Excel@FIT 2015
http://excel.fit.vutbr.cz

Kartézské genetické programování s LUT

Karolína Hajná*

Abstract
Tato práce se zabývá problematikou návrhu obvodů pomocí kartézského genetického programování na úrovni třívstupových logických členů. Je představena metoda bitově paralelní akcelerace výpočtu fitness pro obvody s třívstupovými hradly. Porovnáním jejích vlastnosti s nejpoužívanější variantou kartézského genetického programování, které používá dvouvstupová hradla, jsou ukázány výhody a nevýhody této metody. Výsledky by mohly přinést nové možnosti při návrhu obvodů.

Keywords: Kartézské genetické programování — třívstupové LUT — návrh obvodů

*xhajna00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Evoluční algoritmy (EA) představují jednu z možností pro řešení úloh, kde konvenční algoritmy neposkytují kvalitní výsledky nebo úplně selhávají. Jedná se především o úlohy prohledávání prostoru možných řešení, který může nabývat obrovských rozměrů. EA se inspirují přírodou a využívají operátory jako je křížení a mutace. Stejně jako v přírodě i v těchto algoritmech mají nejvyšší šanci přežít a reprodukovat se nejsilnější jedinci (nejlepší řešení úlohy) v dané populaci. Použití genetických operátorů a paralelní přístup k prohledávání umožňuje překonávat lokální extrémy, které jsou značnou překážkou pro mnohé jiné metody. Předpokladem k použití EA je schopnost zakódování potenciálních řešení do řetězce (chromozomu). Způsob kódování závisí na řešené úloze a je klíčový pro úspěšnost celého algoritmu. Dalším důležitým aspektem je ohodnocování jedinců. K tomuto účelu se používají rozmanité fitness funkce. Stejně jako v případě kódování jsou závislé na typu úlohy. Obecně EA tvoří opakující se kroky: (i) vytvoření populace, (ii) ohodnocení jedinců v populaci a (iii) vytvoření nové populace z vybraných jedinců. Jedním z možných uplatnění EA je návrh a optimalizace obvodů. Literatura uvádí mnoho úspěšných příkladů v této oblasti [1]. Pro evoluční návrh kombinačních obvodů se využívá kartézské genetické programování (CGP)[1][4], o kterém následující text pojednává. V současné době se obvody nejčastěji navrhují na úrovní základních logických členů – dvouvstupových hradel. Snaha využít v CGP hradla s více vstupy zatím selhávala z hlediska efektivity. Cílem následujícího textu je odhalit výhody návrhu obvodů na úrovní třívstupových logických členů, které mohou být implementovány pomocí look-up tabulek (LUT).

V druhé kapitole je vysvětlen samotný algoritmus CGP. V kapitole třetí je popsán způsob urychlení algoritmu, který se aktuálně v CGP používá, a jeho rozšíření pro výpočet s třívstupovými LUT. Čtvrtá kapitola popisuje experimenty provedené na obou variantách CGP, porovnává je mezi sebou a hledá souvislosti, které by mohly vést k využití návrhu obvodů na úrovni LUT. V závěru jsou shrnuty dosažené výsledky a rozebrány možnosti pokračování výzkumu v tomto směru.

2. Kartézské genetické programování

Jedná se o variantu genetického programování, která kóduje zadaný problém do acyklické orientované grafové struktury. Obecně nachází uplatnění při návrhu kombinačních obvodů, v umělé inteligenci a strojovém učení, při řešení matematických problémů, v evolučním umění a mnoha dalších oblastech.[2] K prohledávání CGP využívá principu evoluční strategie. Jako jediný genetický operátor používá pouze mutaci. Kandidátní obvod je popsán jako maticové uspořádání

bloků. Velikost pole je $r \times c$, kde r je počet řádků a c počet sloupců matice. Každý blok představuje jeden element obvodu. Jednotlivé elementy mají n vstupů a jeden výstup. Zadán je počet primárních vstupů i a výstupů o navrhovaného obvodu. Cílem algoritmu je nalézt zapojení elementů, které odpovídá obvodu zadanému pravdivostní tabulkou. Vstupy elementů jednoho sloupce mohou být připojeny pouze na výstupy elementů v předcházejících sloupcích nebo na primární vstupy. Důležitým parametrem řídícím vnitřní konektivitu je L-back, který udává počet bezprostředně předcházejících sloupců, na jejichž výstupy je možno element připojit. L-back může nabývat hodnot z intervalu $\langle 1, c \rangle$. V CGP je chromozom zakódován do celočíselných řetězců délky $r \times c \times (n+1) + o$. Všechny elementy a primární vstupy mají přiřazeny unikátní indexy. Nejprve se číslují primární vstupy a poté elementy zleva doprava. Každý blok je reprezentován pomocí $n+1$ hodnot. Prvních n hodnot udává indexy výstupů, kam jsou připojeny vstupy bloku, poslední hodnota je vyhrazena pro reprezentovanou funkci. Poslední část tvoří k-tice s velikostí shodnou s počet primárních výstupů. Chromozom pro CGP s parametry: $r = 3$, $c = 3$, $i = 2$, $o = 2$ může vypadat například následovně. (0, 2, 1)(1, 2, 4)(1, 2, 1)(3, 1, 1)(4, 0, 1)(4, 5, 3)(7, 4, 2)(3, 8, 3)(8, 6, 4)(6, 11) Grafická reprezentace tohoto chromozomu je na obrázku 1.

