentováno jako globální fázový vektor.

5.2.3 Genotyp QR-dekompozice

Další reprezentace navržená v rámci této DP je založená na QR dekompozici. Pomocí této metody lze generovat unitární matici z téměř libovolné posloupnosti komplexních čísel. Nenalezl jsem jinou práci, ve které by tato metoda již byla použita, proto jsem ji zahrnul do výzkumu a porovnal s předchozími metodami.

QR dekompozice

Definice 5.2.1. [14] Nechť A je matice $m \times n$ komplexních čísel s hodností matice $rank(A) = n$. Potom matice A může být rozložena na součin

$$A \ = QR, \tag{5.6}$$

kde Q je $n \times n$ unitární matice a R je horní trojúhelníková matice $m \times n$.

QR dekompozice (nebo také QR faktorizace) tedy umožňuje rozložit libovolnou $m \times n$ matici s n nezávislými sloupci na $n \times n$ unitární matici Q a $m \times n$ horní trojúhelníkovou matici R. Pro účely této práce položím $n = m$. Potom jsou A a R čtvercové matice velikosti $n \times n$.

QR faktorizace má využití mimo jiné v matematice v problému nejmenších čtverců, kde pomáhá řešit soustavu lineárních rovnic

$$Ax = b. \tag{5.7}$$

Aplikováním QR rozkladu a vynásobení celé rovnice adjungovanou maticí $Q^\dagger$[1] lze vytvořit rovnici

$$Rx = Q^\dagger b \tag{5.8}$$

a tu potom řešit například metodou zpětné substituce [39]. Další využití má při hledání vlastních čísel a vektorů matice A [14].

Algoritmů QR dekompozice existuje několik [12, 14, 39]. V práci byla implementována metoda Householder, která pracuje na principu postupného nulování prvků pod hlavní diagonálou [39]. Rovnici 5.6 můžeme přepsat do tvaru

$$Q^\dagger A \ = R. \tag{5.9}$$

Následně lze $Q^\dagger$ rozložit na součin Housholderových matic

$$Q^\dagger = H_n H_{n-1} \dots H_1, \tag{5.10}$$

[1] Protože $QQ^\dagger = I$ (viz definice 2.5.6), rovnici $QRx = b$ můžeme přepsat na $Rx = Q^\dagger b$.

kde H_i je zkonstruována tak, aby vynulovala řádky s indexem větším než i. Tedy pro libovolný vektor u musí platit [39]

$$H_i u = \begin{pmatrix} v_1 \\ \vdots \\ v_i \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{5.11}$$

Chromozom

Se znalostí QR dekompozice může být každá $n \times n$ unitární matice Q reprezentována $n \times n$ komplexní maticí A. Chromozom je tedy zakódován do sekvence $2n^2$ reálných čísel, kde každá dvojice po sobě jdoucích čísel a a b tvoří komplexní číslo $a + bi$. Takto se vytvoří n^2 komplexních čísel reprezentující matici A.

Výhoda tohoto genotypu je v tom, že mapování genotypu na fenotyp je surjektivní zobrazení. To může být pro evoluční algoritmus výhodnější, než bijektivní zobrazení, protože hledané řešení může mít více než jednu podobu. Na druhou stranu to může mít i negativní vliv, když se pro křížení vyberou rodiče každý velmi blízko jinému z možných řešení. Největší nevýhodou této reprezentace je podmínka, že matice A v QR dekompozici musí mít všechny své sloupce lineárně nezávislé. Při generování nových jedinců v evoluci tak mohou vznikat neplatné chromozomy. To se projeví více u větších matic. I přes nevýhody má tato reprezentace i své výhody, díky kterým by mohla v evolučním algoritmu dávat dobré výsledky.

5.3 Definice úlohy

Pro potřeby této práce je úloha návrhu kvantového obvodu pomocí evolučního algoritmu definována následujícím způsobem. Mějme dánu dvojici $a = (|i\rangle, |o\rangle)$, kde $|i\rangle$ je vstupní stav a $|o\rangle$ je výstupní stav. S využitím reprezentací ze sekce 5.2 v evolučních algoritmech ze sekce 5.1 je cílem najít takovou unitární matici U, pro kterou platí, že

$$U|i\rangle = |o\rangle. \tag{5.12}$$

Pro specifikaci úlohy může být zadána i množina dvojic vstup-výstup $TS = \{a_1 \dots a_n\}$. Potom hledaná matice U musí splnit podmínku 5.12 pro každou dvojici $a_i \in TS$. Množinu TS budu dále nazývat trénovací množina. Jedna z možných trénovacích množin pro hradlo X je ukázána v příkladu 5.3.1.

Příklad 5.3.1. Mějme úlohu, ve které hledáme unitární matici hradla X podle specifikace v sekci 5.3. Trénovací množina pak může být $TS = \{(\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}), (\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix})\}$

5.4 Fitness funkce

Fitness funkce slouží k posouzení, do jaké míry odpovídá kandidátní řešení definici úlohy. Je počítána podle následujícího vzorce:

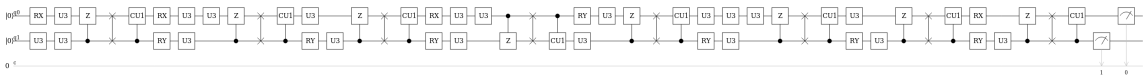
$$fitness(U) = \frac{\sum_{j=0}^{M} \sum_{i=0}^{N} ||out_j\rangle_i - U|in_j\rangle_i|}{MN} \tag{5.13}$$

kde M je velikost trénovací množiny, N je velikost stavového vektoru, $|out_j\rangle$ je j-tý výstupní vektor trénovací množiny a $|in_j\rangle$ je příslušný vstupní vektor. $|out_j\rangle_i$ značí i-tý prvek vektoru $|out_j\rangle$. Čím je hodnota fitness funkce blíže k nule, tím je řešení více vyhovující. Výsledná fitness je vydělena součinem MN. Hodnota se díky tomu nezvětšuje s rostoucí velikostí unitární matice.

Stavový vektor $U|in_j\rangle$ se vypočítá pomocí simulátoru kvantového počítače, který dokáže inicializovat stav $|in_j\rangle$ a aplikovat na něj hradlo U. Kvantové simulátory mají navíc tu výhodu, že mohou zprostředkovat celý stavový vektor a ne jenom hodnoty získané měřením, jak je tomu u kvantových počítačů.

5.5 Simulace kvantového počítače

Kvantové počítače dnes ještě nejsou příliš dostupné. Některé společnosti k nim umožňují vzdálený přístup přes API [23], ale pouze v omezeném množství. Proto bylo potřeba sáhnout po simulátoru. Využívá se ho při aplikaci kvantového operátoru na stav. Pěkný přehled kvantových simulátorů je dostupný v [11]. Při výběru simulátoru jsem provedl srovnání několika velkých hráčů: Cirq od Google [15], Qiskit od IBM [1], PyQuill od Rigetti Computing [48] a QuEST vytvořený na Oxfordské univerzitě [25]. Původně seznam obsahoval ještě XACC [36], který z uživatelského hlediska neodpovídal mým potřebám. Nad vybranými simulátory jsem provedl sérii experimentů. Vytvořil jsem obvody pro 2, 4, 8 a 16 qubitů které jsem v simulátorech spouštěl. Obvod pro 2 qubity je ukázán na obrázku 5.1. Obvod byl vygenerován v nástroji Quantum Programming Studio [40]. Obvody pro více qubitů měly stejný počet hradel, pouze jejich aplikace probíhala nad více qubity.



Obrázek 5.1: Testovací obvod pro 2 qubity.

Obrázek 5.2 vyobrazuje graf časů běhu jednotlivých simulací. Protože měly obvody stejný počet hradel, časová složitost simulací byla, díky dobře zvládnuté paralelizaci, téměř stejná. Zvětšovala se spíše prostorová složitost.

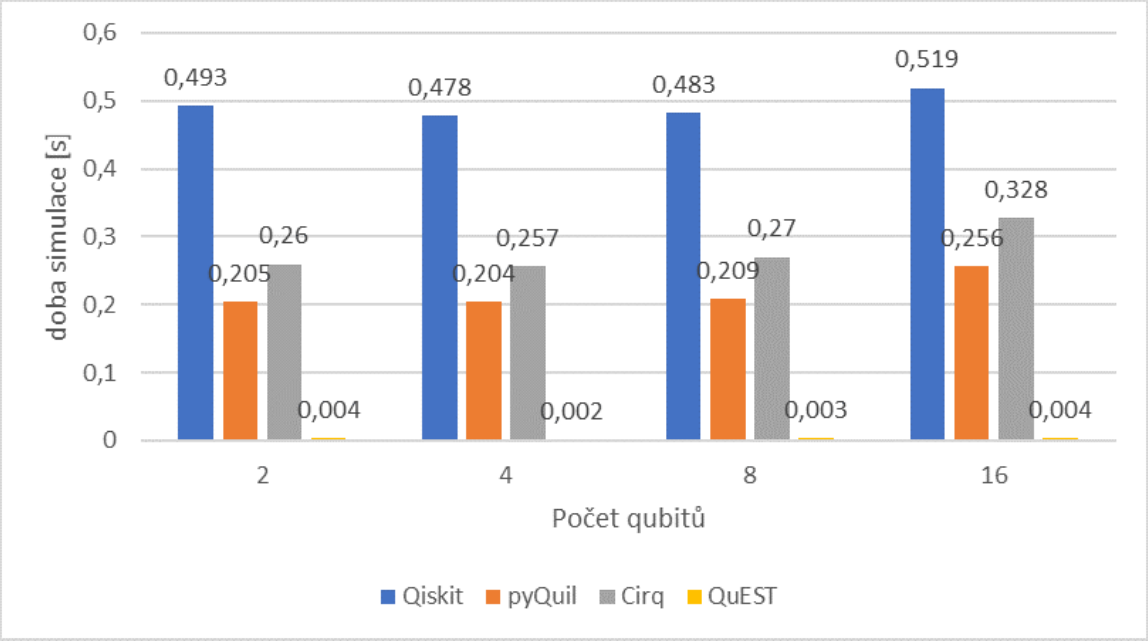
Z grafu vidíme, že simulátor QuEST si vedl řádově lépe. Jedním z důvodů je, že jako jediný je napsaný v C++, ostatní běží v pythonu. QuEST navíc využívá knihovny OpenMP, MPI a CUDA pro paralelizaci a optimalizace pomocí GPU [26].

Simulátor QuEST vyhovuje i funkčním požadavkům, protože umožňuje inicializovat libovolný počáteční stav a lze v něm definovat libovolný operátor pomocí unitární matice.

Z důvodů efektivity byl tedy vybrán simulátor QuEST [25]. Lze ho i snadno integrovat do evolučního systému také napsaném v C++.

5.6 Generátor náhodných čísel

Kvantové počítače disponují schopností absolutně náhodného generování čísel [28]. I kdyby byla možnost toho využít, v této práci by se tak nestalo. Pro evoluční algoritmy je důležité, aby je šlo zopakovat. K tomuto účelu je výhodnější, použít generátor pseudonáhodných čísel, který při inicializaci stejným "semínkem" produkuje stejná čísla. Experimenty tak mohou být zopakovány a demonstrovány. V této práci byl použit pseudonáhodný generátor typu MT19937ar [57].



Obrázek 5.2: Doba běhu simulací.

Kapitola 6

Experimentální výsledky

Evoluční systém byl otestován na několika experimentech. Každý experiment v každém nastavení byl spuštěn 96krát. Výsledky jsou ukázány na konvergenčních grafech a krabicových grafech. Konvergenční křivka ukazuje vývoj nejlepší fitness během evoluce pro každý běh. Krabicové grafy znázorňují statistické zhodnocení kvality získaných řešení ze všech 96 běhů. Nakonec ještě připomenu, že čím nižší fitness, tím kvalitnější jedinec.

Ukončující podmínka v experimentech byla nastavena pouze počtem generací. Důvodem je, aby při porovnání použitých metod byla dodržena metrika `Počet vyhodnocení fitness` uvedena v rovnici 4.12. Avšak stále byl sledován práh fitness, při kterém považuji experiment za úspěšný. Všechna zaznamenaná data a výsledky jsou s přesností na 10 desetinných míst.

Označení experimentu se skládá ze 3 částí.

<evoluční algoritmus> - <genotyp> - <konfigurace> (6.1)

Ty mohou nabývat následujících hod:

- <evoluční algoritmus>
  - ES = evoluční strategie
  - GA = genetický algoritmus
- <genotyp>
  - HG = genotyp Hutsell a Greenwood
  - M = genotyp MacKinnon
  - QR = genotyp QR dekompozice
- <konfigurace>
  - M = GA pouze s mutací
  - MK = GA s mutací a křížením
  - MKN = GA s mutací a křížením a novou krví
  - MN = GA s mutací a novou krví
  - 1+10 = ES s $(1 + 10)$ strategií
  - 1+50 = ES s $(1 + 50)$ strategií
  - 4+8 = ES s $(4 + 8)$ strategií

– 8+16 = ES s $(8 + 16)$ strategií

Například evoluční strategie (1+10) s genotypem MacKinnon je pojmenovaná `ES-M-1+10`. V následujících podkapitolách budou genotypy a konfigurace nazývány podle tohoto označení.

V prvním experimentu jsou vyhodnoceny všechny výše uvedené konfigurace. Z nich je do následujících experimentů vybráno jedno nastavení ES a jedno nastavení GA, které z prvního experimentu vyhodnotím jako nejlepší.

6.1 Experiment 1: Hadamardovo hradlo

Cílem prvního experimentu bylo navrhnout Hadamardovo hradlo H. Jeho podoba je pro připomenutí ukázána v rovnici 6.2.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{6.2}$$

Trénovací množina (6.3) tohoto problému má dvě položky. Jednu pro transformaci z $|0\rangle$ na $|+\rangle$, druhou pro přechod z $|1\rangle$ na $|-\rangle$.

$$\left\{ \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \right) \right\} \tag{6.3}$$

Ukončující podmínka experimentu byla nastavena u GA o populaci 100 jedinců na 20 generací. Pro evoluční strategie pak byla dopočítána podle příslušné konfigurace.

6.1.1 Výsledky

Experiment byl proveden na všech osmi variantách evolučních algoritmů uvedených v sekci 5.1 (4 $\times$ GA a 4 $\times$ ES). Konvergenční křivky ES respektive GA jsou zobrazeny na obrázku 6.1 respektive 6.2. Na těchto grafech je zobrazen celý běh evoluce. Detailní zobrazení prvních čtvrtin běhů je pak na obrázku 6.3 pro ES a 6.4 pro GA. Pro snadné porovnání mají grafy v daném obrázku stejné měřítko osy y. Na všech obrázcích je v prvním řádku genotyp M, v druhém řádku genotyp QR a ve třetím genotyp HG. Po sloupcích jsou pak jednotlivé konfigurace. Můžeme tak snadno ve sloupcích srovnávat vlastnosti genotypů v dané konfiguraci a v řádcích vlastnosti konfigurace v daném genotypu.

Statistické zhodnocení výsledků na obrázcích 6.5 - 6.7 jsou uspořádány trochu rozdílně. Řádek představuje evoluční algoritmus, sloupec genotyp. V každém grafu jsou pak 4 krabicové grafy, jeden pro každou konfiguraci. Grafy na obrázku 6.5 jsou nastaveny tak, aby byly vidět všechny hodnoty. Pozor na různá měřítka osy y. V obrázku 6.6 jsou ty stejné grafy, ale tentokrát bez odlehlých hodnot. Tomu je upraveno i měřítko. Nakonec obrázek 6.7 zobrazuje všechny grafy ve stejném měřítku pro snadné porovnání.

V experimentech jsem dále sledoval úspěšnost řešení. V tomto experimentu jsem řešení považoval za úspěšné při výsledné fitness nižší než 0,01. V tabulce 6.1 je ukázána procentuální úspěšnost jednotlivých konfigurací. Při stoprocentní úspěšnosti i počet vyhodnocení fitness nutný k úspěchu všech běhů experimentu. Maximální počet vyhodnocení fitness byl v tomto experimentu dán ukončující podmínkou na 2000.

Z konvergenčních křivek je na první pohled vidět, že genotyp QR ve všech konfiguracích konvergoval rychleji, než zbylé dva. Na statistickém zhodnocení je také vidět, že výsledné hodnoty fitness byly u genotypu QR řádově lepší. Dokonce na grafu *ES-QR* v obrázku 6.6 jsou konečné

fitness rovny 0 s přesností na 10 desetinných míst. Dále lze v konvergenčních grafech ES u genotypů M a HG pozorovat tendence uváznutí v lokálních extrémech. Na druhou stranu je vidět i celkem úspěšná schopnost vyproštění se z tohoto stavu. Což je klíčová vlastnost evolučních algoritmů v porovnání například s horolezeckými algoritmy. Nakonec podle tabulky 6.1 je genotyp QR jediný, který vždy úspěšně nalezl hledanou matici. Na základě těchto pozorování mohu říct, že si v tomto experimentu nově navržený genotyp vedl nejlépe.

Při srovnání ES a GA vidíme, že ES mají strmější konvergenční křivku. Ze statistických zhodnocení vidíme, že i většina výsledných hodnot fitness jsou u ES odhadem poloviční než u GA a to ve všech genotypech. I procentuální úspěšnost je u ES vyšší. Na druhou stranu se u ES vyskytují odlehlé hodnoty fitness řádově vyšší, než jaké jsou hodnoty fitness u GA. To je proto, že u GA nedochází k uváznutí v lokálních extrémech. I přesto podle mého v tomto experimentu ES předčili GA.

U GA můžeme pozorovat, že zavedení genetyckého operátoru *nová krev* většinou zhoršilo výsledky i konvergenci. Stejně tak bylo dosaženo horších výsledků při nepoužití operátoru *křížení*. Dalších experimentech tedy budou probíhat pouze s touto konfigurací GA. Z ES jsem nakonec vybral strategii $(1 + 10)$. Hlavním důvodem je rychlejší konvergence i přes častější tendenci uváznutí.

