České vysoké učení technické v Praze
Fakulta elektrotechnická

Diplomová práce

Použití genetických algoritmů pro syntézu celulárních automatů pro testování logických obvodů

Petr Hašlar

Vedoucí práce: Ing. Petr Fišer

Studijní program: Elektrotechnika a informatika dobíhající magisterský

Obor: Informatika a výpočetní technika

květen 2007

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu diplomové práce, Ing. Petru Fišerovi, a svým rodičům, bez jejichž vydatné podpory by tato práce nikdy nevznikla.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 23.5. 2007 .....

Abstract

This thesis evaluates genetic algorithms with regards to their possible application to the problem of cellular automaton synthesis, where the resulting automaton should be able to act as a test pattern generator for combinational logic circuits. The thesis proposes a binary genetic algorithm suitable for such a task and evaluates its implementation in comparison with a fully randomized approach for solving the given problem.
The thesis also contains a brief introduction to commonly used methods of logic circuit testing with the focus on the usage of cellular automata as pseudorandom test pattern generators within built-in self-test devices.
Keywords: genetic algorithms, built-in self-test device, test pattern generator, cellular automaton

Anotace

Tato práce se zabývá použitím genetických algoritmů pro řešení problému syntézy celulárních automatů vhodných k testování kombinačních logických obvodů. Práce předkládá návrh vhodné varianty binárního genetického algoritmu a výsledná implementace takového algoritmu je pak experimentálně vyhodnocena srovnáním s plně randomizovaným přístupem k řešení zadaného problému.
Práce dále obsahuje stručný přehled problematiky testování logických obvodů a možností využití celulárních automatů v rámci prostředků vestavěné diagnostiky jako generátorů posloupností pseudonáhodných testovacích vektorů.
Klíčová slova: genetické algoritmy, vestavěná diagnostika, generátor testovacích vektorů, celulární automat

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Jedním z typických rysů dnešní doby je stále rostoucí tempo rozvoje elektroniky a elektronických zařízení obecně. Tento rozvoj se projevuje především rychle se zvyšující mírou nasazení elektroniky takřka ve všech oborech lidské činnosti a prudce rostoucí komplexností používaných zařízení. S tím také souvisí rostoucí potřeba efektivního způsobu testování, které by umožnilo včasné rozpoznání porouchaných zařízení a ve svém konečném důsledku i minimalizaci ekonomických dopadů a bezpečnostních rizik spojených s výrobou a používáním takovýchto zařízení. Obecně se používá testování během různých fází výrobního procesu a nezřídkakdy i za samotného provozu finálního výrobku. K testování se typicky využívá tzv. *vnějšího testovacího zařízení* (angl. Automated Test Equipment, dále jen *ATE*), jehož princip činnosti spočívá v *buzení testovaného zařízení* předem stanoveným způsobem a v následném vyhodnocení takto získaných *odezev*.

V praxi se však ukazuje, že čím dál tím větší část nákladů vynaložených na vývoj a výrobu spolkne samotné testování. To je z velké části způsobeno tím, že trendy vývoje ATE do značné míry kopírují trendy ve vývoji jimi testovaných zařízení, především co do růstu výkonu a komplexnosti. Čím složitější jsou testovaná zařízení, tím vyšší nároky jsou kladeny na ATE z hlediska rychlosti komunikace s testovaným zařízením a doby vyhodnocování získaných odezev. Vyšší složitost testovaných zařízení se také negativně projevuje na obtížnosti sestavení testů samotných a jejich celkové délce.

Částečné východisko z této situace nabízí přizpůsobení návrhu zařízení potřebám testování. Tento přístup k návrhu se nazývá *návrh pro snadnou diagnostiku* (angl. Design for Testability, dále jen *DFT*) a jeho cílem je zjednodušení sestavování testů a jejich následné aplikace [1].

Samotný přístup DFT se skládá z široké palety vzájemně se doplňujících technik, se kterými volně souvisí jeden prvek návrhu zařízení, který nás bude v dalším textu zajímat, a tím je tzv. *vestavěná diagnostika* (angl. Built-in Self-test, dále jen *BIST*).

Tato technika ve zkratce spočívá v zabudování podstatné části testovacího zařízení dovnitř obvodu navrhovaného zařízení. Těžištěm tohoto textu pak bude prozkoumání možností návrhu jedné z částí takto zabudovávaného testovacího zařízení, a to *generátoru stimulů* pro buzení navrhovaného zařízení.

2 Záměr práce

Jak jsem již v úvodu naznačil, důležitou částí BIST obvodů je generátor stimulů testovaného obvodu. Výstup generátoru, nazývaný též *vstupní* nebo *testovací vektor*, tvoří spolu s očekávanou odezvou testovaného obvodu ve formě *výstupního vektoru* tzv. *krok testu*. Posloupnost takovýchto kroků pak tvoří samotný *test* [1].
Cílem této práce je prozkoumat možnosti využití genetických algoritmů pro syntézu celulárních automatů použitelných jako generátory testovacích vektorů pro kombinační logické obvody.
Genetické algoritmy budeme v rámci práce používat k evoluci optimální struktury vhodného celulárního automatu. Ta může být popsána formou *pravidel*, kterými se jednotlivé buňky automatu řídí, nebo jeho *charakteristickou maticí*. Takový popis struktury bude navíc doplněn o počáteční nastavení buněk, tzv. *seed*, který bude také předmětem evoluce (pro definici výše zmíněných pojmů viz kap. 3.1.3).
Použitelnost celulárního automatu určité struktury jako generátoru testovacích vektorů bude vyhodnocována především prostřednictvím simulátoru poruch *fsim* [2].
Vhodnost různých implementací genetických algoritmů k tomuto účelu bude posuzována experimentálně, porovnáním s plně randomizovaným přístupem. Plně randomizovaným přístupem se v textu rozumí zcela náhodné generování celulárních automatů, při kterém se vždy udržuje nejlepší takto nalezený automat.
Samotný text je rozčleněn do celem sedmi kapitol, pricemž první dvě tvoří samotný úvod a záměr práce.
Další kapitoly jsou koncipovány takto:

- Třetí kapitola pokrývá teoretický základ práce. Jsou v ní postupně probrány existující metody pro testování logických obvodů, teorie celulárních automatů, obecné principy fungování genetických algoritmů a jejich různé varianty.
- Čtvrtá kapitola se zabývá návrhem genetického algoritmu pro zadaný problém syntézy optimálního celulárního automatu.
- Pátá kapitola pak popisuje samotnou implementaci tohoto algoritmu s důrazem na její nestandardní části.
- V šesté kapitole jsou uvedeny výsledky měření finálního programu. Cílem měření je zjistit vliv různých parametrů implementovaného algoritmu na kvalitu výsledného celulárního automatu, tj. jeho vhodnosti jako generátoru testovacích vektorů.
- Naměřené výsledky a přínos práce shrnuje závěrečná, sedmá kapitola.

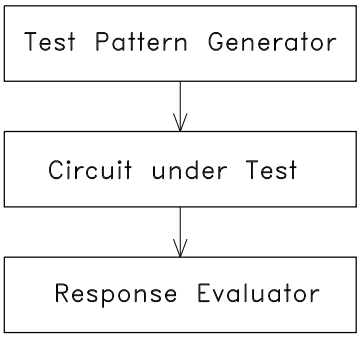
3 Vhled do problematiky

3.1 Testování logických obvodů

Na začátek bych rád předeslal, že tento text si v žádném případě neklade za cíl čtenáře dopodrobna seznámit s problematikou testování logických obvodů, která je navíc vyčerpávajícím způsobem popsána například v [1] a v řadě dalších odborných publikací, ale jeho hlavním cílem je úvod do této problematiky v rozsahu nezbytně nutném pro účely této práce jako takové.

3.1.1 Prostředky vestavěné diagnostiky

Jak již bylo v úvodu práce zmíněno, v dnešní době, tj. k první polovině roku 2007, se běžně vyrábějí zařízení obsahující logické obvody s řádově desítkami až stovkami miliónů transistorů. Taková zařízení jsou standardními prostředky ATE testovatelná jen velice obtížně, a to vždy za cenu vysokých nákladů na testování v podobě velkého množství času a komplikovaného testovacího zařízení, které je k tomu účelu potřeba.
Výrobci takových zařízení se tento problém snaží řešit začleněním prostředků vestavěné diagnostiky (BIST) přímo do návrhu zařízení samotných. Takto navržená zařízení pak k vlastnímu testování zpravidla nevyžadují součinnost s ATE, a pokud tomu tak přeci jen je, bývají nároky na ATE výrazně nižší, než je tomu v případech, kdy zařízení BIST část zcela postrádají.
Bylo vypracováno mnoho metodik pro návrh BIST, z nichž většina se spoléhá na *generátor pseudonáhodných vektorů* (angl. Pseudo-random Pattern Generator, dále jen *PRPG*) pro testování lehce detekovatelných poruch, které často představují zhruba 90% z celkového počtu všech možných poruch testovaného obvodu [3]. Zbývající poruchy jsou pak testovány deterministicky sestavenou posloupností testovacích vektorů, kterou generuje buď samotný BIST, nebo externí ATE.
Jako generátoru pseudonáhodných vektorů se ponejvíce využívá *lineárního zpětnovazebního posuvného registru* (angl. Linear Feedback Shift Register, dále jen *LFSR*) a *celulárního automatu* (angl. Cellular Automaton, dále jen *CA*). Tyto obvody jsou používány především z důvodů jejich jednoduché struktury, s tím souvisejících nízkých prostorových nároků a dobrého *diagnostického pokrytí*, které vyjadřuje percentuální podíl poruch detekovaných testovací posloupností získanou z takového generátoru ku celkovému počtu detekovatelných poruch testovaného obvodu [1].
Schéma základní struktury BIST je uvedeno na obrázku 3.1. Samotný BIST se skládá z *generátoru testovacích vektorů* (angl. Test Pattern Generator, dále jen *TPG*), který produkuje testovací vektory postupně přiváděné na vstupy *testovaného obvodu* (angl. Circuit Under Test, dále jen *CUT*) a *příznakový analyzátor* (angl. Response Evaluator, dále jen *RE*), který má za úkol vyhodnotit odezvu testovaného obvodu na takto získané testovací vektory [3]. Z výše uvedeného je zřejmé, že TPG je realizován pomocí PRPG, který může být dále upraven za účelem zlepšení jeho vlastností.
Design TPG hraje v návrhu BIST klíčovou roli z hlediska jeho výsledných vlastností, především pak dosahovaného diagnostického pokrytí a prostorových nároků části obvodu implementující samotný generátor. U strukturně jednoduchých TPG je zpravidla dosti obtížné dosáhnout dostatečného pokrytí, a tak je ve většině případů výhodné základní strukturu generátoru modifikovat způsobem, o kterém se zmíníme dále v textu.
Typický TPG je ve své podstatě jednoduchý sekvenční logický obvod, který generuje slova cyklického kódu definovaného *generujícím polynomem*. Tato jsou přiváděna na vstup testovaného obvodu buď přímo, nebo po modifikaci kombinační logikou. V takovém případě pak z pochopitelných důvodů při návrhu usilujeme o to, aby byla tato přidaná kombinační logika co možná nejjednodušší, a tedy co nejméně náročná z hlediska plochy potřebné k její realizaci.



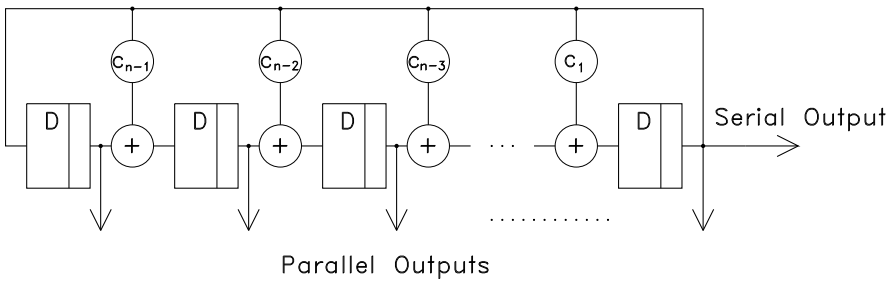
Obrázek 3.1: Obecné schéma BIST

Na výše popsaném principu doplnění TPG kombinační logikou je postaven tzv. *mixed-mode BIST* [3]. Takto navržený BIST pracuje ve dvou fázích:

1. Pseudonáhodné testovací vektory jsou z TPG přiváděny přímo na vstup testovaného obvodu, aby tak testovaly co nejvíce lehce detekovatelných poruch.
2. Vektory z TPG jsou modifikovány přidanou kombinační logikou tak, aby testovaly poruchy nepokryté v první fázi. Někdy je za tímto účelem namísto kombinační logiky použita přímo tabulka předgenerovaných vektorů uložená buď v paměti BIST nebo ATE. Takové řešení je výhodné pouze v případě, kdy je vektorů potřebných ve druhé fázi relativně malý počet.

3.1.2 Lineární zpětnovazební posuvné registry

Jedna varianta struktury lineárního zpětnovazbního posuvného registru, LFSR-I, je uvedena na obrázku 3.2.



Obrázek 3.2: Základní struktura LFSR

LFSR-I se skládá z klopných obvodů typu D, které slouží k uchování stavu celého registru, a z interních logických hradel typu XOR tvořících zpětnou vazbu [3]. Počáteční stav klopných obvodů se nazývá *seed* a jednoznačně určuje podobu posloupnosti stavů, kterými registr dané struktury postupně prochází.
Paralelní výstupy registru odpovídají přímo výstupům jeho klopných obvodů a odebírají se z nich testovací vektory.
Jako *sériový výstup* je vyveden výstup posledního klopného obvodu. Sériové vstupy a výstupy se při testování často používají z důvodu velice omezeného počtu vývodů na čipu použitelných k tomuto účelu a umožňují relativně snadnou inicializaci registrů a jejich pozdější čtení. Navíc se tyto vstupy a výstupy dají použít pro propojení více registrů do jednoho dlouhého řetězce, což má za následek další úsporu použitých vývodů.

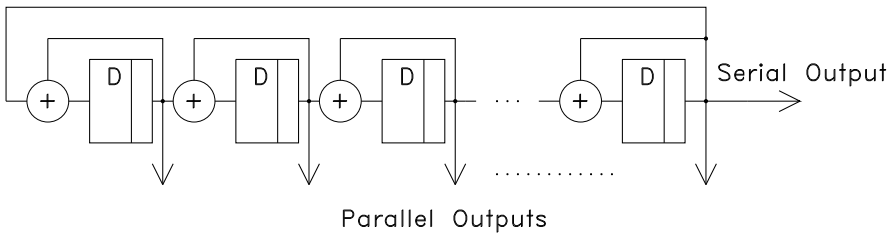
Sekvence kódových slov produkovaných takovým registrem je dána *generujícím polynomem* nad Galoisovým polem $GF(2^n)$

$$g(x) = x^n + c_{n-1}x^{n-1} + ... + c_1x^1 + 1, \tag{3.1}$$

kde koeficienty c_1 až c_{n-1} vyjadřují, jestli je do vstupu odpovídajícího klopného obvodu přivedena zpětná vazba a n je délka registru [3].
Pokud je navíc takový polynom *primitivní*, LFSR jím popsané struktury má maximální periodu $2^n - 1$, tj. je schopen generovat celkem $2^n - 1$ různých testovacích vektorů [3].
Druhá varianta registru, LFSR-II, se od toho prvního liší ve způsobu uzavření zpětné vazby. V tomto případě nejsou hradla XOR přidružena k jednotlivým klopným obvodům, ale jsou soustředěna u vstupu prvního klopného obvodu. Můžeme je pak nahradit jediným hradlem o odpovídajícím počtu vstupů, které budou buzeny výstupy klopných obvodů tak, jak to popisuje generující polynom.

3.1.3 Celulární automaty

Celulární automaty (CA) používané jako generátory testovacích vektorů mají obdobnou strukturu jako LFSR, ale používají zcela jinak utvořenou zpětnou vazbu, která je v tomto případě silně lokálního charakteru (viz obr. 3.3) [3].



Obrázek 3.3: Struktura homogenního CA používajícího pravidlo 90

Tyto automaty jsou ve své podstatě matematické idealizace přírodních systémů, které se vyznačují tou vlastností, že jejich komplexita je výsledkem dlouhodobě opakované aplikace jednoduchých pravidel. Původním účelem CA bylo modelování takových systémů a biologických procesů v nich probíhajících [4].
Typický CA se skládá z konečného počtu *buňek*, které mohou nabývat konečného počtu hodnot, resp. *stavů*. Funkce, která popisuje stav jednotlivých buňek v závislosti na čase, je definována jako

$$S : CA \times T \to \Sigma, \tag{3.2}$$

kde $CA = C_i$ je množina všech buňek, T je množina diskrétních časových kroků a Σ je konečná abeceda symbolů reprezentujících stavy buňek [4].
Dá se říci, že každá buňka automatu představuje určitou část modelovaného systému, ať už se jedná o část fyzickou nebo čistě logickou. Žádná buňka v automatu přitom neexistuje sama o sobě a její stav závisí převážně na stavu buňek v jejím *okolí*. Toto je popsáno funkcí

$$N : CA \to CA^n, \tag{3.3}$$

kde n je počet buňek. Velikost takového okolí, tj. počet sousedních buňek, je zpravidla pro každou buňku stejná. Budeme ji značit jako $k = |N(C_i)|$ [4].

Vývoj buňek automatu v diskrétním čase je určen pevně danou sadou *pravidel* popisujících chování jednotlivých buňek. Stav každé buňky v následujícím časovém kroce je jednoznačně určen jí přiřazeným pravidlem na základě stavu buňek v jejím okolí. Pravidla mají obecně tvar

$$R : CA \times \Sigma^k \rightarrow \Sigma, \tag{3.4}$$

kde k je už dříve zmíněná velikost okolí buňek [4].
CA lze dělit podle dvou základních kritérií [4]:

- podle *počtu unikátních pravidel* na *automaty hybridní* a *uniformní*. Pokud je automat uniformní, všechny jeho buňky se řídí stejným pravidlem.
- podle *typu pravidel* na *automaty lineární* a *nelineární*. Lineární automaty obsahují pouze lineární pravidla, tj. pravidla, která je možná popsat lineární kombinací dílčích pravidel.

Automaty, které jsou jak hybridní tak lineární, jsou zpravidla v literatuře označovány jako LHCA [4].
Pro účely testování se nejčastěji používají jednodimenzionální LHCA automaty s cyklickými hranicemi, kde levý soused první buňky je poslední buňka a naopak. Samotné automaty jsou pak implementovány logickými obvody složenými, obdobně jako LFSR, z klopných obvodů typu D a logických hradel typu XOR.
Strukturu CA, jehož buňky mohou nabývat pouze dvou logických hodnot, můžeme obecně popsat jedním ze dvou následujících způsobů:

- *výčtem pravidel*, kterými se jednotlivé buňky CA řídí.

  Pravidla popisující chování buněk jsou ve své podstatě boolské funkce okolí popisovaných buněk, které je možné vyjádřit ve formě pravdivostní tabulky s 2^k řádky, kde k je velikost okolí těchto buněk. Každý řádek takové tabulky tedy přiřazuje popisované buňce nový stav v závislosti na stavu buněk v jejím okolí.

  Pokud se na poslední sloupec tabulky, který představuje samotný výstup pravidla pro různé stavy okolí popisované buňky, budeme dívat jako na číslo zapsané v binárním kódu, můžeme pak pravidlo vyjádřené touto tabulkou reprezentovat oním číslem převedeným do dekadické soustavy (viz tab. 3.1, která obsahuje linearizované pravdivostní tabulky některých pravidel).

  To nám pak umožní se na pravidlo jednoduchým způsobem odkazovat, viz např. pravidlo 90, které nastaví buňku do logické jedničky právě tehdy, když má tuto hodnotu pouze jeden z jejích sousedů ($S(C_{i-1}) \oplus S(C_{i+1})$ pro buňku C_i).

Smysl pravidla	Stav okolí								Číslo pravidla
	111	110	101	100	011	010	001	000	
$S(C_{i-1})$	1	1	1	1	0	0	0	0	240
$S(C_{i-1}) \oplus S(C_i)$	0	0	1	1	1	1	0	0	60
$S(C_{i-1}) \oplus S(C_{i+1})$	0	1	0	1	1	0	1	0	90
$S(C_{i-1}) \oplus S(C_i) \oplus S(C_{i+1})$	1	0	0	1	0	1	1	0	150

Tabulka 3.1: Příklad pravidel CA

V případě popisu CA výčtem pravidel chování jednotlivých buňek automatu lze získat nový stav libovolné buňky jako funkční hodnotu pravidla pro aktuální stav buněk v okolí uvažované buňky.

- *charakteristickou maticí CA.*

  Tato matice je čtvercového tvaru, kde počet řádků i sloupců odpovídá celkovému počtu buňek CA. Každý její řádek popisuje okolí jedné z buňek a hodnoty jeho prvků udávají přítomnost vazeb mezi uvažovanou buňkou a buňkami odpovídajícími jednotlivým prvkům řádku. Jednotková hodnota prvku znamená, že tato vazba existuje a pro nulovou hodnotu platí opačné tvrzení.

  Příklad takové matice pro CA tvořený šesti buňkami se strukturou odpovídající obrázku 3.3 představuje obrázek 3.4.

$$\left|\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right|$$

Obrázek 3.4: Příklad charakteristické matice CA

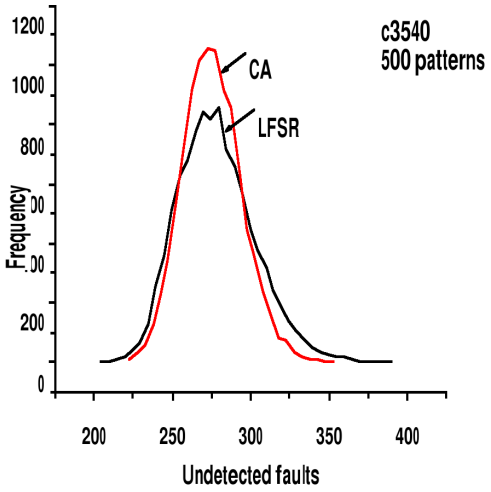
  V případě popisu CA jeho charakteristickou maticí lze získat nový stav libovolné buňky automatu tak, že po určení okolí uvažované buňky z příslušného řádku matice provedeme součet modulo 2 stavů všech buňek tvořících toto okolí. Výsledná hodnota pak odpovídá novému stavu uvažované buňky.

3.1.4 Srovnání celulárních automatů s LFSR

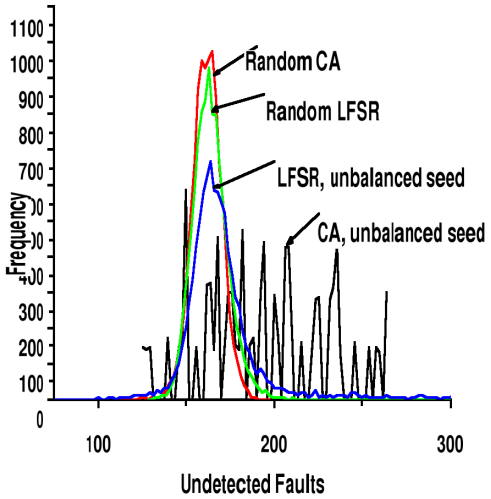
Pokud používáme LFSR pro generování pseudonáhodných testovacích vektorů ve smyslu kapitoly 3.1.2, kde vektory odebíráme přímo z paralelních výstupů registru, takto získaná posloupnost vektorů bude vykazovat poměrně výrazné znaky pravidelnosti. Ty budou obvzláště patrné v případě, že struktura registru bude popsána primitivním polynomem.
Takové chování je projevem korelace bitových posloupností tvořených v čase stavy jednotlivých klopných obvodů registru [5].
Oproti LFSR mají CA tu výhodu, že jimi produkované posloupnosti vektorů mají více náhodný charakter. To je důsledek podstatně redukované korelace bitových posloupností klopných obvodů, za kterou však zaplatíme zkrácením periody automatu [5].
Za účelem porovnání vlastností CA a LFSR z hlediska vhodnosti jako generátoru testovacích vektorů v rámci BIST byla vypracována celá řada studií, kromě [5] se jedná např. o [6], [3] a další.
Na obrázku 3.5 převzatém z [3] je uveden výsledek jednoho takového měření srovnávajícího vlastnosti CA a LFSR pro obvod c3540.
V rámci tohoto měření byly porovnávány CA s náhodně generovaným seedem a kompletně náhodně vytvářené LFSR, u kterých byl kromě seedu náhodně stanoven i generující polynom. V obou případech bylo takto získanými CA a LFSR generováno 500 testovacích vektorů, které byly posléze aplikovány na testovaný obvod a byl zaznamenán počet nedetekovaných poruch. Celý proces se pak opakoval celkem 10000-krát.
Na obrázku samotném je znázorněno srovnání dosaženého pokrytí poruch testovaného obvodu oběma generátory. Je z něj dobře patrné, že rozdělení počtu nedetekovaných poruch oběma generátory má stejnou střední hodnotu, ale v případě CA má menší standardní odchylku.