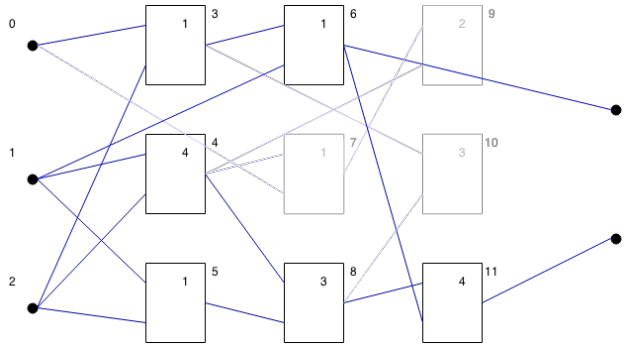

Figure 1. Grafická reprezentace obvodu v CGP

Prohledávací algoritmus CGP vychází z evoluční strategie $(1+\lambda)$ a používá pouze mutaci. Selekce je postavena na principu elitismu. Mutace je definována jako náhodná změna genu. Celý algoritmus sestává z následujících kroků.

1. Vygenerování $1+\lambda$ náhodných jedinců pro inicializaci populace
2. Ohodnocení všech jedinců populace pomocí fitness funkce
3. Nalezení nejlépe ohodnoceného jedince (neutralita je podporována)
4. Vygenerování λ potomků mutací nejlepšího jedince
5. Nalezený nejlepší jedinec společně s potomky tvoří novou populaci
6. Pokud není splněna podmínka ukončení, pokračuje se krokem 2

Hlavním kritériem pro hodnocení obvodu ve fitness funkci je míra funkčnosti. Výpočet probíhá přivedením všech možných kombinací vstupů na primární vstupy a porovnáním výstupů ohodnocovaného jedince s očekávanými výstupy. Fitness je potom definována jako počet bitů, které kandidátní obvod správně vyprodukoval.

3. Akcelerovaná simulace obvodů s LUT

Pro ohodnocení kandidátního obvodu s k vstupy je třeba vygenerovat a získat odezvu pro 2^k vektorů. Vyhodnocovat kombinace vstupů jednotlivě by bylo značně neefektivní, protože by vedlo na 2^k průchodů obvodem. Z tohoto důvodu se používá bitově-paralelní simulace. Namísto jednotlivých bitů se ke každému primárnímu vstupu přivede vektor vstupů o délce x bitů, který kóduje příslušné bity pravdivostní tabulky, a celé ohodnocení se tak urychlí x-krát. Stupeň paralelizace je omezen architekturou procesoru, přičemž x udává počet bitů FX operandu na dané architektuře (na obr. 2 je příklad pro $x = 16$).

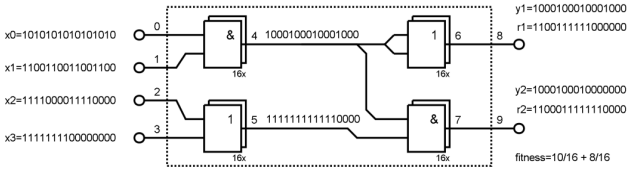

Figure 2. Paralelní simulace, převzato z [1]

Vyvstává zde otázka jak realizaci funkcí implementovat. U dvouvstupových hradel se používá jednoduchý výčet všech 16 funkcí a výběr pomocí switch. Funkce jsou realizovány pomocí bitových operací nad datovým typem unsigned int (o x bitech). U vícevstupových LUT vzniká problém exponenciálního nárůstu počtu funkcí v závislosti na počtu vstupů do prvku. Dále je kvůli paralelní simulaci nutné převést logickou funkci s několika operandy popsanou kódem v LUT na posloupnost elementárních logických funkcí pracujících se dvěma operandy. Pro paralelní simulaci je kritickou částí způsob vyhledání příslušné funkce a realizace bitové operace nad celým vektorem. U vyššího počtu vstupů je neefektivní výčtem popsat všechny funkce (pro n=3 je to 256 funkcí, pro $n = 4$ je to 65536 funkcí atd.) tak, jak je to používáno dvouvstupových hradel. Nicméně pro účely třívstupových LUT byl postup založený na výčtu vyhodnocen jako nejvhodnější. Pro všech 256 možných funkcí byla manuálně nalezena jejich realizace pomocí binárních