6.1.2 Nejlepší řešení

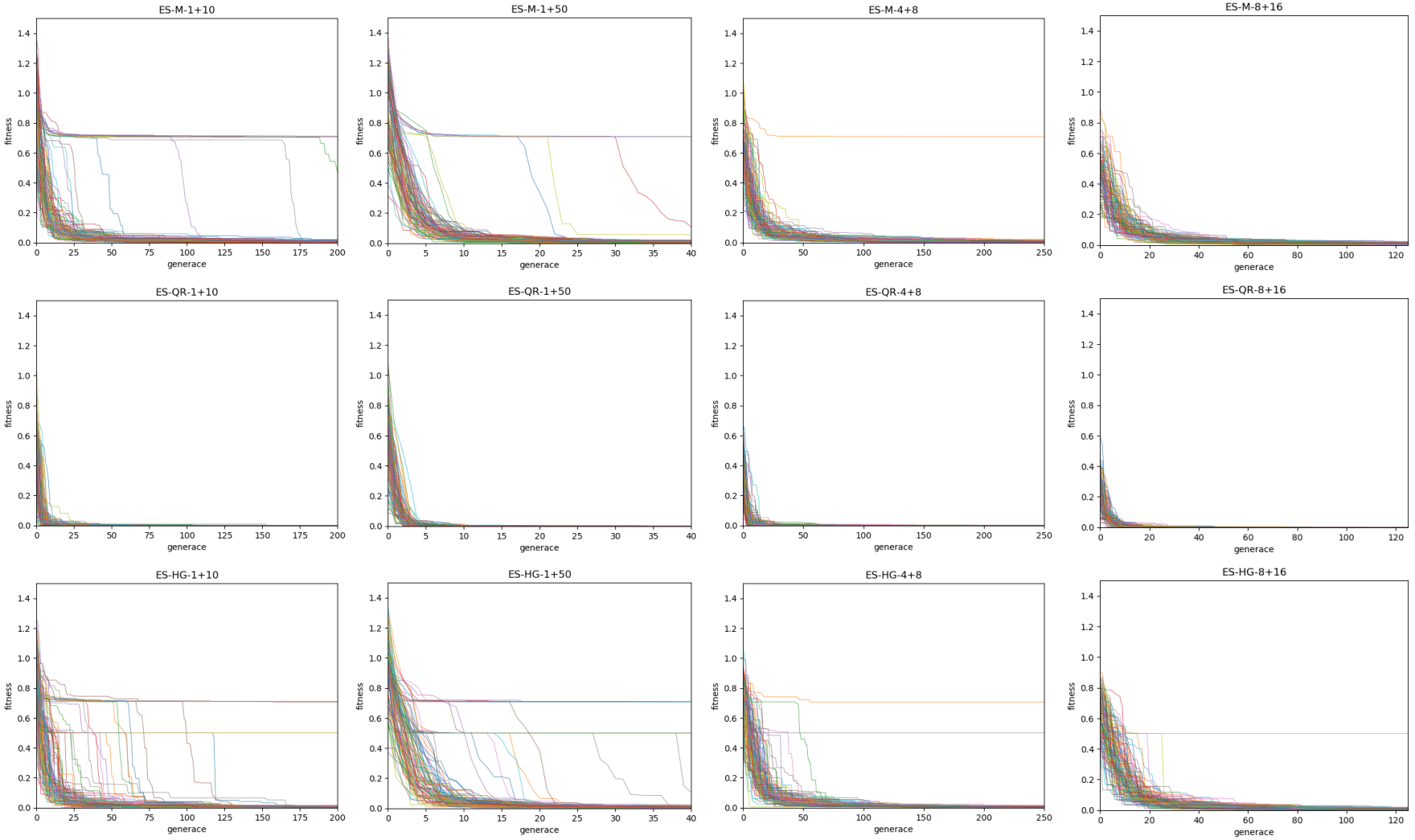
Nejlepší nalezené řešení dosáhlo hodnoty fitness rovné 0 s přesností na 10 desetinných míst. Vygeneroval ho evoluční algoritmus s nastavením *ES-QR-1+10*. Jeho tvar je zobrazen v 6.4. V porovnání s referenční maticí 6.2 jde o velmi přesný výsledek.

$$\begin{pmatrix} 0.70710678119 & 0.70710678119 \\ 0.70710678119 & -0.70710678119 \end{pmatrix} \tag{6.4}$$

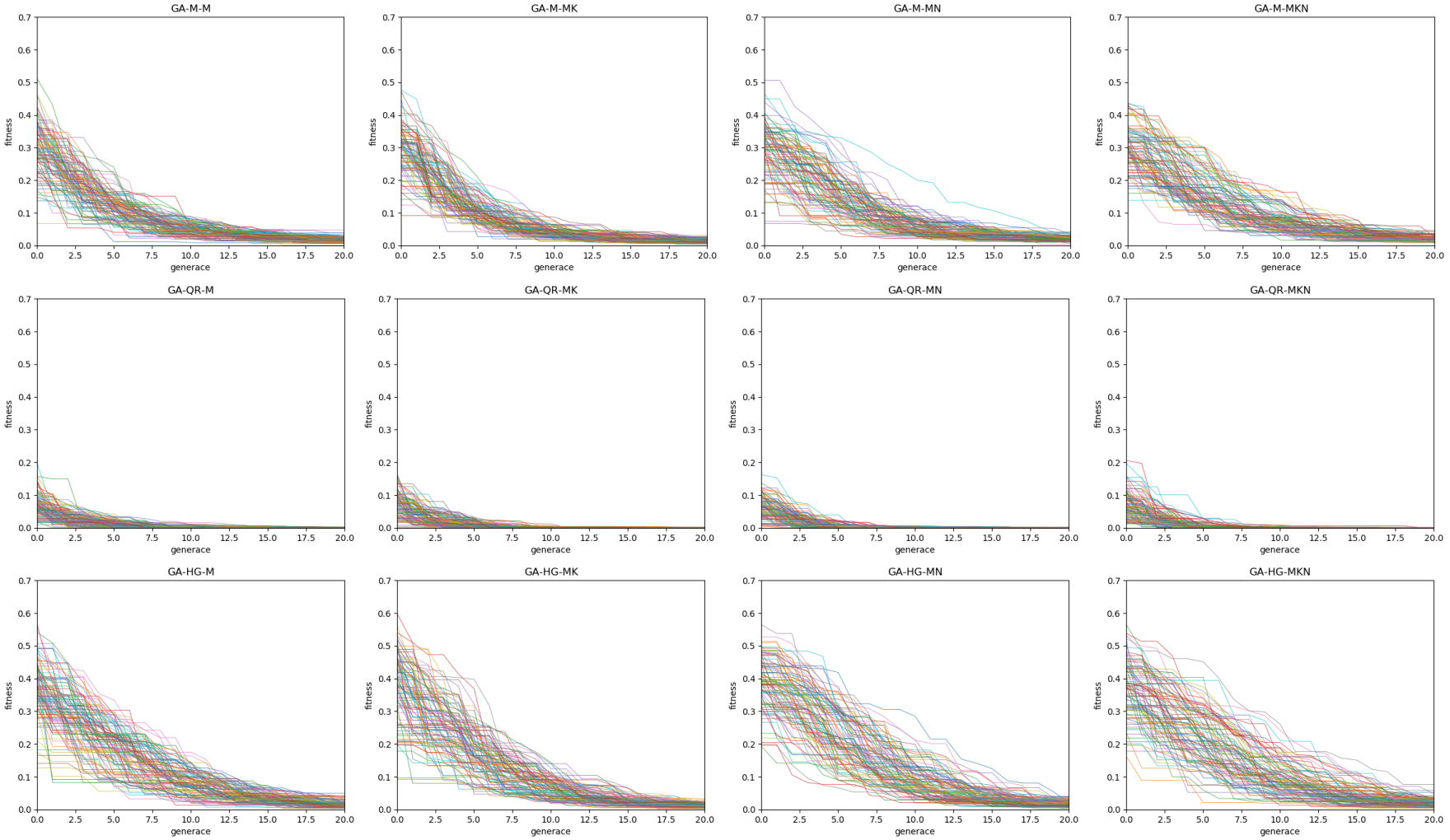
	ES											
	1+10			1+50			4+8			8+16		
	M	QR	HG	M	QR	HG	M	QR	HG	M	QR	HG
úspěšnost [%]	72,9	100	86,5	69,8	100	69,8	83,3	100	91,6	75	100	79,2
min. počet vyhodnocení fitness		430			550			520			448	

	GA											
	M			MK			MKN			MN		
	M	QR	HG	M	QR	HG	M	QR	HG	M	QR	HG
úspěšnost [%]	15,6	100	28,1	26	100	44,8	5,2	100	13,5	4,2	100	14,5
min. počet vyhodnocení fitness		1300			1100			1000			800	

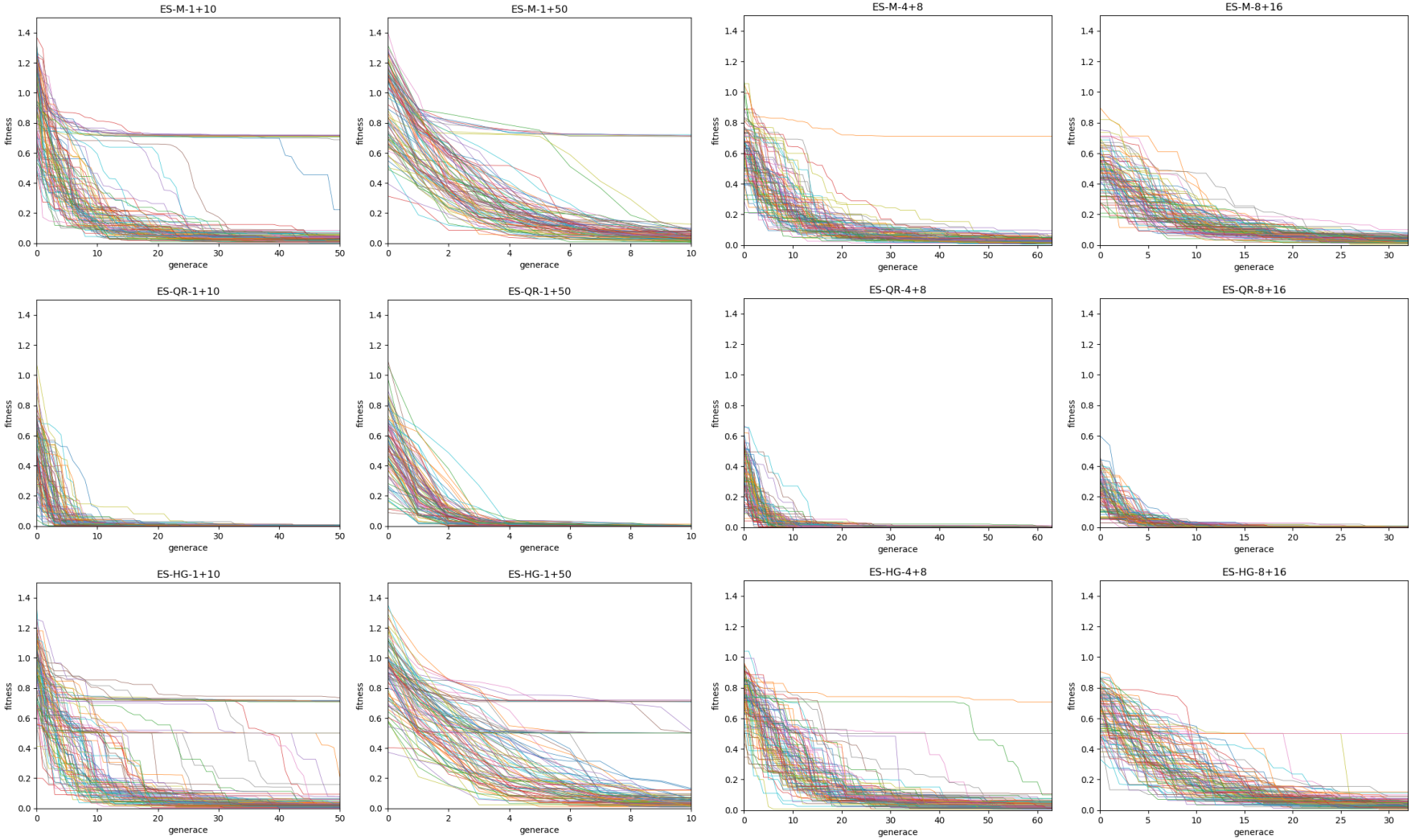
Tabulka 6.1: Procentuální úspěšnost a minimální počet vyhodnocení fitness k nalezení řešení s fitness menší než 0,01.



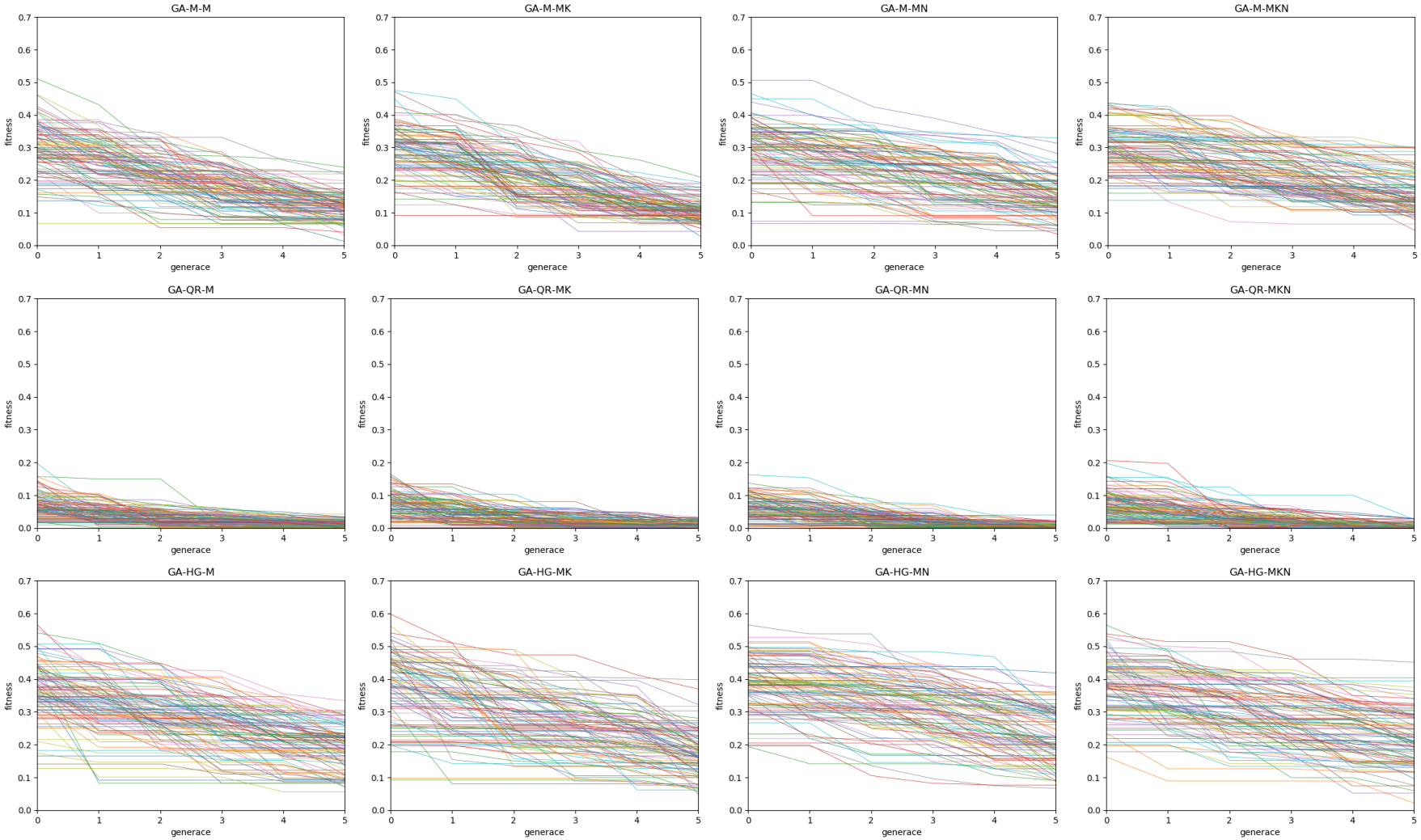
Obrázek 6.1: Experiment 1. Konvergenční křivky ES. V řádcích shora genotypy M, QR, HG. Ve sloupcích zleva strategie 1+10, 1+50, 4+8, 8+16.