3.1.5 Celulární automaty s rozváženým seedem

V předchozí kapitole jsme uvažovali generátory pseudonáhodných testovacích vektorů realizované pomocí CA a LFSR, jejichž seed byl stanoven zcela náhodně a výskyt nul a jedniček v takovém seedu se řídil rovnoměrným rozložením pravděpodobnosti. Takové generátory jsou díky svým dobrým vlastnostem pro testování obvodů vhodné pro použití v BIST.
V některých případech však bývá testování obvodů komplikováno výskytem relativně malého počtu *těžce detekovatelných poruch*, kterých bývá typicky kolem 10% z celkového počtu všech detekovatelných poruch testovaného obvodu [3]. Tyto poruchy lze detekovat pouze a jedině jedním nebo několika málo specifickými testovacími vektory. Až dosud uvažované generátory však generují různé testovací vektory se stejnou pravděpodobností. S přihlédnutím k tomuto faktu a vzhledem k mohutnosti prostoru testovacích vektorů, je u takových generátorů šance na vygenerování posloupnosti vektorů obsahující těchto několik konkrétních vektorů velice nízká. U obvodů obsahujících takové poruchy je tedy nutné sáhnout po upraveném způsobu testování, tzv. *váženém testování* [5].
Podstata tohoto způsobu testování spočívá ve změně rozložení pravděpodobnosti výskytu nul a jedniček v různých částech generovaných testovacích vektorů. Toho se dá kromě modifikace vektorů přidanou kombinační logikou, jako je tomu například v případě mixed-mode BIST, dosáhnout také vhodnou volbou seedu jejich generátoru [5]. Takto nastavený generátor má pak výrazně vyšší pravděpodobnost vygenerování vektorů, potřebných k detekci těžce detekovatelných poruch.
Jeden možný způsob, jak takový seed zvolit je popsán v [3]. V rámci experimentu byly porovnávány CA a LFSR s náhodně vytvořeným seedem a tytéž generátory s *rozváženým seedem*, tvořeným samými nulami a jednou, náhodně umístěnou jedničkou. Výsledky celkem čtyř měření, během kterých bylo na obvod s838 aplikováno 500 testovacích vektorů získaných z takto nastavených generátorů, jsou uvedeny na obrázku 3.6.
Z obrázku je patrné, že v některých případech dosahovaly rozvážené CA podstatně lepšího pokrytí poruch testovaného obvodu než stejným způsobem rozvážené LFSR. Z toho vyplývá, že pokud se pro testovaný obvod povede zvolit ten správný seed, pak takové CA dosahují vynikajících výsledků.



Obrázek 3.5: Porovnání pokrytí dosahovaného CA a LFSR



Obrázek 3.6: Pokrytí dosahované rozváženými CA a LFSR

3.2 Úvod do optimalizace

Optimalizace je obecně vazto proces úpravy stávajícího řešení nějakého problému za účelem zlepšení jeho vlastností. Při optimalizaci zkoušíme různé varianty řešení a znalostí, kterých během toho nabýváme, se snažíme využívat tak, abychom postupně získávali stále lepší a lepší řešení [7]. Pojmy „problém“ a „řešení“ v tomto případě chápejme pouze v co nejširším možném slova smyslu.

Definice 3.2.1 *Uvažujme nyní následující definici problému: nechť* $X = \{x_1, x_2, ..., x_n\}$ *je množina* vstupních proměnných problému P, $Y = \{y_1, y_2, ..., y_m\}$ *je množina* konfiguračních proměnných problému P *a* $Z = \{z_1, z_2, ..., z_p\}$ *je množina* výstupních proměnných problému P. *Dále nechť* $C(I, Y)$ *je* omezující podmínka problému P *a* $W(I, Y)$ *je* optimalizační kritérium problému P, *kde* I *je* instance problému P *daná ohodnocením vstupních proměnných tohoto problému. Pak* problém P *definujeme jako* $P = \{X, Y, Z, C(I, Y), W(I, Y)\}$ *[8].*

Definice 3.2.2 *Dále platí, že* konfigurace Y instance I problému P *je ohodnocení konfiguračních proměnných tohoto problému a* řešení Y instance I problému P *je přípustná konfigurace* Y, *tj. platí* $C(I, Y)$. *Za* optimální řešení Y instance I problému P *budeme považovat takové řešení* Y, *které bude mít ze všech možných řešení nejvyšší hodnotu optimalizačního kritéria.* Suboptimální řešení Y instance I problému P *je pak takové řešení* Y, *jehož hodnota optimalizačního kritéria je hlediska aplikace stále ještě přijatelná [8].*

Množina vstupních proměnných určuje konkrétní instanci problému, dále jen instanci, a upřesňuje tak jeho zadání. Konfigurační proměnné popisují potenciální řešení instance a platnost takto popsaného řešení nám udává omezující podmínka. Množina výstupních proměnných pak popisuje samotný výstup instance. Z výše uvedeného je zřejmé, že množina výstupních proměnných je často podmnožinou množiny konfiguračních proměnných a v řadě případů jsou obě množiny totožné. Pokud tomu tak není, je třeba důsledně rozlišovat mezi *výstupem instance* a řešením instance. Poslední prvek definice, optimalizační kritérium, určuje *kvalitu řešení* instance v kontextu zadání celého problému. Pro nás budou dále nejdůležitější konfigurační proměnné, protože jsou to právě ony, které podrobíme optimalizaci.
Pozorný čtenář si jistě všimne toho, že takováto definice problému je velice blízká definci *problému kombinatorického*. Liší se od ní v podstatě jen tím, že kombinatorický problém je popsán s pomocí proměnných, které mají diskrétní obor hodnot. Kombinatorický problém je tak vlastně jen speciálním případem výše popsaného problému. K tomu se však ještě dále v textu vrátíme.

Definice 3.2.3 *Uvažujme nyní* algoritmus A řešící problém P, *který během výpočtu používá kromě konfiguračních proměnných problému* P, $X = \{x_1, x_2, ..., x_n\}$ *i* pracovní proměnné, $V = \{v_1, v_2, ..., v_q\}$ *a nazvěme sjednocení obou množin proměnných* vnitřními proměnnými, $U = X \cup V$. Stavem s algoritmu A řešícího problém P *budeme potom rozumět jakékoliv ohodnocení vnitřních proměnných algoritmu [8].*

Definice 3.2.4 *Nechť* S *je množina všech stavů algoritmu* A *řešícího problém* P *a nechť* $Q = \{q_1, q_2, ..., q_r\}$ *je množina operací* $S \to S$ *takových, že* $q(s) \neq s$ *pro žádné* $s \in S$. *Pak* stavový prostor algoritmu A řešícího problém P *je dvojice* (S, Q) *[8].*

Jedná se tedy o množinu všech stavů algoritmu dovybavenou o množinu operací, které umožňují přechod mezi různými stavy algoritmu a tedy i pohyb jeho stavovým prostorem.
K výše uvedené definici je třeba poznamenat, že pracovní proměnné nesou často informaci o tom, zda je ta či ona konfigurační proměnná platná, a tedy zda-li už byla ohodnocena. V takovém případě se rozlišuje stavový prostor daný pouze konfiguračními proměnnými problému od *prostoru hledání* odpovídajícímu shora uvedené definici. V dalším textu budeme pracovat se stavovým prostorem ve smyslu znění tohoto odstavce.

3.2.1 Základní přístupy k optimalizaci

Teď, když jsme si zavedli pojem „problém“ a pojmy s ním související, můžeme přikročit k dalším výkladu o optimalizaci.
Přístupy k optimalizaci můžeme podle jejich různých aspektů rozdělit celkem do šesti základních kategorií [7]. Jejich hranice jsou však vytýčeny jen přibližně a kategorie se tak v mnoha případech vzájemně překrývají:

1. V první kategorii rozlišujeme přístup *metodou pokus-omyl* a *funkční optimalizaci*.

   - Metoda pokus-omyl se při zkoumání kvality různých řešení z velké části spoléhá na intuitivní volbu řešení. Činíme tak, aniž bychom blíže znali optimalizační kritérium, což se v praxi stává velice často.
   - Naproti tomu, při funkční optimalizaci dokážeme optimalizační kritérium vyjádřit pomocí matematické funkce a optimalizace spočívá v hledání *extrémů této funkce* buď pomocí příslušného matematického aparátu nebo numericky.

2. Na *počet dimenzí* optimalizovaného problému klademe důraz v druhé kategorii.

   Je zřejmé, že s růstem počtu dimenzí prudce stoupá i složitost optimalizace. Nárůst složitosti je zpravidla exponenciálního charakteru.

3. Do třetí kategorie pak spadá *statická* a *dynamická optimalizace*.

   Při dynamické optimalizaci, na rozdíl od optimalizace statické, uvažujeme i čas. Tím nám přibyde jedna dimenze a složitost optimalizace se tak podstatně zvýší. Velké množství problémů z reálného světa vede právě na tento typ optimalizace. Otázkou zůstává, jakým způsobem budeme při optimalizaci s časem pracovat.

4. Nejen tím se zabývá čtvrtá kategorie, která rozlišuje optimalizaci z hlediska použití proměnných s *diskrétním* nebo *spojitým oborem hodnot*.

   - V případě diskrétních proměnných mluvíme o optimalizaci kombinatorických problémů. Tato třída problémů má konečně mnoho diskrétních řešení, která je možné všechna prozkoumat s použitím hrubé síly. Díky tomu je vhodná pro řešení s pomocí prostředků výpočetní techniky.
   - Pokud však hledáme extrém matematické funkce, jedná se zpravidla o optimalizaci spojitou.

5. Pátá kategorie řeší, zda použít proměnné s *omezeným* či *neomezeným oborem hodnot*.

   V případě proměnných s omezeným oborem hodnot to znamená, že optimalizační kritérium svážeme podmínkami, typicky ve formě rovnic. To nám samozřejmě celý proces zkomplikuje a proto se v případě, že je takovýto postup nezbytný, snažíme transformací optimalizačního kritéria podmínek zbavit.

   Pokud například budeme optimalizovat funkci $f(x)$ na intervalu $-1 \leq x \leq 1$, bude pro nás výhodné zavést substituci $x = sin(u)$ a dále optimalizovat funkci $f(sin(u))$ [7].

6. Poslední úhel pohledu na optimalizaci představuje šestá kategorie, ve které rozlišujeme mezi optimalizací *deterministickou* a *nedeterministickou*.

   - Řada optimalizačních algoritmů pro různé problémy vychází z pevně dané sady počátečních řešení těchto problémů, kterou pak tyto algoritmy transformují deterministickou posloupností operací. Díky tomu jsou zpravidla při procházení stavového prostoru velice rychlé, ale mají sklon k uváznutí v lokálních extrémech.

- Algoritmy, které naproti tomu nejprve navzorkují stavový prostor z velké části náhodným způsobem a dále jej pak procházejí nedeterministicky, tj. jejich přechodová funkce se řídí určitým rozdělením pravděpodobnosti, ve výsledku sice pomaleji konvergují k optimálním řešením, ale mají zato mnohem vyšší pravděpodobnost nalezení globálního optima. To je dáno především jejich výrazně vyšší odolností vůči uváznutí v lokálních extrémech ve srovnání se třídou algoritmů popsaných v předchozím odstavci.

Všechny výše popsané přístupy k optimalizaci mají však jedno společné: hledají pomocí vhodného algoritmu optimální řešení zadaného problému. Pokud takové řešení nedokážeme dopředu specifikovat, tj. neznáme horní nebo spodní mez optimalizačního kritéria, nebo je jeho hledání příliš zdlouhavé, je cílem optimalizace nalézt alespoň suboptimální řešení, které by vyhovovalo potřebám dané aplikace.

3.2.2 Minimum-seeking algoritmy

Tato třída algoritmů má společný základ, kterým je více či méně systematické prohledávání stavového prostoru algoritmu s ohledem na extrémy optimalizačního kritéria. Existuje mnoho různých způsobů, jakými se lze ve stavovém prostoru pohybovat a hledat tyto extrémy, ať už se jedná o minimální či maximální hodnoty optimalizačního kritéria. Ve většině případů však zpravidla nejsme schopni určit, zda-li je nalezený extrém lokálního nebo globálního charakteru. Nalezení globálního extrému je pro algoritmy tohoto typu obecně velice obtížný úkol, s vyjímkou situací, kdy má *povrch stavového prostoru* jednoduchý tvar. Povrchem stavového prostoru je zde myšlena plocha definovaná optimalizačním kritériem.
Typickým příkladem algoritmu tohoto typu je algoritmus *úplného prohledávání stavového prostoru*. Ten při hledání globálního optima ve formě extrému optimalizačního kritéria postupuje zhruba následujícím způsobem:

1. Nejprve provede dostatečně jemné vzorkování stavového prostoru, které obnáší vyhodnocení optimalizačního kritéria v jeho zvolených bodech. Pokud se jedná o diskrétní stavový prostor, jako je tomu v případě kombinatorických problémů, zvolené body odpovídají přímo množině všech stavů tvořících tento prostor.

2. Poté projde algoritmus takto získanou množinu vzorků hrubou silou, tj. postupně zkoumá jeden vzorek za druhým. Za globální optimum pak zvolí takový bod stavového prostoru, který odpovídá vzorku s nejnižší či, podle zadání problému, naopak nejvyšší hodnotou.

Z výše uvedeného je zřejmé, že pokud je počet takto zkoumaných vzorků vzhledem ke složitosti stavového prostoru příliš malý, globální optimum nám v případě spojitého prostoru s vysokou pravděpodobností proklouzne sítem výběru, a my tak v nejlepším případě získáme suboptimální řešení daného problému. Pokud uvažujeme diskrétní prostor, tak máme nalezení globálního optima vždy zaručeno.
Jiným příkladem je už dříve zmíněná *funkční či analytická optimalizace*, která se od úplného prohledávání stavového prostoru liší především způsobem výběru vzorků:

1. Tento přístup nahrazuje první krok úplného prohledávání stavového prostoru, vzorkování samotného prostoru, analytickým nebo numerickým určením extrémů optimalizačního kritéria. Dále pak pokračuje obdobným způsobem.

2. Ve druhém kroce pak algoritmus pracuje s množinou vzorků, která by, díky způsobu jejich výběru, měla obsahovat i globální optimum. Navíc má výsledná množina prohledávaných vzorků z principu podstatně menší mohutnost, než je tomu v případě předchozího algoritmu.

Analytická optimalizace je nesporně elegantější než prosté prohledávání stavového prostoru, nicméně složitost jejího prvního kroku přímo závisí na dimenzi uvažovaného stavového prostoru a také na tvaru jeho povrchu.
Pokud obsahuje povrch trhliny a zlomy, vyžaduje tento přístup úpravu spočívající v použití *Lagrangeových multiplikátorů*, které umožňují nahradit optimalizační kritérium se všemi jeho omezeními funkcí, která tato omezení obsahuje inherentně [7]. Podrobnější popis této techniky však přesahuje rámec tohoto textu.

3.2.3 Algoritmy inspirované přírodními procesy

Zatímco klasické optimalizační algoritmy se spoléhají na deterministicky řízené prohledávání stavového prostoru, v osmdesátých letech minulého století se začaly objevovat nové typy algoritmů, souhrnně nazývané *evoluční techniky*, jejichž tvůrci čerpali inspiraci ve fungování přírodních procesů.
Mezi tyto algoritmy se řadí například [9]:

- *genetické algoritmy* (angl. Genetic Algorithms),
- *genetické programování* (angl. Genetic Programming),
- *evoluční programování* (angl. Evolutionary Programming).

Mezi jim příbuzné algoritmy pak patří

- *simulované žíhání* (někdy také uváděné jako *simulované ochlazování*, angl. Simulated Annealing),
- *optimalizace pomocí roje částic* (angl. Particle Swarm Optimization),
- *optimalizace pomocí kolonie mravenců* (angl. Ant Colony Optimization).

Všechny výše uvedené algoritmy se při řešení problémů pohybují stavovým prostorem tak, že počáteční sadu řešení transformují pomocí specifických operátorů a přesouvají se tak do jiných oblastí stavového prostoru. Aplikace těchto operátorů se pak řídí různými statistickými metodami.
Tímto způsobem dokáží efektivně prohledat rozsáhlé stavové prostory a nevadí jim přitom ani použití diskrétních proměnných ve specifikaci problému, ani případné nespojitosti optimalizačního kritéria.
Na přírodu samotnou a procesy, které se v ní odehrávají, je možné nahlížet jako na obrovský optimalizační algoritmus jehož předmětem optimalizace jsou celé *živočišné druhy*. Kvalita jednotlivých druhů je v něm určena mírou, na kolik se každý druh dokázal přizpůsobit svému habitatu [7].
Tato veličina, definovaná na celé *živočišné populaci*, tak formuje zvlněnou plochu, jejíž míra elevace v jednotlivých oblastech prostoru této populace odpovídá dříve uvedené míře adaptace živočišných druhů vyskytujících se v té které oblasti prostoru.
Jinými slovy si tento prostor můžeme představit jako hornatou krajinu, jejíž vyvýšené části obývají jednotlivé živočišné druhy. Rozloha těchto částí pak udává diverzitu populací druhů, které je obývají. Diverzitou v tomto případě rozumíme podíl *jedinců* s unikátními rysy v rámci celé populace daného druhu. Čím je rozloha takové části menší, tím vyšší je stupeň adaptace vyžadovaný pro přežití druhu obývajícího tuto část.
V předchozím odstavci jsme narazili na pojem „rys jedince", aniž bychom jej nějak blíže specifikovali. Rys jedince je možné považovat za vnější projev *dědičné informace*, která je v každém jedinci uložena a z větší části se mezi jedinci předává z generace na generaci. O tom, jakým způsobem je toho dosaženo, bude pojednávat další část tohoto textu.

Tím se dostáváme ke způsobu, jakým je v jedinci uložena dědičná informace a jakým způsobem se tato informace projevuje na morfologické stavbě jedince, jeho vlastnostech a chování v rámci okolního prostředí [10]:

- Základní jednotkou informace organismu každého jedince je *gen*, což je segment nukleové kyseliny, který definuje přesnou strukturu jedné bílkoviny a řídí způsob její produkce a jejího dalšího metabolismu v organismu. Soubor všech genů v rámci celého organismu se pak nazývá *genom*.

  Vzhledem k faktu, že sám gen nemá atomickou strukturu a je tvořen kombinací čtveřice nukleotidů, existuje pro každý gen více možných kódování–tzv. *allel*. Allely jsou tedy různé variace jednoho genu.

- Geny jsou seskupeny do *chromosomu*, kterých se může v jedné buňce organismu vyskytovat větší počet. V lidském organismu se pak typicky jedná o jeden pár chromosomů do ktereréhо přispěl každý z rodičů právě jedním chromosomem. Místu, které gen v rámci chromosomu obsazuje se říká *locus*.

  Chromosom je tedy tvořen velice dlouhou posloupností genů, která řídí vlastnosti a funkci celého organismu. Každá buňka organismu navíc obsahuje kompletní sadu chromosomů a tento fakt je z hlediska správné funkce organismu nesmírně důležitý, protože umožňuje decentralizované řízení jeho základní látkové výměny.

Při *pohlavním rozmnožování* jedinců dojde mezi dvěma jedinci k vzájemné výměně genetické informace. V pohlavních buňkách každého jedince je obsažen pouze jeden chromosom, ze kterého se při rozmnožování vytvoří identická kopie. Tyto kopie si pak oba jedinci vymění, takže každý z nich bude mít dále k dispozici jeden chromosom původní a jeden od svého protějšku. Nakonec každý jedinec oba chromosomy rekombinuje tak, že se tyto nejdříve v jednom bodě spojí, vzájemně si prohodí koncové části a nakonec se od sebe zase oddělí. Této operaci se říká *operace křížení*.
V celé populaci druhu dochází v průběhu času k rozmnožování jedinců výše popsaným způsobem. Tento proces však není zcela náhodný, ale řídí se několika několika základními pravidly, která v polovině 19. století postuloval ve své práci nazvané *„O původu druhů"* britský přírodovědec Charles R. Darwin.
Tato pravidla se dají shrnout následujícím způsobem:

1. Potomek dědí výraznou část svých vlastností po rodičích. Tím je zajištěna krátkodobá stabilita populace a výrazná mezigenerační změna vlastností populace je bez vnějšího zásahu velice nepravděpodobná.

2. V populaci pohlavně se rozmnožujících jedinců nejsou žádní dva jedinci stejní. Dále platí, že většina této rozmanitosti přechází i do další generace takovéto populace.

3. Jedinci jsou obecně velice plodní a vyprodukují výrazně větší množství potomků, než nakonec dosáhne dospělosti a může se začít rozmnožovat.

4. Přežití jedince a pravděpodobnost, s jakou dosáhne dospělosti a začne se rozmnožovat, přímo závisí na vlastnostech tohoto jedince a na tom, jak dalece se dokázal přizpůsobit danému prostředí.

Výše uvedená pravidla ve své podstatě popisují jev, který Darwin nazval *přírodním výběrem* [11].
Tento jev je charakteristický tím, že jakkoliv drobné variace v genomu jedince, které mu v daném prostředí poskytnou konkurenční výhodu před ostatními jedinci, a tudíž i zvýší jeho šanci na přežití, mají tendenci přecházet na jeho potomky. To má v dlouhodobém horizontu za následek adaptaci populace druhu jako celku na prostředí, ve kterém se tato populace vyskytuje.

3.3 Klasické genetické algoritmy

Genetický algoritmus, dále jen *GA*, je jednou z pokročilých optimalizačních technik, které jsou insipirovány principy genetiky a dříve popsaným mechanismem přírodního výběru.
Tento algoritmus je řazen mezi globální iterativní *heuristiky*, tedy metody pro řízené prohledávání stavového prostoru určité třídy problémů. Jejich účelem je co nejrychlejší nalezení použitelného řešení nebo naopak nalezení řešení, které se bude co nejvíce blížit řešení optimálnímu, bez ohledu na čas strávený jeho hledáním [12].
Základním principem funkce GA je udržování konstatně veliké populace řešení daného problému a její evoluce v diskrétním čase za použití principů přírodního výběru (viz obr. 3.7) [7]. Diskrétním časem se zde myslí tok času odměřovaný v pevně stanovených intervalech a vyjádřený ve formě posloupnosti generací uvažované populace.
Tímto způsobem napodobuje algoritmus vzorce chování, které můžeme pozorovat v přírodě a ve své podstatě modeluje proces adaptace populace řešení daného problému na prostředí, které je popsáno omezující podmínkou a optimalizačním kritériem tohoto problému.
Omezující podmínku zde uvádím výslovně pro to, že z určitého úhlu pohledu ji můžeme považovat za doplnění optimalizačního kritéria. O jedincích, kteří jí nevyhovují a nejsou tak platným řešením, můžeme totiž prohlásit, že jejich kvalita je z hlediska optimalizačního kritéria nulová nebo dokonce záporná. I přes tento fakt můžeme takové jedince v populaci ponechat a nevyřazovat je tak zcela ze soutěže o přežití. V některých případech je pak takový postup dokonce výhodný a má pozitivní vliv na vývoj celé populace.
Jako první zformuloval základní myšlenky genetického algoritmu v sedmdesátých letech minulého století americký vědec John H. Holland. Tuto myšlenku pak mnozí, včetně Hollanda samotného a některých jeho studentů [7], dále rozvíjeli a takto vzniklé implementace genetického algoritmu pak aplikovali s různou měrou úspěšnosti na širokou škálu optimalizačních problémů.
Klíčové charakteristiky genetického algoritmu zahrnují [7]:

- Pro GA není důležité, zda pracuje s vnitřními proměnnými diskrétního nebo spojitého charakteru, a jeho výkonnost tedy není přímo závislá na typu proměnných použitých při formulaci řešeného problému.

  Z toho vyplývá, že GA lze použít téměř na libovolný problém, jehož množiny charakteristických proměnných mají konečnou mohutnost.

