BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2021
http://excel.fit.vutbr.cz

Porovnání kódování problému v evolučním návrhu kombinačních obvodů

Adam Sedláček*

Abstrakt
Práce porovnává dva odlišné přístupy k zakódování kombinačních obvodů při automatizovaném návrhu obvodů, který využívá evolučních algoritmů. Porovnání proběhlo mezi kartézským genetickým programováním (CGP) a obvodem reprezentovaným v algebraické normální formě (ANF). Obě metody byly demonstrovány na čtyřech obvodech. Pro urychlení hodnocení kvality obvodů bylo využito paralelní simulace. Výhody a nevýhody obou metod zakódování jsou pak shrnuty v závěru této práce.

Klíčová slova: Kartézské genetické programování — Algebraická normální forma — Evoluční návrh — Kombinační obvod

Přiložené materiály: *N/A*

*xsedla1e@fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Evoluční algoritmy (EA) jsou optimalizační algoritmy, které stochasticky prohledávají prostor kandidátních řešení. Jsou inspirovány Darwinovou evoluční teorií, kde se uplatňuje přirozený výběr (přežití) nejsilnějších jedinců do další generace. EA využívá selekce, mutace a rekombinace. Prohledávaný prostor může být obrovský. Konvenční techniky pak selhávají nebo ani není možné je aplikovat. Nalezené řešení nemusí nutně být nejlepší možné, ale hledáme dostatečně kvalitní řešení [1].

Aby mohl být evoluční algoritmus využit, je třeba vhodně zakódovat řešení, v našem případě obvod, do řetězce symbolů, který se nazývá chromozom. Takto zakódovaný jedinec je ohodnocen pomocí fitness funkce, která určuje jeho kvalitu. Dobře navržená fitness funkce je klíčová pro efektivní řešení daného problému [1]. Evoluční algoritmus pak následně pracuje na základě opakování následujících kroků: (i) Vytvoření nové množiny kandidátních řešení (populace), (ii) Ohodnocení populace.

Jednou z variant EA je kartézské genetické programování (CGP), které je popsáno v kapitole 2. CGP se osvědčilo hlavně při návrhu kombinačních obvodů, ale má využití i v jiných oblastech, například ve strojovém učení, symbolické regresi, evolučním umění apod. [2]. V CGP se nejčastěji se sestavují obvody pomocí dvouvstupých hradel, nicméně je na uživateli, jak zvolit vhodně stavební bloky pro řešení.

Jinou alternativou pro zakódování obvodu je použití algebraické normální formy (ANF), tj. jako termy spojené operací xor (kapitola 3).

Tyto dva odlišné přístupy k reprezentaci obvodů byly porovnány v úloze evolučního návrhu obvodů (v kapitole 5) na zvolených obvodech. Ty jsou blíže popsány v kapitole 5, která také obsahuje výsledky experimentů. Závěrečná kapitola 6 prezentuje shrnutí této práce.

2. Kartézské genetické programování

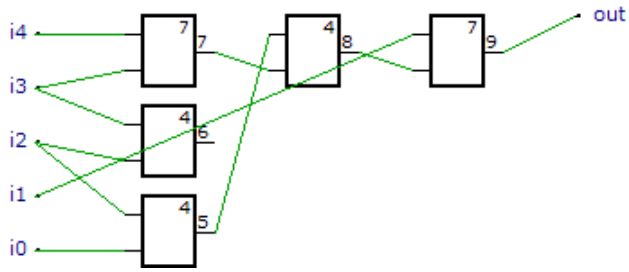
CGP kóduje acyklický orientovaný graf pomocí řetězce celočíselných hodnot. Jedinec je reprezentován jako mřížka, která má pevný počet řádků a sloupců. Velikost mřížky zůstává po celou dobu běhu konstantní, a vypočítá se jako $c \times r$, kde c je rovno počtu sloupců (column) a r je počet řádků (row). Každý uzel reprezentuje hradlo o n vstupech s právě jedním výstupem o. Uzel realizuje jednu z pevně daných funkcí nad danými vstupy [3, 4]. V této práci se jedná o dvouvstupová hradla, do uzlu přichází maximálně 2 vstupy.