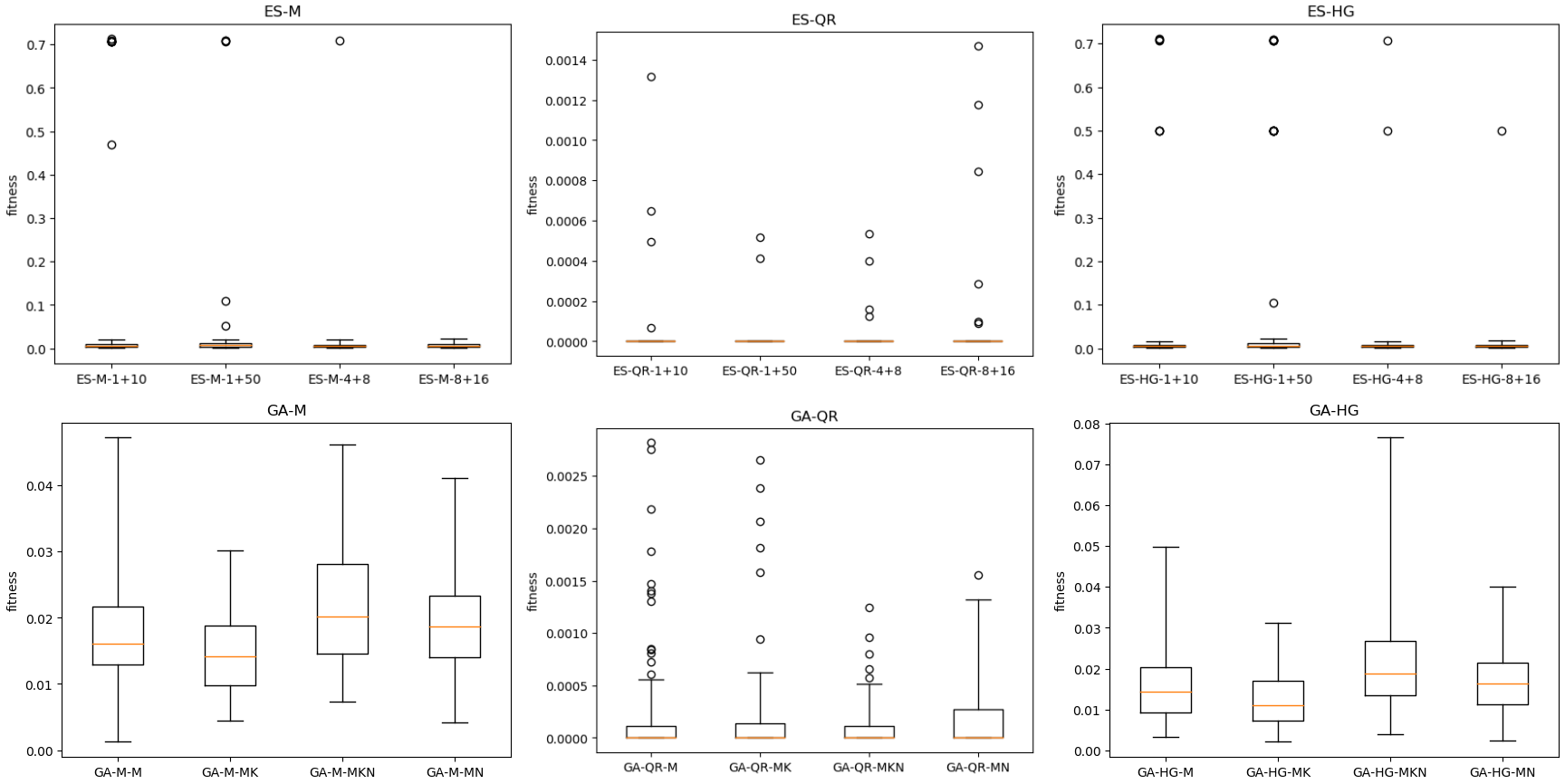
Obrázek 6.2: Experiment 1. Konvergenční křivky GA. V řádcích shora genotypy M, QR, HG. Ve sloupcích zleva techniky M, MK, MN, MKN.



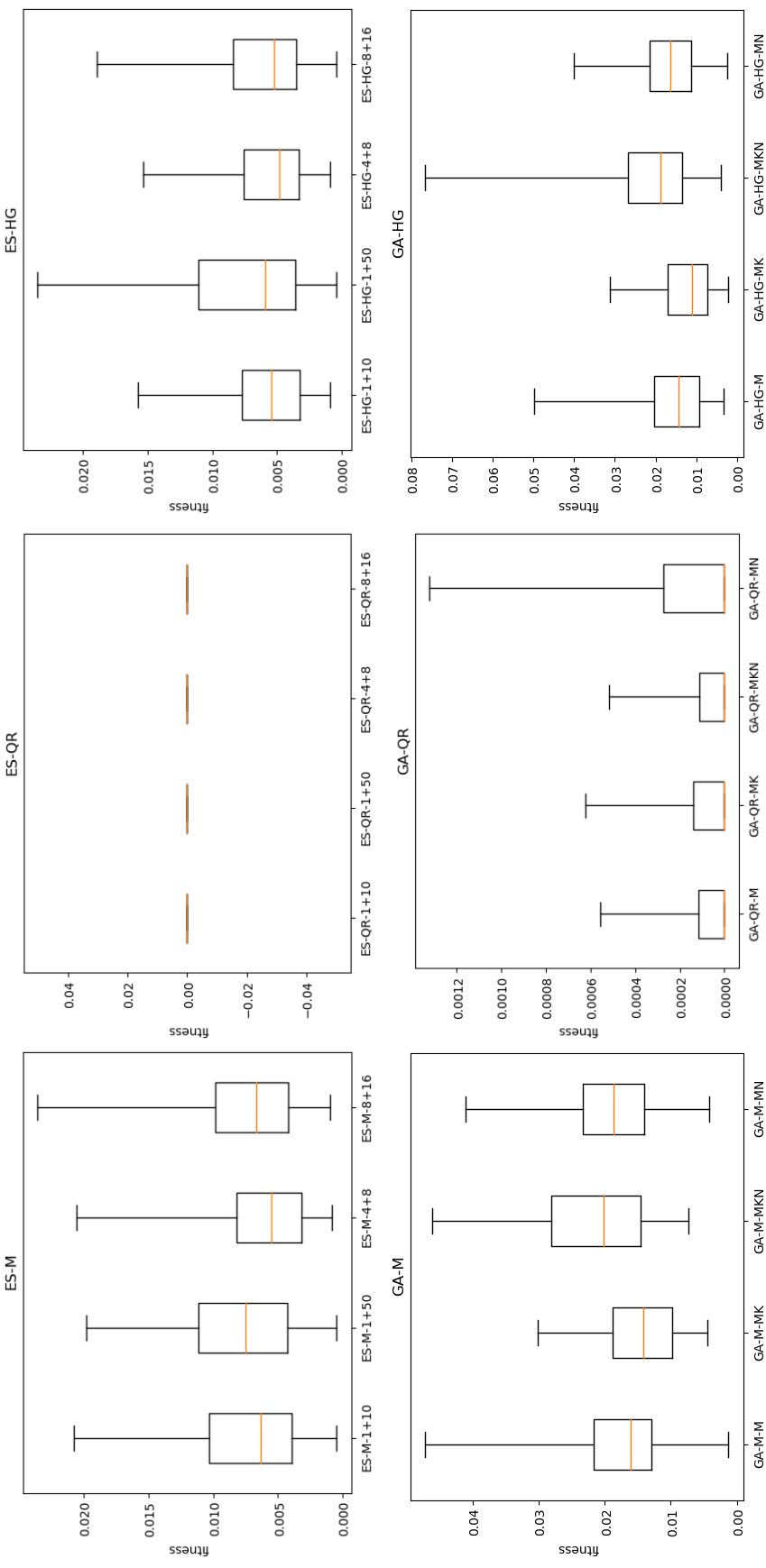
Obrázek 6.3: Experiment 1. Konvergenční křivky ES. Detail. V řádcích shora genotypy M, QR, HG. Ve sloupcích zleva strategie 1+10, 1+50, 4+8, 8+16.



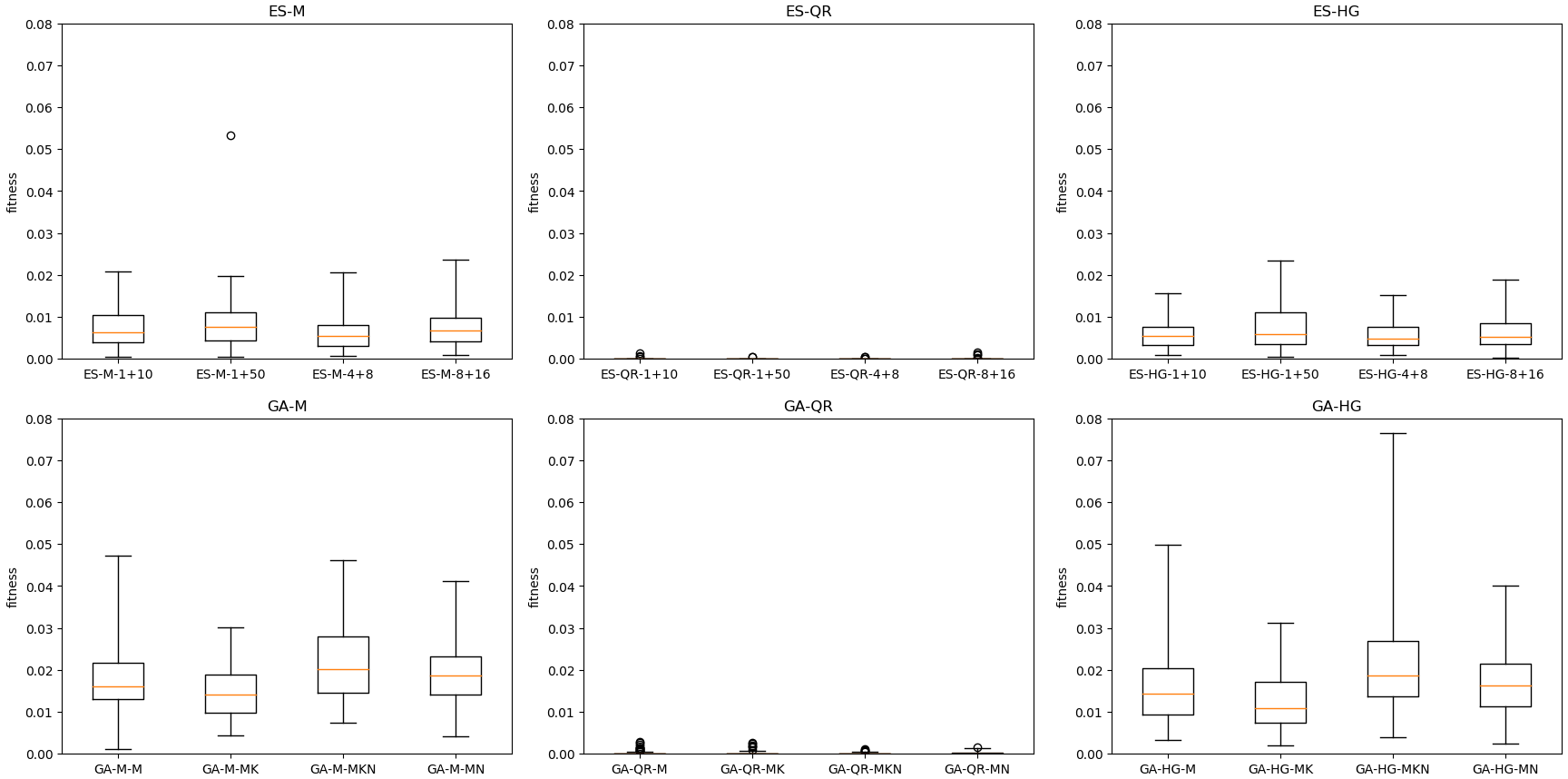
Obrázek 6.4: Experiment 1. Konvergenční křivky GA. Detail. V řádcích shora genotypy M, QR, HG. Ve sloupcích zleva techniky M, MK, MN, MKN.



Obrázek 6.5: Experiment 1. Statistické zhodnocení experimentů. Všechny hodnoty. V řádcích shora evoluční algoritmus ES, GA. Ve sloupcích zleva genotypy M, QR, HG.



Obrázek 6.6: Experiment 1. Statistické zhodnocení experimentů. Bez odlehlých hodnot. V řádcích shora evoluční algoritmus ES, GA. Ve sloupcích zleva genotypy M, QR, HG.



Obrázek 6.7: Experiment 1. Statistické zhodnocení experimentů. Stejné měřítko. V řádcích shora evoluční algoritmus ES, GA. Ve sloupcích zleva genotypy M, QR, HG.

6.2 Experiment 2: CNOT hradlo

V druhém experimentu jsem se snažil nalézt $CNOT$ hradlo. Toto hradlo pracuje nad dvěma qubity. Velikost genotypu se oproti předchozímu experimentu exponenciálně zvětšuje. Hledaná matice hradla $CNOT$ je

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{6.5}$$

Trénovací množina pro tento experiment obsahuje 4 páry

$$\left\{ \left(\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right) \right\}. \tag{6.6}$$

Protože se jedná o dvouqubitové hradlo, genotyp je větší a evoluce je tedy náročnější. Ukončující podmínka byla tentokrát nastavena na 300 generací u GA s populací 100 jedinců a na 3000 generací u ES 1+10. Hranice úspěšnosti byla zvýšena na 0,05.

Experimenty byly tentokrát zredukovány na 2 evoluční algoritmy: ES (1+10) a GA Mutace + Křížení. Vybral jsem ty, které v předchozím experimentu vycházeli nejlépe.

6.2.1 Výsledky

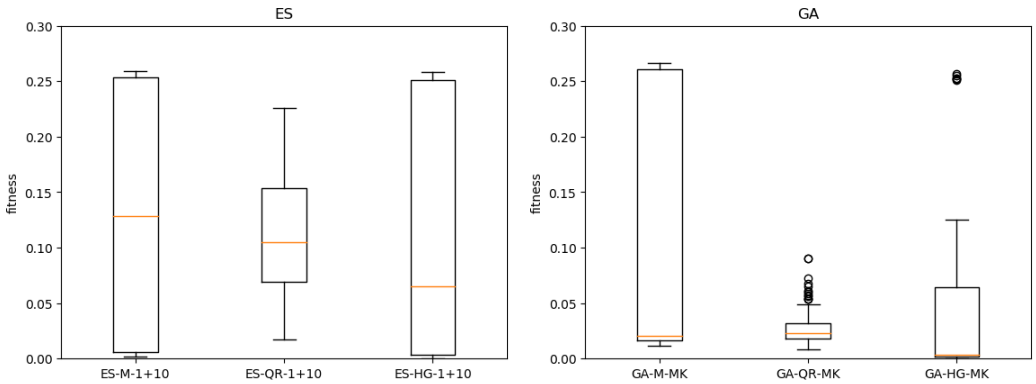
Výsledky jsou prezentovány stejnou formou jako v předchozím experimentu. Konvergenční křivky jsou na obrázku 6.9, v detailu na první osminu generací na obrázku 6.10. Protože je v tomto experimentu použito méně konfigurací než v předchozím, uspořádání grafů je mírně rozdílné. Stále je však zachováno stejné měřítko pro snadné porovnání. V řádcích jsou evoluční algoritmy v použité konfiguraci, ve sloupcích jsou genotypy. Statistické zhodnocení výsledků ve formě krabicových grafů je na obrázku 6.8. Zde jsou v levém grafu evoluční strategie a v pravém genetické algoritmy. Nakonec opět v tabulce 6.2 je spočítána úspěšnost. Tedy procento běhů, jejichž konečné řešení mělo fitness nižší než 0,05.

V tomto experimentu už fitness nedosahuje tak nízkých hodnot jako v předchozím. Experimenty byly opět úspěšné, převážně při použití GA, který v této metrice tentokrát výrazně převyšuje EA. Nově navržený genotyp QR má v GA největší úspěšnost avšak v ES nejnižší. To je vidět i na konvergenčních křivkách. Genotyp QR očividně nemá problém s uváznutím v lokálních extrémech, ale při větších přesnostech špatně konverguje. V tomto experimentu nejrychleji konverguje a má i nejnižší medián výsledků i nejlepší výsledek genotyp HG i přesto, že jeho úspěšnost je nejnižší. Důvodem je největší tendence uváznutí v lokálním extrému avšak rychlá konvergence i ve vysokých přesnostech. Pro tento experiment považuji genotyp HG jako nejvhodnější.