- GA není příliš citlivý na počet konfiguračních proměnných řešeného problému, a tudíž ani na dimenzi odpovídajícího stavového prostoru.

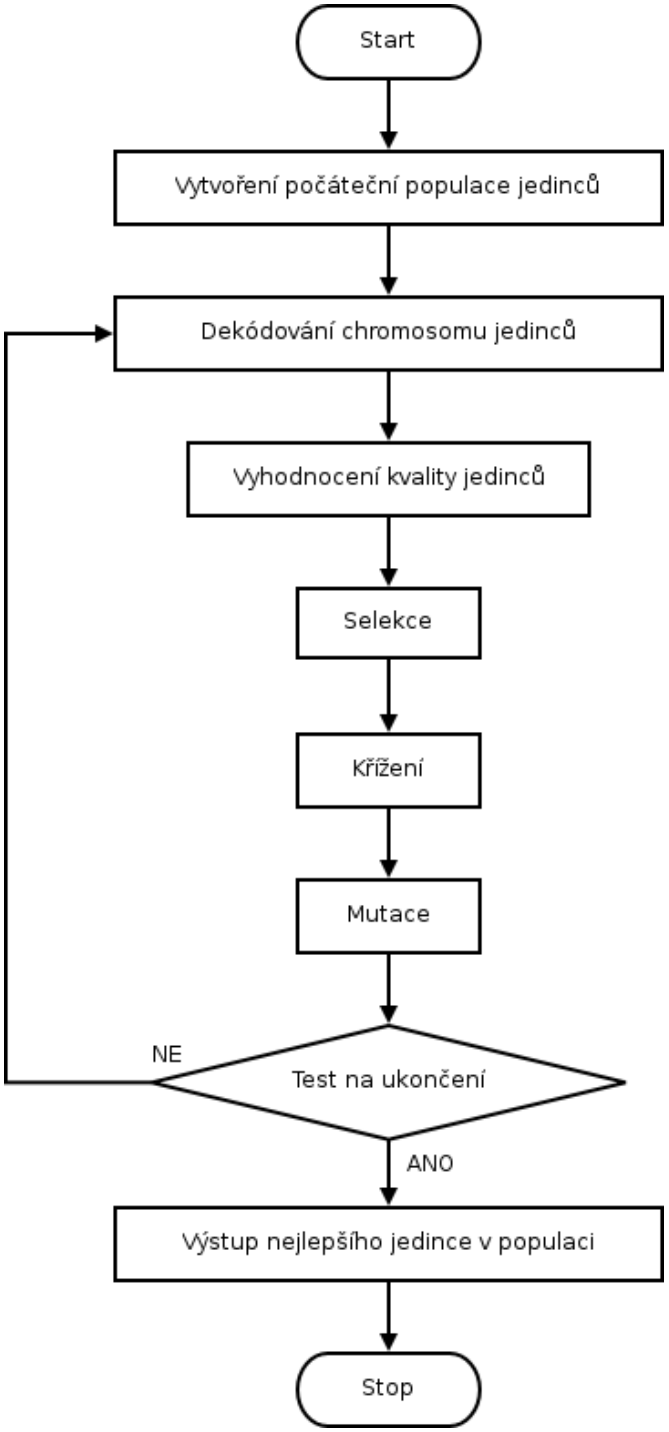
  S rostoucím počtem konfiguračních proměnných ovšem koreluje dimenze stavového prostoru a jeho velikost se zvětšuje zpravidla exponenciálně. Tento fakt je nutné vzít v úvahu při volbě velikosti počáteční populace a minimálního počtu generací, po který bude probíhat evoluce.

- Pro GA je také typické, že současně prohledává více rozsáhlých oblastí stavového prostoru. Konkrétní tvar, velikost a počet těchto oblastí je na začátku algoritmu určen volbou počáteční sady řešení a v průběhu samotného výpočtu algoritmu se pak dále vyvíjí.

  Díky této charakteristice jsou tyto algoritmy vhodné k paralelizaci, což je v dnešní době, kdy se stále více využívají paralelní výpočetní systémy, velice žádoucí vlastnost.

- S předchozím bodem také souvisí jedna důležitá vlastnost, a tou je možnost výstupu několika optimálních, příp. suboptimálních řešení daného problému na konci algoritmu.

  Tato vlastnost nám může být užitečná například v situacích, kdy na výsledné řešení máme i jiné požadavky, než jsme do algoritmu vložili ve formě optimalizačního kritéria.



Obrázek 3.7: Obecné schéma genetického algoritmu

Taková situace může nastat v případě, když v době formulace problému nejsou plně známy všechny požadavky na kvalitu výsledného řešení nebo jsou-li takové požadavky do optimalizačního kritéria jen velice obtížně zapracovatelné.

- S ohledem na výše popsaný způsob prohledávání stavového prostoru je zřejmé, že GA je zcela imunní vůči případným nespojitostem optimalizačního kritéria řešeného problému a navadí mu ani extrémně složitá plocha stavového prostoru.

- Poslední z vlastností GA, kterou v tomto výčtu uvedu, je možnost transformace konfiguračních proměnných řešeného problému.

  Pokud nám typ těchto proměnných nebo podmínky, kterými jsou svázány, z nějakého důvodu nevyhovují, můžeme proměnné nahradit za takové, se kterými se nám bude snáze pracovat.

  Říkáme pak, že jsme konfigurační proměnné zakódovali pomocí kódu, který přiřazuje vnitřním proměnným algoritmu sémantický význam v rámci kontextu řešeného problému právě ve formě konfiguračních proměnných.

Je důležité si uvědomit, že ačkoliv dávají výše uvedené vlastnosti genetickým algoritmům při řešení řady problémů výhodu před jinými typy heuristik, mohou také za určitých podmínek hledání optimálního řešení znatelně zkomplikovat.
Jedním z příkladů takového chování je situace, kdy má plocha stavového prostoru velice jednoduchý, např. konkávní tvar (ve formě prohlubně). V takovém případě zpravidla optimální řešení naleznou dříve heuristiky založené např. na sledování gradientu než genetické algoritmy a jim podobné.
Tento jev je důsledkem způsobu, jakým GA pracuje s populací. Tu je totiž nutné s každou novou generací znovu a znovu vyhodnotit, jedince po jedinci, řešení po řešení. Časová náročnost tohoto kroku pak obecně záleží na složitosti vyhodnocení optimalizačního kritéria.
Může se tak stát, že zatímco GA stále ještě vyhodnocuje některou z počátečních generací řešení uvažovaného problému, nalezne optimální řešení výše zmíněná heuristika.
Proto je vždy nutné dobře posoudit, jestli je na daný problém vhodnější nasadit GA nebo nějakou jinou heuristiku lokálnějšího charakteru, popř. zvolit kombinaci obou výše jmenovaných.

3.3.1 Binární genetický algoritmus

V této části kapitoly se zaměříme na jednu konkrétní variantu GA, a to na *binární genetický algoritmus*, který je ve své podstatě mimořádně vhodný k implementaci na počítači.
Tato varianta, mimochodem úplně první varianta GA, která kdy vznikla, pracuje s čistě *binárně kódovaným chromosomem*, tj. s chromosomem složeným z pouze jedniček a nul. Je zřejmé, že takto definovaný chromosom lze snadno reprezentovat posloupností bitů v binárním kódu, se kterým interně pracují prakticky všechny běžně používané výpočetní systémy.
Navíc budeme v dalším textu uvažovat čistě obecný algoritmus, na kterém budeme zkoumat jeho základní stavbu, aniž bychom se přitom soustředili na konkrétní problém, který tento algoritmus řeší. To mimo jiné znamená, že budeme uvažovat *populaci složenou z obecných jedinců* obsahujících binární choromosom, namísto z řešení konrétního problému.

3.3.2 Výběr množiny kódovaných parametrů

Při výběru *parametrů pro reprezentaci informace uložené v jedinci* se snažíme najít takové parametry řešeného problému, které by jej plně popsaly na zvolené úrovni abstrakce. Obecně platí, že bychom měli vybrat co nejmenší počet parametrů tak, abychom je mohli předat optimalizačnímu kritériu jako vstupní hodnoty. Pokud máme navíc k dispozici formální zadání problému,

pak je samozřejmě nejjednodušší vyjít přímo z množiny konfiguračních proměnných obsažených v takovém zadání.
Tím se dostáváme k volbě *optimalizačního kritéria*, které nám při evoluci slouží jako měřítko kvality zkoumaných jedinců. Toto kritérium by mělo plně využívat informaci obsaženou v jedinci, tj. použít při výpočtu všechny parametry v něm zakódované. Z opačného úhlu pohledu nám pak může optimalizační kritérium posloužit při výběru nezbytné množiny parametrů, a tedy i stanovení příslušné úrovně abstrakce:

- V řadě případů představuje optimalizační kritérium *matematická funkce* s omezeným definičním oborem. Každé takové omezení pak ve svém důsledku prodražuje výpočet její funkční hodnoty. V některých případech je však možné funkci omezení alespoň částečně zbavit pomocí transformace a někdy je dokonce možné část parametrů zcela zanedbat jako nevýznamné a funkci tak dále zjednodušit.

- Další možností stanovení optimalizačního kritéria je *využití naměřených experimentálních hodnot*. Tím je myšleno porovnání informace obsažené v jedinci s výsledky předešlých experimentů, které v takovém případě doplňují zadání řešeného problému.

  Typickým příkladem takové situace je pokus o sestavení klasifikátoru, kde nás zajímá, zda chování jedincem specifikovaného klasifikátoru odpovídá chování zjištěnému experimantálně. Jinými slovy se jedná o to, zda klasifikátor dává v konkrétních situacích výsledky, které odpovídají skutečným hodnotám pro tyto situace změřeným.

- Poslední způsob, úzce související s předcházejícím příkladem, představuje *nahrazení matematické funkce experimentem*.

  Tento způsob je z hlediska času potřebného k vyhodnocení jedince zdaleka nejnáročnější, ale pokud nejsme s to přesně formulovat podmínky optimality jedince, tak pro nás představuje jedno z možných východisek z takové situace.

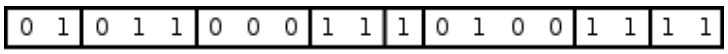
3.3.3 Kódování a dekódování chromosomu

Kódování, resp. k němu inverzní *dekódování*, je operace spočívající v převodu parametrů vybraných způsobem popsaným v předchozí kapitole do interního kódu, v našem případě do kódu binárního.
Způsob kódování parametrů závisí na jejich typu:

- Pokud kódujeme *parametry ze spojité domény*, musíme jejich skutečné hodnoty vhodným způsobem aproximovat ve zvoleném kódu. Tato činnost se obvykle nazývá *amplitudová kvantizace* a je myslím natolik známá, že ji zde není nutné dále rozvádět.

- V případě *parametrů z diskrétní domény* stačí obvykle najít způsob přepisu hodnot do zvoleného kódu. To znamená, že buď jejich hodnotu v tomto kódu přímo vyjádříme nebo vytvoříme slovník s pomocí kterého hodnoty převádíme. Slovníkem se zde myslí vyjádření bijektivního zobrazení z množiny hodnot parametru do množiny kódových slov zvoleného kódu.

Takto zakódované parametry, tvořené *bloky bitů*, pak odpovídají genům, ze kterých je celý chromosom sestaven (viz obr. 3.8). Při další práci s takovým chromosomem je zpravidla výhodné dbát na hranice genů v rámci chromosomu, a pracovat tak s geny jako s celky.



Obrázek 3.8: Struktura chromosomu

3.3.4 Vytváření počáteční populace

Populace je množina jedinců s konečnou mohutností, která je vybavena operacemi definovanými nad jedinci. Tyto operace slouží k manipulaci jedinců, a umožňují tak vývoj populace v čase. Existuje několik jednoduchých způsobů jak takovou populaci vytvořit:

- Prvním z nich je *náhodné generování chromosomu každého jedince v populaci*, a to buď ve smyslu řetězce genů nebo přímo řetězce bitů.

  Tento způsob je zdaleka nejjednodušší a umožňuje prohledat největší část stavového prostoru. Díky těmto vlastnostem je také nejpoužívanější.

- Další způsob spočívá ve *vložení předem vytvořené skupiny jedinců do populace* a doplnění zbytku populace metodou popsanou v předchozím bodě.

  Tímto způsobem se lze zaměřit na určité oblasti stavového prostoru, o kterých už dopředu tušíme, že budou s největší pravděpodobností obsahovat velice kvalitní jedince.

  Existuje také modifikace výše popsaného přístupu, kdy skupina vkládaných jedinců není definována exaktně, ale je namísto toho charakterizována pomocí *specifické vlastnosti*, kterou musí jedinci tvořící tuto skupinu splňovat. Jako příklad nám může posloužit poměr jedniček a nul v binárním řetězci reprezentujícím chromosom.

- Poslední možností je *naplnit populaci zcela deterministicky vytvořenými jedinci.*

  Tato možnost přichází v úvahu zpravidla v situacích, kdy už jsme nějakým způsobem, např. příbuznou evoluční technikou, jedince získali a potřebovali bychom evolucí dále zlepšit jejich vlastnosti.

3.3.5 Selekce jedinců

Selekce jedinců je proces inspirovaný přírodním výběrem a ve své podstatě slouží k určení pravděpodobnosti, s jakou se pak jednotliví jedinci v populaci účastní páření.
Selekce pracuje ve dvou fázích, z nichž v první dochází k samotné selekci a v druhé pak páření takto vybraných jedinců:

1. V první fázi pracuje selekce s *celou populací* a v závislosti na konkrétní implementaci může nabývat těchto podob:

   - Z populace je vyčleněna *skupina nejkvalitnějších jedinců*, kterým bude dovoleno spářit se. Mohutnost této skupiny je zpravidla předem stanovena ve formě konstanty.

     Nevýhoda tohoto postupu spočívá v nutnosti celou populaci předem seřadit podle kvality jedinců, což je zpravidla velice náročné na čas.

   - Z populace jsou vybráni všichni *jedinci, jejichž kvalita překračuje předem stanovenou mez.* Tato se může v průběhu samotné evoluce dynamicky měnit podle aktuální potřeby.

     To znamená, že pokud nám v důsledku rostoucí průměrné kvality populace projdou sítem téměř všichni jedinci, je načase mez selekce odpovídajícím způsobem upravit. V opačném případě musíme naopak takto zdecimovanou populaci doplnit čerstvými jedinci.

Tento způsob selekce se asi nejvíce blíží pravidlům přírodního výběru tak, jak je popsal Charles Darwin.

- Do další fáze přejde *celá populace* bez ohledu na kvalitu jednotlivých jedinců, tj. tento krok populaci nijak nezmění a ponechá ji v původním stavu.

  To jednak výrazně zjednoduší udržování konstantní velikosti populace a navíc tak z celého procesu úplně nevyřadíme slabší jedince. Ti pak mohou díky náhodě získat možnost spářit se, což má v některých případech za následek oživení jinak zdegenerované populace, ve které převládá elita propojená četnými příbuzenskými vztahy.

Na tomto místě je třeba podotknout, že výše uvedené varianty prvního kroku selekce se liší ještě v jedné důležité vlastnosti, a tou je způsob vytváření nové generace populace.

První dvě varianty, které ze stávající populace vybírají k páření pouze určitou skupinu jedinců, vytvářejí novou generaci tak, že do ní nejdříve umístí tuto skupinu. Zbytek nové generace pak doplní potomky vzešlými ze vzájemného páření jedinců z této skupiny tak, jak je to popsáno v následujícím, druhém kroku selekce.

Poslední varianta se od obou předchozích liší v tom, že novou generaci bezezbytku vyplní právě až v druhém kroku selekce. Existuje však jedna vyjímka z tohoto pravidla, té se ale dotkneme až v kapitole 3.3.8 o elitismu.

2. Druhou fází selekce je *páření skupiny jedinců* vybraných z populace v předchozí fázi s tím, že se jedinci páří vždy ve dvojici.

   Tyto dvojice lze sestavit několika různými způsoby:

   - *Párování do dvojic* postupuje systematicky od jednoho konce skupiny ke druhému, a takto vytvořené dvojice se vždy skládají z jedince z lichým pořadovým číslem v rámci skupiny a z jedince s pořadovým číslem sudým.

     Tento přístup je z hlediska implementace velice jednoduchý, ale příliš neodpovídá chování přírodních procesů, které selekci sloužily modelem.

   - *Náhodné párování* má oproti párování do dvojic tu výhodu, že se při vytváření dvojic z pseudonáhodně vybraných jedinců skupina promíchá a tím máme větší šanci na spáření dvou výrazně se lišících jedinců. To nám pak umožní prozkoumat i dosud neobjevené oblasti stavového prostoru.

     V této variantě párování mají všichni jedinci ve skupině zajištěnou stejnou pravděpodobnost páření, ale v předchozímu kroku selekce se do skupiny dostali z populace pouze silnější jedinci. Tím de facto dojde k protěžování silnějších jedinců přesně tak, jak je tomu v přírodě.

   - *Vážené náhodné párování* se od párování čistě náhodného liší tím, že jedinci ve skupině už nemají stejnou pravděpodobnost páření, ale tato je naopak přímo odvislá od jejich kvality.

     Jinými slovy jsme zúžili výběr jedinců k páření pouze na tu kvalitnější část populace a při samotném páření jedinců z této skupiny navíc preferujeme jedince s vyšší kvalitou na úkor těch méně kvalitních. Touto cestou tak dále zesilujeme *selekční tlak*, tj. míru protěžování těch nejkvalitnějších jedinců ve skupině.

     Vážení samotné je možné realizovat:

     – stanovením *kvalitativní třídy jedince* ve skupině.
       Jedinci jsou ve skupině sestupně seřazeni podle kvality a následně vzestupně očíslováni od nejlepšího po nejhoršího. Číslo, které jim bylo takto přiřazeno, nazveme kvalitativní třída. Podle této třídy je pak stanovena *pravděpodobnost výběru jedince*

$$P_n = \frac{N_k - C_n + 1}{\sum_{i=1}^{N_k} C_i}, \tag{3.5}$$

kde C_n je třída uvažovaného jedince a N_k je mohutnost seřazené skupiny jedinců, a tedy i celkový počet tříd [7].
Kumulativní pravděpodobnost výběru jedince pak stanovíme tak, že sečteme kumulativní pravděpodobosti jedinců předcházejících v seřazené skupině jedinci uvažovanému a pravděpodobnost výběru jedince samotného. Kumulativní pravděpodobnost výběru prvního jedince v takové skupině je přitom rovna přímo pravděpodobnosti jeho výběru.

– výpočtem *normalizované kvality jedince* v rámci skupiny, kterou spočítáme tak, že od kvality každého jedince ve skupině odečteme nejvyšší kvalitu jedince, který neprošel prvním kolem selekce.
Pravděpodobnost výběru jedince je pak

$$P_n = \left| \frac{Q_n}{\sum_{i=1}^{N_k} Q_i} \right|, \tag{3.6}$$

kde Q_n je normalizovaná kvalita uvažovaného jedince a N_k je mohutnost skupiny jedinců [7].
Kumulativní pravděpodobnost výběru jedince pak stanovíme obdobně jako při výpočtu kvalitativní třídy jedince.

Výběr jedince do dvojice pak provedeme tak, že vygenerujeme pseudonáhodné číslo z intervalu $\langle 0, 1\rangle$ a hledáme ve skupině jedinců takového jedince, jehož kumulativní pravděpodobnost výběru je zdola nejblíže vygenerovanému číslu.

- *Turnajový výběr* asi nejvěrněji kopíruje princip přírodního výběru a je také jednou z nejčastěji používaných metod výběru páru.
Funguje tak, že každý jedinec ve výsledné dvojici vzejde z miniaturního turnaje či *klání* několika, zpravidla dvou nebo tří jedinců. Vítěz takového klání je samozřejmě ten nejkvalitnější z jeho účastníků.

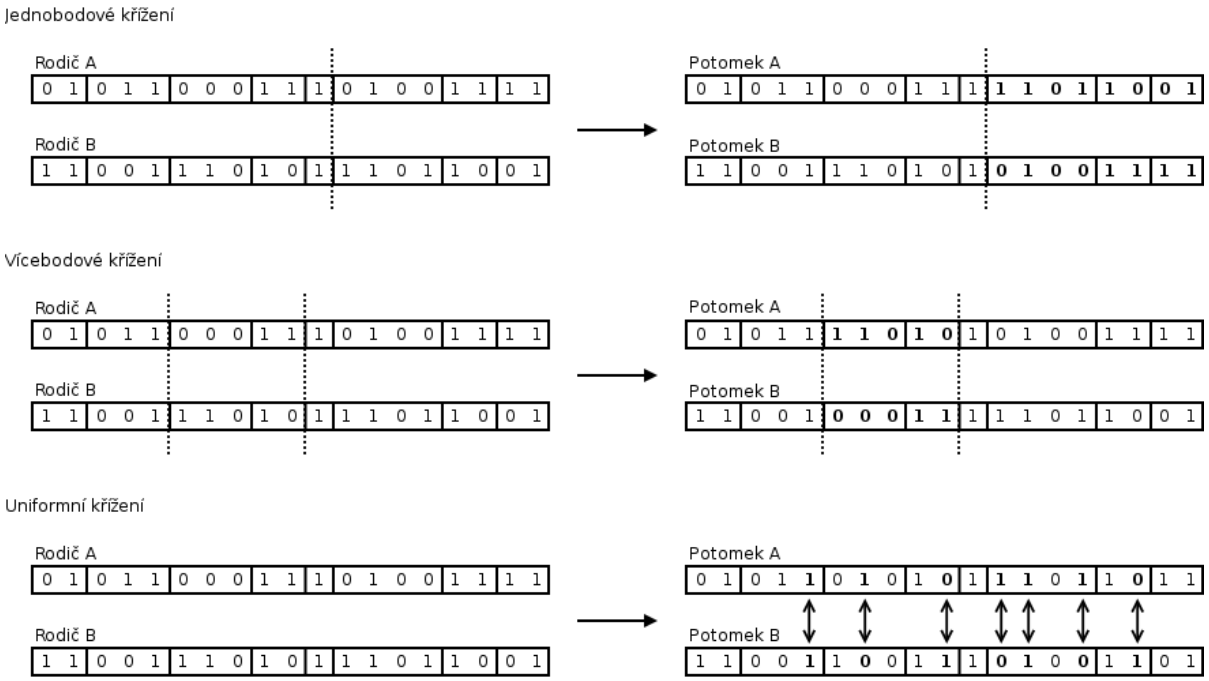
Každá takto získaná dvojice jedinců je posléze spářena za použití níže popsaných operací křížení a mutace. Takto získaní potomci jsou pak začleněni do nové generace populace. Způsob, jakým se to běžně realizuje bude rozveden v kapitole 3.3.4 o vytváření nové populace.

Nezbývá než podotknout, že pokud je dvojice tvořena buď shodnými jedinci nebo jedincem dvakrát po sobě vybraným, můžeme takovou dvojici považovat za platnou, anebo ji vytvořit znovu. V takovém případě je vhodné upravit mechanismus výběru tak, aby k takové situaci pokud možno nedocházelo vůbec, nebo alespoň zřídkakdy. Jedno z možných řešení je provádět výběr jedinců bez náhrady.

3.3.6 Křížení jedinců

Operace *křížení jedinců* je velice přibližný model procesu ke kterému dochází v přírodě při pohlavním rozmnožování buňek haploidních organismů. Při tomto procesu dochází mezi rodičovskými jedinci k výměně genetické informace uložené v chromosomu tak, že si oba rodiče část svého chromosomu prohodí (viz obr. 3.9).
Nejčastěji se používá několik typů křížení:

- U *jednobodového křížení* se v chromosomu náhodně zvolí *bod křížení* (tzv. kinetochora). Tento bod je v chromosomech obou rodičů shodný a zpravidla se volí na rozhraní dvou



Obrázek 3.9: Základní typy křížení

genů tak, abychom neporušili jejich strukturu. Oba rodiče si pak vzájemně prohodí část chromosomu od tohoto bodu do konce chromosomu.

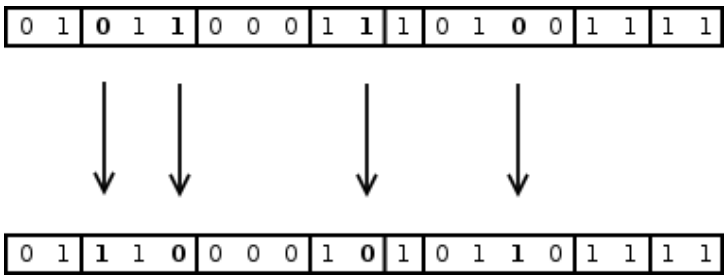
- *Vícebodové křížení*, často dvoubodové, funguje obdobným způsobem. Zvolíme příslušný počet bodů křížení, což nám chromosom rozdělí do několika segmentů. Rodiče si pak vždy prohodí sudé segmenty chromosomu.