Nad takto definovanou maticí se posléze hledá vhodné propojení uzlů. Vnitřní propojení mezi uzly je řízeno tzv. L-back parametrem. Ten určuje, které uzly mohou být připojeny na vstup dalšího uzlu a to tak, aby nedocházelo ke smyčkám (zpětné vazbě). L-back se volí v rozmezí $l \in < 1, c >$. Pokud $l = 1$, tak se uzly napojují pouze na výstupy uzlů z předchozího sloupce [3].

Kandidátní řešení v CGP se zakódovávají do celočíselných řetězců délky $r \times c \times (n+1) + o$. Každý uzel v matici má přiřazeno unikátní hodnotu, a to včetně primárních vstupů, která umožňuje definovat propojení. Zakódování jednoho uzlu se následně skládá z n celých čísel, které určují napojené vstupy (v této práci vždy $n = 2$), a z kódu logické funkce daného uzlu (opět celočíselně, například $0 = and, 1 = or, 2 = nand$ atd). Poslední část chromozomu tvoří k-tice, kde o je počet primárních výstupů.

Zakódování 5bitové parity, které je zobrazeno na obr. 1, by následně vypadalo v CGP s parametry $c = 5, r = 1, n = 2, o = 1$ a množinou hradel (hradlo *IN* pouze replikuje první vstup na výstup, tj. dá se chápat jako vodič) {IN, AND, OR, XOR, NOT, NAND, NOR, NOT XOR} takto:

$$(0,2,4)(2,3,4)(3,4,7)(5,7,4)(1,8,7)(9)$$



Obrázek 1. 5bit parita s nadbytečným hradlem č. 6

3. Algebraická normální forma

Algebraická normální forma (ANF) je jeden ze způsobů reprezentace logické funkce. Oproti CGP, které dovoluje prakticky libovolné propojení hradel, zavádí značné restrikce.

Formule, které popisují funkci obvodu sestávají z termů, které jsou mezi sebou exkluzivně sečteny. Termy pak sestávají z literálů, které jsou spojeny konjunkcí. Literál může být přímý nebo negovaný.

Počet literálů v daném termu je vždy nejvýše počtu vstupů. Počet termů pak závisí na tom, jakou funkci chceme realizovat. Například pokud bychom se snažili popsat 5ti bitovou paritu pouze se třemi termy, pak takovou formuli nelze sestavit. Dalším zásadním parametrem je arita, která udává, kolik vstupů může být v jeden moment aktivních v jednom termu. Term s aritou například 3 je ekvivalentní třivstupovému hradlu *AND*. Obvod s m výstupy modelujeme pomocí m formulí. Je zřejmé, že některé logické funkce budou v ANF zakódovány úsporně, jiné zase velmi nevýhodně.

Jako příklad uvažme obvod s $n = 3$ vstupy a $m = 2$ výstupy, jehož chování popisují formule:

$$y_0 = x_1 x_2 \oplus x_1 \bar{x_2} x_3 \oplus x_3$$
$$y_1 = x_1 x_2 \oplus \bar{x_1} \bar{x_2} \bar{x_3}$$

Za předpokladu, že je ve formuli možné použít $t = 3$ termy, tento obvod bude zakódován řetězcem:

$$y_0 = 1101 - 11001$$
$$y_1 = 011 - 1 - 1 - 1000$$

Kde 0, 1 a -1 popisují nepoužitou proměnnou, přítomnou proměnnou a negovanou proměnou. Délka chromozomu je potom $m \cdot n \cdot t = 2 \cdot 2 \cdot 3 = 12$ genů. Pomocí zavedení arity je možné redukovat počet použitých proměnných v termu.

4. Použitý algoritmus

V obou případech se používá pro prohledávání stavového prostoru stejný algoritmus, konkrétně evoluční strategie, která je typická zejména pro kartézské genetické programování [3], ale osvědčila se i pro obvody reprezentované algebraickou normální formou.

Algoritmus je založený na evoluční strategii $\lambda + 1$, kde λ udává počet mutovaných jedinců v generaci, většinou $\lambda = 4$. Algoritmus podporuje elitismus. Z této populace se pak následně vybere nejlepší jedinec a ten přejde do další generace jako rodič beze změny [1].

Parametr mutace v tomto případě neudává procentuální šanci na změnu, ale kolik hodnot chromozomu se bude měnit [3].