Nejlepší řešení v tomto experimentu je z konfigurace GA-HG-MK a je ukázáno v matici 6.7. Vidíme, že referenční matici je velmi blízko.

$$\begin{pmatrix} 1 + 0,000078i & 0 & 0,000486 + 0,000396i & 0 \\ 0 & 1 + 0,000495i & 0 & 0 \\ 0 & 0 & 0,000501 + 0,000143i & 0,999998 - 0,002069i \\ -0,000485 + 0,000397i & 0 & 0,999999 - 0,001086i & -00005 + 0,000145i \end{pmatrix} \tag{6.7}$$

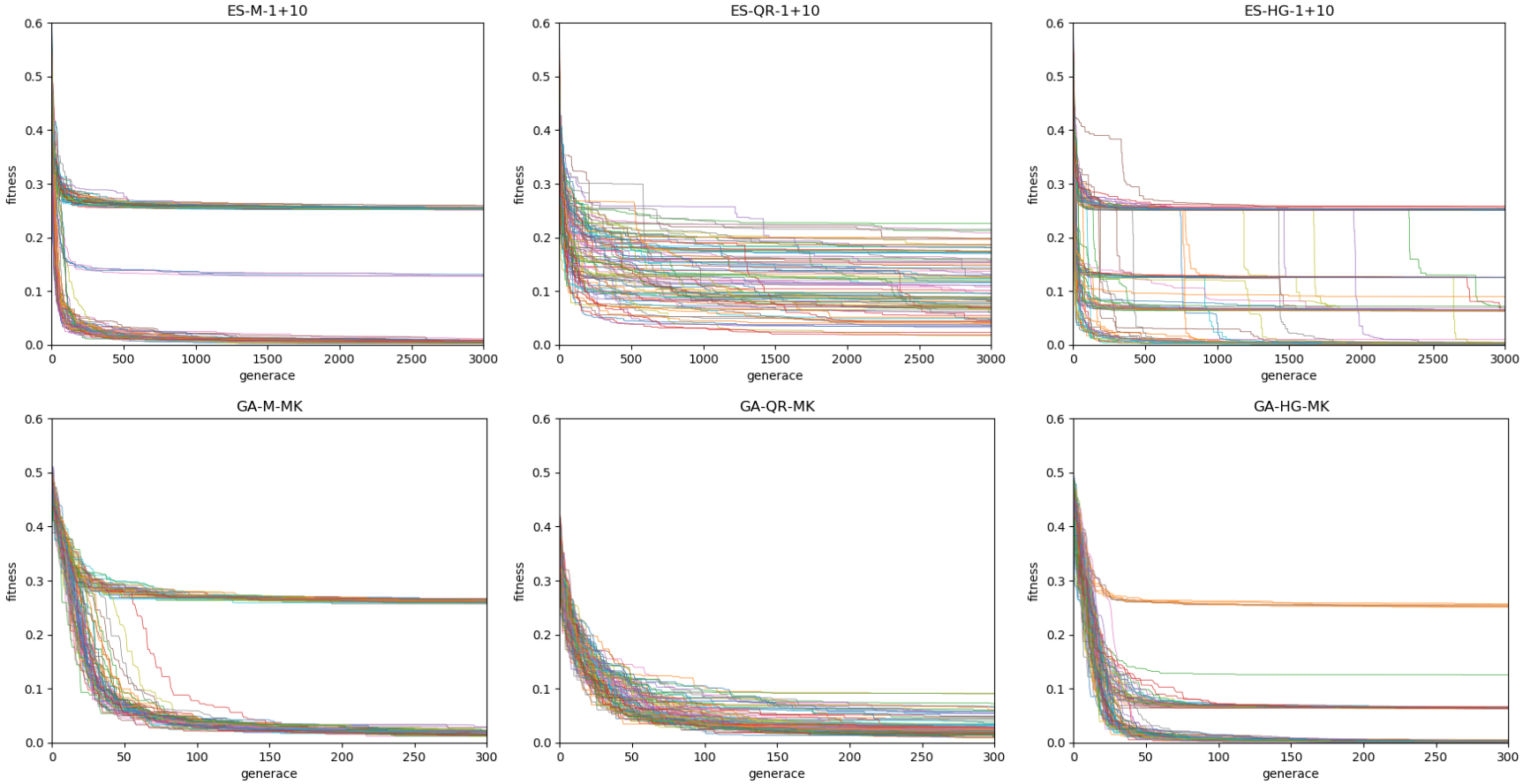
Tentokrát při srovnání ES a GA vidíme lepší úspěšnost GA i lepší konečné hodnoty fitness ve statistickém zhodnocení. I přes rychlejší počáteční konvergenci ES při velkých přesnostech špatně konvergují. To je dáno koeficientem mutace σ v evoluční strategii. Tady by dobře zafungovaly techniky adaptace nebo samoadaptace tohoto parametru.



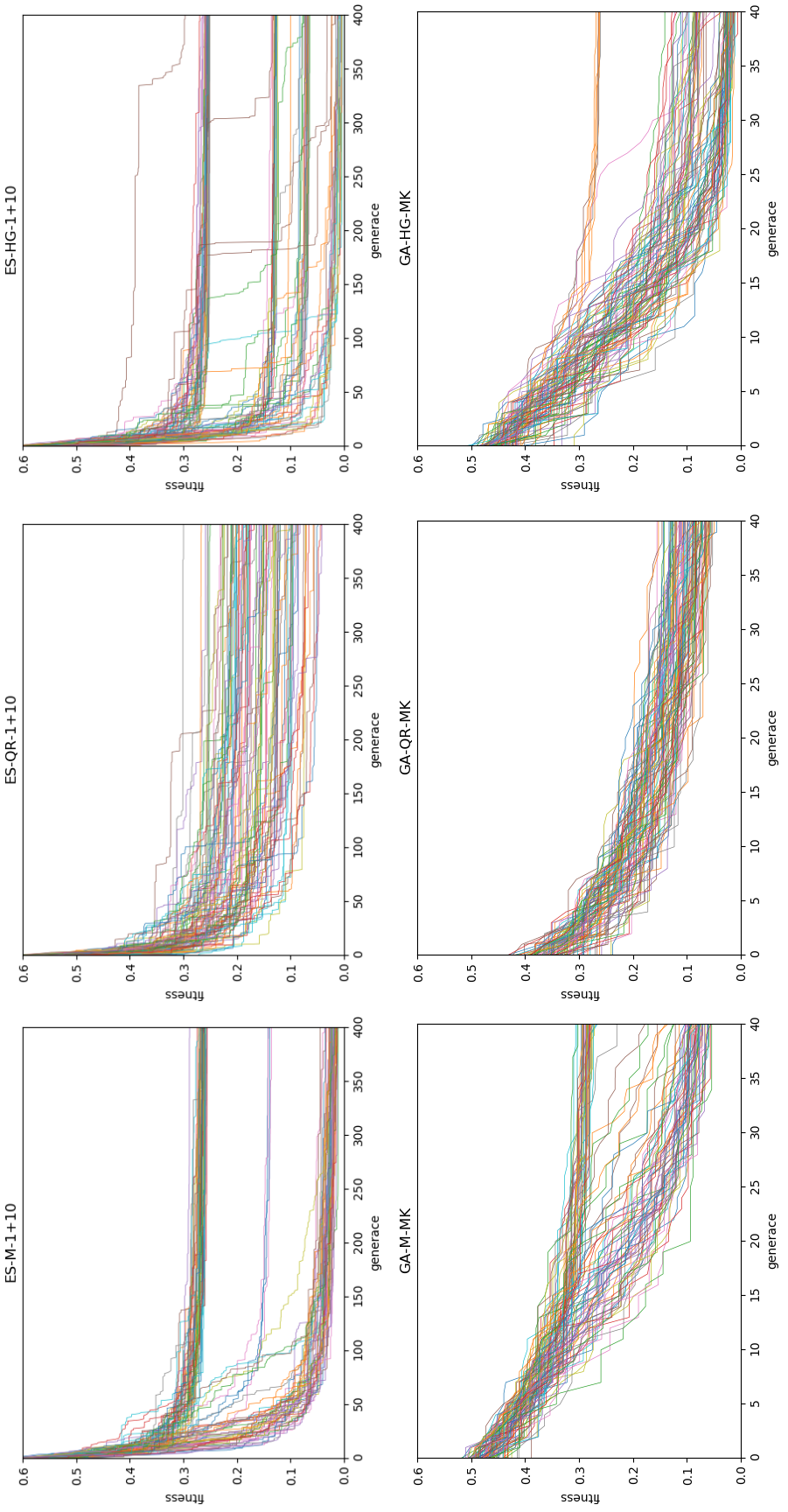
Obrázek 6.8: Experiment 2. Statistické zhodnocení výsledků. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	47,9	15,6	31,3	64,6	86,5	57,3

Tabulka 6.2: Procentuální úspěšnost nalezení řešení s fitness menší než 0,05.



Obrázek 6.9: Experiment 2. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.10: Experiment 2. Konvergenční křivky. Detail. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.

6.3 Experiment 3: Provázání

Ve třetím experimentu jsem generoval hradla pro maximální provázání 2 a 3 qubitů, tedy matici, která ze stavu $|0\ldots0\rangle$ udělá stav $1/\sqrt{2}(|0\ldots0\rangle + |1\ldots1\rangle)$. Stejně byla nastavena i trénovací množina. Tento experiment jsem zahrnul, protože ho provedli ve svých pracích i Hutsell a Greenwood i MacKinnon. Dalším důvodem pro tento experiment je otestování škálovatelnosti evolučního systému.

Pro provedení požadované operace pro 2 qubity lze použít matici U_2 (6.8) a pro 3 qubity matici U_3 (6.9), ve kterých $\times$ značí libovolná komplexní čísla.

$$U_2 = \begin{pmatrix} 1/\sqrt{2} & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 1/\sqrt{2} & \times & \times & \times \end{pmatrix}, \tag{6.8}$$

$$U_3 = \begin{pmatrix} 1/\sqrt{2} & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 1/\sqrt{2} & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}, \tag{6.9}$$

Esperimenty pro 2 qubity proběhli na 5000 vyhodnocení fitness s prahem úspěšnosti 0,01. Experimenty se 3 qubity skončily po 10000 vyhodnocení fitness. Za práh úspěšnosti byla zvolena hodnota 0,02.

6.3.1 Výsledky

Obrázky 6.12 až 6.11 a tabulka 6.3 zobrazují data pro experiment se 2 qubity. Konvergenční křivky na obrázku 6.12 jsou opět pro porovnání ve stejném měřítku. Detailní grafy na obrázku 6.13 jsou tentokrát ve stejném měřítku osy x pouze po řádcích. Obrázek 6.11 ukazuje opět statistické vyhodnocení výsledků formou krabicových grafů. V tabulce 6.3 přibyl řádek minimálního počet vyhodnocení fitness při kterém všechny běhy skončily úspěšně. Obdobně jsou zobrazeny i výsledky experimentu se 3 qubity: konvergenční křivky celého běhu na obrázku 6.14, detail konvergenční křivky na obrázku 6.15, statistické zhodnocení výsledků formou krabicových grafů na obrázku 6.16, detail krabicových grafů na obrázku 6.17 a tabulka 6.4 s procentuální úspěšností běhů.

V experimentu se dvěma qubity vidíme veliký úspěch u většiny konfigurací. Jediná konfigurace GA-M-MK se v konečných hodnotách fitness úspěchu pouze přiblížila. Ve statistickém zhodnocení vidíme, že v obou algoritmech dosahuje nejnižšího mediánu genotyp QR. Druhý řádek tabulky 6.3 ukazuje, že z maximálního počtu vyhodnocení fitness funkce, který byl dán ukončující podmínkou na 5000 stačilo v ES při použití genotypu QR pouze 1490, což je z použitých genotypů nejméně.

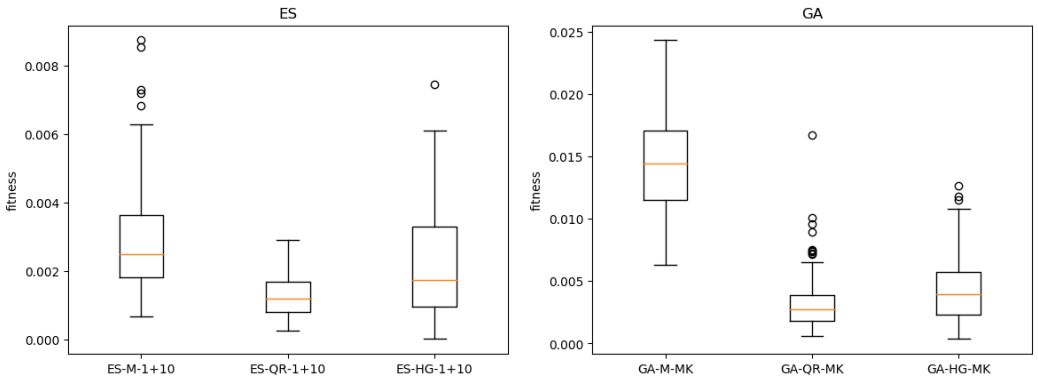
Experiment se třemi qubity už nebyl z celkového pohled tolik úspěšný. V ES dosáhl genotyp QR stoprocentní úspěšnosti spolu s genotypem HG. V GA se pouze genotyp QR přiblížil 90%. Genotyp HG zůstal pod 20% úspěšnosti a genotyp M ji měl dokonce nulovou. I přesto, že ES s QR má menší minimální počet vyhodnocení fitness pro stoprocentní úspěšnost, Z detailního záběr statistického zhodnocení na obrázku 6.17 vidíme, že konečná fitness po celém běhu byla nižší u genotypu HG. V GA je však kvalita genotypu QR oproti ostatním nepochybná.

V konvergenčních křivkách je vidět slabá schopnost vyproštění se z lokálního extrému. Hlavně pak u GA, jejichž hlavním genetickým operátorem je křížení. To je v tomto experimentu absolutně nevhodné. Ve výsledku tohoto experimentu na mnoha hodnotách výsledné matice vůbec nezáleží, tudíž variování s jejich hodnotami nepřináší zlepšení. U Genotypu QR dochází při převodu genotypu na fenotyp k větší interakci mezi jednotlivými hodnotami v genotypu a nedochází proto k uváznutí. V tomto experimentu je to velkou výhodou. ES naproti tomu ukazují schopnost tuto bariéru překonávat.

Nejlepší nalezená řešení jsou vyobrazena v rovnicích 6.10 a 6.11. Pro úsporu místa jsou čísla neovlivňující fitness zaokrouhleny na 2 desetinná místa. Na nich je vidět, jak přesné jsou výsledné matice vůči referenčním (6.8, 6.9).

$$U_2 = \begin{pmatrix} 0.707084 & 0 & -0.139003 & 0,693333 \\ 0 & -0,850841 & -0,515172i & -0,103285i \\ 0 & 0,525423i & 0,83424 & 0,167253 \\ 0,70713 & 0 & 0,138994 & -0,693288 \end{pmatrix} \tag{6.10}$$

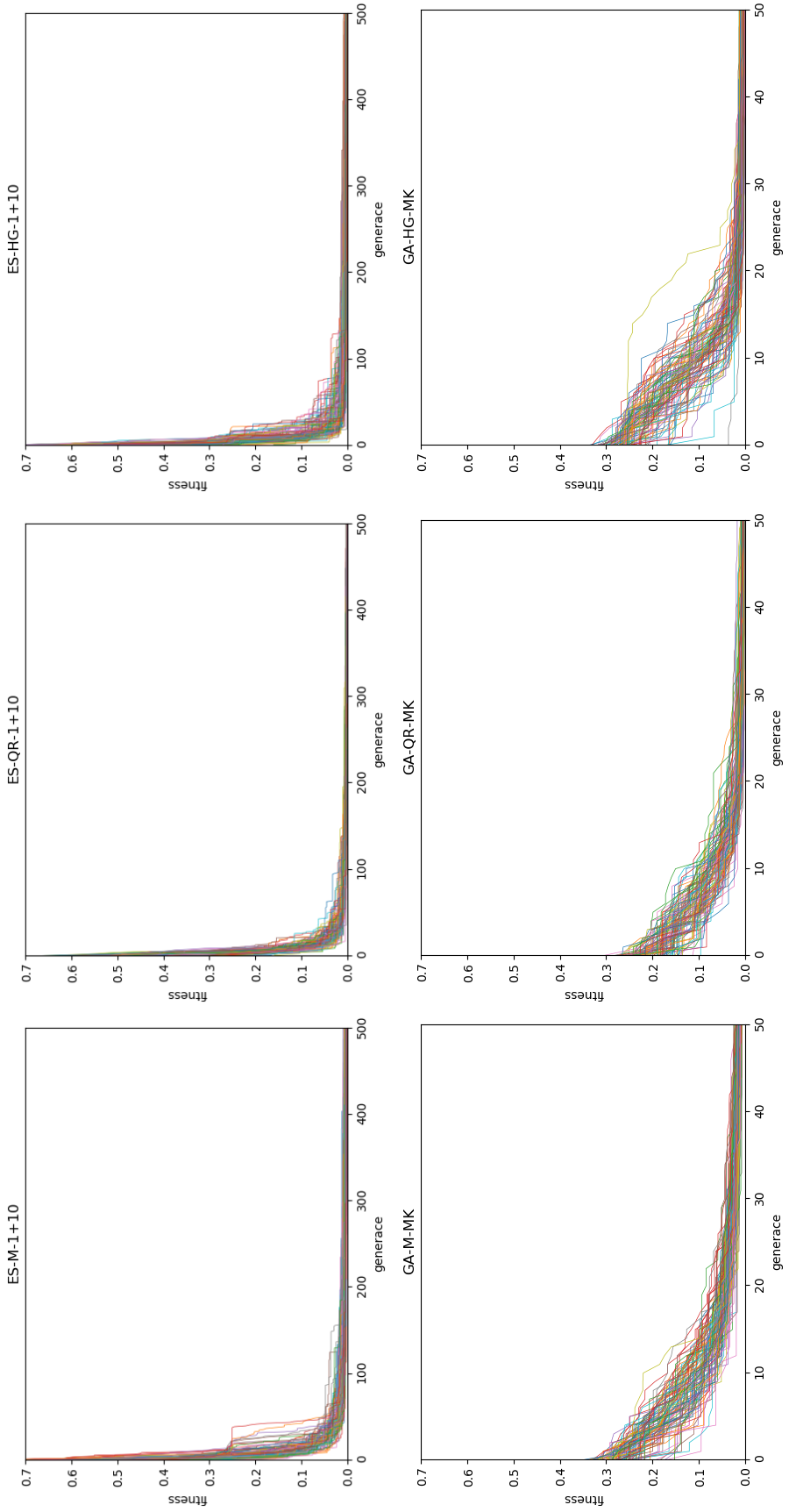
$$\begin{pmatrix} 0,708085 & 0 & 0 & 0,91-0,53i & 0,04-0,23i & -0.18-0.34i & 0.02-0.01i & 0.01+0.04i \\ 0 & 0,29+0,09i & -0,01-0,05i & -0,22-0,38i & 0,22+0,09i & 0,43+0,33i & 0,47-0,24i & -0,17+0,22i \\ 0 & 0,09+0,11i & 0,01-0,03i & 0,01-0,18i & -0,02+0,32i & 0,13+0,04i & -0,01+0,18i & 0,89+0,01i \\ 0 & -0,06+0,05i & 0,01 & 0,07-0,03i & -0,74+0,34i & 0,19-0,33i & 0,29-0,24i & -0,1-0,17i \\ 0 & -0,18+0,15i & 0,04+0,01i & 0,33-0,02i & -0,01-0,2i & -0,21+0,36i & -0,01-0,73i & 0,23-0,18i \\ 0 & -0,51-0,63i & -0,21+0,36i & -0,26i & 0,01+0,17i & 0,14+0,21i & -0,04+0,02i & -0,01-0,02i \\ 0 & 0,26+0,32i & -0,45+0,78i & 0,01+0,07i & -0,01-0,05i & -0,05-0,06i & 0,01-0,01i & 0,01 \\ 0.706127 & 0 & 0 & -0,09+0,54i & -0,04+0,23i & 0,18+0,35i & -0,02+0,01i & -0,01-0,04i \end{pmatrix} \tag{6.11}$$