- *Uniformní křížení*, narozdíl od předchozích dvou typů křížení, nedělí chromosom v jednotlivých bodech, ale prochází oba chromosomy současně a s předem zvolenou pravděpodobností prohazuje jejich odpovídající části, ať už se jedná o celé geny nebo přímo o samotné bity.

Tímto způsobem je zajištěna výměna genetické informace mezi oběma rodiči, která je motivována skutečností, že alespoň jeden takto získaný potomek může s poměrně vysokou pravděpodobností zdědit nejlepší vlastnosti rodičů a dosáhnout vyšší kvality než kterýkoliv z jeho rodičů.
Účinnost této operace závisí na určité, předem stanovené pravděpodobnosti, se kterou ke křížení dochází. To znamená, že někdy ke spáření obou rodičů nemusí vůbec dojít.

3.3.7 Mutace jedinců

Výše popsaná operace křížení je jedním ze dvou hlavních způsobů jak se může GA pohybovat stavovým prostorem a prozkoumávat tak jeho povrch. Druhou možnost představuje operace *mutace*.
Mutace spočívá v náhodné změně hodnoty několika málo bodů v chromosomu a zpravidla pracuje přímo na úrovni základních stavebních prvků chromosomu, v našem případě tedy na úrovni bitů (viz obr. 3.10).
Touto formou je při prohledávání stavového prostoru posílen vliv náhody, což má za následek širší pokrytí jeho povrchu a menší pravděpodobnost degenerace celé populace. Dají se tak náhodně



Obrázek 3.10: Princip mutace

objevit nové vlastnosti perspektivní v daném prostředí a současně zabránit přílišnému sbližování genomů jedinců v populaci, které má za následek ztrátu její variability.
I tato operace je, stejně jako je tomu v případě operace křížení, účinná jen s určitou, pevně danou pravděpodobností.

3.3.8 Elitismus

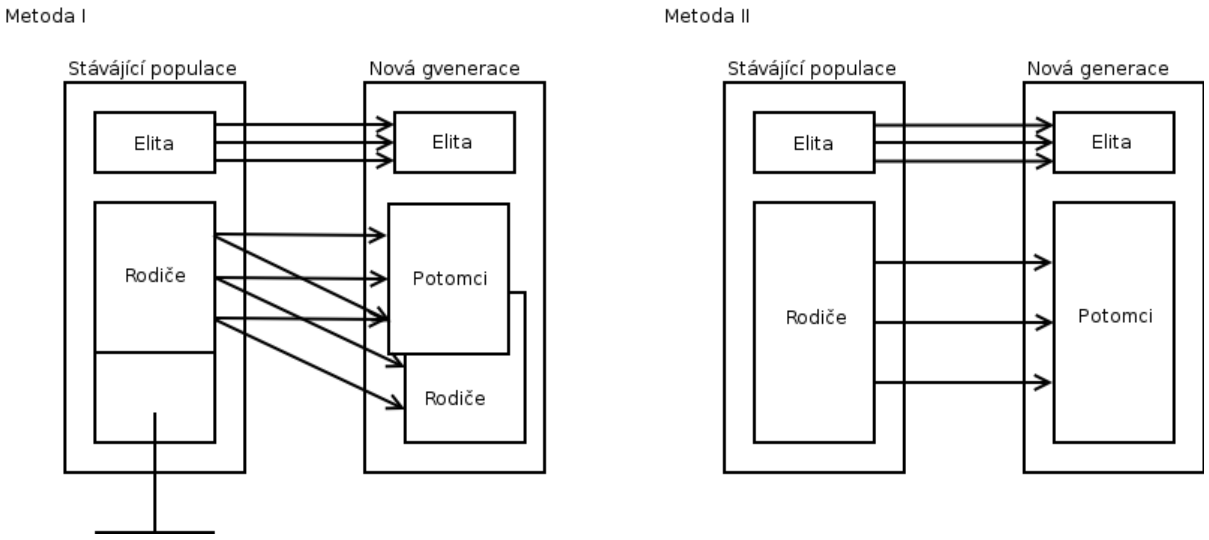
Elitismus je důležitá vlastnost populace, která urychluje její konvergenci k oblastem nejkvalitnějších jedinců. To umožňuje variantám GA, které tuto vlastnost implementují, dosahovat vyšší efektivity ve srovnání s klasickým přístupem, který tuto vlastnost nezahrnuje.
Princip elitismu spočívá v tom, že z populace vybereme v každé generaci pevně stanovený počet nejkvalitnějších jedinců, tzv. *elitu*, a vytvoříme z ní jádro generace nové. Zbytek nové generace pak doplníme přesně tak, jak jsme si to uvedli v kapitole 3.3.5 o selekci.
Elitním jedincům je přitom dovoleno dále se pářit, takže do nové generace přejdou společně se svými potomky, zatímco ostatní jedinci budou ve většině případů svými potomky nahrazeni. Tímto způsobem jednoduše zajistíme, abychom neštastnou náhodou nepřišli o nejkvalitnější jedince v důsledku křížení a mutací.
Průměrná kvalita skupiny elitních jedinců se tak může pouze zvyšovat a pokud vůbec existuje z hlediska kvality optimální jedinec, bude zaručeně členem této skupiny.
Elitní jedinci zpravidla přežívají po mnoho generací, a mají tak výrazně větší počet příležitostí k páření. Díky tomu mají nezanedbatelný vliv na vývoj celé populace, který je tím větší, čím větší je jejich počet. Tento je pak v průběhu evoluce zpravidla konstantní.

3.3.9 Vytváření nové generace

V kapitole 3.3.5 o selekci jsme narazili na *způsob vytváření nové generace*. Nyní, když už máme vybudovaný potřebný aparát k výběru perspektivích jedinců a jejich páření, můžeme tuto část textu uzavřít popisem možných způsobů, jakými se dá z jedné generace populace vytvořit generace nová.
Rozeznáváme dva základní přístupy, které mají společný ten fakt, že v obou nejprve přejde do nové generace případná elita. Zbytek generace se pak doplní jedním ze dvou následujících způsobů (viz obr. 3.11):

- Metodou, kterou jsme si uvedli v kapitole o selekci, nejprve vybereme z populace skupinu perspektivních jedinců, a tu pak po dvojicích spáříme.

  Dostaneme tak dvě přibližně stejně mohutné skupiny jedinců, skupinu rodičů a skupinu potomků. Obě skupiny nemají stejnou mohutnost, protože někteří rodiče nemuseli mít vůbec žádné potomky. V každém případě má každá z obou skupin menší mohutnost než je velikost populace, a tedy i velikost nové generace této populace.



Obrázek 3.11: Způsoby vytváření nové generace

Vzhledem k faktu, že nově vytvářená generace už může obsahovat elitu, je velice pravděpodobné, že se nám do této generace nepovede bezezbytku umístit jak skupinu rodičů, tak skupinu potomků.

Tuto situaci lze vyřešit tak, že do nové generace umístíme tolik potomků, kolik to jen bude možné a pokud nám v této generaci stále jestě zbyde volné místo, obsadíme ho rodiči. Přitom postupujme tak, že přednost při umisťování do nové generace mají kvalitnější rodiče. Méně kvalitní rodiče tedy uvolní místo některým jedincům ze skupiny potomků.

- Druhou možností je vyvořit novou generaci pouze z elity a potomků, kteří v této generaci obsadí místo svých rodičů. Z toho je zřejmé, že rodič, který není členem elity, se může do nové generace dostat pouze v případě, že nemá žádného potomka.

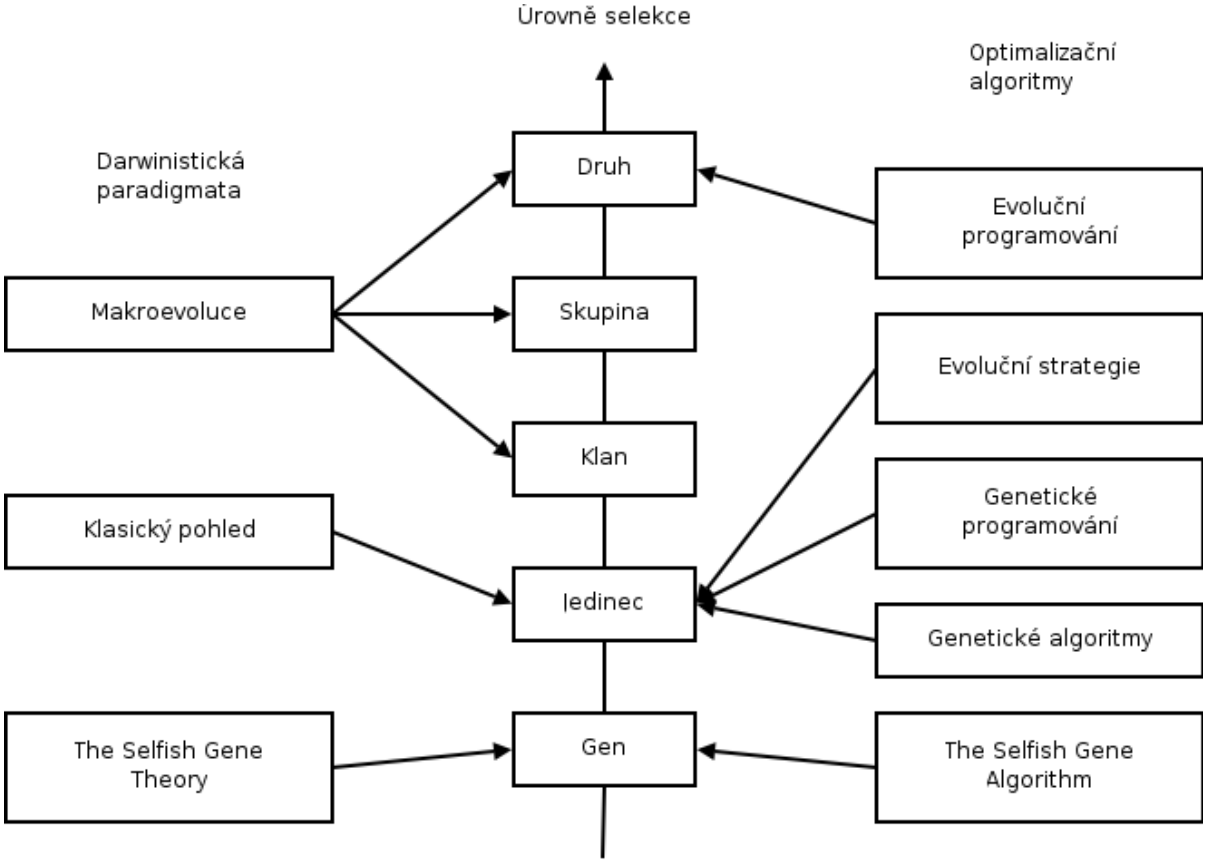
3.3.10 Stanovení některých parametrů evoluce

Aplikace genetických algoritmů není v žádném případě exaktní věda a blíží se spíše umění, kde výsledky do značné míry závisejí na autorově citu pro danou problematiku. Mohu tak závěrem nabídnout několik doporučení, která se z větší části zakládají na mé vlastní zkušenosti.
Rád bych se zmínil o několika dosti podstatných aspektech GA, a to o volbě velikosti populace, pravděpodobnosti úspěchu operací křížení a mutace a v neposlední řadě i určení délky samotné evoluce.
Velikost populace nám udává rozlohu současně prozkoumávané oblasti stavového prostoru. Naším cílem je vždy zvolit takovou velikost populace, která by představovala rozumný kompromis z hlediska poměru velikosti takto určené oblasti stavového prostoru a času, který algoritmus potřebuje k vyhodnocení všech jedinců vyskytujících se v této oblasti. Podle mých zkušeností se příliš nevyplatí zvyšovat velikost populace nad několik málo stovek jedinců.
Pravděpodobnost úspěchu operací křížení a mutace, tedy pravděpodobnost, s jakou se během páření skutečně provedou, by neměla překračovat cca. 70% v případě křížení a jednotky procent v případě mutace. Čím vyšší hodnoty bychom volili, tím více bychom se přibližovali náhodnému prohledávání stavového prostoru a evoluce by tak ztrácela smysl.
Délka procesu evoluce je vyjádřena počtem generací, které algoritmu povolíme vytvořit. Tento počet můžeme na začátku evoluce pevně stanovit, nebo jej přímo za běhu dynamicky upravovat v závislosti na vývoji stavu populace. Pokud například ani po pěti stech generacích nedokáže

algoritmus vyvinout lepšího jedince než nejlepšího dosud známého, je načase evoluci ukončit. Jiná taková situace může nastat v případě, že se kvalita nejlepšího známého jedince dostatečně přiblížila našim požadavkům, a nemá tedy už cenu v evoluci dále pokračovat.
Tím jsme prošli všechny fáze tvorby GA, od sestavení chromosomu jedince, přes generování počáteční populace jedinců, až po vytváření nových generací této populace za pomoci operací křížení a mutace. Takto popsaný GA je pak více či méně přesným modelem procesu přírodního výběru a můžeme jej, po přizpůsobení stavby jedince konkrétní třídě problémů, použít k řešení různých optimalizačních problémů.

3.4 Speciální genetické algoritmy–The Selfish Gene Algorithm

Dosud popisovaný genetický algoritmus (GA) pracoval na základě principů Darwinovského přírodního výběru, který předpokládá, že pouze ti nejkvalitnější jedinci v rámci populace jsou schopni dlouhodobého přežití a vydobytí si práva dále se rozmnožovat. V takovém algoritmu je pak základní jednotkou selekce jedinec v kontextu populace svého druhu.
Od 60-tých let minulého století se však začaly objevovat i jiné modely evoluce, ve kterých už nebyl základní jednotkou selekce jedinec jako takový, ale byl jí například klan nebo skupina jedinců a v některých případech i celý druh. Kromě zmíněných vyšších úrovní selekce se také uvažovalo o selekci přímo na úrovni genů (viz obr. 3.12) [4].



Obrázek 3.12: Přehled úrovní selekce

3.4.1 Úrovně selekce

Klan je skupina jedinců úzce propojených příbuzenskými vazbami. V takové skupině se předpokládá, že každý její člen bude mít iminentní zájem na tom, aby ostatní členové skupiny přežili. V přírodě může tato snaha nabývat různých podob a nejvíce je patrná u některých druhů hmyzu, které budují kolonie, jako je tomu např. u mravenců, vos nebo včel.
V těchto společenstvech spolu jedinci za účelem přežití navzájem spolupracují tak, že si mezi sebou rozdělí jednotlivé role v rámci kolonie. V důsledku toho pak někteří jedinci zcela přijdou o možnost rozmnožování, a ta tak zůstane pouze několika vybraným jedincům, např. trubcům a královně.
Princip *skupinové selekce* poprvé zformuloval v 60-tých letech minulého století britský zoolog Vero C. Wynne-Edwards [13, 4]. Základní premisou tohoto typu selekce je, že některé geneticky

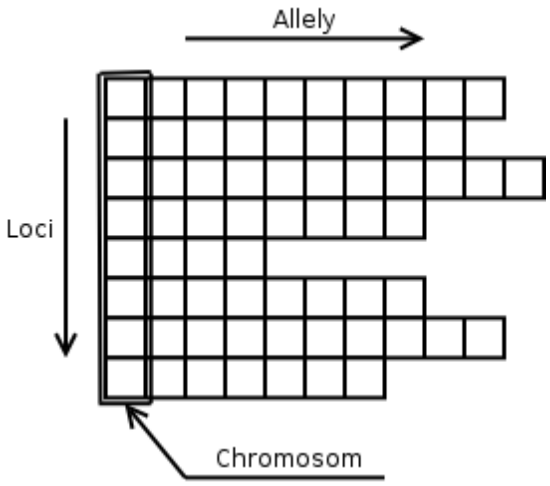
zakódované vlastnosti jedince mají pozitivní přínos pro celou skupinu jedinců koexistujících v určitém prostředí, aniž by nutně představovaly konkurenční výhodu pro jedince jako takového. Tyto vlastnosti jsou pak v průběhu evoluce preferovány, a prospěch celé skupiny je tak postaven nad roveň prospěchu samotných jedinců.
Typickým příkladem takového chování je omezení reprodukční schopnosti jedinců ve skupině, ke kterému dochází v situacích vůči této skupině nepříznivých. V přírodě můžeme takové chování pozorovat například u populací virů nebo parazitů, kteří jsou okolnostmi nuceni zbrzdit tempo své reprodukce, aby tak zcela nezahubili svého hostitele.
V případě *druhové selekce* spolu o přežití soutěží celé druhy a je přitom kladen ještě větší důraz na chování jedinců prospěšné druhu, než je tomu v případě selekce skupinové. To je způsobeno především nutností vyvážit velice silný vliv *sexuální selekce* na úrovni jedinců, jejíž tlak jde v většině případů proti tlaku druhové selekce.
Druhy, které si vyvinuly nějakou formu samo-omezujících se mechanismů, mají v dlouhodobém horizontu obecně menší pravděpodobnost vyhynutí, protože se zpravidla dokáží lépe adaptovat na prostředí, ve kterém se vyskytují.
Současně se zkoumáním stále vyšších a vyšších úrovní selekce se také objevil pokus nahlížet na selekci, jako na proces pracující na té nejnižší možné úrovni–na úrovni samotných genů. Tento přístup rozpracoval ve své knize *„Sobecký gen“* britský biolog Richard Dawkins [14].
Dawkins ve své knize považuje za základní jednotku selekce gen a organismy pak považuje za pouhé kontejnery genů, *genetické stroje* [14], jejichž hlavním účelem je odstínit geny v nich uložené od nepříznivých okolních vlivů. Geny pak v rámci organismu tvoří kolonii, která při pohledu z vnějšku funguje jako jeden celek. Tělo organismu navíc představuje pouze určitou formu rozhraní takové kolonie genů s okolním prostředím a umožňuje často velice sofistikovanou interakci této kolonie s okolím za účelem získání potravy, reprodukce atp.
Na kolonie genů obývajících různé organismy se tak můžeme z určitého úhlu pohledu dívat jako na strojvůdce řídící komplikované stroje, které mezi sebou buď soutěží o přežití, nebo spolu naopak, byť nepřímo, spolupracují.

3.4.2 Virtuální populace

Výše uvedený pohled na selekci vedl k vytvoření varianty GA nazývané *The Selfish Gene Algorithm*, dále jen *SGA*. Tento algoritmus přistupuje k populaci jedinců pouze jako ke skladu s genetickým materálem. Kvalita jednotlivých genů, které se zde vyskytují, pak přímo závisí na kontextu ostatních genů se kterými momentálně tvoří uvažovaný gen kolonii v rámci nějakého organismu [4].
Takto definovaný algoritmus tedy nepracuje přímo s populací jedinců, jako tomu bylo např. v případě binárního GA, ale udržuje si tzv. *virtuální populaci*. Tato je logicky rozčleněna do několika dílčích částí, z nichž každá odpovídá jednomu locusu chromosomu jedinců, resp. organismů, které geny ve skutečnosti obývají. Všechny geny, umístěné v libovolné z částí virtuální populace, jsou tak vlastně allelami z hlediska locusu odpovídajícímu uvažované části populace (viz obr. 3.13).
Další výrazný rozdíl SGA oproti už zmiňovanému binárnímu GA je fakt, že velikost ani skladba populace se v rámci samotného procesu evoluce nijak nemění. Co se však v průběhu evoluce mění je pravděpodobnost, s jakou různé allely v rámci každé části populace obsazují jim odpovídající loci (pl. od locus).

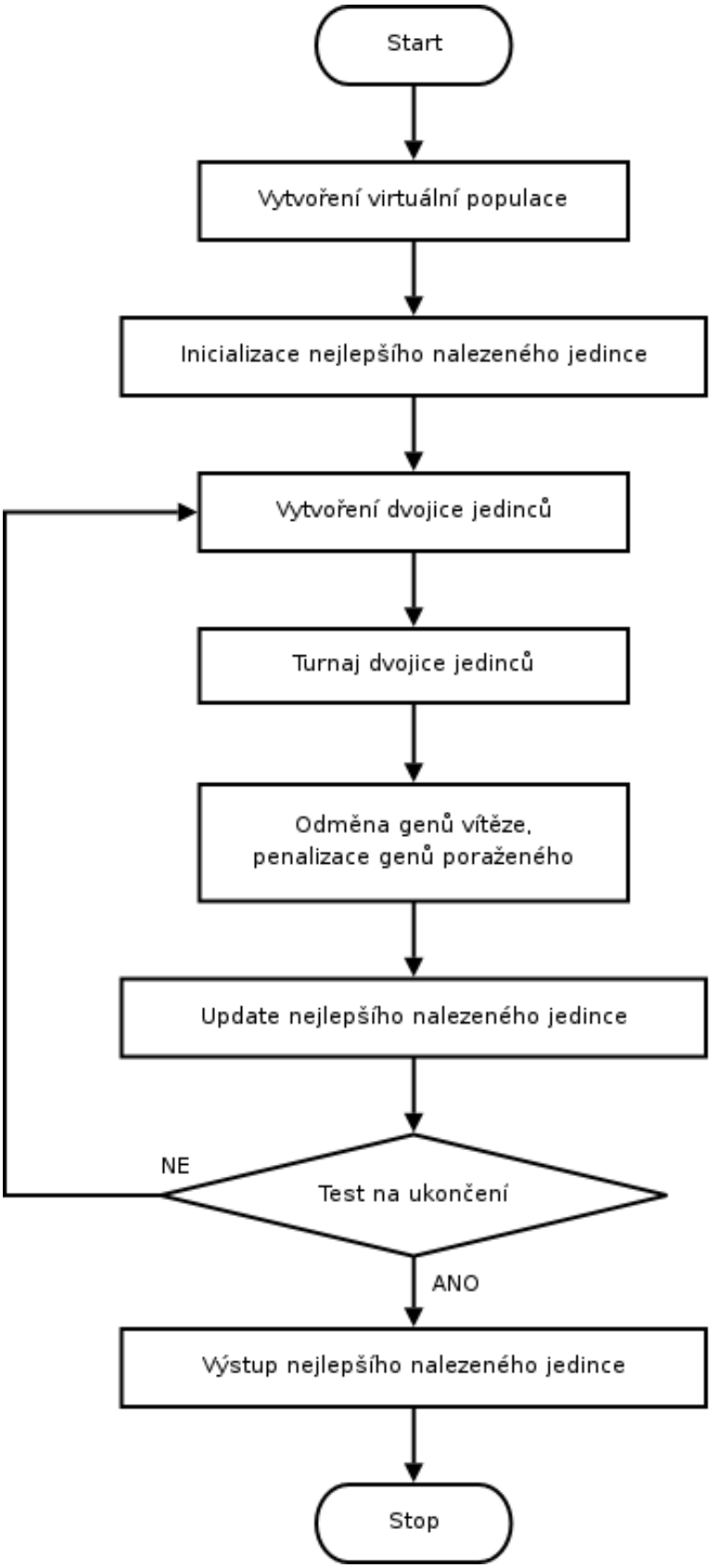
3.4.3 Turnaj

Samotné schéma SGA je velice jednoduché (viz obr. 3.14) a jeho hlavní částí je *turnaj*. Ten se v principu liší od turnaje popsaném v kapitole 3.3.5 tím, že se v něm utkávají vždy pouze dva jedinci. Tito jedinci jsou vytvořeni na základě virtuální populace tak, že dva prázdné chromosomy



Obrázek 3.13: Struktura virtuální populace

jsou obsazeny locus po locusu allelami z částí populace odpovídajícím obsazovaným loci. Allely vítězného jedince jsou pak odměněny tak, že jim zvýšíme pravděpodobnost s jakou budou pro příště obsazovat právě ten locus, který už obsadily v chromosomu vítězného jedince.



Obrázek 3.14: Obecné schéma algoritmu SGA

4 Analýza a návrh řešení

Záměrem této kapitoly je analýza jednotlivých částí zadaného problému a návrh jejich optimálního řešení. Optimalita řešení bude posuzována především z hlediska operační složitosti a chování výsledného algoritmu řešícího zadaný problém.
Naším cílem je navrhnout takový binární genetický algoritmus (GA), který by vyvíjel celulární automaty (CA) použitelné jako generátory testovacích vektorů pro logické obvody. Pro jednoduchost se v našem rozboru omezíme pouze na logické obvody kombinačního charakteru.