obvod	úspěšnost	Konečná fitness			Počet uzlů		
9b sudá parita	100%	MIN 512	MAX 512	AVG 512	MIN 8	MAX 10	AVG 8,24
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 9 972	MAX 674 686	AVG 151 535	MIN 8	MAX 19	AVG 11,82
		Konečná fitness			**Počet uzlů**		
4b + 4b sčítačka	100%	MIN 1 280	MAX 1 280	AVG 1 280	MIN 17	MAX 29	AVG 21,08
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 174 625	MAX 6 320 783	AVG 1 427 922	MIN 23	MAX 37	AVG 28,86
		Konečná fitness			**Počet uzlů**		
4b x 3b násobička	12%	MIN 885	MAX 896	AVG 893,02	MIN 50	MAX 57	AVG 53,5
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 7 965 508	MAX 17 433 799	AVG 11 613 796	MIN 50	MAX 57	AVG 54,33

Table 1. CGP s hradly a plnou množinou funkcí

obvod	úspěšnost	Konečná fitness			Počet uzlů		
9b sudá parita	100%	MIN 512	MAX 512	AVG 512	MIN 4	MAX 10	AVG 6,7
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 1 269	MAX 76 583	AVG 19 787	MIN 7	MAX 28	AVG 11,1
		Konečná fitness			**Počet uzlů**		
4b + 4b sčítačka	100%	MIN 1 280	MAX 1 280	AVG 1 280	MIN 10	MAX 18	AVG 13,92
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 38 494	MAX 4 283 582	AVG 869 665	MIN 17	MAX 43	AVG 26,5
		Konečná fitness			**Počet uzlů**		
4b x 3b násobička	100%	MIN 896	MAX 896	AVG 896	MIN 32	MAX 48	AVG 39
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 1 052 151	MAX 19 243 508	AVG 7 253 217	MIN 33	MAX 52	AVG 44,86

Table 2. CGP s třívstupovými LUT a plnou množinou funkcí

logických operátorů. Rozdíl oproti CGP s dvouvstupovými hradly je zejména v počtu bitových operací, které je nutné vykonat na jednu funkci. Namísto jedné operace pro dvouvstupá hradla může počet operací dosahovat až pěti. Velká část funkcí se třemi vstupy však tohoto maximálního počtu nedosahuje a průměrný počet binárních operací je 3,5.

Pro získání kompletního ohodnocení kandidátního obvodu bude počet binárních operací rovný počtu průchodů celým obvodem $(2^k/x)$ násobený průměrným počtem binárních operací na funkci. Ve srovnání s obvody, které používají dvouvstupová hradla, lze předpokládat, že obvody tvořené třívstupovými LUT budou mít méně prvků, protože počítají s více vstupy na prvek. Z tohoto bude důvodu postačovat k nalezení řešení menší mřížka CGP.

4. Experimenty

Byly provedeny dvě sady experimentů, které porovnávají CGP s dvouvstupovými hradly a CGP s třívstupovými LUT. V první sadě byla použita plná množina funkcí, v druhé množina redukovaná na vybrané logické funkce. Všechny experimenty byly provedeny s následujícími parametry:

- Počet řádků = 10
- Počet sloupců = 7
- L-back = 7

obvod	úspěšnost	Konečná fitness			Počet uzlů		
9b sudá parita	100%	MIN 512	MAX 512	AVG 512	MIN 8	MAX 10	AVG 8,16
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 4 702	MAX 305 475	AVG 104 601	MIN 8	MAX 18	AVG 11,84
		Konečná fitness			**Počet uzlů**		
4b + 4b sčítačka	100%	MIN 1 280	MAX 1 280	AVG 1 280	MIN 17	MAX 32	AVG 20,42
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 161 906	MAX 3 283 491	AVG 1 193 690	MIN 21	MAX 37	AVG 27,7
		Konečná fitness			**Počet uzlů**		
4b x 3b násobička	18%	MIN 886	MAX 896	AVG 892,92	MIN 47	MAX 56	AVG 52,22
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 2 876 793	MAX 17 207 766	AVG 11 613 796	MIN 51	MAX 57	AVG 54,22