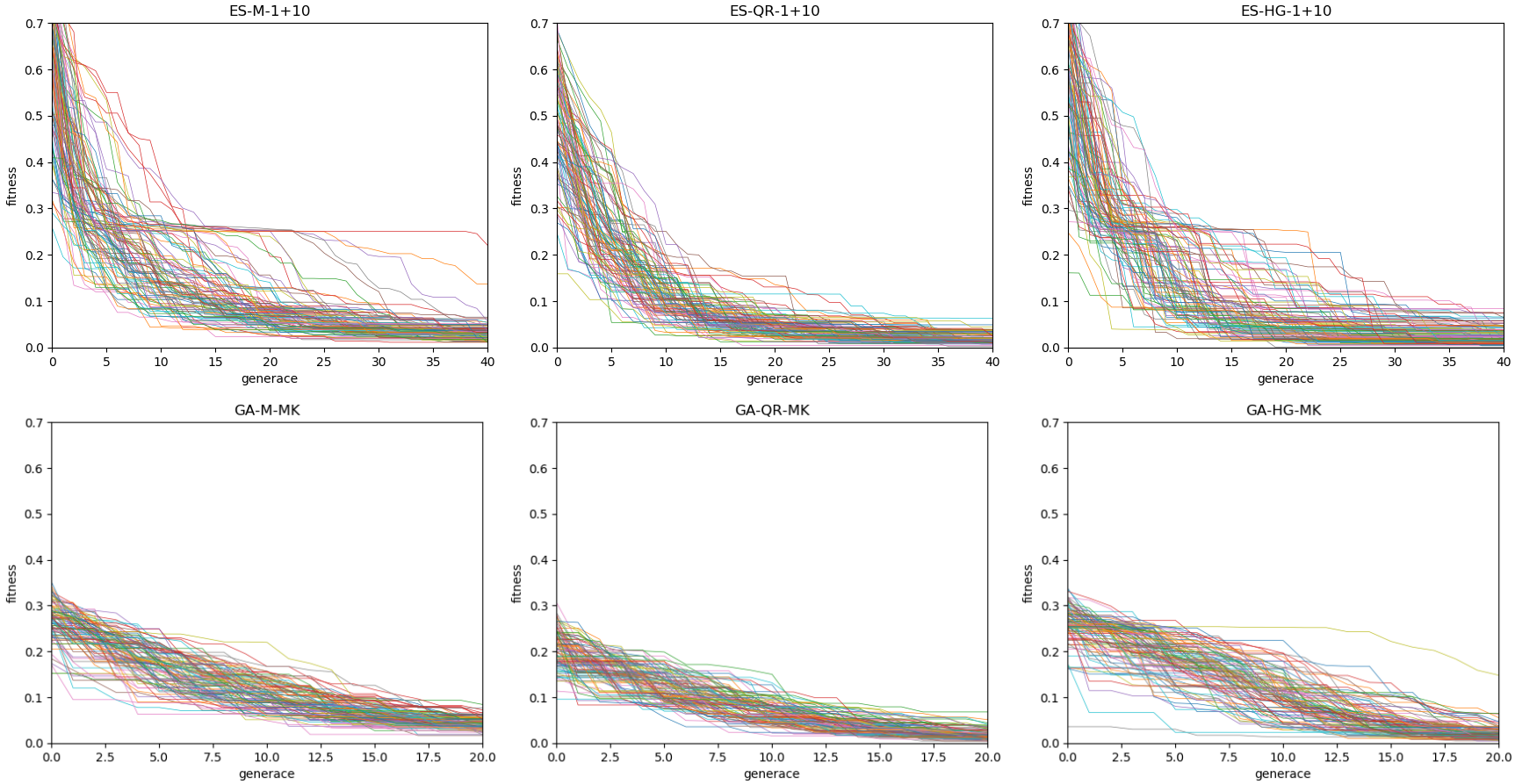
Obrázek 6.11: Experiment 3 pro 2 qubity. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	100	100	100	17,7	97,9	95,8
min. počet vyhodnocení fitness	4050	1490	3430			

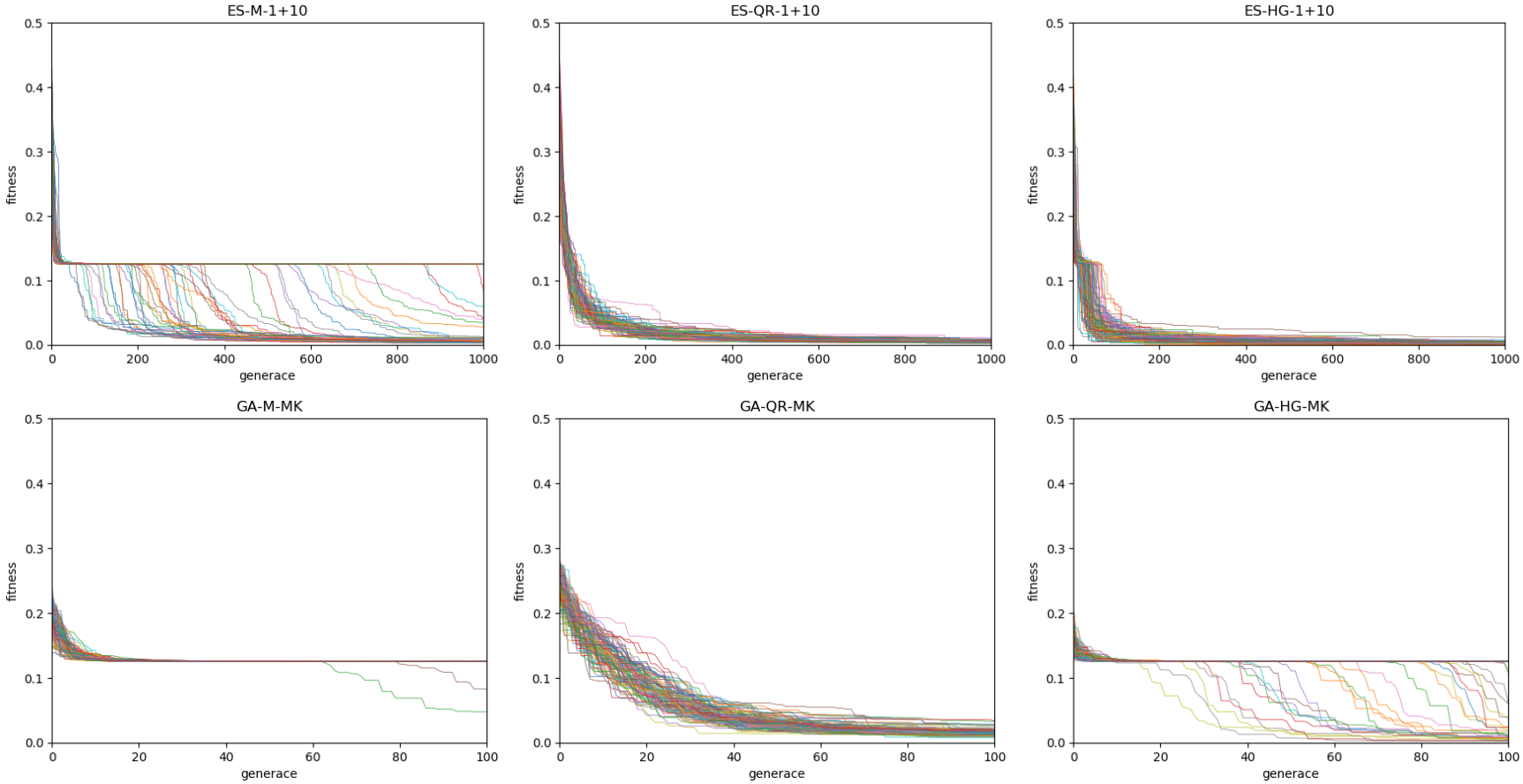
Tabulka 6.3: Experiment 3 pro 2 qubity. Procentuální úspěšnost nalezení řešení s fitness menší než 0,01.



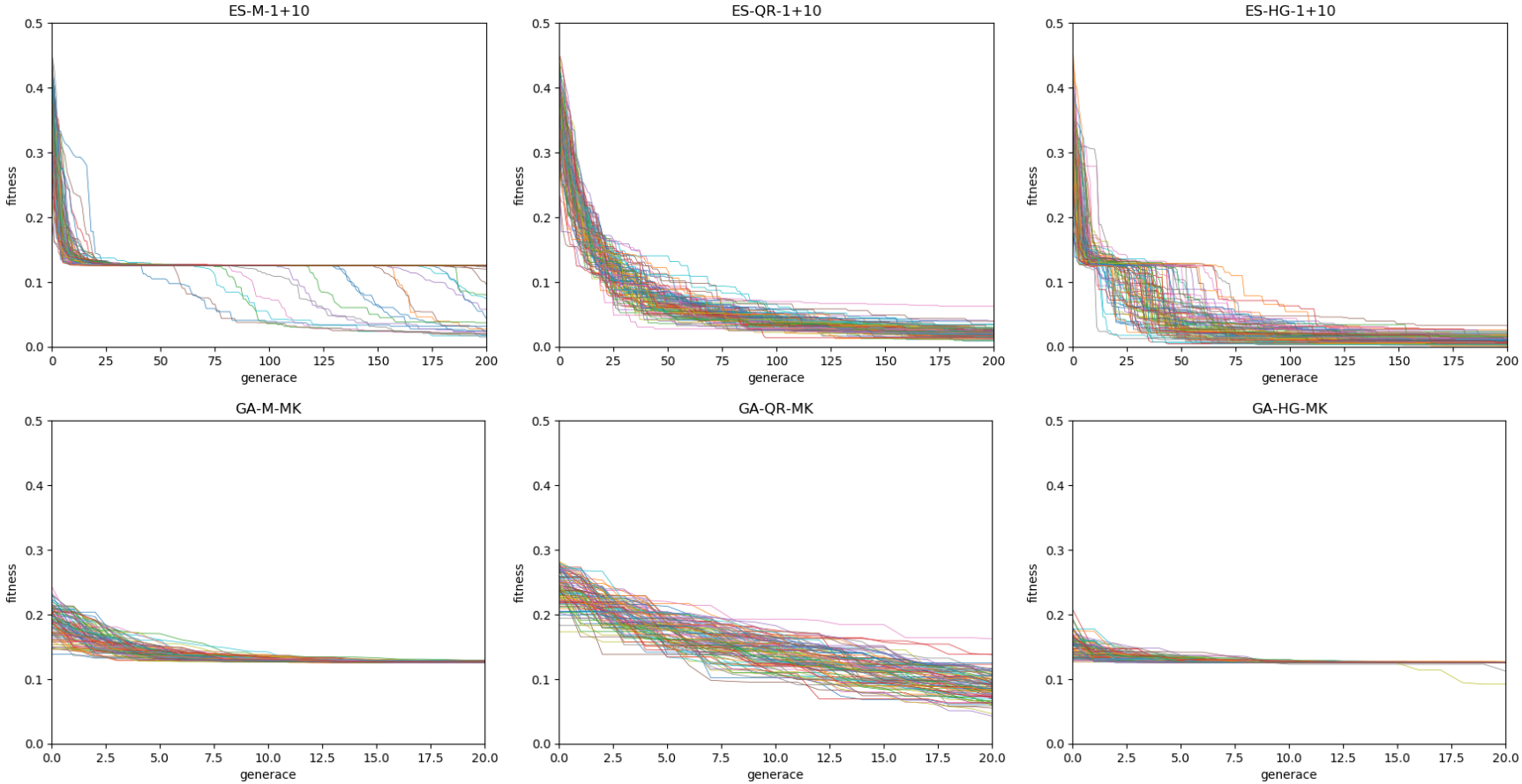
Obrázek 6.12: Experiment 3 pro 2 qubity. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



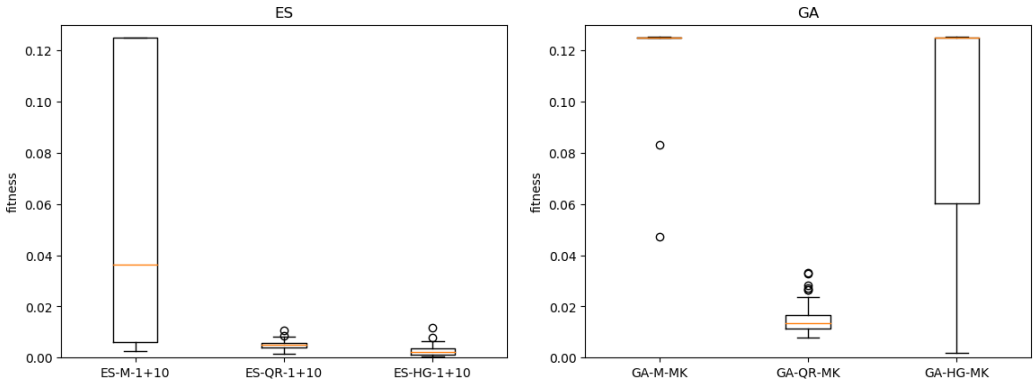
Obrázek 6.13: Experiment 3 pro 2 qubity. Konvergenční křivky. Detail. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



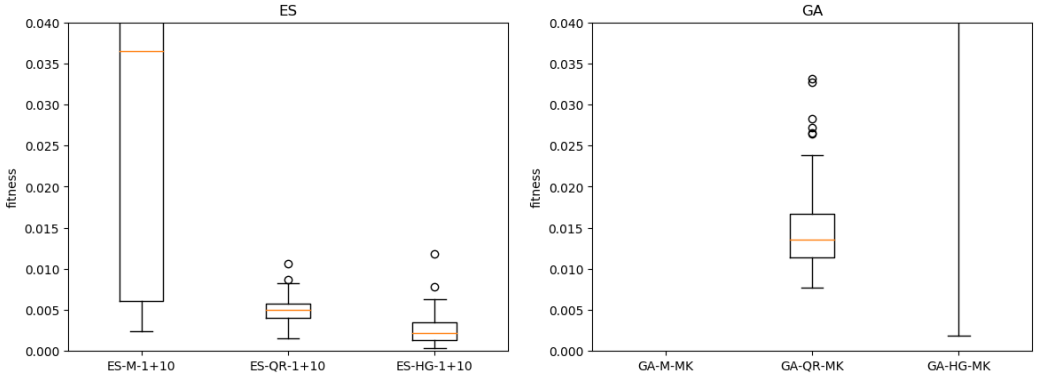
Obrázek 6.14: Experiment 3 pro 3 qubity. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.15: Experiment 3 pro 3 qubity. Konvergenční křivky. Detail. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.16: Experiment 3 pro 3 qubity. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.



Obrázek 6.17: Experiment 3 pro 3 qubity. Statistické zhodnocení experimentů. Detail. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	47,9	100	100	0	89,6	19,8
min. počet vyhodnocení fitness		4870	5270			

Tabulka 6.4: Experiment 3 pro 3 qubity. Procentuální úspěšnost nalezení řešení s fitness menší než 0,02.

6.4 Experiment 4: Klonování

Tento experiment je převzatý od Gregora, který se inspiroval "no-clonning"teorémem. Ten říká, že neexistuje hradlo U takové, že

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle, \tag{6.12}$$

pro libovolný stav $|\psi\rangle$. [58] V tomto experimentu jsem navrhl dvě varianty. První varianta nazvaná *Cloning* je shodná s Gregorovým postupem. Do trénovací množiny vygeneruji jeden náhodný stav a budu hledat matici která jej naklonuje podle rovnice 6.12.

V druhé variantě nazvané *No-cloning* vygeneruju 10 náhodných stavů a budu hledat univerzální matici, která by zvládla naklonovat všech deset. Pokud je "no-cloning"teorém správný, takovou matici by se nemělo podařit najít.

Pro variantu 1 jsem nastavil ukončující podmínku na 5000 vyhodnocení fitness. Algoritmy ve variantě 2 jsem nechal při hledání teoreticky neexistující matice běžet až po dobu 20000 vyhodnocení fitness. Hranice úspěšnosti byla nastavena na 0,01.