4.1 Základní koncepce algoritmu

Námi navrhovaný algoritmus se bude držet koncepce klasického GA, a jeho struktura tedy bude vypadat následujícím způsobem:

1. Vytvoření počáteční populace jedinců obsahující extremisty (viz kap. 4.5)
2. Volba páru jedinců z populace podle principů přírodního výběru
3. Případné zkřížení obou vybraných jedinců
4. Případná mutace obou potomků vzešlých z předchozího kroku
5. Vložení obou potomků upravených v předchozím kroce do nové populace
6. Pokud není nová populace plná, skok na krok 2
7. Nahrazení stávající populace populací novou
8. Pokud je dosaženo vytýčeného cíle, ukončení procesu, jinak skok na krok 2

Kontrolní mechanismus bude zajištěn formou algoritmu pro náhodnou syntézu CA, jehož struktura je definována takto:

1. Nastavení nejlepšího známého jedince na jedince čerstvě vygenerovaného
2. Pokud vypršel povolený časový limit, ukončení procesu
3. Vygenerování nového jedince
4. Pokud je nový jedinec lepší než nejlepší dosud známý jedinec, update nejlepšího známého jedince
5. Skok na krok 2

4.2 Návrh struktury jedince

Jedním z nejdůležitějších aspektů návrhu GA je způsob reprezentace řešení uvažovaného problému, v našem případě pak popisu struktury jednodimenzionálního CA, přesněji LHCA.
V zásadě je možné samotný CA reprezentovat dvěma způsoby:

- linearizovanou *charakteristickou maticí a k ní připojeným seedem.*

  Tento způsob reprezentace vychází z naivní představy, že každé buňce CA nejdříve charakteristickou maticí přesně určíme ty buňky, které budou tvořit její okolí, a pak jí seedem přiřadíme nějaký počáteční stav. Okolí každé z buněk je definováno jedním řádkem matice,

jehož délka odpovídá počtu všech buňek. Do tohoto okolí pak patří všechny buňky, které odpovídají sloupcům obsazeným v takovém řádku jedničkami.

Velice podstatnou nevýhodou této reprezentace je její příliš vysoká paměťová složitost a s ní korelující operační složitost výpočtu kvality jedince, který je postaven na bázi takové reprezentace. Její asymptotická paměťová složitost totiž řádově roste s druhou mocninou počtu buňek.

- *výčtem pravidel*, kterými se jednotlivé buňky řídí, *a k němu připojeným seedem*.

  Rád bych na tomto místě připomněl, že pravidla slouží k popisu chování buňek v rámci CA. Každá buňka má jednoznačně určené okolí, tvořené jejím levým a pravým sousedem a dále pak buňkou samotnou. Pokud navíc známe pravidlo, kterým se konkrétní buňka řídí, jsme také schopni určit z tabulky pravidel nový stav této buňky na základě znalosti tohoto pravidla a stavu buňek v jejím okolí.

  Taková reprezentace má podstatně lepší vlastnosti, než je tomu v předchozím případě. Její asymptotická paměťová složitost je lineární funkcí celkového počtu buňek.

Libovolná z obou výše definovaných variant reprezentace CA může sloužit jako základ struktury jedince, se kterým pak výsledný GA dále pracuje. V rámci této práce jsme se rozhodli vyzkoušet struktury obě. Jak přesně je budeme implementovat rozebereme až v následující kapitole.

4.3 Simulace celulárního automatu

Abychom mohli CA použít pro generování posloupnosti testovacích vektorů, musíme ho být nejdříve schopni vůbec odsimulovat. Způsoby, jakými to můžeme provést, se liší podle použité reprezentace automatu:

- Pokud použijeme *linearizovanou charakteristickou matici doplněnou seedem*, začneme tak, že všechny buňky uvažovaného CA inicializujeme podle seedu. Poté budeme stav buňek transformovat po stanovený počet kroků následujícím postupem:

  Každé buňce CA určíme z charakteristické matice všechny buňky tvořící její okolí a provedeme součet modulo 2 stavů těchto buňek. Tak získáme nový stav každé buňky a současně i jeden testovací vektor.

- I v situaci, kdy uvažovaný CA reprezentujeme *výčtem pravidel doplněným o seed*, bude náš první krok stejný jako předchozím případě. Všechny buňky automatu nastavíme seedem do jejich výchozího stavu, který dále transformujeme po předepsaný počet kroků následujícím způsobem:

  Okolí libovolné buňky CA určíme způsobem naznačeným např. v kapitole 4.2. Na základě stavu buňek v takto definovaném okolí a pravidla, kterým se uvažovaná buňka řídí, určíme z tabulky pravidel nový stav této buňky. Tímto způsobem pak určíme nový stav každé buňce a zároveň obdržíme i jeden testovací vektor.

4.4 Vyhodnocení kvality jedince

Operace vyhodnocení kvality jedince je úhelným kamenem každého GA. Od způsobu, jakým kvalitu jedince určujeme, se odvíjí operační složitost celého algoritmu. Je proto nasnadě, že naším cílem je nalezení co nejefektivnější implementaci této operace.
V našem případě pracujeme s jedinci reprezentujícími CA a takové jedince lze tedy ohodnotit jen na základě vlastností jimi specifikovaných automatů. Ty zjistíme tak, že každý takový automat nejdříve necháme vygenerovat posloupnost testovacích vektorů určité délky, většinou se jedná o

několik málo tisíc vektorů, a takto získanou posloupnost pak dále ohodnotíme jedním ze dvou následujících způsobů:

- *určením diagnostického pokrytí testovaného obvodu*, kdy simulací prostřednictvím simulátoru poruch *fsim* [2] ověříme diagnostické pokrytí testovaného obvodu námi uvažovanou posloupností.

  Takto lze získat nejpřesnější možnou informaci o vhodnosti konkrétního CA pro zamýšlený účel, ale bohužel zato zaplatíme vysokou časovou náročností celého procesu.

- *určením pokrytí předgenerované tabulky vektorů* námi uvažovanou posloupností. Zmíněná tabulka vektorů, získaná za pomoci ATPG nástroje *Atalanta* [15] způsobem popsaným v kapitole 6.2, se zpravidla skládá pouze z vektorů testujících těžko detekovatelné poruchy testovaného obvodu.

  Tyto poruchy jsou detekovatelné pouze a jedině jedním nebo několika málo testovacími vektory. Vzhledem k tak nízkému počtu použitelných vektorů existuje obecně jen velice malá pravděpodobnost, že posloupnost pseudonáhodných testovacích vektorů generovaná CA bude obsahovat mimo jiné právě tyto vektory. Z tohoto důvodu si zvláště ceníme automatů, které jsou schopny takové vektory generovat, a tyto poruchy tak detekovat.

  Tento způsob ohodnocení posloupnosti představuje zpravidla podstatně rychlejší cestu k posouzení vlastností CA, než je poruchová simulace. Vyžaduje však jistou dodatečnou práci spočívající ve vygenerování zmíněné tabulky vektorů.

Rozhodli jsme se implementovat oba výše uvedené způsoby vyhodnocení kvality jedince s tím, že během evoluce jako takové budeme z důvodů menší časové náročnosti používat pokrývání předgenerované tabulky vektorů. U jedinců získaných jako výsledek evoluce na závěr dodatečně určíme diagnostické pokrytí testovaného obvodu pro jimi specifikované CA, abychom si tak mohli udělat lepší představu o reálných vlastnostech takových automatů.

4.5 Vytváření počáteční populace

Při volbě způsobu vytváření počáteční populace jsme sáhli po přístupu popsaném ve stejnojmenné kapitole 3.3.4, a to vložení skupiny *extrémních jedinců* do populace a jejím doplněním jedinci generovanými náhodně. Extrémní jedinci jsou takoví jedinci, v jejichž chromosomu se nevyskytují nuly a jedničky se stejnou pravděpodobností.
Takto definovaní extrémní jedinci jsou speciálním případem jevu v literatuře někdy označovaného jako *polarizace chromosomu* [4]. Kompletně polarizovaný chromosom má každý locus okupován se 100% pravděpodobností právě jedním genem a pravděpodobnost, že locus obsadí některá z allel tohoto genu je nulová. Oproti tomu, ve zcela nepolarizovaném chromosomu může každý locus obsadit libovolná z přípustných allel se stejnou pravděpodobností. V našem případě uvažujeme pouze dvě allely, nulu a jedničku, jejichž pravděpodobnost obsazení určitého locusu je přes celý chromosom konstantní.
Na evoluci populace složené z jedinců obsahujících takové chromosomy je možné nahlížet tak, že na začátku je populace polarizována kolem určité sady více či méně kvalitních řešení, která se v optimálním případě postupem času přemění na řešení ještě kvalitnější. Za takových podmínek se nám celý proces evoluce optimálních, resp. suboptimálních řešení může znatelně urychlit.
Jiný pohled na extrémní jedince je takový, že tito jedinci specifikují CA s rozváženým seedem, které byly popsány v kapitole 3.1.5. Posloupnosti testovacích vektorů generované takovými automaty v některých případech dosahují výrazně lepšího diagnostického pokrytí testovaných obvodů, než posloupnosti generované automaty, jejichž seed je čistě náhodný [4, 6, 3].

4.6 Vhodný typ selekce

Během specifikace problému jsme se rozhodli pro implementaci dvou z nejpoužívanějších metod selekce, a to tzv. *ruletového výběru*, což je speciální případ váženého náhodného párování s výpočtem normalizované kvality jedince (viz kap. 3.3.5), a turnajového výběru. Ruletový výběr představuje ideální volbu z hlediska snadnosti implementace a turnajový výběr je taktéž vhodným kandidátem k implementaci, protože věrně kopíruje proces přírodního výběru.

4.7 Vhodný typ křížení a mutace

Stejně tak jako jsme volili implementaci dvou různých metod selekce, rozhodli jsme se implementovat celkem tři různé typy křížení: jednobodové, dvoubodové a uniformní. Co se mutace týče, k implementaci jsme vybrali mutaci jednobodovou.

4.8 Vytváření nové generace

Novou generaci stávající populace budeme vytvářet tak, že do nové, prázdné populace nejprve převedem ze stávající populace elitu a zbytek doplníme potomky vzešlými pářením jedinců vybraných ze stávající populace jedním z typů selekce zmíněných v kapitole 4.6.

5 Realizace řešení

V této kapitole si stručně rozebereme hlavní části implementace binárního genetického algoritmu (GA) navrženého v předchozí kapitole. Vzhledem k tomu, že se nejedná o práci vyloženě implementačního charakteru, omezíme se jen na slovní popis nejdůležitějších aspektů té které části implementace.
Závěrem této kapitoly naznačíme princip implementace plně randomizovaného přístupu, který budeme v rámci testování výsledné implementace GA používat k získání představy o použitelnosti GA pro řešení zadaného problému.

5.1 Základní koncepce programu

Výsledný program implementující navržený GA pracuje buď v dávkovém módu, nebo v módu interaktivním, kdy s uživatelem komunikuje prostřednictvím jednoduchého uživatelského rozhraní typu CLI přes terminál.
V dávkovém módu program očekává všechny vstupní údaje specifikované formou hodnot příslušných parametrů. Pokud však nějaký údaj schází, program se na něj uživatele přímo dotáže a základním způsobem přitom kontroluje typ zadávané informace.
Při práci v interaktivním módu program funguje následovně:

1. Po svém spuštění si program od uživatele vyžádá cestu k souborům obsahujícím předgenerovanou tabulku vektorů a popis struktury testovaného obvodu ve formátu ISCAS-89.

   Uživatel je také požádán o cestu k souboru, kam má uložit statistická data získaná v průběhu měření. Ta se skládají z průměrných hodnot kvality populace, kvality nejlepšího jedince v rámci populace, diversity populace a konečně pak skutečného podílu elity v populaci pro každou odsimulovanou generaci.

2. Program se uživatele dále dotáže na parametry procesu evoluce CA, mezi které patří například typ jedince, velikost populace, podíl elity a extrémních jedinců v populaci, typ selekce a křížení atd.

3. Následuje samotný proces evoluce CA podle zadaných parametrů a po jeho skončení pak stejně dlouhý běh části programu implementující plně randomizovaný přístup, který slouží jako měřítko výkonnosti procesu evoluce CA (viz kap. 5.8).

   Tento krok je pak prováděn po předem stanovený počet opakování, a takto získané výsledky jsou po zprůměrování uloženy do souboru zadaného uživatelem v kroku 2. Na terminál je navíc vytištěn stručný přehled dosažených výsledků.

5.2 Implementace struktury jedince

V rámci předběžné přípravy jsme implementovali obě varianty struktury jedince zmíněné v kapitole 4.2 a prakticky tak ověřili, že implementace první varianty má výrazně vyšší operační složitost. Z tohoto důvodu jsme se v další práci rozhodli soustředit na variantu druhou, založenou na reprezentaci celulárního automatu (CA) výčtem pravidel popisujících chování jeho buněk doplněným o seed. Taková reprezentace automatu přímo vybízí k implementaci chromosomu na této reprezentaci založeného ve formě jednorozměrného pole pevné délky.
Každý prvek takového pole pak odpovídá jednomu páru nukleotidů reálného chromosomu a několik prvků dohromady tvoří samotný gen. Pro všechny prvky také platí, že mohou nabývat pouze dvou hodnot: nuly a jedničky.

Takto definované pole je možné logicky rozdělit na dvě samostatné, na sebe navazující části (viz obr. 5.1):

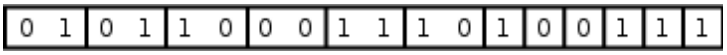
1. *oblast genů* skládajících se z několika prvků pole, *jejímž fenotypem je samotná struktura CA*.

   Každý gen z této oblasti kóduje číslo pravidla, kterým se řídí jemu odpovídající buňka CA, v tabulce všech dostupných pravidel. Toto číslo je pak reprezentováno v binárním kódu posloupností prvků tvořících uvažovaný gen.

   V našem případě jsme použili celkem čtyři různá pravidla: 60, 90, 150 a 240 (viz tab. 3.1). Díky tomu jsou tyto geny tvořeny pouze dvěma prvky, které plně postačují na zakódování čísla pravidla.

2. *oblast genů kódujících seed CA*.

   Geny z této oblasti jsou tvořeny pouze jedním prvkem pole a hodnota těchto genů přímo určuje počáteční stav jim odpovídajících buňek CA.



Obrázek 5.1: Implementace struktury jedince

Výše popsaná implementace chromosomu je, společně s operacemi nad takovým chromosomem definovanými, základem samotné implementace výsledného jedince.

5.3 Vyhodnocení kvality jedince

Námi implementovaný GA používá pro výpočet kvality jedinců jak diagnostické pokrytí testovaného obvodu, tak pokrytí předgenerované tabulky vektorů jimi specifikovanými CA:

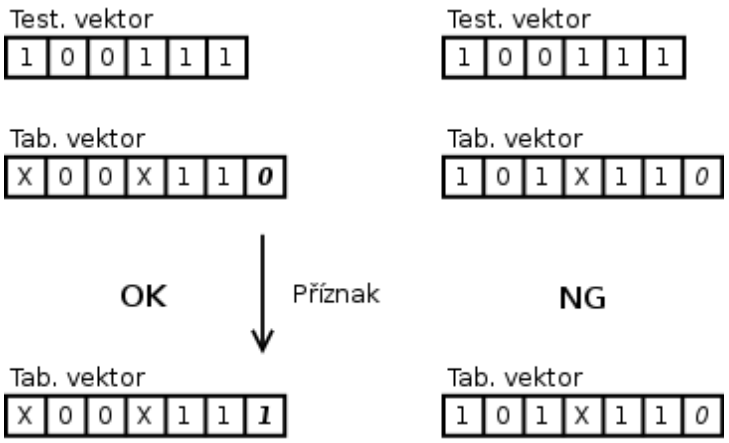
- *Diagnostické pokrytí testovaného obvodu* určíme za pomoci externího simulátoru poruch *fsim*, který dokáže simulovat poruchy kombinačních logických obvodů.

  V první fázi nejdříve vygenerujeme stanovený počet testovacích vektorů vyhodnocovaným CA a ty uložíme do souboru. Ten pak ve druhé fázi předáme simulátoru a počkáme si na výsledek.

- *Předgenerovaná tabulka vektorů* se zpravidla skládá pouze z vektorů testujících těžko detekovatelné poruchy testovaného obvodu (viz kap. 4.4). Pokrytí této tabulky implementujeme tak, že z vyhodnocovaného CA odebíráme postupně jeden testovací vektor za druhým a zkoušíme jimi postupně pokrývat dosud nepokryté vektory v tabulce.

  Samotné pokrytí vektoru je ve své podstatě ekvivalentní operaci porovnání dvou řetězců při kterém uvažujeme tzv. *don't-care znaky* s tím rozdílem, že v případě shody obou vektorů si tuto informaci uložíme ve formě příznaku bezprostředně připojeného k vektorům v tabulce (viz obr. 5.2).

  Jakmile tímto způsobem zpracujeme všechny vektory odebrané z automatu, spočítame si jako výsledek percentuální podíl počtu pokrytých vektorů ku celkovému počtu vektorů v tabulce.



Obrázek 5.2: Příklad pokrytí vektoru v tabulce

5.4 Vytváření počáteční populace

Během generování chromosomu jedinců počáteční populace rozlišujeme mezi normálními jedinci a jedinci extrémními. Rozdíl spočívá v pravděpodobnosti s jakou se v jejich chromosomech vyskytují jedničky. U normálních jedinců je tato pravděpodobnost stejná jak pro výskyt jedničky, tak pro výskyt nuly, a je rovna 50%. U extrémních jedinců je pravděpodobnost výskytu jedničky specifikována uživatelem při spouštění programu.
Při samotném vytváření počáteční populace pak postupujeme tak, že do prázdné populace nejprve vložíme uživatelem specifikovaný počet extrémních jedinců na náhodně zvolená místa v populaci. Zbývající prázdné místo v populaci pak vyplníme náhodně vygenerovanými normálními jedinci.

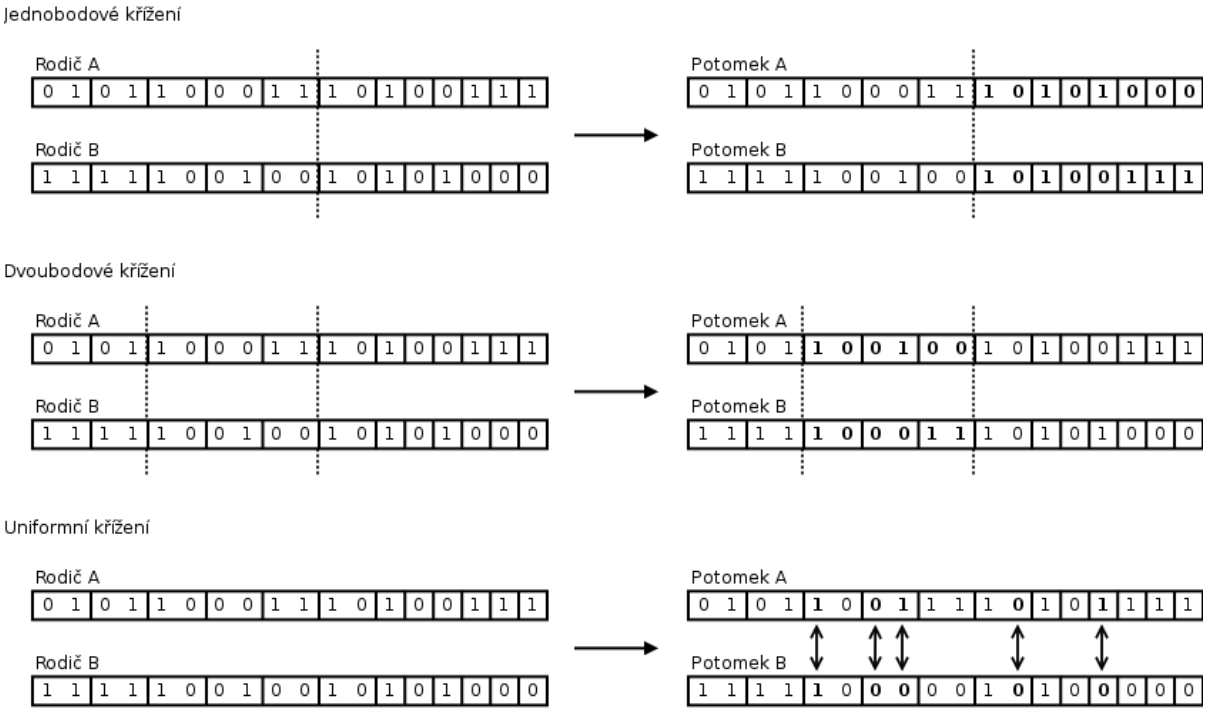
5.5 Implementace křížení a mutace

Jak jsme již v předchozí kapitole naznačili, implementovali jsme celkem tři varianty křížení: jednobodové, dvoubodové a uniformní (viz obr. 5.3). Všechny tři se pak chovají přesně tak, jak bylo popsáno v kapitole 3.3.6. Vzhledem k námi použité implementaci chromosomu jsme navíc do těchto operací zabudovali mechanismus, který zajišťuje, aby bod dělení nerozbil gen skládající se z více prvků. Takové geny obsahuje oblast chromosomu, která kóduje strukturu CA.
Tento mechanismus pak pracuje tak, že náhodně zvolený bod dělení představuje index prvku v poli implementujícím chromosom. Tomuto indexu se pak vynuluje počet nejméně významných bitů, který je roven horní celé části dvojkového logaritmu počtu dostupných pravidel. Tak získáme index prvního prvku genu určeného bodem dělení a můžeme s genem dále pracovat jako s celkem.
Jednobodovou mutaci jsme implementovali tak, že pracuje přímo na úrovni základních stavebních bloků genu, tj. prvků pole implementujícího chromosom, kterým pak invertuje hodnotu podle obrázku 5.4.

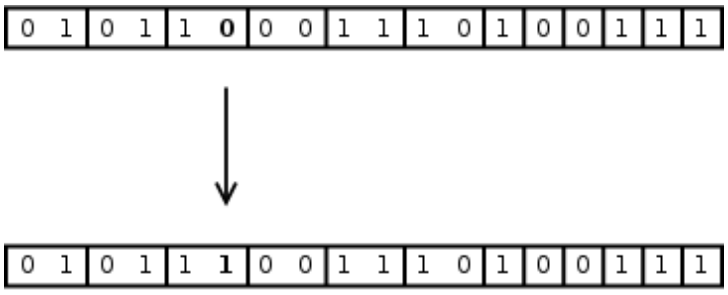
5.6 Vytváření nové generace

Novou generaci vytváříme ve dvou fázích, při kterých používáme pomocnou populaci:

1. Vytvoříme prázdnou pomocnou populaci a okopírujeme do ní ze stávající populace elitu, kterou pro jednoduchost její další manipulace umisťujeme vždy na začátek pomocné popu-



Obrázek 5.3: Příklad implementace křížení



Obrázek 5.4: Příklad implementace mutace

lace. Zbytek této populace doplníme ze stávající populace uživatelsky zvolenou metodou selekce a přitom předpokládáme, že náhodný výběr jedinců v rámci selekce nám zajistí dostatečné promíchání pomocné populace.

Samotný výběr jedinců ze stávající populace je během selekce prováděn bez náhrady. To znamená, že jedinec už jednou vybraný za kandidáta k páření nemůže být až do odvolání znovu vybrán a zůstává tak nadále pasivní. Populace pak implementuje funkci, která umožní všechny jedince znovu naráz aktivovat.

Takové chování selekce nám umožňuje implementovat turnajový výběr, který se skládá z jednotlivých *turnajových kol*.

V rámci jednoho kola turnaje vybíráme ze stávající populace jedince ke klání tak dlouho, až už nemáme k dispozici žádné aktivní jedince nebo je pomocná populace zcela zaplněná. V prvním případě pak za použití aktivační funkce zmíněné v předchozím odstavci provedeme reaktivaci celé populace a zahájíme nové kolo turnaje.

Důležitým důsledkem takto definovaného způsobu selekce je fakt, že velikost klání, tj. počet jedinců do něj vstupujících, může nepřímo ovlivnit selekční tlak na propagaci těch nejkvalitnějších jedinců v populaci do její další generace.