Table 3. CGP s hradly a redukovanou množinou funkcí

obvod	úspěšnost	Konečná fitness			Počet uzlů		
9b sudá parita	100%	MIN 512	MAX 512	AVG 512	MIN 4	MAX 7	AVG 8,24
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 1 587	MAX 14 555	AVG 4 968	MIN 5	MAX 24	AVG 13,5
		Konečná fitness			**Počet uzlů**		
4b + 4b sčítačka	100%	MIN 1 280	MAX 1 280	AVG 1 280	MIN 16	MAX 26	AVG 20,1
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 24 353	MAX 4 475 586	AVG 997 799	MIN 20	MAX 40	AVG 30,12
		Konečná fitness			**Počet uzlů**		
4b x 3b násobička	62%	MIN 877	MAX 896	AVG 894,86	MIN 40	MAX 59	AVG 49,68
		Generace prvního řešení			**Počet uzlů prvního řešení**		
		MIN 1 286 033	MAX 19 913 574	AVG 10 191 054	MIN 44	MAX 62	AVG 53,03

Table 4. CGP s třívstupovými LUT a redukovanou množinou funkcí

- Počet generací = 20 000 000
- Velikost populace = 5
- Počet mutovaných genů = 1

Velikost mřížky, počet generací a počet mutovaných genů byly zvoleny na základě již provedených experimentů. Stejným způsobem byla nastavena hodnota l-back na maximální možnou konektivitu, která umožňuje nejsnažší dosažení řešení. Po dosažení plně funkčního řešení byl optimalizován počet prvků obvodu. V tabulkách 1, 2, 3 a 4 jsou shrnuty výsledky padesáti spuštění algoritmu pro každou jeho variantu a každý obvod. Úspěšnost značí procento případů, ve kterých se povedlo nalézt plně funkční řešení. Dále jsou v tabulce uvedeny minimální, maximální a průmerné hodnoty pro konečněhou hodnotu fitness funkce, počet uzlů optimalizovaného řešení, generaci, ve které bylo nalezeno první plně funkční řešení a počet uzlů prvního nalezeného řešení.

Z tabulek 1 a 2 je patrné, že CGP s třívstupovými LUT nachází funkční řešení podstatně snáze než jeho používanější varianta s dvouvstupovými hradly. S cílem redukovat dobu evolučního návrhu je potřeba hledat způsob, jak množinu funkcí zredukovat pouze na funkce, které povedou k řešení úlohy a současně umožní zredukovat prostor možných řešení. U CGP s dvouvstpovými hradly se běžně používá množina AND, OR, XOR, NOT, NAND, NOR, XNOR, identita. Případně se tato množina dále redukuje k navržení jed-

noho konkrétního obvodu na základě předcházejících zkušeností. Stejný přístup byl zvolen pro druhou sadu experimentů. Pro CGP s třívstupovými LUT byla pro prvotní experimenty, které jsou prezentovány v tomto článku, zvolena stejná množina funkcí jako u verze s hradly a byla doplněna o funkce AND3, OR3, XOR3, NAND3, NOR3, XNOR31.

Po redukci množiny (tabulky 3 a 4) CGP s třívstupovými LUT nachází řešení hůře než s plnou množinou, ale stále je v tomto ohledu účinnější než CGP s dvouvstupovými hradly. Je však nutno podotknout, že nalezená řešení jsou méně kvalitní z hlediska počtu prvků než u CGP s dvouvstupovými hradly. Je možné, že je tento nedostatek způsoben velikostí mřížky, která (jak již bylo řečeno v kapitole 2) by mohla být menší. Dalším důvodem může být, že použitá podmnožina funkcí nebyla zvolena správně.

5. Závěr

V článku byl představen způsob akcelerace ohodnocení kandidátních řešení pro CGP, které využívá třívstupové LUT. Experimenty odhalily, že CGP na úrovni třívstupových LUT nachází řešení zadaných úloh snáze než na úrovni dvouvstupových hradel. Této vlastnosti by se dalo využít v kombinaci s běžně používanou variantou CGP. Tato práce bude pokračovat v hledání možností, jak množinu funkcí efektivním způsobem redukovat dynamicky, aby se sama přizpůsobila řešenému problému.

References

[1] SEKANINA, Lukáš. *Evoluční hardware: od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Vyd. 1. Praha: Academia, 2009, 321 s. ISBN 978-80-200-1729-1.

[2] *Cartesian Genetic Programming* [online]. 2013 [cit. 2015-04-01]. Dostupné z: http://www.cartesiangp.co.uk

[3] MILLER, Julian. *Cartesian genetic programming*. New York: Springer, c2011, xxii, 344 p. Natural computing series. ISBN 9783642173103

[4] MILLER, Julian F., Dominic JOB a Vesselin K. VASSILEV. *Principles in the Evolutionary Design of Digital Circuits – Part I*. Genetic Programming and Evolvable Machines. 2000, č. 1.

[5] KAUFMANN, Paul, Christian PLESSL, Marco PLATZNER. EvoCaches: Application-specific Adaptation of Cache Mappings. 2009 NASA/ESA Conference on Adaptive Hardware and Systems. 2009. DOI: 10.1109/ahs.2009.26.