6.4.1 Výsledky

Varianta 1: Cloning

Výsledky tohoto experimentu jsou opět graficky znázorněny v konvergenčních křivkách na obrázku 6.18. Statistické zhodnocení všech hodnot ve formě krabicových grafů je na grafech 6.19. Na obrázku 6.20 jsou tyto krabicové grafy obrány o odlehlé hodnoty a detailněji přiblíženy k hodnotám blízkým nule. I tento experiment hodnotím úspěšně. Nejlepšího řešení dosáhla konfigurace ES-M-1+10, ve kterém se hledala matice U splňující provádějící operaci zobrazenou v rovnici 6.11 na náhodně vygenerovaném stavu také ukázaném v této rovnici.

$$U \begin{pmatrix} 0.718349 + 0.688145i \\ 0.036327 + 0.095454i \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.0424817 + 0.988657i \\ -0.0395908 + 0.0935675i \\ -0.0395908 + 0.0935675i \\ -0.00779182 + 0.00693511i \end{pmatrix} \tag{6.13}$$

Výsledná matice má tvar

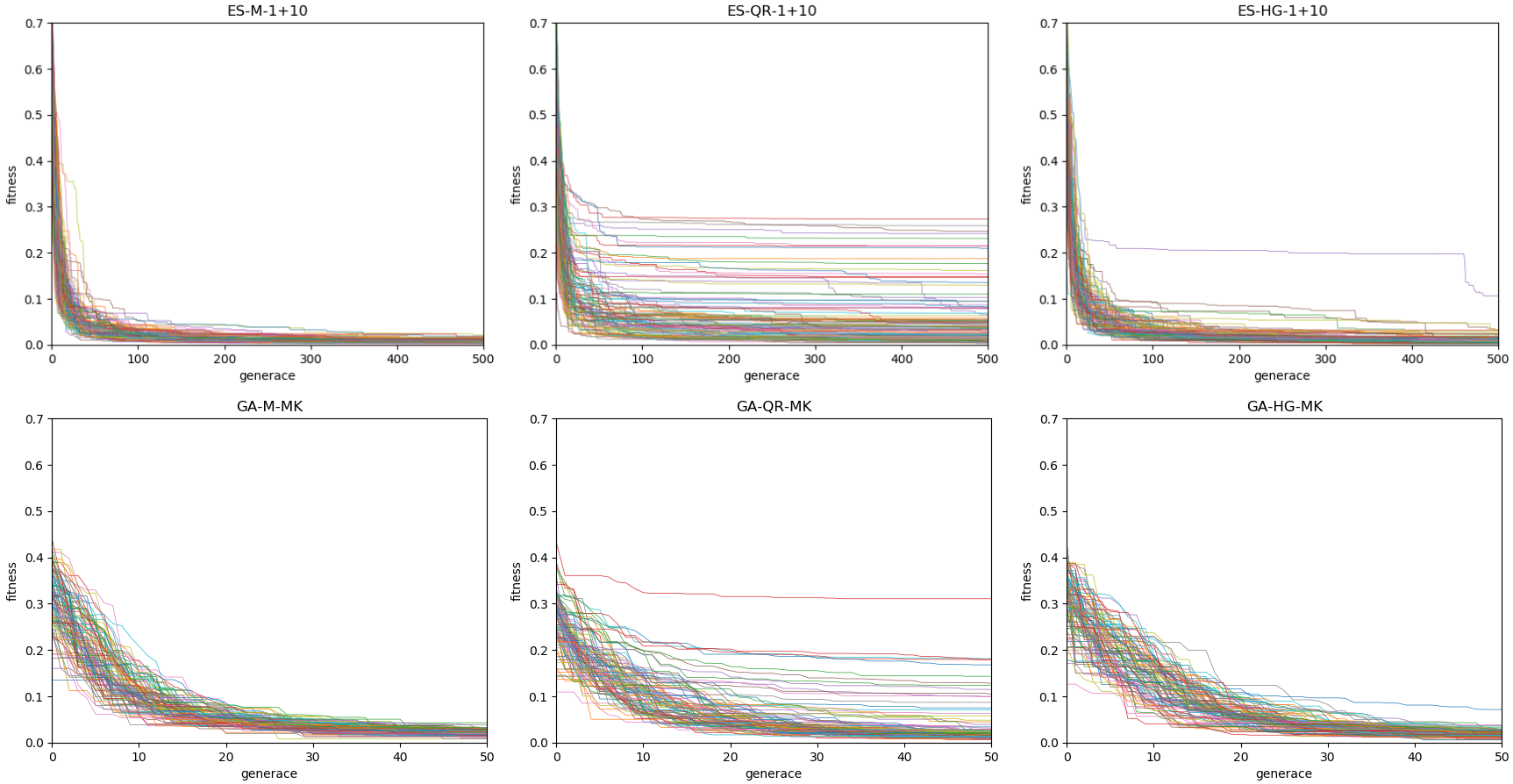
$$U = \begin{pmatrix} 0.718227 + 0.680729i & 0.0322104 + 0.0654202i & 0.0671712 - 0.0607362i & -0.046992 + 0.0709351i \\ 0.0211489 + 0.0865274i & 0.174397 + 0.0122076i & -0.0356817 + 0.205131i & 0.899789 - 0.32944i \\ -0.00438312 + 0.074529i & 0.45216 - 0.00033837i & -0.855396 + 0.0881824i & -0.191672 - 0.11731i \\ -0.0788032 - 0.032252i & 0.869652 + 0.0581014i & 0.454445 - 0.0494891i & -0.088332 + 0.1276998i \end{pmatrix} \tag{6.14}$$

a její aplikací na vstupní stav v rovnici 6.13 dostáváme výstupní stav

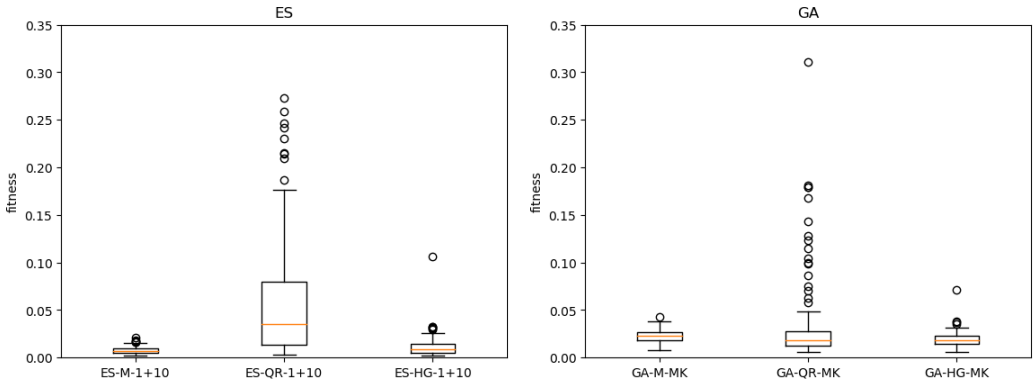
$$\begin{pmatrix} 0.0424228 + 0.9886964i \\ -0.0391810 + 0.0938007i \\ -0.0379774 + 0.0936697i \\ -0.0083683 + 0.0077261i \end{pmatrix} \tag{6.15}$$

Při porovnání s referenčním výstupním stavem dostáváme shodu na 2 desetinná místa. Můžu tedy říct, že se klonovací hradlo podařilo vygenerovat. Pro jeho přesnější podobu by bylo potřeba využít pokročilejších technik evolučního počítání. No cloning teorém samozřejmě porušen nebyl, protože vygenerované hradlo není schopné klonovat libovolný stav.

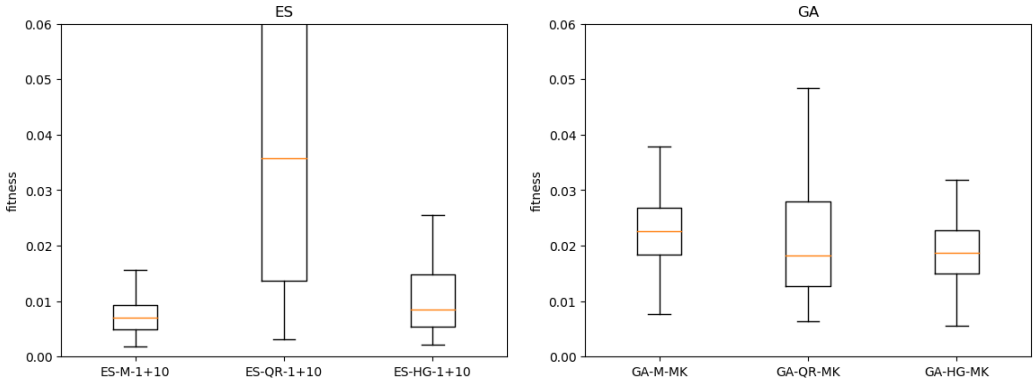
V tomto experimentu si genotyp QR vedl rozhodně nejhůře. Medián hodnot fitness dosažených výsledků v ES je u QR řádově vyšší než u ostatních genotypů. V obou algoritmech GA i ES genotyp QR hůře konvergoval.



Obrázek 6.18: Experiment 4 varianta Cloning. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.19: Experiment 4 varianta Cloning. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.



Obrázek 6.20: Experiment 4 varianta Cloning. Statistické zhodnocení experimentů. Detail. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	77,1	17,7	58,3	3,1	9,4	7,3

Tabulka 6.5: Experiment 4 varianta Cloning. Procentuální úspěšnost nalezení řešení s fitness menší než 0,01.

Varianta 2: No-cloning

V tomto experimentu už šlo o snahu vyvrátit nebo potvrdit no-cloning teorém. Cílem bylo najít operátor, který by zvládl klonovat 10 různých, náhodně vygenerovaných stavů. Výsledky jsou opět vyjádřeny pomocí konvergenčních křivek a statistického zhodnocení krabicovými grafy v obrázcích 6.22 a 6.21. Tento experiment považuji za úspěšný. Přestože se nepodařilo najít požadovaný operátor, evoluční algoritmy našli nejlepší možné řešení celkem v krátkém počtu vyhodnocení fitness. Z konvergenčních křivek vidíme, že po 2500 vyhodnocení fitness už se řešení téměř nezlepšilo. Ze statistického zhodnocení pak vidíme, že nejlepší výsledky mají fitness v okolí hodnoty 0,1, medián vša kolísá kolem hodnoty 0,15. To je od hranice úspěšnosti 0,02 daleko. Nakonec následující ukázka

matice s nejnižší fitness s aplikací na jeden z náhodných stavů nám také potvrdí, že no-cloning teorém nebyl překonán. Vstupní stav I a referenční výstupní stav O jsou

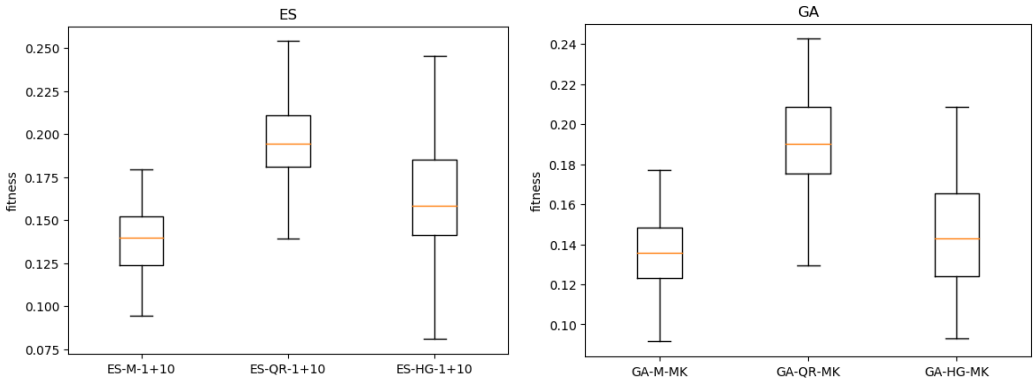
$$I = \begin{pmatrix} 0.38457 + 0.596706i \\ 0.543271 + 0.448224i \\ 0 \\ 0 \end{pmatrix}, O = \begin{pmatrix} -0.208163 + 0.45895i \\ -0.0585323 + 0.496546i \\ -0.0585323 + 0.496546i \\ 0.0942381 + 0.487015i \end{pmatrix} \tag{6.16}$$

Vygenerovaná matice U má podobu

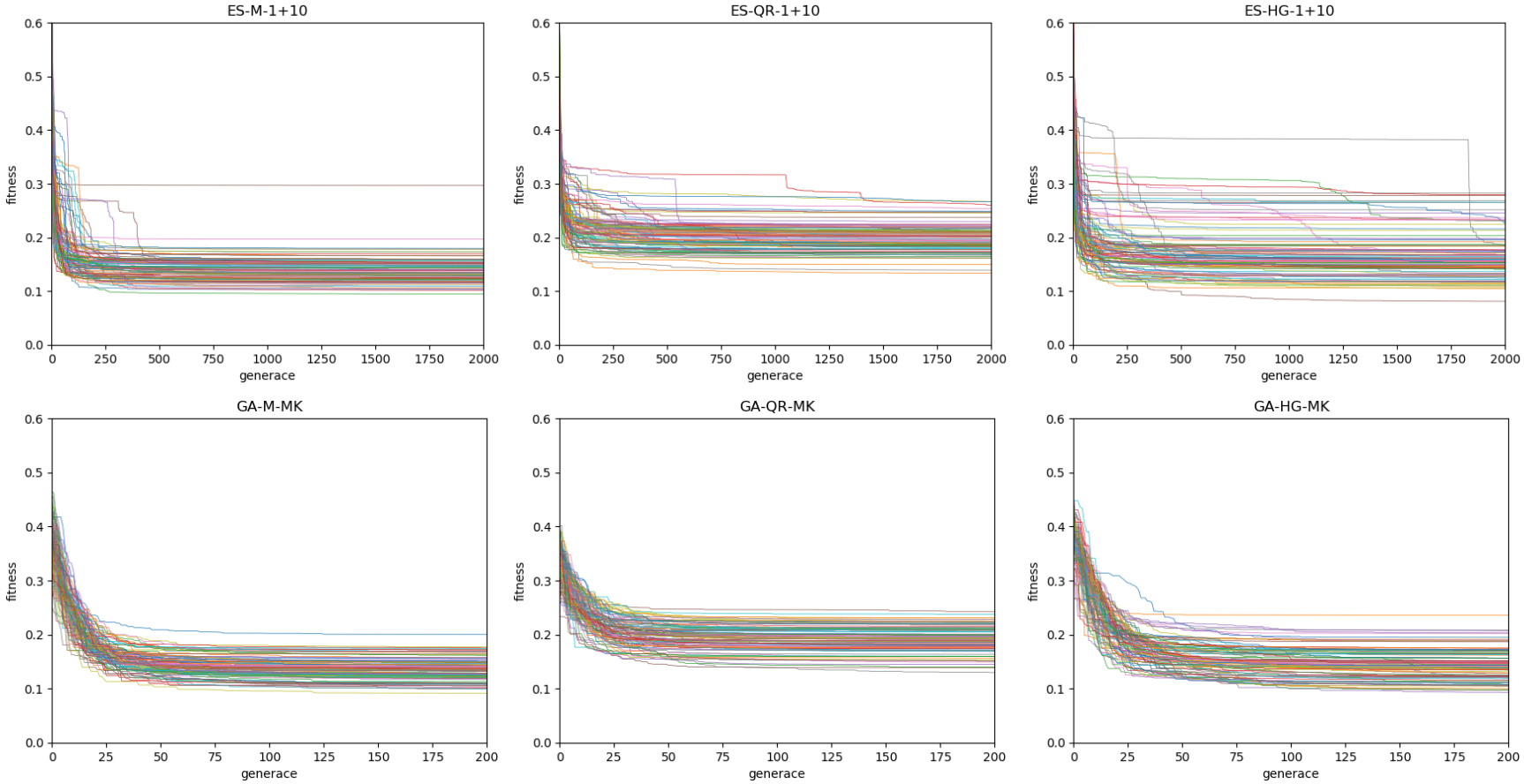
$$U = \begin{pmatrix} 0.545119 + 0.532073i & -0.0634638 - 0.10318i & 0.208088 + 0.323328i & 0.206293 + 0.463327i \\ 0.356394 + 0.334263i & 0.164511 + 0.13418i & 0.315408 - 0.130545i & -0.528489 - 0.566002i \\ 0.286658 + 0.273529i & 0.235476 + 0.205956i & -0.703354 - 0.400406i & 0.281806 - 0.103421i \\ -0.0879211 - 0.127573i & 0.690897 + 0.604196i & 0.204673 + 0.196752i & 0.0657443 + 0.220634i \end{pmatrix}. \tag{6.17}$$

Aplikací U na stav I vznikne stav

$$\begin{pmatrix} -0.096085 + 0.445394i \\ -0.033166 + 0.487843i \\ -0.017363 + 0.493677i \\ 0.146840 + 0.536394i \end{pmatrix} \tag{6.18}$$



Obrázek 6.21: Experiment 4 varianta No-Cloning. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.



Obrázek 6.22: Experiment 4 varianta No-Cloning. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.

6.5 Experiment 5: Maximalizace

V posledním experimentu jde o maximalizaci prvku s nejvyšší amplitudou. Tedy například pro stavový vektor

$$\begin{pmatrix} 1/2 \\ 0 \\ 1/\sqrt{2} \\ 1/2 \end{pmatrix} \tag{6.19}$$

bude cílový stav

$$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{6.20}$$

Pro každý experiment byl opět prvně náhodně vytvořen stav, na nějž byly generované matice aplikovány. Experiment byl proveden pro registry velikosti 2 a 3 qubity. Při 2 qubitech byla ukončující podmínka na 5000 vyhodnocení fitness a práh úspěšnosti na 0,02. Experimenty se 3 qubity pak běželi po dobu 20000 vyhodnocení fitness a hranice úspěchu byla nastavena na 0,05.

6.5.1 Výsledky

Grafy na obrázcích 6.23 a 6.26 zobrazují konvergenční křivky běhů evolučních algoritmů. Na těch je v obou těchto experimentech vidět větší rozptyl konečných řešení. Následné obrázky 6.24 a 6.26 ukazují statistické vyhodnocení výsledků prostřednictvím krabicových grafů. Ke obrázku 6.26 je přiřazen i obrázek 6.25 s detailním zobrazením těchto grafů. I zde je vidět, že výsledky jsou hodně rozptýlené. Tyto úlohy se jsou pro evoluční systémy náročné, proto jsem zvýšil hranici úspěchu oproti předchozím experimentům.