Čím více jedincům totiž povolíme vstup do každého klání, tím méně klání se může v rámci jednoho kola turnaje uskutečnit. Výsledkem bude pochopitelně menší počet vítězů na jedno kolo, a počet kol nutných k doplnění pomocné populace tak stoupne. Protože se každý jedinec může v rámci jednoho kola zúčastnit nejvýše jednoho klání, je toto zvýšení počtu kol výhodné především pro elitní jedince, jejichž pravděpodobnost výběru do klání a nakonec i postupu do pomocné populace je velice vysoká.

Získáme tak další prostředek, kterým můžeme řídit vliv elity na populaci a zabránit tomu, aby elita v populaci převládla společně se svými potomky.

2. Z pomocné populace, kterou jsem naplnili v předešlé fázi, přeneseme do nové populace celou elitu. V pomocné populaci nám tak zůstane přesně tolik rodičů, kolik potomků je potřeba pro doplnění nové populace. Nyní pomocí genetických operací postupně, po dvojicích sousedů, spáříme zbylé jedince z pomocné populace a vložením jejich potomků do nové populace získáme další generaci stávající populace.

   Páření po dvojicích sousedů jsme zvolili pro nižší operační složitost jeho implementace a předpokládáme přitom, jak už bylo zmíněno v předchozím bodě, že pomocná populace je už dostatečně promíchána z první fáze, a tento způsob páření by tedy neměl mít žádné negativní dopady na vývoj populace jako takové.

   Jeden z možných pohledů na tuto fázi je takový, že namísto abychom přímo spářili vítěze dvou po sobě jdoucích klání, případně náhodně vybraného jedince a jeho partnera vzešlého z vítězného klání, uložíme si oba takové jedince pro pozdější zpracování do pomocné populace. Takový přístup nám potom snadno umožní implementovat výše zmíněný vícekolový turnaj a důsledné oddělení fáze selekce od fáze páření nám dává větší volnost při volbě konkrétní implementace obou fází.

5.7 Určování diversity populace

Funkce pro určování diversity populace slouží ke dvěma účelům. Jednak zjišťuje samotnou diversitu populace a jednak počítá skutečný podíl elity v populaci.
Diversitu populace zjišťujeme tak, že si nejdříve vytvoříme seznam všech jedinců v populaci. Tento seznam pak od začátku postupně probíráme, jedince po jedinci, přičemž každého vybraného jedince započítáme a pak ho ze seznamu odstraníme spolu se všemi jeho kopiemi. Po

zpracování celého seznamu nám tak zůstane počet unikátních jedinců v rámci celé populace, na jehož základě pak stanovíme diversitu populace jako percentuální podíl počtu unikátních jedinců ku počtu všech jedinců populace.
Za kopii jedince považujeme každého jedince, který má s uvažovaným jedincem zcela shodný chromosom. Tyto kopie nejčastěji vznikají pokud při páření dvou elitních jedinců nedojde ke křížení ani k mutaci. Takto vytvořené kopie pak přejdou do další generace po boku svých rodičů.
Určení *skutečného podílu elity v populaci* probíhá v podstatě stejným způsobem jako v případě zjišťování diversity populace, ale při vyřazování jedinců ze seznamu počítáme pouze elitní jedince, a to včetně všech jejich kopií. Podíl elity v populaci pak počítáme obdobně jako diversitu populace.

5.8 Plně randomizovaný přístup

Jako měřítko chování naší implementace GA nám slouží algoritmus řešící identický problém náhodným prohledáváním svého stavového prostoru.
Jinými slovy, necháme náhodně generovat jedince po dobu, kterou trvala evoluce a udržujeme si přitom dosud nejlepšího takto nalezeného jedince. Jím specifikovaný CA pak po vypršení časového limitu prohlásíme za řešení problému.

5.9 Zvolený jazyk implementace

K implementaci jsme zvolili ANSI/ISO C++ z důvodů dobré výkonnosti, relativně snadné přenositelnosti a možnosti případné další paralelizace v rámci systému MPI [16].

6 Testování řešení

V této kapitole se zabýváme experimentálním vyhodnocením implementace námi navrženého binárního genetického algoritmu (GA), použitého pro návrh celulárních automatů (CA) vhodných jako generátory testovacích vektorů pro kombinační logické obvody.
Jako měřítko výkonnosti výsledné implementace GA nám pak slouží plně randomizovaný přístup popsaný v kapitole 5.8.

6.1 Metodika testování

V rámci měření jsme pro řadu testovaných kombinačních obvodů provedli srovnání obou metod z hlediska kvality jimi produkovaných CA. Obě metody, tj. GA a randomizovaný přístup, jsme vždy nechali pracovat stejně dlouho. Pro každý testovaný obvod jsme provedli sérii měření vlivu několika vybraných parametrů na kvalitu výsledných CA.
Postup měření pro jednu hodnotu měřeného parametru se dá shrnout následujícím způsobem:

1. Volba hodnoty měřeného parametru
2. Nastavení hodnot ostatních parametrů na výchozí hodnoty (viz níže uvedené kapitoly)
3. Běh evoluce; na závěr vyhodnocení získaného CA poruchovou simulací
4. Běh randomizovaného přístupu po stejnou dobu, jako trval běh evoluce; na závěr vyhodnocení získaného CA poruchovou simulací
5. Kroky 3 a 4 opakovat celkem 5x; výsledky z kroků 3 a 4 průběžně průměrovat a takto získané průměrné hodnoty výsledků na závěr uložit do souboru

Výše uvedeným způsobem jsme měřili tyto parametry (v závorce je uveden seznam měřených hodnot, kurzívou jsou pak vyznačeny *výchozí hodnoty*):

- Podíl extremistů v populaci (*50%*, 60%, 70%, 80%, 90%)
- Polarizace chromosomu extremistů (*5%*, 95%)
- Typ selekce (*Turnajový výběr*, Ruletový výběr)
- Typ křížení (Jednobodové křížení, Dvoubodové křížení, *Uniformní křížení*)

Ostatní parametry jsme pak zafixovali na následujících hodnotách (viz níže uvedené kapitoly):

- Velikost populace: 150
- Podíl elity v populaci: 1%
- Počet jedinců vstupujících do klání: 2
- Pravděpodobnost křížení: 70%
- Pravděpodobnost mutace: 10%
- Maximální počet generací: 1500

Testované obvody jsme vybírali jednak s ohledem na průměrný počet pseudonáhodných testovacích vektorů, které jsou k jejich úplnému otestování potřeba (viz. [3]), a jednak podle počtu jejich vstupů, který určuje velikost potřebného CA, a tedy i časovou náročnost samotného měření. Pro obvody s několika stovkami vstupů, jako je např. c2670, se časová náročnost měření pohybuje v řádu týdnů procesorového času.
Při měření jsme použili testované obvody uvedené v tabulce 6.1. V případě sekvenčních obvodů jsme pak testovali jejich full-scan verze. Generovanými vektory se v tabulce rozumí počet testovacích vektorů generovaných CA při vyhodnocování kvality jim odpovídajících jedinců.

Testovaný obvod	**Počet vstupů**	**Potřebné vektory**	**Generované vektory**
c3540	50	32 K	2 K
c880	60	13 K	500
c2670	233	4.4 M	2 K
s386	13	3600	300
s420	34	1.4 M	1 K
s953	45	46 K	1 K
s713	54	1 M	1 K
s838	67	až 100 M	2 K
b07	50	3 M	500

Tabulka 6.1: Tabulka testovaných obvodů

6.2 Generování tabulky vektorů

Tabulku vektorů detekujících těžko detekovatelné poruchy (viz kap. 4.4) jsme pro každý testovaný obvod sestavili tak, že jsme tento obvod nejdříve otestovali posloupností testovacích vektorů z náhodně sestaveného LFSR za pomoci ATPG nástroje Atalanta [15]. Tímto nástrojem jsme pak určili poruchy nepokryté takovou posloupností a pro tyto poruchy jsme, opět pomocí Atalanty, vygenerovali příslušné testovací vektory.
Shora uvedeným postupem jsme tedy získali tabulku testovacích vektorů, které detekují poruchy nepokryté posloupností testovacích vektorů ze zcela náhodně vytvořeného LFSR. Výsledná velikost tabulky pak přímo závisela na délce zmíněné posloupnosti. Na základě výsledků v [17] bylo naším cílem získat tabulku o cca. 900 vektorech, které se však ne vždy podařilo získat.
To se zpravidla stávalo v případech jednodušších obvodů, např. u s386, u kterých byla délka úplných testů menší, než požadovaná velikost tabulky. V takových případech jsme pak použili největší možnou tabulku, jejíž velikost byla stále ještě menší než délka úplného testu takového obvodu.
U obvodu b07 jsme pak zjišťovali vliv velikosti tabulky na kvalitu získaných CA, takže jsme velikost volili v pokud možno pravidelných rozestupech.

6.3 Vhodný druh polarizace

Obecně platí, že pro vetšinu obvodů dávají dobré výsledky CA s jednoduchou strukturou a seedem obsahujícím pouze několik málo jedniček [4]. Takové automaty tedy můžeme považovat za slibná počáteční řešení. Na základě tohoto předpokladu jsme zvolili standardní hodnotu pravděpodobnosti obsazení locusu jedničkou u extrémních jedinců jako 5% a pro úplnost také provedeme některá měření s touto pravděpodobností rovnou 95%.
Podíl extrémních jedinců v populaci se pak bude během měření pohybovat od standardních 50% až do 90%, vždy v 10% skocích, abychom tak zjistili vliv velikosti tohoto podílu na průběh

evoluce samotné populace.

6.4 Vhodné parametry selekce

V případě turnajového výběru jsme se přiklonili k omezení počtu účastníků klání na dva, abychom tak potlačili vliv elity, která zpočátku představuje 1% jedinců v populaci. Důvodem pro takové rozhodnutí bylo zjištění v rámci předběžných měření, že elita má při nárůstu jejího podílu v populaci nad 5% velice silnou tendenci celou populaci záhy ovládnout. Pokud bychom to dopustili, přišli bychom tím takřka úplně o variabilitu populace, která by tak rychle zkonvergovala k lokálním extrémům.

6.5 Vhodné parametry křížení a mutace

V průběhu měření budeme standardně používat křížení uniformní, abychom tak co nejdéle zachovali počáteční variabilitu populace. Pravděpodobnost, že ke zkřížení vybraných jedinců skutečně dojde nastavíme na 70% [18, 7].
Pravděpodobnost jednobodové mutace potomků vzešlých z křížení dvou jedinců pak zafixujeme na 10%. Tuto poměrně vysokou hodnotu jsme se rozhodli zvolit z důvodu velice rychlé degenerace populace, kterou jsme pozorovali v předběžných měřeních s nižšími hodnotami tohoto parametru.

6.6 Ostatní parametry evoluce

Ostatními parametry evoluce rozumíme především velikost populace a délku evoluce. Způsobu jakým co nejlépe stanovit jejich hodnotu jsme se dotkli už v kapitole 3.3.10.
V rámci předběžných měření jsme zjistili, že desetinásobné zvětšení populace, konkrétně z 50 na 500 jedinců, nám typicky přineslo nárůst diagnostického pokrytí benchmarku pro CA specifikované výslednými jedinci pouze o necelé 1%. Na základě tohoto poznatku jsme pak velikost populace omezili na 150 jedinců.
To nám stále ještě umožňuje pracovat s dostatečně rozmanitou populací, aniž bychom však genetický algoritmus (GA) přespříliš zahltili nadměrným počtem jedinců, které je třeba s každou další generací znovu a znovu vyhodnocovat.
Při stanovení délky evoluce jsme se rozhodli horní mez počtu generací, po které evoluce probíhá, implementovat dynamicky tak, že pokud nedokáže výsledný GA během 500 generací najít řešení lepší než to nejlepší dosud známé, evoluce bude automaticky ukončena.
Tak zabráníme plýtvání časem v případě, že se populace stala v podstatě nehybnou, ať už v důsledku degenerace nebo emergence optimálního jedince, a zároveň neznemožníme čile se měnící populaci další vývoj.

6.7 Naměřené výsledky

V níže uvedených tabulkách jsou shrnuty výsledky GA s různým nastavením ve smyslu kvality nejlepších takto nalezených jedinců (pro použité způsoby vyhodnocování viz legendu v kap. 6.7.1), včetně doby potřebné k evoluci těchto jedinců. Pro srovnání jsou pak v tabulkách uvedeny výsledky, kterých dosahoval za stejnou dobu plně randomizovaný přístup.
Všechny výsledky jsou uváděny v procentech maximální kvality jedince, tj. úplného pokrytí předgenerované tabulky vektorů pro daný testovaný obvod, resp. úplného diagnostického pokrytí testovaného obvodu získaného poruchovou simulací.

6.7.1 Tabulky naměřených výsledků pro testované obvody

Legenda k tabulkám naměřených výsledků:

Evoluce (tab.) Evoluce s vyhodnocováním kvality jedinců pokrýváním předgenerované tabulky vektorů

Evoluce (sim.) Evoluce s vyhodnocováním kvality jedinců poruchovou simulací

Rand. přístup (tab.) Randomizovaný přístup s vyhodnocováním kvality jedinců pokrýváním předgenerované tabulky vektorů

Rand. přístup (sim.) Randomizovaný přístup s vyhodnocováním kvality jedinců poruchovou simulací

Std. Standardní nastavení GA

Extr. Podíl extremistů v populaci

Polar. Polarizace chromosomu extremistů

Ruleta Ruletový výběr

1-B kříž. Jednobodové křížení

2-B kříž. Dvoubodové křížení

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	60.77	60.51	60.57	60.33	61.13	60.41	60.72	60.077	60.13
Evoluce (sim.)	95.56	95.58	95.55	95.51	95.58	95.62	95.54	95.58	95.58
Rand. přístup (tab.)	60.57	60.57	60.72	60.57	60.75	60.67	60.64	60.57	60.92
Rand. přístup (sim.)	95.54	95.62	95.60	95.55	95.64	95.59	95.64	95.53	95.62
Prům. čas [s]	3972	4084	4482	4516	4918	3630	4719	3474	3266

Tabulka 6.2: Výsledky měření benchmarku c3540 pro tabulku 779 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	37.84	39.30	39.30	39.30	38.64	36.43	36.78	37.84	37.39
Evoluce (sim.)	97.58	97.43	97.71	97.67	97.64	97.71	97.83	97.58	97.83
Rand. přístup (tab.)	36.48	36.28	35.98	36.080	36.58	35.58	36.080	36.48	36.13
Rand. přístup (sim.)	97.18	97.35	97.30	97.37	97.43	97.47	97.58	97.18	97.72
Prům. čas [s]	1220	1477	1536	1509	1431	1254	1508	991	960

Tabulka 6.3: Výsledky měření benchmarku c880 pro tabulku 398 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	12.14	11.68	12.066	11.86	11.94	11.89	11.84	11.30	11.84
Evoluce (sim.)	84.35	84.35	84.32	84.37	84.31	84.35	84.27	84.36	84.38
Rand. přístup (tab.)	11.76	11.76	11.91	11.71	11.81	11.81	11.68	11.71	11.68
Rand. přístup (sim.)	84.30	84.33	84.27	84.35	84.33	84.34	84.40	84.30	84.35
Prům. čas [s]	19968	19485	15363	17204	18007	12468	26432	9768	15431

Tabulka 6.4: Výsledky měření benchmarku c2670 pro tabulku 784 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	87.25	86.10	85.63	85.83	85.49	87.12	87.73	84.68	85.56
Evoluce (sim.)	94.010	92.92	92.40	92.87	91.77	93.91	93.54	92.14	92.97
Rand. přístup (tab.)	87.12	86.98	86.51	86.92	86.85	87.32	86.92	87.39	86.92
Rand. přístup (sim.)	93.49	93.59	93.54	93.59	93.65	93.70	93.80	93.80	93.70
Prům. čas [s]	255	300	322	314	335	249	651	261	225

Tabulka 6.5: Výsledky měření benchmarku s386 pro tabulku 295 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	85.051	85.32	85.12	85.32	85.59	79.66	80.74	84.65	84.92
Evoluce (sim.)	97.21	97.35	97.26	96.84	97.49	93.91	94.56	96.79	97.21
Rand. přístup (tab.)	80.067	79.87	80.00	80.34	80.067	80.00	79.93	80.74	80.067
Rand. přístup (sim.)	94.093	94.047	93.81	95.26	93.77	93.77	92.98	94.23	93.58
Prům. čas [s]	723	1024	1212	958	950	730	976	641	674

Tabulka 6.6: Výsledky měření benchmarku s420 pro tabulku 297 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	78.44	78.64	79.25	79.026	78.025	77.64	79.72	77.41	77.97
Evoluce (sim.)	93.62	94.050	94.12	94.14	93.75	94.68	94.24	94.087	94.27
Rand. přístup (tab.)	78.78	78.89	78.92	78.80	78.72	78.92	78.75	78.44	78.58
Rand. přístup (sim.)	93.62	94.31	94.61	94.35	93.77	94.25	94.42	93.75	93.72
Prům. čas [s]	2465	2903	2644	2537	2603	2277	4323	2038	2255

Tabulka 6.7: Výsledky měření benchmarku s953 pro tabulku 719 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	88.11	87.76	87.97	87.97	87.62	88.11	88.11	87.19	87.26
Evoluce (sim.)	91.12	90.91	91.15	90.67	90.88	90.81	91.26	90.74	90.81
Rand. přístup (tab.)	88.11	88.26	88.11	88.043	88.19	87.90	88.11	88.11	88.043
Rand. přístup (sim.)	90.98	90.84	91.15	90.81	91.015	90.91	91.26	91.015	90.95
Prům. čas [s]	608	848	820	868	628	592	1101	529	536

Tabulka 6.8: Výsledky měření benchmarku s713 pro tabulku 281 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	79.85	82.91	83.062	85.27	84.92	74.15	77.75	83.41	84.34
Evoluce (sim.)	93.56	95.45	95.80	97.90	97.50	86.95	91.23	96.48	97.11
Rand. přístup (tab.)	74.42	74.38	74.38	74.23	74.11	74.26	74.42	74.30	74.19
Rand. přístup (sim.)	87.70	88.098	88.098	87.56	87.42	87.40	87.40	87.96	87.35
Prům. čas [s]	3330	5172	4952	3354	3192	4026	5305	3563	3977

Tabulka 6.9: Výsledky měření benchmarku s838 pro tabulku 516 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	17.69	22.31	23.85	20.00	27.69	16.92	12.31	13.077	16.15
Evoluce (sim.)	93.91	94.60	94.77	93.87	95.18	94.065	93.76	93.75	94.67
Rand. přístup (tab.)	20.00	19.23	19.23	22.31	20.77	18.46	18.46	19.23	19.23
Rand. přístup (sim.)	93.71	94.62	94.69	94.19	94.81	94.048	93.59	94.53	94.42
Prům. čas [s]	147	173	166	184	159	124	126	111	112

Tabulka 6.10: Výsledky měření benchmarku b07 pro tabulku 26 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	23.33	25.42	30.83	30.42	32.083	24.17	23.75	28.75	29.58
Evoluce (sim.)	93.45	95.32	95.91	96.00	96.00	94.95	95.34	93.91	95.73
Rand. přístup (tab.)	22.92	21.17	22.50	22.92	21.25	22.50	21.25	23.75	21.67
Rand. přístup (sim.)	95.32	95.0042	94.95	94.97	94.77	94.99	94.69	94.81	95.0046
Prům. čas [s]	193	266	251	252	236	213	200	229	159

Tabulka 6.11: Výsledky měření benchmarku b07 pro tabulku 48 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	30.99	36.056	33.52	38.028	39.44	22.54	23.38	32.39	35.49
Evoluce (sim.)	94.33	95.64	91.16	94.44	92.81	95.22	95.54	90.080	94.51
Rand. přístup (tab.)	23.66	23.38	26.76	23.94	22.82	23.10	23.66	23.94	23.94
Rand. přístup (sim.)	95.39	94.95	95.29	95.29	95.075	94.90	95.18	95.0044	95.093
Prům. čas [s]	258	246	333	308	302	271	388	252	239

Tabulka 6.12: Výsledky měření benchmarku b07 pro tabulku 71 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	39.80	41.37	41.18	45.49	42.55	32.16	36.27	39.22	38.82
Evoluce (sim.)	96.085	94.97	93.48	90.72	90.22	95.57	94.030	89.14	94.26
Rand. přístup (tab.)	32.35	33.53	32.75	32.55	33.73	32.94	33.14	33.33	32.75
Rand. přístup (sim.)	95.52	95.62	95.27	95.39	95.50	95.52	95.32	95.22	95.62
Prům. čas [s]	352	429	401	381	401	346	394	274	265

Tabulka 6.13: Výsledky měření benchmarku b07 pro tabulku 102 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	61.11	59.68	62.22	62.064	61.35	58.81	59.52	58.81	60.24
Evoluce (sim.)	95.50	94.95	95.71	95.77	96.014	95.022	95.52	95.075	95.75
Rand. přístup (tab.)	59.29	58.97	58.65	58.97	59.048	59.29	59.52	58.41	59.21
Rand. přístup (sim.)	95.38	95.0044	95.25	94.79	95.70	94.92	95.70	95.38	95.29
Prům. čas [s]	637	817	775	849	722	612	868	581	562

Tabulka 6.14: Výsledky měření benchmarku b07 pro tabulku 252 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	63.99	64.71	67.00	66.77	66.88	63.76	64.83	65.44	64.75
Evoluce (sim.)	93.69	95.43	95.75	95.55	95.48	94.63	95.15	94.95	95.093
Rand. přístup (tab.)	64.60	64.30	64.15	64.60	63.76	63.34	64.26	63.76	64.49
Rand. přístup (sim.)	94.97	94.90	94.44	95.38	94.95	95.32	94.85	95.25	95.11
Prům. čas [s]	1257	1455	1484	1418	1358	1139	1769	1099	978

Tabulka 6.15: Výsledky měření benchmarku b07 pro tabulku 526 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	72.86	73.73	70.89	74.57	73.90	70.08	70.44	71.99	72.65
Evoluce (sim.)	95.13	95.41	94.83	95.47	95.50	94.53	94.93	94.83	94.92
Rand. přístup (tab.)	70.61	70.49	70.79	70.49	70.64	70.20	70.54	70.87	70.38
Rand. přístup (sim.)	95.25	95.32	94.81	94.93	95.15	95.38	90.51	94.42	94.62
Prům. čas [s]	1606	1913	1817	1758	1749	1445	1688	1334	1273

Tabulka 6.16: Výsledky měření benchmarku b07 pro tabulku 784 vektorů

Měření	**Std.**	**Extr. 60%**	**Extr. 70%**	**Extr. 80%**	**Extr. 90%**	**Polar. 95%**	**Ruleta**	**1-B kříž.**	**2-B kříž.**
Evoluce (tab.)	73.65	74.75	74.91	75.77	76.039	72.11	72.67	73.58	73.95
Evoluce (sim.)	95.61	95.29	95.77	95.78	95.94	94.79	94.95	95.54	94.95
Rand. přístup (tab.)	72.42	72.62	72.69	72.83	72.50	72.28	72.65	72.65	72.48
Rand. přístup (sim.)	94.048	94.62	94.92	94.72	95.15	94.58	94.99	95.057	94.56
Prům. čas [s]	1982	1688	2439	2051	2140	1755	2228	1753	1931

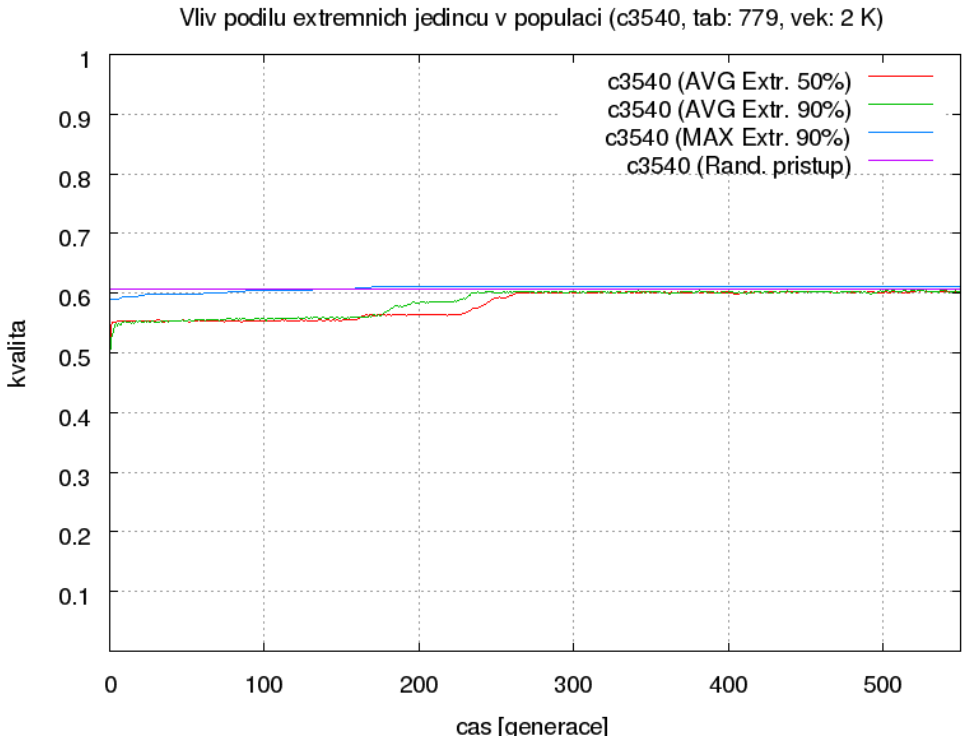
Tabulka 6.17: Výsledky měření benchmarku b07 pro tabulku 1021 vektorů