Fitness je definována jako Hammingova vzdálenost mezi pravdivostní tabulkou kandidátního obvodu a cílovou pravdivostní tabulkou. Pro akcelerace je použito paralelní simulace. Algoritmus pracuje v těchto krocích:

1. Vygenerování $\lambda + 1$ náhodných jedinců (chromozomů) pro inicializaci populace.
2. Ohodnocení všech jedinců populace pomocí fitness funkce.
3. Nalezení nejlépe ohodnoceného jedince — nejnižší fitness.
4. Vygenerování λ potomků pomocí operátoru mutace aplikovaného na nejlepšího nalezeného jedince.

5. Nejlepší nalezený jedinec společně s jeho λ potomky tvoří novou populaci.
6. Ohodnocení všech jedinců populace pomocí fitness funkce.
7. Není-li splněna podmínka ukončení, pokračuje se krokem 3.

5. Experimenty

Cílem evolučního návrhu je najít plně funkční obvody založené na z náhodně generované počáteční populace. Porovnáme reprezentaci obvodů CGP a ANF. Budeme zjišťovat, kolik generací je potřeba pro nalezení obvodu. Optimalizace velikosti obvodů není cílem této studie.

Provedené experimenty proběhly nad čtyřmi různými obvody. Obvody byly zvoleny tak, aby se ukázalo, kde je vhodné použít CGP a kde naopak ANF. Parametry EA byly zvoleny na základě řady zkušebních experimentů, jejich vliv na výsledek není předmětem této studie. Použitý evoluční algoritmus byl již popsán v kap. 4. Pro každý z obvodů bylo provedeno 31 běhů v obou kódováních.

Parametry pro všechny obvody zakódované v CGP:

- Počet řádků = 1
- Počet sloupců = 70
- Mutace = 3 %
- L-back = 70
- Počet generací = 5000000
- Množina funkcí = {IN, AND, OR, XOR, NOT, NAND, NOR, NOT XOR}

Parametry pro všechny obvody zakódované v ANF:

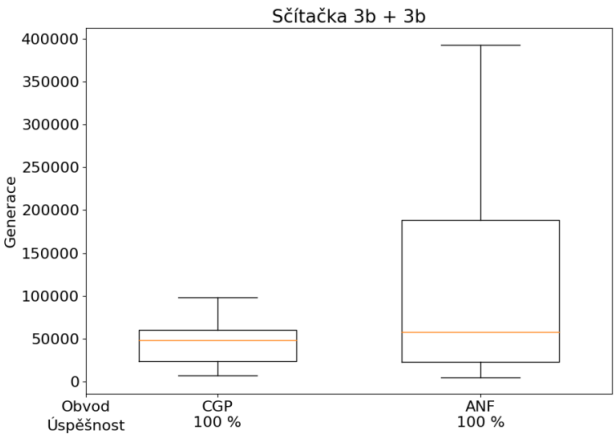
- Počet termů = 12
- Mutace = 2
- Arita = počtu vstupů daného obvodu
- Počet generací = 5000000

5.1 Sčítačka 3b + 3b

U 3bitové sčítačky jak v CGP ani ANF nebyl žádný problém najít plně funkční obvod. Statistické vyhodnocení počtu generací potřebných k nalezení plně funkčního obvodu zobrazuje obr. 2. Nicméně z maximálního a průměrného počtu generací vyplývá, že CGP si vedlo lépe. Toto může být zapříčiněno tím, že u sčítačky je potřeba použít hradlo *OR*, které v ANF přímo není. CGP má přímo hradlo *OR* v množině funkcí.

5.2 Sudá parita 9b

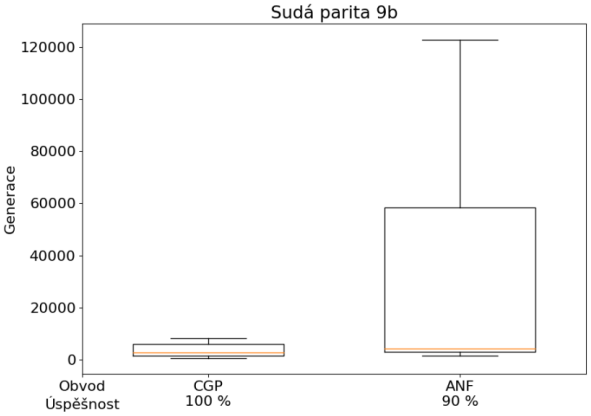
Parita je jedním z klasických obvodů, kterým se ověřuje, zda nám vůbec správně pracuje evoluční návrh. Struktura ANF přímo odpovídá realizaci parity (sudé nebo liché) ze své podstaty.