2 qubity

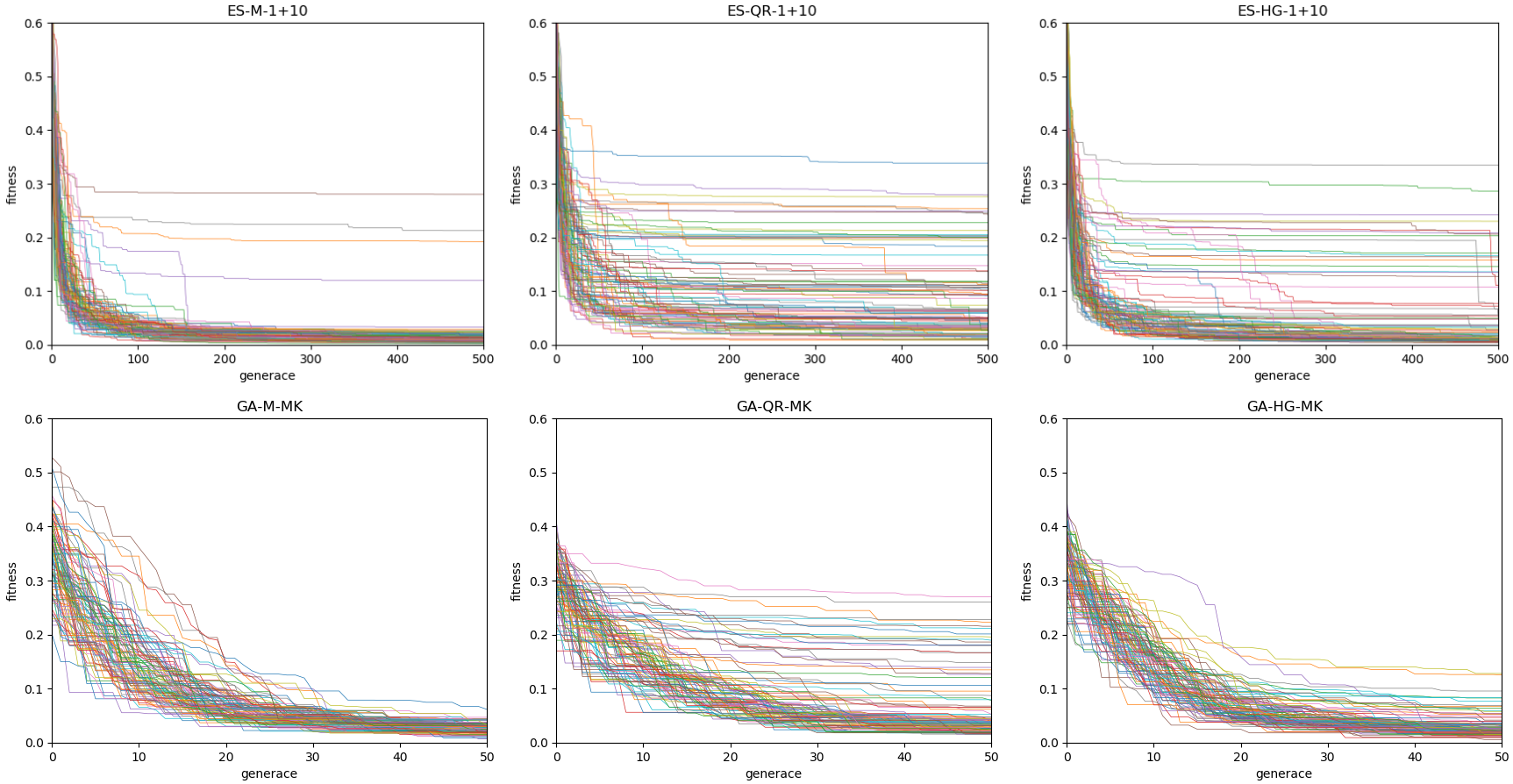
V tomto experimentu byly úspěšné převážně genotypy M a HG. Genotyp QR dosáhl pouze na nízké desítky procentuální úspěšnosti. Nejlepší hodnotu fitness měla matice U vygenerovaná algoritmem ES-M-1+10 zobrazená v rovnici 6.21. Její trénovací množina TS měla podobu 6.22.

$$U = \begin{pmatrix} 0.20392 + 0.321256i & 0.608343 + 0.169812i & -0.499061 - 0.415674i & 0.141233 + 0.120417i \\ -0.529204 + 0.271637i & -0.353321 - 0.133713i & -0.343073 - 0.078249i & 0.575002 - 0.221344i \\ -0.0950648 - 0.690061i & 0.0787123 + 0.53915i & -0.2909 + 0.0427976i & 0.111729 - 0.344909i \\ 0.0738961 - 0.103373i & 0.172477 - 0.362962i & 0.317582 - 0.516739i & -0.00192657 - 0.674154i \end{pmatrix} \tag{6.21}$$

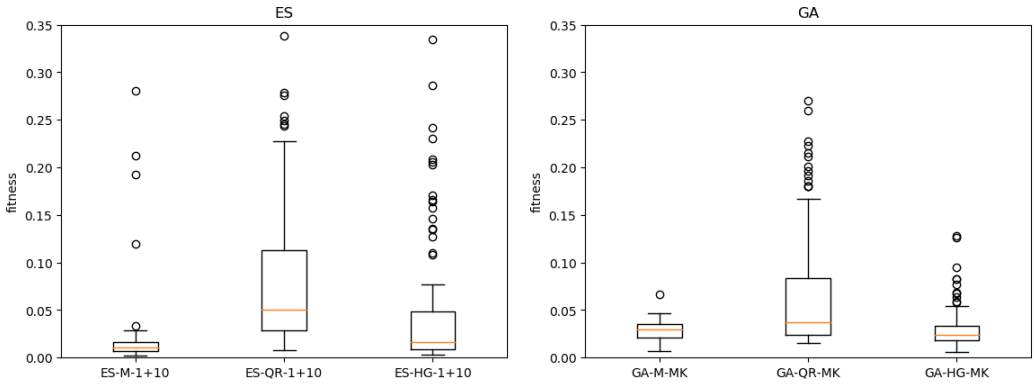
$$TS = \{(\begin{pmatrix} 0.0784175 + 0.110147i \\ 0.168799 + 0.357699i \\ 0.314552 + 0.518355i \\ 0.00159575 + 0.676491i \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix})\} \tag{6.22}$$

Aplikací matice na počáteční stav z trénovací množiny dostaneme stav

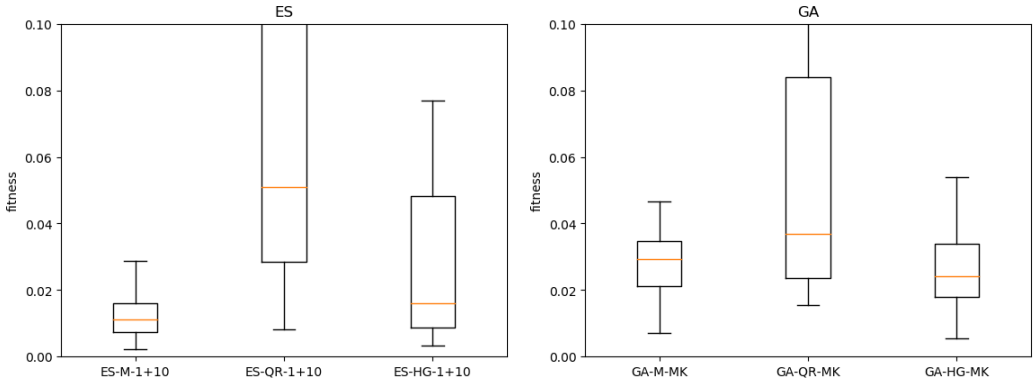
$$\begin{pmatrix} -0.000197 + 0.000214i \\ 0.000071 + 0.000241i \\ 0.008805 - 0.007714i \\ 0.999932 + 0.000160i \end{pmatrix}. \tag{6.23}$$



Obrázek 6.23: Experiment 5 pro 2 qubity. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.24: Experiment 5 pro 2 qubity. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.



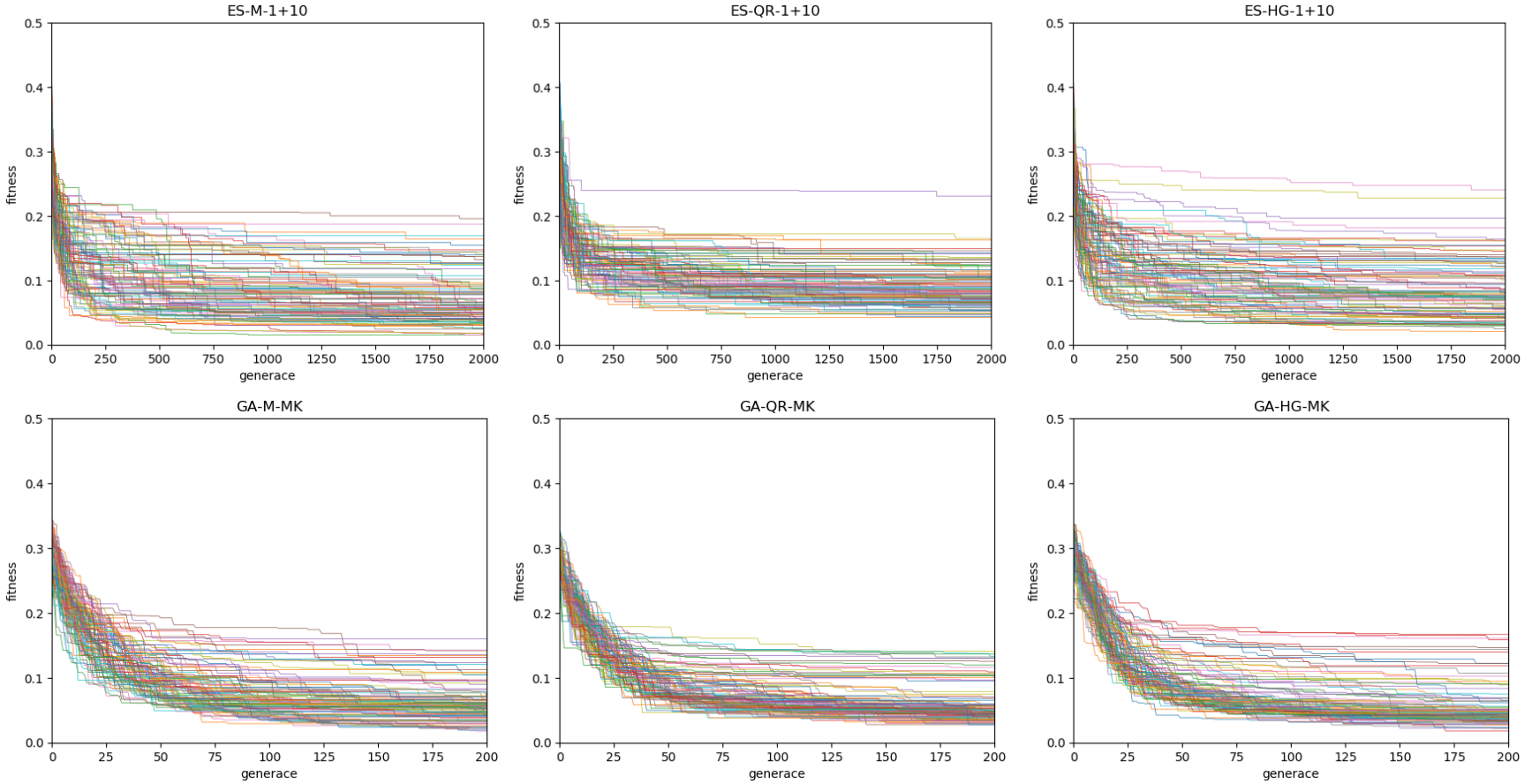
Obrázek 6.25: Experiment 5 pro 2 qubity. Statistické zhodnocení experimentů. Detial. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	86,5	15,6	58,3	21,9	14,6	36,5

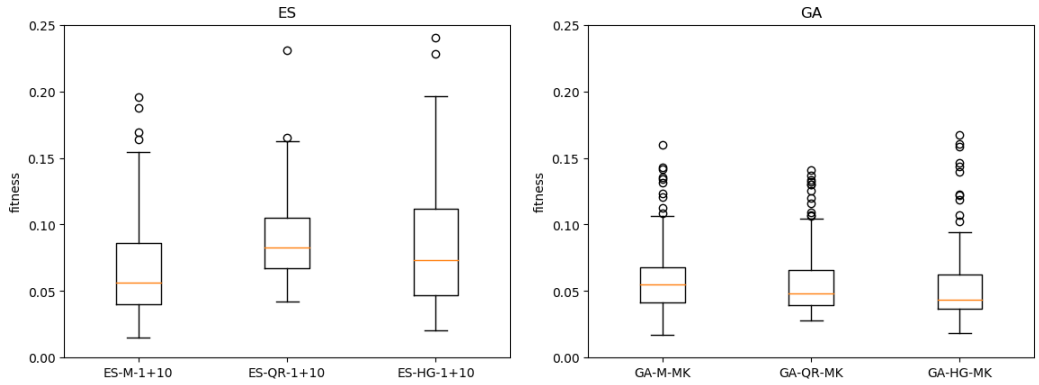
Tabulka 6.6: Experiment 5 pro 2 qubity. Procentuální úspěšnost nalezení řešení s fitness menší než 0,02.

3 qubity

V této variantě experimentu už opět genotyp QR dosahuje srovnatelných výsledků jako ostatní. Stále však není nejlepším z nich. V krabicových grafech na obrázku 6.27 je vidět že v tomto experimentu GA dosáhl lepších výsledků avšak s mnohem pomalejší konvergencí (obrázek 6.26).



Obrázek 6.26: Experiment 5 pro 3 qubity. Konvergenční křivky. V řádcích shora algoritmus ES, GA. Ve sloupcích zleva genotyp M, QR, HG.



Obrázek 6.27: Experiment 5 pro 3 qubity. Statistické zhodnocení experimentů. Zleva evoluční algoritmus ES, GA.

	ES-1+10			GA-MK		
	M	QR	HG	M	QR	HG
úspěšnost [%]	40,6	5,2	29,2	41,7	57,3	64,6

Tabulka 6.7: Experiment 5 pro 3 qubity. Procentuální úspěšnost nalezení řešení s fitness menší než 0,05.

6.6 Shrnutí experimentů

V práci bylo provedeno několik experimentů. Některé pro pevný počet qubitů, jiné ve více variantách pro různé počty qubitů. Z celkového pohledu proběhly experimenty úspěšně. Vždy se povedlo alespoň jednou konfigurací získat řešení s fitness funkcí blízko nule. Nově navržený genotyp QR si vedl nejlépe v prvním experimentu, kde přinesl velmi přesné výsledky a ve třetím experimentu - provázání. V dalších experimentech byl srovnatelný a někdy i horší než ostatní genotypy. Při porovnání evoluční strategie a genetického algoritmu vycházela většinou lépe evoluční strategie. V případě, kdy tomu tak nebylo by šla ES vylepšit pokročilejšími technikami adaptace a samoadapatace parametru σ. Pak by lépe konvergovali i v pozdějších fázích evoluce. Konvergenční křivka ES byla vždy strmější, než u GA. Z těchto důvodů shledávám použití ES nejen za vhodné, ale i výhodné oproti GA. Tímto se potvrdila i hypotéza z kapitoly 5.1.

Kapitola 7

Závěr

V práci byl implementován evoluční systém pro návrh kvantových operátorů. Systém úspěšně a v rozumném čase generuje operátory pro jeden a dva qubity. Pro více qubitů jsou výsledky méně přesné a časově náročnější.

V systému lze použít genetický algoritmus nebo evoluční strategii. Dále byla představena nová reprezentace genotypu založená na QR dekompozici. V systému lze zvolit mezi genotypem použitým v práci Hutsella a Greenwooda, genotypem vymyšleným MacKinnonem a nově představeným genotypem. Všechny tyto reprezentace i oba typy evolučních algoritmů byly v práci porovnány. Výsledky ukazují, že evoluční strategie mají v této úloze lepší vlastnosti než genetické algoritmy, stejně jako nově představený genotyp dává v některých případech lepší výsledky než zbylé dva.

Práci lze rozšířit o adaptaci nebo samoadaptaci parametru σ v evoluční strategii. Dále by se dalo vylepšit oba dva převzaté genotypy o myšlenky použité v práci MacKinnona. Poté by mohly tyto reprezentace předčít nový genotyp. Nakonec by se dalo podle vzoru Gregora rozšířit systém na evoluci více unitárních matic oddělených hradlem orákula nebo kvantovým měřením. Poté by šlo systém aplikovat třeba na groverův algoritmus.

Tato práce nepřináší nové algoritmy ani vylepšení těch stávajících. Její význam je v ukázání nových možností a jejich porovnání se stávajícími. Vůbec poprvé je v práci použita evoluční strategie pro generování unitárních matic. Dále je zde představen nový způsob zakódování genotypu a jeho převod na fenotyp.

Literatura

[1] ABRAHAM, H., AKHALWAYA, I. Y., ALEKSANDROWICZ, G., ALEXANDER, T., ALEXANDROWICS, G. et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2019.

[2] BANG, J. a YOO, S. A genetic-algorithm-based method to find unitary transformations for any desired quantum computation and application to a one-bit oracle decision problem. *Journal of the Korean Physical Society*. 2014, roč. 65, s. 2001–2008.

[3] BENNETT, C. H., BERNSTEIN, E., BRASSARD, G. a VAZIRANI, U. Strengths and Weaknesses of Quantum Computing. *Society for Industrial and Applied Mathematics. Journal on Computing*. [online]. 1997, roč. 26, s. 1510–1523. Dostupné z: https://arxiv.org/pdf/quant-ph/9701001.

[4] BERNSTEIN, E. a VAZIRANI, U. Quantum Complexity Theory. *Society for Industrial and Applied Mathematics. Journal on Computing*. 1997, roč. 26, s. 1411–1473.

[5] BIDLO, M. *Evoluční návrh řadicího algoritmu*. Brno, CZ, 2004. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/2031/.

[6] BRABAZON, A., O'NEILL, M. a MCGARRAGHY, S. *Natural Computing Algorithms*. Springer, Berlin, Heidelberg. ISBN 978-3-662-43631-8.

[7] DEUTSCH, D. a PENROSE, R. *Quantum theory, the Church–Turing principle and the universal quantum computer*. The Royal Society of London, 1997, s. 97–117. Series A, sv. 400.

[8] EIBEN, A. E. a SMITH, J. E. *Introduction to Evolutionary Computing*. Second edition. Springer-Verlag, 2015. ISBN 978-3-662-44873-1.