6.7.2 Výsledky pro obvod c3540

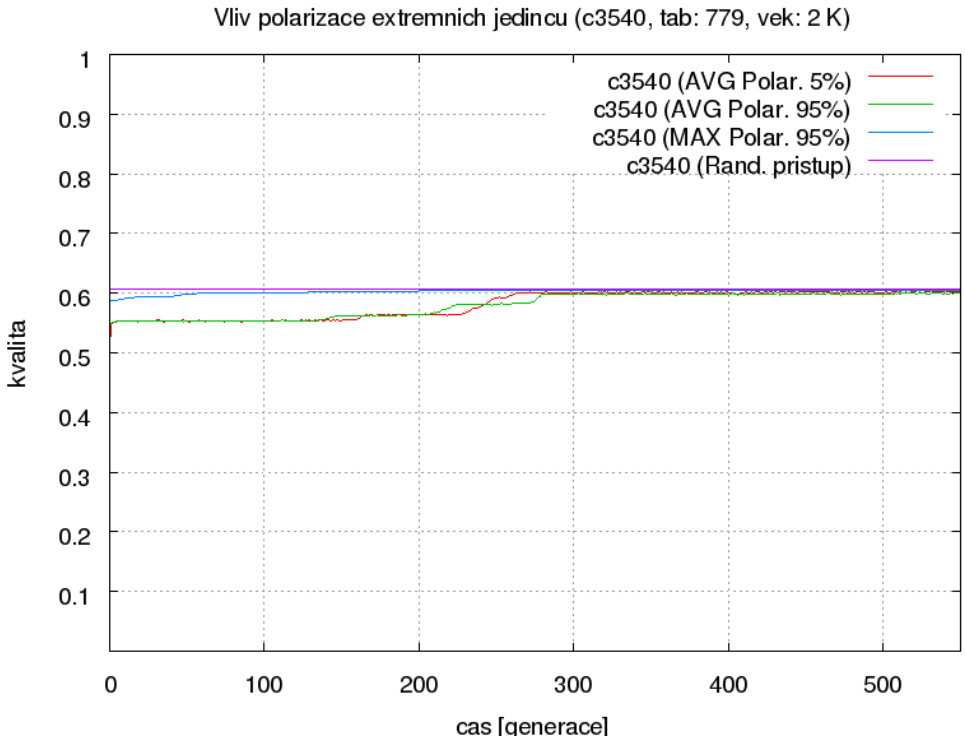
Při porovnávání s randomizovaným přístupem, dále jen *RS* podle Random Search, budeme za výsledek GA považovat kvalitu nejlepšího jedince v populaci po dosažení 550. generace této populace. Při zkoumání vlivu hodnoty parametrů algoritmu na průběh evoluce nás pak bude zajímat spíše průměrná kvalita jedinců v populaci. Takto lépe podchytíme trend vývoje uvažované populace v závislosti na hodnotě sledovaného parametru. Mez 550. generace jsme pak zvolili proto, že v řadě srovnávaných případů došlo v populaci k rychlé degeneraci a populace se ustálila natolik, že nemělo smysl v měření dále pokračovat.
Příkladem testovaného obvodu, u kterého se neprojevil takřka žádný rozdíl mezi výsledky dosahovanými GA a RS, je obvod c3540. Drobné fluktuace ve výsledcích jsou s vysokou pravděpodobností dílem náhody, a lze je tak považovat za statistické chyby měření.
Na výsledky dosahované GA pro obvod c3540 nemá podle grafů na obrázcích 6.1 a 6.2 prakticky žádný vliv polarizace extrémních jedinců ani jejich podíl v populaci. V obou případech jsou pak výsledky GA srovnatelné s výsledky RS.
Polarizací jako takovou rozumíme v tomto kontextu míru polarizace ve prospěch počtu jedniček v chromosomu extrémních jedinců.
Z grafu na obrázku 6.3 je navíc dobře patrné, že použitá implementace ruletového výběru nedokáže zajistit zdravý vývoj populace. Rozdíl průměrných kvalit takové populace a populace prořezávané turnajovým výběrem se blíží 5%. I přes to však GA dosahuje za použití tohoto typu výběru výsledků srovnatelných s RS.
Za použití dvoubodového křížení pak populace sice dříve konverguje ke kvalitním jedincům, ale z dlouhodobého hlediska má tento typ křížení na průměrnou kvalitu populace stejný vliv, jako standardně používané křížení uniformní (viz obr. 6.4). Výsledky obou metod, tedy GA a RS, jsou pak opět plně srovnatelné.

6.7.3 Výsledky pro obvod s838

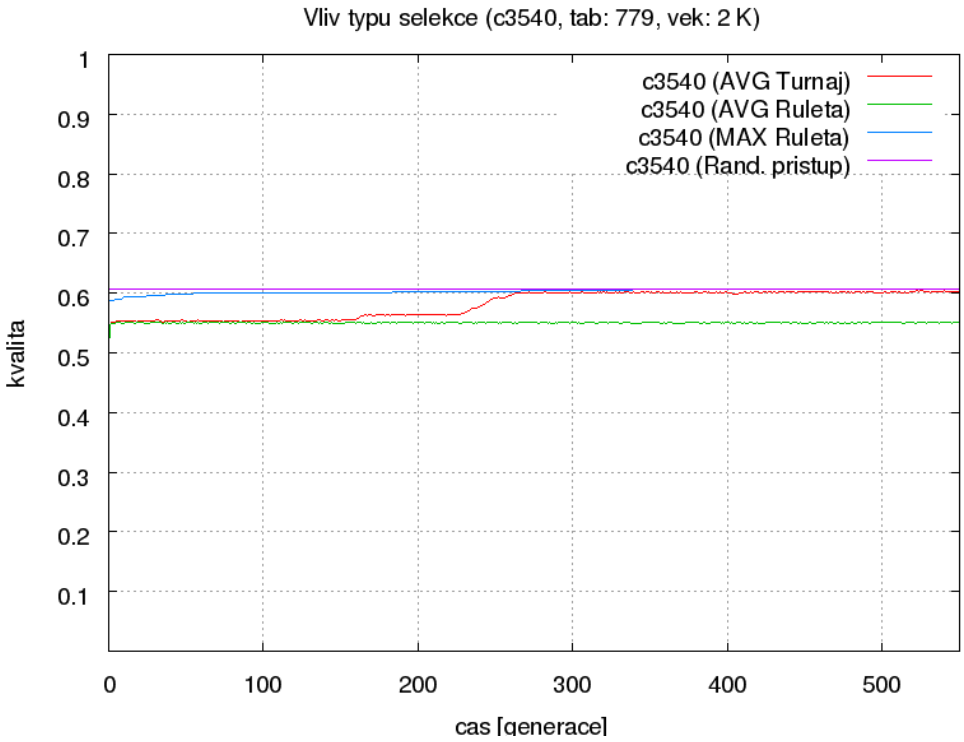
Opačná situace nastala v případě obvodu s838. U tohoto obvodu dosahoval GA ve srovnání s RS v některých případech až o 10% lepších výsledků (viz graf na obr. 6.5).
Na tomto grafu je dobře vidět vliv podílu extrémních jedinců v populaci na její průměrnou kvalitu. Zvýšením tohoto podílu ze standardních 50 na 90% dosahuje průměrná kvalita populace hodnot o necelých 5% vyšších. Tento rozdíl navíc kolem 50. generace dosahuje krátkodobě téměř 10%. Oproti RS pak dosahuje GA v případě 90% zastoupení extrémních jedinců v populaci přibližně o 10% lepších výsledků.
Graf na obrázku 6.6 popisuje vývoj průměrné kvality populace v případě, že je populace z 50% tvořena extrémními jedinci, jejichž chromosom je z naprosté většiny vyplněn samými jedničkami. Pro všechny loci chromosomu takových jedinců platí, že pravděpodobnost jejich obsazení jedničkou je v tomto případě rovna 95%. Z grafu je pak vidět, že takový způsob polarizace chromosomu extrémních jedinců je méně výhodný, než standardně používaná 5% polarizace. Rozdíl průměrné kvality populací dosahované v obou případech překračuje 5%. Pro uvažovanou 95% polarizaci pak GA dosahuje srovnatelných výsledků s RS.
Stejně jako tomu bylo u obvodu c3540, platí i v tomto případě, že použitá implementace ruletového výběru není příliš efektivní z hlediska konvergence populace ke kvalitnějším jedincům. Průměrná kvalita populace zde stoupá jen velice pozvolna (viz obr. 6.7). Rozdíl oproti standardně používanému turnajovému výběru dosahuje téměř 10%. I přes výše zmíněné nedostatky však výsledky GA s ruletovým výběrem převyšují výsledky RS o několik málo procent.



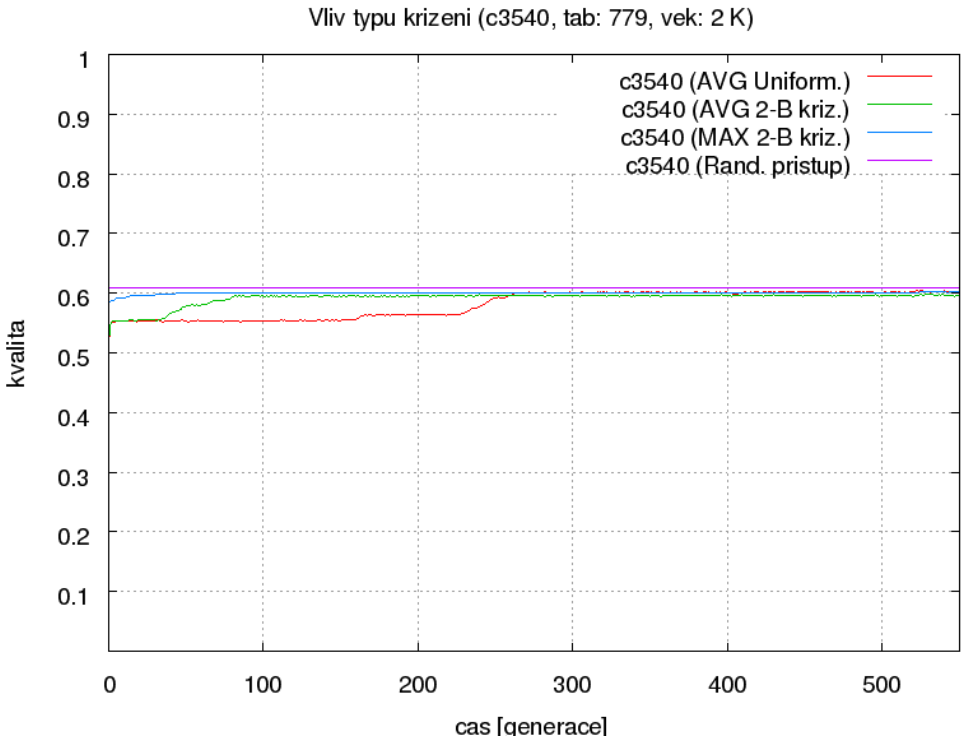
Obrázek 6.1: Vliv podílu extremistů pro obvod c3540



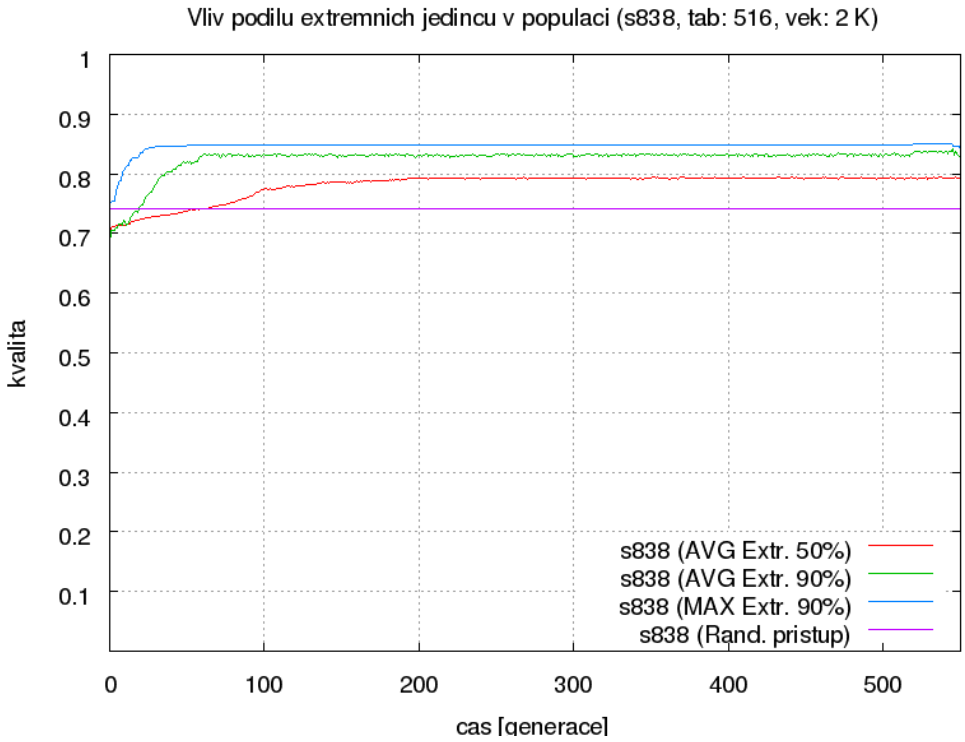
Obrázek 6.2: Vliv polarizace extremistů pro obvod c3540



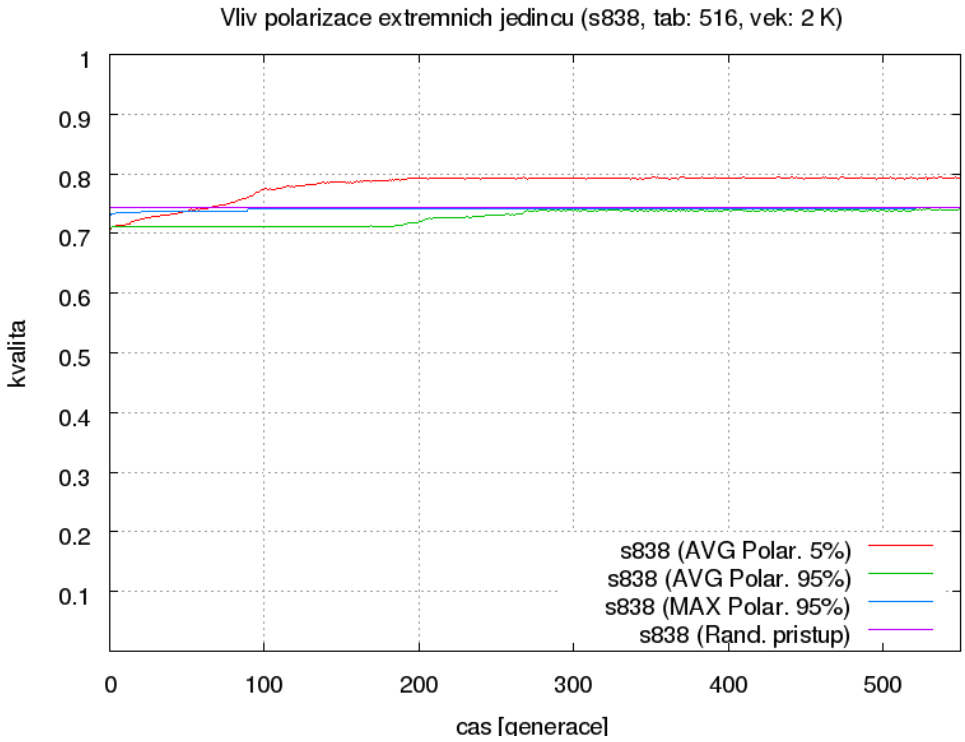
Obrázek 6.3: Vliv typu selekce pro obvod c3540



Obrázek 6.4: Vliv typu křížení pro obvod c3540



Obrázek 6.5: Vliv podílu extremistů pro obvod s838

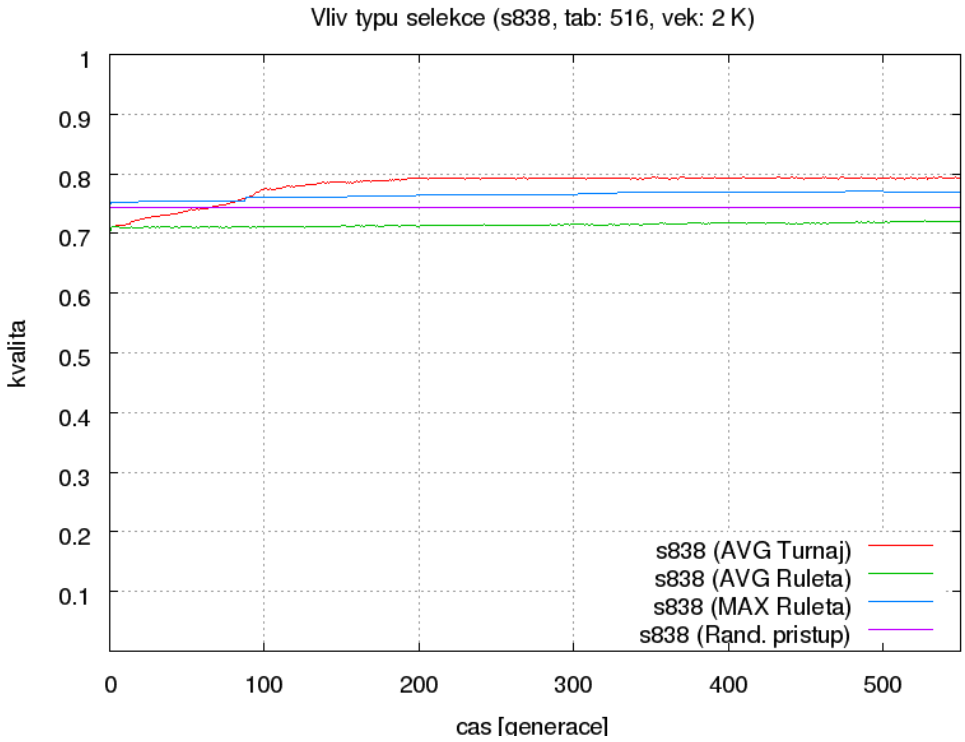


Obrázek 6.6: Vliv polarizace extremistů pro obvod s838

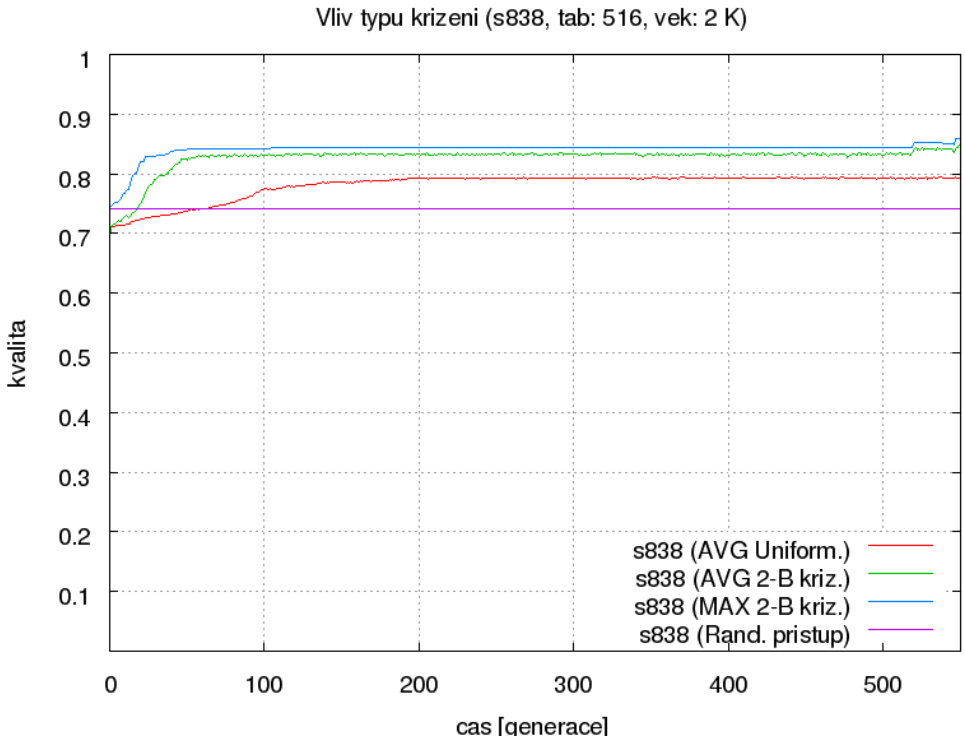
Vliv typu křížení na průměrnou kvalitu populace zachycuje graf na obrázku 6.8. Je na něm patrná výhoda dvoubodového křížení oproti standardně používanému křížení uniformnímu. Průměrně dosahovaná kvalita populace je v případě prvně jmenovaného typu křížení o necelých 5% vyšší než u uniformního křížení. Výsledky GA používajícího dvoubodové křížení jsou pak zhruba o 10% lepší než výsledky RS.

6.7.4 Výsledky pro obvod b07

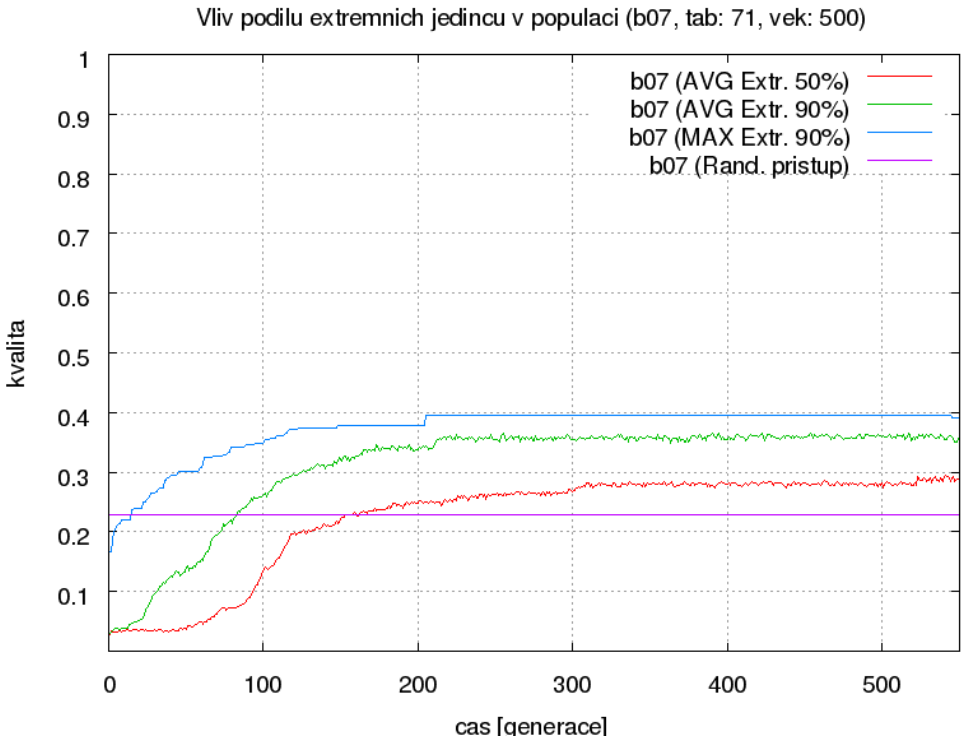
Posledním z testovaných obvodů, u kterých si podrobněji popíšeme výsledky měření, je obvod b07. Tento obvod je do značné míry speciální tím, že jsme u něj zjišťovali vliv velikosti předgenerované tabulky vektorů, používané při vyhodnocování kvality jedinců, na vývoj populace jedinců v průběhu evoluce.
Nejzajímavějších výsledků jsme pak dosáhli pro poměrně malou tabulku, čítající celkem 71 vektorů.
Rozdíl průměrné kvality populace obsahující standardních 50% extrémních jedinců a populace s 90% takových jedinců se pohybuje v rozmezí 5 a 10% (viz obr. 6.9). Podstatnější je však fakt, že GA pracující s takovouto populací dosahuje téměř o 20% lepších výsledků než RS, což představuje oproti RS téměř dvojnásobné zlepšení.
Graf na obrázku 6.10 popisuje mimo jiné i vývoj průměrné kvality populace, která obsahuje jedince s 95% polarizací chromosomu. Je z něj jasně patrný propad této kvality ve srovnání s kvalitou dosahovanou v populaci, která používá standardní 5% polarizaci. Tento pokles dosahuje kolem 150. generace téměř 20% a v další generacích se ustaluje zhruba na 10%. Co se výsledků GA s takovou populací týče, ty jsou pak plně srovnatelné s výsledky dosahovanými RS.
V případě ruletového výběru se i u tohoto obvodu opakuje situace popsaná u obou dříve zmíněných obvodů. Průměrná kvalita populace je při použití tohoto typu selekce přibližně o 10% nižší, než když zvolíme turnajový výběr (viz obr. 6.11). Je zajímavé, že posledně jmenovaný typ selekce je z výše uvedeného hlediska v prvních asi sto generacích výrazně horší a ruletový výběr překonává teprve při překročení přibližně 125. generace. Jinak jsou GA a RS opět srovnatelné z pohledu nejlepších jimi dosažených výsledků.
Na obrázku 6.12 je uveden graf vlivu typu křížení na vývoj průměrné kvality populace. I v tomto případě se dvoubodové křížení chová obdobným způsobem jako u předchozích dvou testovaných obvodů. Hlavní rozdíl spočívá v tom, že diference průměrné kvality populací u obou typů křížení, tj. dvoubodového a standardně používaného uniformního, je mnohem více patrná, především pak v počátcích samotné evoluce. Tato diference lehce překračuje hranici 5%. GA využívající tento typ křížení pak dosahuje o více jak 10% lepších výsledků než RS.