Obrázek 2. Porovnání CGP a ANF: 3bit sčítačka

V tomto konkrétním případě měla metoda založená na ANF úspěšnost 90 %. Důvodů je hned několik. Parametr arity byl nastaven na počet vstupů (tj. 9 literálů v jednom termu). Toto nastavení způsobí značné ztížení při hledání funkčního obvodu. Jednak se musí vyhodnotit mnoho kombinací, ale hlavně evoluce musí deaktivovat nadbytečné literály, které nepřispívají do ohodnocení obvodu. Výsledky pro toto nastavení jsou na obr. 3.

Druhý parametr, který značně ovlivní složitost hledání, je počet termů. V tomto konkrétním případě bylo nastaveno 12 termů. Pokud o daném obvodu víme nějaké informace *navíc*, je velmi vhodné nastavit parametry pro ANF (platí i pro CGP) tak, abychom zbytečně nekomplikovali prohledávání. Z tohoto důvodu jsem provedl dalších 31 běhů pro každé zakódování s co možná nejlepším nastavení parametrů. Výsledky jsou zobrazeny na obr. 4. Níže jsou uvedené nalezené nejvhodnější parametry pro ANF (dále označené jako ANF-O), pro CGP a pro CGP s redukovanou množinou funkcí (CGP-R).



Obrázek 3. Porovnání CGP a ANF: 9bit sudá parita (demonstrace nevhodného nastavení ANF)

Parametry pro ANF-O:

- Počet termů = 9
- Mutace = 2
- Arita = 1
- Počet generací = 5000000

Parametry pro CGP:

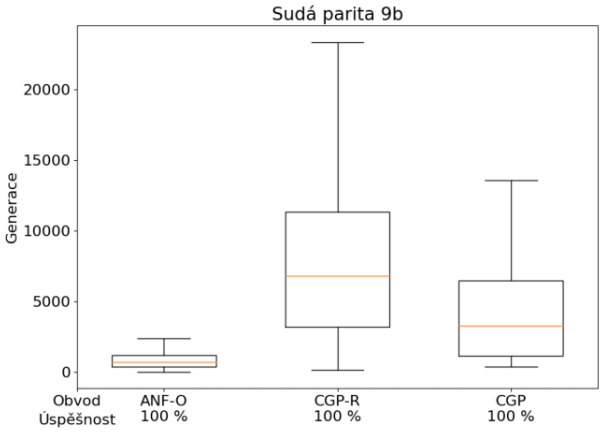
- Počet řádků = 1
- Počet sloupců = 80
- Mutace = 2 %
- L-back = 80
- Počet generací = 5000000
- Množina funkcí = {IN, AND, OR, XOR, NOT, NAND, NOR, NOT XOR}

Parametry pro CGP-R:

- Počet řádků = 1
- Počet sloupců = 80
- Mutace = 2 %
- L-back = 80
- Počet generací = 5000000
- Množina funkcí = {NOT, AND, XOR}

Parametry pro CGP-R jsou stejné jako pro CGP, pouze došlo k redukci množiny funkcí na {AND, XOR, NOT}, čímž jsem se snažil napodobit ANF.

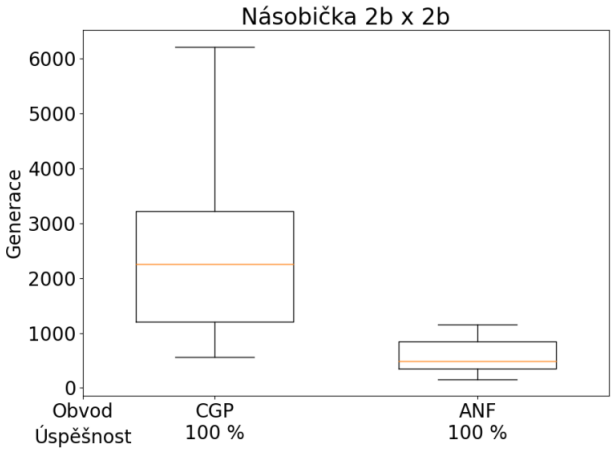
Vidíme, že v případě upravených parametrů ANF-O zcela překonalo CGP i CGP-R. Překvapením může být CGP-R s redukovanou množinou hradel, kdy může docházet k tomu, že i tak je celkem složité najít správnou strukturu parity (propojení). Zatímco CGP může najít stejnou funkcionalitu pomocí jiných hradel.