[9] FARHI, E., GOLDSTONE, J. a GUTMANN, S. *A Quantum Approximate Optimization Algorithm* [online]. 2014. Dostupné z: https://arxiv.org/abs/1411.4028.

[10] FEYNMAN, R. P. Simulating physics with computers. *International Journal of Theoretical Physics*. 1982, roč. 21, č. 6, s. 467–488. ISSN 1572-9575.

[11] FOUNDATION, Q. O. S. *Open-Source Quantum Software Projects* [online]. 2020. Dostupné z: https://github.com/qosf/awesome-quantum-software.

[12] GANDER, W. *Algorithms for the QR-Decomposition: Research report*. 1980.

[13] GEPP, A. a STOCKS, P. A review of procedures to evolve quantum algorithms. *Genetic Programming and Evolvable Machines*. Červen 2009, roč. 10, s. 181–228.

[14] GOLUB, G. H. a VAN LOAN, C. F. *Matrix Computations*. 3rd Edition. Johns Hopkins University Press, 1996. ISBN 0801854148.

[15] GOOGLE. *Cirq* [online]. 2019. Dostupné z: https://github.com/quantumlib/Cirq.

[16] GREGOR, C. A. G. *Construction of Unitary Matrices and Bounding Minimal Quantum Gate Fidelity Using Genetic Algorithms*. Guelph, Ontario, Canada, 2018. Diplomová práce. The University of Guelph.

[17] GROVER, L. K. A Fast Quantum Mechanical Algorithm for Database Search. *Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*. New York, NY, USA: Association for Computing Machinery. 1996, s. 212–219.

[18] GRUSKA, J. *Quantum Computing*. McGraw-Hill Book Co Ltd, 2000. Advanced topics in computer science series. ISBN 978-0077095031.

[19] HIOE, F. T. a EBERLY, J. H. N-Level Coherence Vector and Higher Conservation Laws in Quantum Optics and Quantum Mechanics. *Phys. Rev. Lett.* American Physical Society. Sep 1981, roč. 47, s. 838–841. Dostupné z: https://link.aps.org/doi/10.1103/PhysRevLett.47.838.

[20] HIRVENSALO, M. *Quantum Computing*. 2. vyd. Springer-Verlag Berlin Heidelberg, 2004. Natural Computing Series. ISBN 978-3-642-07383-0.

[21] HOLLAND, J. H. *Adaptation in natural and artificial systems*. Unibersity of Michigen Press, 1975.

[22] HUTSELL, S. a GREENWOOD, G. Applying evolutionary techniques to quantum computing problems. *2007 IEEE Congress on Evolutionary Computation, CEC 2007*. Říjen 2007, s. 4081 – 4085.

[23] IBM. *IBM Quantum Experience* [online]. 2019. Dostupné z: https://quantum-computing.ibm.com/.

[24] JONES, H. F. *Groups, Representations and Physics*. 1. vyd. CRC Press, 1998. 340 s. ISBN 9780367805791.

[25] JONES, T., BROWN, A., BUSH, I. a BENJAMIN, S. *QuEST: Quantum Exact Simulation Toolkit* [online]. 2018. Dostupné z: https://quest.qtechtheory.org/.

[26] JONES, T., BROWN, A., BUSH, I. a BENJAMIN, S. QuEST and High Performance Simulation of Quantum Computers,. *Scientific Reports*. 2019, s. 2045–2322. Dostupné z: https://doi.org/10.1038/s41598-019-47174-9.

[27] KRAWEC, W. An algorithm for evolving multiple quantum operators for arbitrary quantum computational problems. *Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO Comp '14)*. 2014, s. 59–60.

[28] KUPČA, V. *Teorie a perspektiva kvantových počítačů* [online]. 2001. Dostupné z: http://www.karlin.mff.cuni.cz/~holub/soubory/qc/qc.html.

[29] KVASNIČKA, V., POSPÍCHAL, J. a TIŇO, P. *Evolučné algoritmy*. Slovenská technická univerzita. Edícia vysokoškolských učebnic. ISBN 80-227-1377-5.

[30] LEIER, A. *Evolution of Quantum Algorithms using Genetic Programming*. Dortmund, 2004. Disertační práce. DisUniversity of Dortmund.

[31] LUKAC, M. a PERKOWSKI, M. Evolving quantum circuits using genetic algorithm. 2002, s. 177–185.

[32] MACKINNON, D. *Evolving Quantum Algorithm with Genetic Programming*. Guelph, Ontario, Canada, 2017. Diplomová práce. The University of Guelph.

[33] MASSEY, P., CLARK, J. a STEPNEY, S. Evolving quantum circuits and programs through genetic programming. *Genetic and Evolutionary Computation Conference (GECCO 2004*. Springer. 2004, s. 569–580.

[34] MASSEY, P., CLARK, J. a STEPNEY, S. Evolution of a human-competitive Quantum Fourier Transform algorithm using genetic programming. *Genetic and Evolutionary Computation Conference (GECCO 2005*. ACM Press. 2005, s. 1657–1664.

[35] MASSEY, P. *Searching for Quantum Software*. York, 2006. Disertační práce. University of York.

[36] MCCASKEY, A. J., LYAKH, D. I., DUMITRESCU, E. F., POWERS, S. S. a HUMBLE, T. S. XACC: A System-Level Software Infrastructure for Heterogeneous Quantum-Classical Computing. *ArXiv e-prints*. Nov 2019, s. arXiv:1911.02452.

[37] NIELSEN, M. A. a CHUANG, I. L. *Quantum computation and quantum information*. New York: Cambridge University Press, 2000. ISBN 05-216-3503-9.

[38] PERUZZO, A., MCCLEAN, J., SHADBOLT, P., YUNG, M.-H., ZHOU, X.-Q. et al. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*. 2014, roč. 5, s. 4213. ISSN 2041-1723.

[39] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. a FLANNERY, B. P. *Numerical Recipes: The Art of Scientific Computing*. 3rd Edition. Cambridge University Press, 2007. ISBN 978-0521880688.

[40] QUANTASTICA. *Quantum Programming Studio* [online]. 2019. Dostupné z: https://quantum-circuit.com/.

[41] RECHENBERG, I. *Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment Library Translation*. Library Translation. Royal Aircraft Establishment, 1965.

[42] RIEFFEL, E. G. a POLAK, W. *An Introduction to Quantum Computing for Non-Physicists* [online]. 1998. Dostupné z: https://arxiv.org/abs/quant-ph/9809016.

[43] RUBINSTEIN, B. Evolving quantum circuits using genetic programming. *2001 IEEE Congress on Evolutionary Computation (CEC2001)*. 2001, s. 114–121.

[44] SCHALLER, R. R. Moore's law: past, present and future. *IEEE Spectrum*. June 1997, roč. 34, č. 6, s. 52–59. ISSN 1939-9340.

[45] SCHWEFEL, H.-P. *Kybernetische Evolution als Strategie der Exprimentellen Forschung in der Strömungstechnik*. Disertační práce.

[46] SERODIO, H. a SJODAHL, M. *Symmetries and Group Theory: Continuous Groups* [online]. 2019. Dostupné z: http://home.thep.lu.se/~malin/FYTN13/GTnotes2019.pdf.

[47] SHOR, P. W. Algorithms for quantum computation: discrete logarithms and factoring. Nov 1994, s. 124–134.

[48] SMITH, R. S., CURTIS, M. J. a ZENG, W. J. *A Practical Quantum Instruction Set Architecture*. 2016.

[49] SPECTOR, L. C. The evolution of arbitrary computational processes. *IEEE Intelligent Systems and their Applications*. IEEE. 2000, roč. 15, s. 80–83.

[50] SPECTOR, L. C., BARNUM, H. a BERNSTEIN, H. J. Genetic programming for quantum computers. *Genetic Programming 1998: Thurd Annual Conference*. 1998, s. 365–374.

[51] SPECTOR, L. C., BARNUM, H., BERNSTEIN, H. J. a SWAMY, N. Finding a better-than-classical quantum AND/OR algorithm using genetic programming. *1999 Congress on Evolutionary Computation*. IEEE. 1999, s. 239–246.

[52] SPECTOR, L. C., BARNUM, H., BERNSTEIN, H. J. a SWAMY, N. Quantum computing applications of genetic programming. *Genetic Programming: Volume 3*. The MIT Press. 1999, s. 135–160.

[53] SPECTOR, L. C. a BERNSTEIN, H. J. Communication capacities of some quantum gates, discovered in part through genetic programming. *6th International Conference on Quantum Communication, Measurement, and Computing (QCMC)*. Rinton Press. 2003, s. 500–503.

[54] SPECTOR, L. C. *Automatic quantum computer programming: a genetic programming approach*. Boston: Kluwer Academic Publishers, 2004. Genetic Programming Series. ISBN 14-020-7894-3.

[55] STADELHOFER, R. *Evolving Blackbox Quantum Algorithms using Genetic Programming*. Dortmund, 2006. Disertační práce. University of Dortmund.

[56] STADELHOFER, R., BANZHAF, W. a SUTER, D. Evolving blackbox quantum algorithms using genetic programming. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Cambridge University Press. 2008, roč. 22, č. 3, s. 285–297.

[57] T. NISHIMURO, M. M. a. *Mersenne Twister with improved initialization* [online]. 1997. Dostupné z: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html.

[58] WEIGERT, S. No-Cloning Theorem. *Compendium of Quantum Physics*. Springer. 2009, s. 404–405.

[59] WILLIAMS, C. a GRAY, A. Automated Design of Quantum Circuits. *Quantum Computing and Quantum Communications. QCQC 1998*. Springer, Berlin, Heidelberg. 1998, roč. 1509, s. 113–125.

[60] YANOFSKY, N. S. a MANNUCCI, M. A. *Quantum computing for computer scientists*. Cambridge: Cambridge University Press, 2008. ISBN 978-0-521-87996-5.

[61] ZYCZKOWSKI, K. a KUS, M. Random unitary matrices. *Journal of Physics A: Mathematical and General*. 1999, roč. 27, s. 4235.

Příloha A

Obsah přiloženého paměťového média

Tato příloha popisuje obsah paměťového média přiloženého k této diplomové práci. Po otevření média nalezeneme následující soubory a adresáře:

conf Adresář s konfiguračními soubory použitými v experimentech.

dip Spustitelný soubor s programem vytvořeným v této práci.

projekt.pdf PDF s technickou zprávou diplomové práce.

src Adresář se zdrojovými soubory.

- **QuEST** Adresář obsahující zdrojové soubory knihovny QuEST.
- **Makefile** Soubor pro kompilaci zdrojových kódu na spustitelný program.
- **src** Adresář se zdrojovými soubory programu vytvořeném v této práci.

tex_src Adresář se zdrojovými soubory pro vysázení technické zprávy v prostředí LaTeX.

Příloha B

Manuál k programu

Součástí této práce je aplikace implementovaná v jazyce C/C++ v prostředí operačního systému Linux. Tato příloha popisuje její překlad, spuštění a obsluhu.

B.1 Požadavky

GCC verze 9.0 nebo vyšší

CMake verze 3.16 nebo vyšší

B.2 Překlad

Překlad proběhne spuštěním příkazu

```
$ make
```

v adresáři `src`. Překladem vznikne spustitelný soubor `dip`.

B.3 Spuštění

Program lze spustit následujícím příkazem s jedním volitelným parametrem a jedním volitelným přepínačem.

```
$ ./dip [-x] [konfiguracni_soubor]
```

Přepínač `-x` nastaví výstup do souborů `.dat` a `.best`, kde jméno souboru je ID procesu. Do souboru `.dat` se vypíší nejlepší fitness z každé generace a do souboru `.best` se uloží výsledná matice. Bez přepínače `-x` se průběh experimentu vypisuje na standardní výstup.

Konfigurační soubor obsahuje veškerou specifikaci a konfiguraci programu. V případě nezadání souboru s konfigurací bude použit soubor jehož jméno je na prvním řádku souboru `params.conf`. V případě absence tohoto souboru je použita výchozí konfigurace programu. Důvodem je umožnění změny konfiguračního souboru bez nutnosti opětovné kompilace zdrojových kódů. Popis konfiguračního souboru je v sekci B.4.

B.4 Konfigurační soubor

Konfigurační soubor se skládá ze sekvence

```
<parametr>=<hodnota>
```

na samostatných řádcích. Prázdné řádky jsou ignorovány. Při použití stejného parametru vícekrát je použita poslední jemu přiřazená hodnota. Nevalidní parametry jsou ignorovány stejně jako validní parametry s nevalidní hodnotou. V parametrech i hodnotách je potřeba rozlišovat malá a velká písmena.

Následují tabulky obsahující výčet všech možných parametrů a k nim přípustných hodnot. Seznam je rozdělen na tabulku obecných parametrů B.1, tabulku parametrů specifických pro GA B.3 a tabulku parametrů specifických pro ES B.2. Hodnoty některých parametrů jsou navíc omezeny obecnou podmínkou[1]. Konfigurační soubory použité na experimenty jsou v adresáři `conf`.

[1] Bez ohledu na použití absolutního, nebo procentuálního počtu součet potomků, nové krve, elity a mutované elity musí být roven velikosti populace.

Tabulka B.1: Tabulka obecných parametrů a jejich hodnot

Parametr	**Přípustné hodnoty**	**Výchozí hodnota**	**Popis**
algorithm	GA, ES	GA	Použitý evoluční algoritmus. ES = evoluční strategie, GA = genetický algoritmus.
encoding	HG, M, QR	QR	Použité zakódování genotypu. HG = genotyp Hutsell a Greenwood, M = genotyp MacKinnon, QR = genotyp QR dekompozice
expriment	HADAMARD_1, CNOT_2, ENTANGLE_N, CLONING_2, NO_CLONING_2, MAX_1_N	HADAMARD_1	Použitý experiment. Číslo na konci popisuje nad kolika qubity je možné experiment spustit. N znamená libovolně mnoho.
qubits	$\mathbb{Z}_{>0}$	1	Počet qubitů udává velikost kvantových registrů i hledané matice. Když se počet qubitů neshoduje s číslem uvedeným v experimentu, program skončí chybou.
populationSize	$\mathbb{Z}_{>0}$	100	Velikost populace v evolučním algoritmu.
maxGenerations	$\mathbb{Z}_{>0}$	1000	Ukončující podmínka maximálního počtu generací v evolučním algoritmu.
bestInRowLimit	$\mathbb{Z}_{>0}$	1000	Ukončující podmínka maximálního počtu generací v řadě, ve kterých se hodnota nejlepší fitness nezměnila.
approximation	$\mathbb{R}_{\geq 0}$	0,01	Ukončující podmínka při hodnotě nejlepší fitness rovné nebo menší než je tato hodnota.

Tabulka B.2: Tabulku parametrů specifických pro GA.

Parametr	**Přípustné hodnoty**	**Výchozí hodnota**	**Popis**
crossoverProbability	$\mathbb{R}_{\geq 0}$	0	Pravděpodobnost aplikace genetického operátoru křížení.
mutationProbability	$\mathbb{R}_{\geq 0}$	0	Pravděpodobnost aplikace genetického operátoru mutace
mutationScale	$\mathbb{R}_{>0}$	1,0	Parametr mutace σ.
selectionMethod	TOURNAMENT, ROULETTE, RANK	TOURNAMENT	Operátor selekce použitý v GA
childPercentage	$\mathbb{R}_{\geq 0}$[I]	0	Počet potomků vygenerovaných v každé generaci daný procentuálním poměrem k velikosti populace.
newBloodPercentage	$\mathbb{R}_{\geq 0}$[I]	0	Počet jedinců nové krve vygenerovaných v každé generaci daný procentuálním poměrem k velikosti populace.
elitePercentage	$\mathbb{R}_{\geq 0}$[I]	0	Počet elitních jedinců v každé generaci, kteří přežijí do následující generace nezměněni, daný procentuálním poměrem k velikosti populace.
mutatedElitePercentage	$\mathbb{R}_{\geq 0}$[I]	0	Počet elitních jedinců v každé generaci, kteří pouze zmutují a přežijí do následující generace, daný procentuálním poměrem k velikosti populace.
childrenCount	$\mathbb{Z}_{\geq 0}$[I]	0	Absolutní počet potomků vygenerovaných v každé generaci. Použití tohoto parametru přepisuje parametr *childPercentage*.
newBloodCount	$\mathbb{Z}_{\geq 0}$[I]	0	Absolutní počet jedinců nové krve vygenerovaných v každé generaci. Použití tohoto parametru přepisuje parametr *newBloodPercentage*.
eliteCount	$\mathbb{Z}_{\geq 0}$[I]	0	Absolutní počet elitních jedinců v každé generaci, kteří přežijí do následující generace nezměněni. Použití tohoto parametru přepisuje parametr *elitePercentage*.
mutatedEliteCount	$\mathbb{Z}_{\geq 0}$[I]	0	Absolutní počet elitních jedinců v každé generaci, kteří pouze zmutují a přežijí do následující generace. Použití tohoto parametru přepisuje parametr *mutatedElitePercentage*.

Tabulka B.3: Tabulku parametrů specifických pro GA.

Parametr	**Přípustné hodnoty**	**Výchozí hodnota**	**Popis**
strategy	COMMA, PLUS	PLUS	Použitá evoluční strategie. COMMA = strategie (μ, λ), PLUS = strategie $(\mu + \lambda)$.
mutationProbability	$\mathbb{R}_{\geq 0}$	0	Pravděpodobnost aplikace genetického operátoru mutace
mutationScale	$\mathbb{R}_{>0}$	1,0	Parametr mutace σ.
childPercentage	$\mathbb{R}_{\geq 0}$	0	Počet potomků vygenerovaných v každé generaci daný procentuálním poměrem k velikosti populace.
childrenCount	$\mathbb{Z}_{\geq 0}$	0	Absolutní počet potomků vygenerovaných v každé generaci. U ES jde o parametr λ. Použití tohoto parametru přepisuje parametr *childPercentage*.