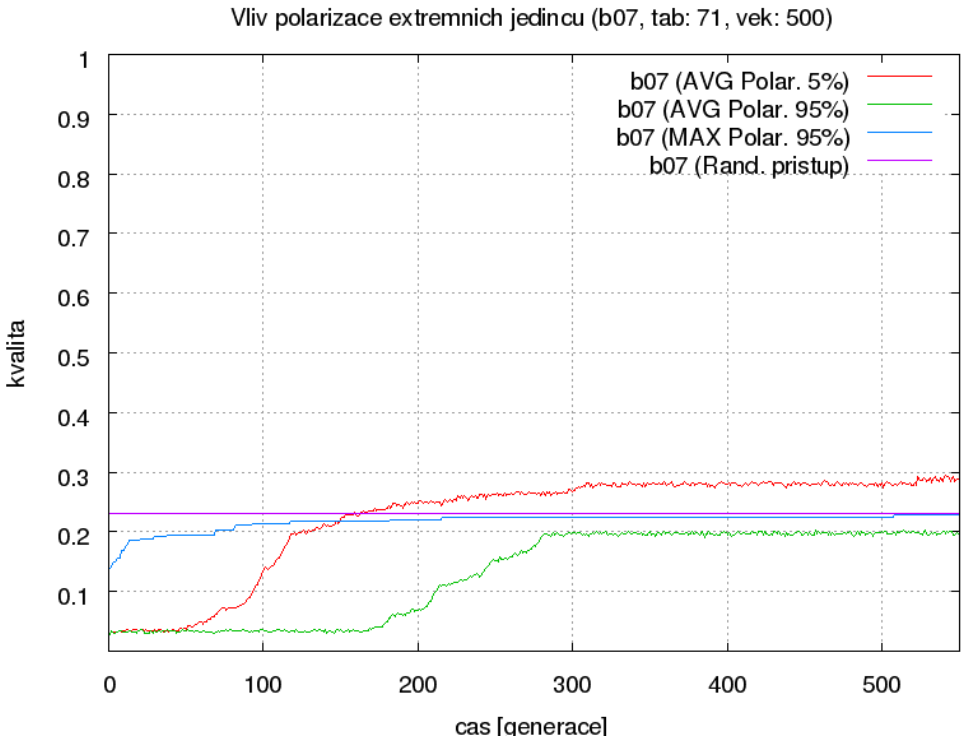
Obrázek 6.7: Vliv typu selekce pro obvod s838



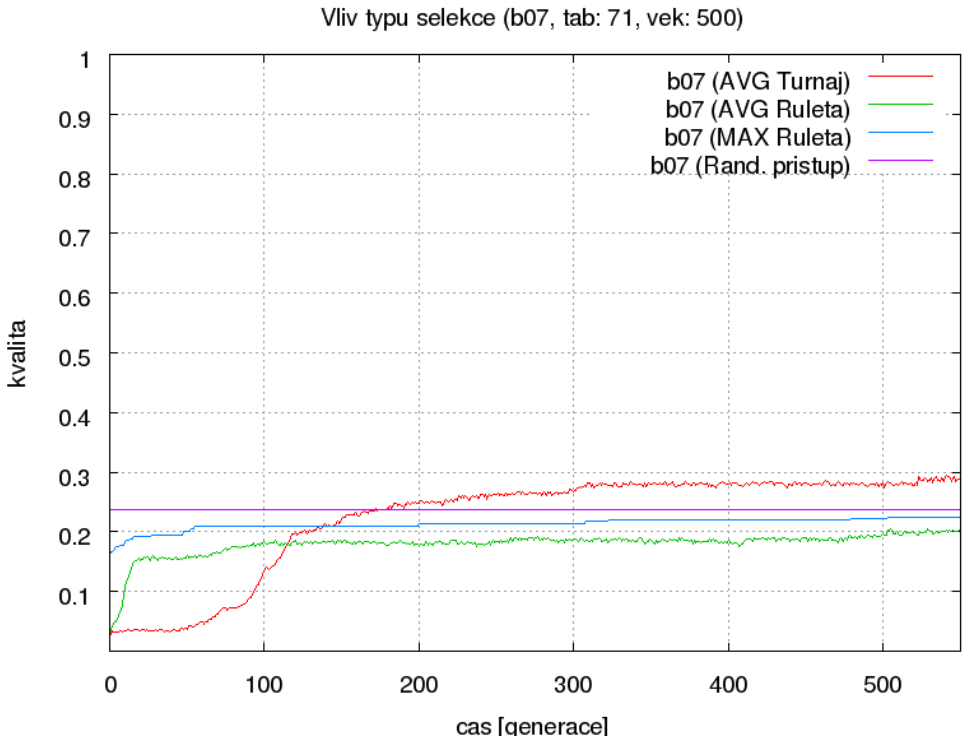
Obrázek 6.8: Vliv typu křížení pro obvod s838



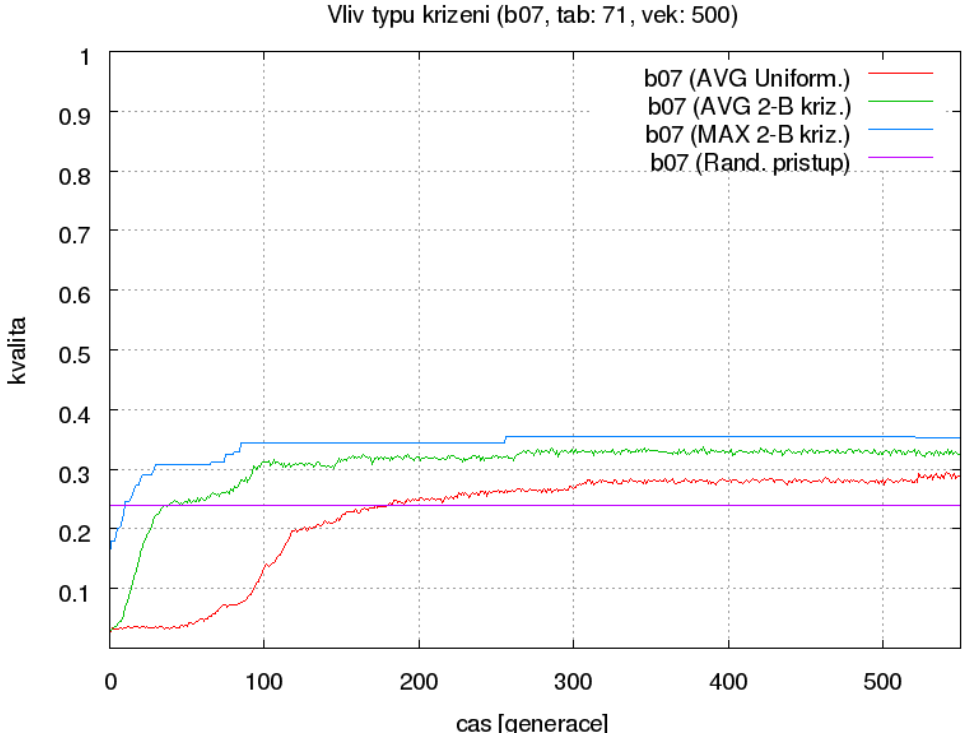
Obrázek 6.9: Vliv podílu extremistů pro obvod b07



Obrázek 6.10: Vliv polarizace extremistů pro obvod b07



Obrázek 6.11: Vliv typu selekce pro obvod b07



Obrázek 6.12: Vliv typu křížení pro obvod b07

6.7.5 Shrnutí naměřených výsledků

I v tomto závěrečném shrnutí budeme pracovat se zmíněnými třemi obvody, tj. s c3540, s838 a b07, protože výsledky pro ně změřené představují poměrně dobrou reprezentaci výsledků naměřených pro všechny testované obvody (pro jejich celkový přehled viz tab. 6.18).
Výše popsané výsledky měření je možné shrnout ze dvou různých úhlů pohledu. Prvním je vliv různých hodnot sledovaných parametrů GA na jeho výkonnost jako takovou a druhým je pak skutečná použitelnost takto nastaveného algoritmu pro řešení zadaného problému. Jako měřítko nám přitom poslouží už tolikráte citovaný plně randomizovaný přístup:

1. Začněme jedním velice zajímavým parametrem GA, kterým je *podíl extrémních jedinců v populaci*.

   Ačkoliv se tento parametr u obvodu c3540 takřka vůbec neprojevil, u zbývajících dvou obvodů se už situace značně lišila. Samotné zvýšení hodnoty tohoto parametru z 50 na 90% se projevilo nárůstem průměrné kvality populace o 5% v případě obvodu s838 a o 10% u obvodu b07.

   Ještě více dramatický je rozdíl výsledků dosahovaných GA s takto zvýšenou hodnotou parametru a RS, který dosahoval více než 10% pro obvod s838 a téměř 20% v případě obvodu b07.

2. Druhým na řadě je možná ještě zajímavější parametr GA, kterým je *polarizace chromosomu extrémních jedinců*.

   V kapitole 3.1.5 o CA s rozváženým seedem jsme se zmínili o důsledcích, jaké má rozvážení seedu pro testování logických obvodů. Co jsme ale pouze naznačili, byla optimální míra takového rozvážení. V řadě experimentů byly zkoumány vlastnosti CA se seedem tvořeným téměř výhradně samými nulami [4, 6, 3]. To byl také důvod, proč jsme takové nastavení parametru použili jako výchozí.

   Možná by zde také bylo na místě jen letmo připomenout, že námi používaní jedinci s polarizací chromosomu 5% de facto reprezentují výše zmíněné CA (viz kap. 6.3).

   V případě obvodu c3540 se nastavení tohoto parametru opět takřka nijak neprojevilo, zato u dalších dvou obvodů jsme mohli pozorovat negativní vliv zvýšení polarizace ve prospěch počtu jedniček v chromosomu extrémních jedinců. Zatímco u obvodu s838 se jednalo o pokles průměrné kvality populace o 5%, v případě obvodu b07 to už představovalo téměř 10%.

   Ve všech třech případech se pak zvýšená polarizace promítla do výsledných vlastností GA tak, že tento dosahoval takřka totožných výsledků s RS.

3. Dalším sledovaným parametrem je *typ selekce*.

   Zde se téměř jednoznačně ukázalo, že u všech zmiňovaných obvodů se ruletový výběr projevil jako nevhodná varianta selekce. Ve všech třech případech se pak průměrná kvalita populace zvyšovala v průběhu evoluce pouze zanedbatelně. Rozdíl mezi ní a průměrnou kvalitou populace dosahovanou za použití turnajového výběru se pohyboval v rozmezí od 5 do 10%, počínaje obvodem c3540 a konče obvodem b07.

   Výsledky GA používajícího ruletový výběr se velice blížily výsledkům dosahovaným RS, od kterých se nikdy nelišily více než o několik málo procent v obou směrech. Jedním z důsledků použití tohoto typu selekce byla takřka konstantní diversita populace v průběhu měření, která se většinou pohybovala kolem 90%. Tak vysoká diversita populace mohla podle mého názoru přiblížit vlastnosti GA s tímto nastavením a RS, a proto jsme pak u obou metod mohli pozorovat obdobné chování.

Testovaný obvod	GA (tab.)	GA (sim.)	RS (tab.)	RS (sim.)	Prům. čas [s]
c3540	61.13	95.58	60.75	95.14	4918
c880	38.64	97.64	36.58	97.43	1431
c2670	11.94	84.31	11.81	84.33	18007
s386	85.49	91.77	86.85	93.65	335
s420	85.59	97.49	80.067	93.77	950
s953	78.025	93.75	78.72	93.77	2603
s713	87.62	90.88	88.19	91.015	628
s838	84.92	97.50	74.11	87.42	3192
b07 (26 vek.)	27.69	95.18	20.77	94.81	159
b07 (48 vek.)	32.083	96.00	21.25	94.77	236
b07 (71 vek.)	39.44	92.81	22.82	95.075	302
b07 (102 vek.)	42.55	90.22	33.73	95.50	401
b07 (252 vek.)	61.35	96.014	59.048	95.70	722
b07 (526 vek.)	66.88	95.48	63.76	94.95	1358
b07 (784 vek.)	73.90	95.50	70.64	95.15	1749
b07 (1021 vek.)	76.039	95.94	72.50	95.15	2140

Tabulka 6.18: Přehled výsledků změřených pro 90% podíl extremistů v populaci

V tabulce 6.19 jsou pak shrnuty výsledky měření pro předgenerovanou tabulku testovacích vektorů o velikosti 100 vektorů. Nastavení GA zůstalo identické jako v případě předchozích měření, s vyjímkou použití 90% podílu extremistů a dvoubodového křížení.

Testovaný obvod	GA (tab.)	GA (sim.)	RS (tab.)	RS (sim.)	Prům. čas [s]
c3540	57.31	95.64	58.85	95.55	565
c880	34.90	98.24	28.43	97.98	236
c2670	0	84.33	0	84.31	1553
s386	87.80	92.19	90.40	92.14	78
s420	68.30	93.49	48.68	92.61	371
s953	52.50	93.88	54.23	94.01	419
s713	75.62	90.81	77.33	90.95	487
s838	60.76	97.85	8.19	87.54	941
b07	43.53	87.65	32.35	95.61	444

Tabulka 6.19: Přehled výsledků změřených pro tabulku 100 vektorů

4. Posledním z probíraných parametrů je *typ křížení.*

   Na rozdíl od parametru popisovaného v předchozím bodě je u tohoto parametru situace pro sledovanou hodnotu přesně opačná. U každého ze zmiňovaných obvodů se projevila změna ze standardně používaného uniformního křížení na křížení dvoubodové pozitivně a ve všech případech se jednalo o cca. 5% nárůst průměrné kvality populace.

   Co se srovnání RS a GA využívajícího dvoubodové křížení týče, tak v případě obvodů s838 a b07 dosahoval takový algoritmus zhruba o 10% lepších výsledků než RS a výsledky pak byly srovnatelné pouze u obvodu c3540.

Na základě výše uvedeného výčtu sledovaných parametrů je možné formulovat následující doporučení: jako nejslibnější nastavení GA se mi jeví použití populace tvořené z 90% extrémními jedinci, jejichž chromosomy obsahují mizivé procento jedniček, spolu s turnajovým výběrem a dvoubodovým křížením.
Závěrem bych se rád zmínil o *vlivu velikosti předgenerované tabulky vektorů* na chování GA, který se nejvýrazněji projevil právě u obvodu b07 pro tabulku o 71 vektorech.
Z měření provedených pro tento obvod výplývá, že s klesající velikostí tabulky se vlastnosti GA, konkrétně dosahované pokrytí této tabulky, ve srovnání s RS zlepšovaly. Nejlépe to pak bylo patrné právě pro již zmíněnou tabulku 71 vektorů. Při dalším poklesu se začaly vlastnosti algoritmu opět pomalu zhoršovat.
To si vysvětluji tím, že v případě velkých tabulek obsahujících i přes tisíc vektorů bude cílový CA vyvíjený pomocí GA přespecifikovaný. Algoritmus pak bude přehlcen daty o takovém automatu, takže pro něj bude těžší takový automat vyvinout.
Pokud však použijeme příliš malé tabulky, čítající pouze několik málo desítek vektorů, poskytneme tak GA nedostatečné množství informací o cílovém CA, a tím mu také zkomplikujeme hledání takového automatu.
Z tohotu úhlu pohledu se pak jako ideální jeví tabulky, jejichž velikost nepřekračuje 150 vektorů. Směrodatnost tohoto tvrzení je ale třeba prokázat dalším měřením pro co nejširší škálu testovaných obvodů.

7 Závěr

Cílem této práce bylo prozkoumat možnosti použití genetických algoritmů pro syntézu celulárních automatů vhodných k testování kombinačních logických obvodů. V rámci práce jsme navrhli a implementovali jednu možnou variantu binárního genetického algoritmu a experimentálně vyhodnotili její vlastnosti porovnáním s plně randomizovaným přístupem.
Z výsledků měření vyplývá, že nasazení genetických algoritmů je účelné pouze u některých typů testovaných obvodů a dosahované výsledky se často liší případ od případu. V některých situacích však mají genetické algoritmy z hlediska kvality jimi syntetizovaných celulárních automatů jasnou převahu nad plně randomizovaným přístupem:

1. Jako nejvýznamnější parametr genetického algoritmu z pohledu kvality jím dosahovaných výsledků se ukazuje především podíl extrémních jedinců v populaci, se kterou algoritmus pracuje, a dále pak polarizace chromosomu takových jedinců ve prospěch podílu nul v chromosomu obsažených.

2. S výkonností genetického algoritmu také velice úzce souvisí jím používaný způsob vyhodnocování kvality jedinců v populaci.

   Na základě výsledků měření je možné prohlásit, že v případě vyhodnocování kvality jedinců pokrýváním předgenerované tabulky vektorů má velice výrazný vliv na kvalitu algoritmem dosahovaných výsledků velikost této tabulky. Ukazuje se, že algoritmus dosahuje velice dobrých výsledků pro relativně malé tabulky.

   Vedlejším efektem použití takových tabulek je podstatné snížení operační složitosti algoritmu, což může dále usnadnit jeho aplikaci v praxi.

Na základě výše shrnutých poznatků bych navrhoval následující postup další práce:

- Podle mého názoru by se na základě bodu 1 výše uvedeného závěru vyplatilo detailněji prozkoumat možnosti vytváření libovolně velké části počáteční populace, se kterou genetický algoritmus pracuje, z předem vygenerovaných jedinců, reprezentujících ať už celulární automaty získané pro konkrétní testovaný obvod nějakou deterministickou metodou, nebo automaty vyvinuté algoritmem pro jiné obvody s příbuznou strukturou.

- Na základě bodu 2 výše uvedeného závěru usuzuji, že by bylo účelné provést další, podstatně rozsáhlejší série měření s cílem potvrdit výhodnost použití malých tabulek vektorů při vyhodnocování kvality jedinců pro co nejširší možnou škálu testovaných obvodů.

- Další variantou postupu je prozkoumání vhodnosti použití speciálních genetických algoritmů, jejichž příklad jsme uvedli v kapitole 3.4, popř. heterogenních paralelních evolučních technik, jejichž princip obecně spočívá v kombinaci různých typů algoritmů inspirovaných přírodními procesy s deterministickými heuristikami, které se pak společně podílí na řešení uvažovaného problému.

8 Literatura

[1] J. Hlavička, *Diagnostika a spolehlivost.* Zikova 4, 166 35, Praha 6, Czech Rep.: Vydavatelství ČVUT, 2nd ed., 1998.

[2] H. K. Lee and D. S. Ha, "An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. of the 1991 International Test Conference*, pp. 946–955, Oct. 1991.

[3] P. Fišer and H. Kubátová, "Pseudorandom testability - study of the effect of the generator type," *CVUT Acta Polytechnica*, vol. 45, Aug. 2005.

[4] F. Corno, M. S. Reorda, and G. Squillero, *Evolutionary Algorithms for Embedded Design*, ch. Built-In Self Test of Sequential Circuits. Kluwer Academic Publishers, 2002.

[5] P. D. Hortensius, R. D. McLeod, and B. W. Podaima, "Cellular automata circuits for built-in self-test," *IBM Journal of Research and Development*, vol. 34, Mar. 1990.

[6] P. Fišer and H. Kubátová, "Improvement of the fault coverage of the pseudo-random phase in column matching bist," in *Proc. of the 31th Euromicro Symposium on Digital Systems Design*, pp. 56–63, Sep. 2005.

[7] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms.* 111 River St., Hoboken, NJ 07030, USA: John Wiley & Sons Inc., 2nd ed., 2004.

[8] P. Fišer and J. Schmidt, "Problémy a algoritmy - webová stránka předmětu." . `http://service.felk.cvut.cz/courses/36PAA/`.

[9] "Wikipedia - evolutionary computation." . `http://en.wikipedia.org/wiki/Evolutionary_computation`.

[10] "Wikipedia - gene." `http://en.wikipedia.org/wiki/Gene`.

[11] "Wikipedia - the origin of species." . `http://en.wikipedia.org/wiki/The_Origin_of_Species`.

[12] "Wikipedia - heuristic (computer science)." . `http://en.wikipedia.org/wiki/Heuristic_(computer_science)`.

[13] "Wikipedia - group selection." `http://en.wikipedia.org/wiki/Group_selection`.

[14] R. Dawkins, *The Selfish Gene.* Great Clarendon Street, Oxford OX2 6DP, UK: Oxford University Press, 1st ed., 1989.

[15] H. K. Lee and D. S. Ha, "Atalanta: an efficient atpg for combinational circuits," tech. rep., Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1993.

[16] "The message passing interface - a definition." `http://www-unix.mcs.anl.gov/mpi/`.

[17] P. Fišer and H. Kubátová, "Influence of the test lengths on area overhead in mixed-mode bist," in *Proc. of the 9th Biennial Baltic Electronics Conference*, pp. 201–204, Oct. 2004.

[18] "Ai-junkie - genetic algorithms in plain english." . `http://www.ai-junkie.com/ga/intro/gat1.html`.

A Seznam použitých zkratek

ANSI American National Standards Institute

ATE Automated Test Equipment

ATPG Automatic Test Pattern Generation

BIST Built-in Self-test

CA Cellular Automaton

CLI Command Line Interface

CUT Circuit Under Test

DFT Design for Testability

GA Genetic Algorithm

ISO International Organization for Standardization

LFSR Linear Feedback Shift Register

LHCA Linear Hybrid Cellular Automaton

MPI Message Passing Interface

PRPG Pseudo-random Pattern Generator

RE Response Evaluator

TPG Test Pattern Generator

XOR Exclusive Disjunction

B Obsah přiloženého CD

```
thesis
  |-bench (ISCAS-89 a ITC-99 benchmarky)
  |-code (zdrojovy kod programu)
  |-fsim (simulator poruch fsim)
  |-intro (uvodni studie)
  |-logs (zaznamy mereni)
  |---b07-102
  |---b07-1021
  |---b07-252
  |---b07-26
  |---b07-48
  |---b07-526
  |---b07-71
  |---b07-784
  |---c2670
  |---c3540
  |---c880
  |---s386
  |---s420
  |---s713
  |---s838
  |---s953
  |-paper (zdrojovy text prace)
  |---figs (grafy a obrazky)
  |-rand (generatory pseudonahodnych cisel)
  |-tables (predgenerovane tabulky vektoru)
```