Obrázek 4. Porovnání CGP a ANF s vyladěnými parametry: 9bit sudá parita

5.3 Násobička 2b x 2b

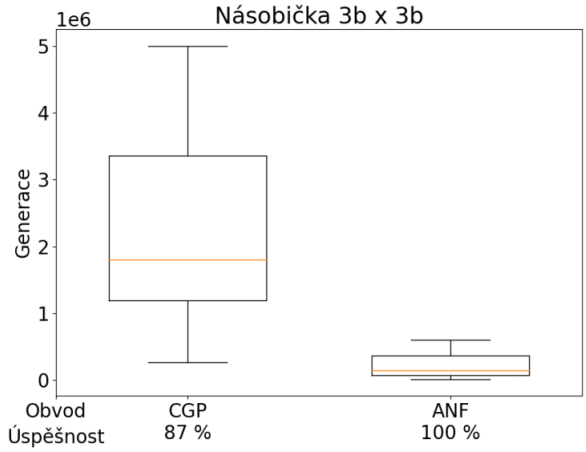
Násobička je jeden z dalších obvodů, kde se hlavně využívá hradel *XOR* a *AND*. Není překvapením, že ANF opět dominovala. V průměru můžeme vidět, že metoda založená na ANF konvergovala zhruba 4krát rychleji než CGP. Na násobičce 2b x 2b to nemusí být zcela patrné, ale u větších násobiček se tento fakt projeví více.



Obrázek 5. Porovnání CGP a ANF: 2bit násobička

5.4 Násobička 3b x 3b

Složitost zapojení násobičky se významně zvyšuje s každým přidaným vstupním bitem. Můžeme si všimnout, že CGP v tomto konkrétním případě mělo úspěšnost 87% viz obr. 6. Jak bylo avizováno u dvoubitové násobičky, je to zapříčiněno složitostí zapojení obvodu, kdy CGP musí najít nejen správná hradla, ale i jejich správné zapojení. V tomto má ANF velkou výhodu, protože struktura obvodu je předepsána.



Obrázek 6. Porovnání CGP a ANF: 3bit násobička

6. Závěr

V článku byly představeny dva odlišné přístupy k zakódování kombinačních obvodů v evolučním návrhu obvodů. Provedené experimenty ukázaly výhody i nevýhody obou přístupů. Metoda založená na ANF se ukázala výhodná zej-ména při návrhu násobiček, kde

konvergence probíhá mnohonásobně rychleji, rovněž s vyšší úspěšností. Toto zjištění je možné využít například při návrhu násobiček s vyšším počtem vstupů, kde CGP naráží na své limity. Naopak CGP bude vhodnější použít, pokud bude kladen důraz i na plochu nebo zpoždění obvodu.

Další práce bude spočívat v experimentování s dalšími obvody tak, aby byly prozatímní závěry důkladněji prověřeny. Rovněž bude provedeno porovnání velikosti evolučně navržených obvodů.

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu prof. Ing. Lukáši Sekaninovi, Ph.D. za rady při psaní této práce.

Literatura

[1] Agoston Eiben and James Smith . *Introduction to Evolutionary Computing*. Natural Computing Series,. Springer Berlin Heidelberg, 2015.

[2] Lukáš Sekanina, editor. *Evoluční hardware : od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Number sv. 4 in Gerstner. Academia, Praha, vyd. 1 edition, 2009.

[3] Julian Miller, editor. *Cartesian genetic programming*. Natural computing series. Springer, Heidelberg : New York, 2011.

[4] Zdeněk Vašíček. Biologií inspirované počítače - kartézské genetické programování, 2021. [Online] Dostupné z http://www.fit.vutbr.cz/~vasicek/courses/bin_lab1/.cs.