MULTIOBJECTIVE CARTESIAN GENETIC PROGRAMMING

Jiří Petrlík
Master Degree Programme (2), FIT BUT
E-mail: xpetrl04@stud.fit.vutbr.cz

Supervised by: Lukáš Sekanina
E-mail: sekanina@fit.vutbr.cz

Abstract: This paper attempts to integrate a multiobjective optimization into cartesian genetic programming. The experimental evaluation is performed in the task of digital circuit evolution on gate level and on function level.

Keywords: NSGAII, multiobjective optimization, cartesian genetic programming

1 ÚVOD

Tento příspěvek je zaměřen na oblast evoluční elektroniky. Cílem je představit metodu pro generování jednoduchých číslicových obvodů za pomocí kartézského genetického programování (CGP), která bude umožňovat optimalizaci výsledných obvodů podle několika kritérií. Například na zpoždění, nebo počet funkčních jednotek a podobně.

Kartézské genetické programování (CGP) [1] využívá pro reprezentaci kandidátních řešení orientovaný acyklický graf, kde uzly reprezentují funkční jednotky a hrany propojení mezi nimi. Tento graf je zakódován jako vektor celých čísel s pevnou délkou. Samotný obvod má tvar dvourozměrného pole uzlů. Každý uzel reprezentuje jednu funkční jednotku. Typ této funkční jednotky a připojení vstupů určují příslušné hodnoty v chromozomu. Primární výstupy obvodu mohou být připojeny na libovolný uzel v poli. Prohledávání probíhá pomocí evoluční strategie $(1+\lambda)$, která využívá pouze operátor mutace.

2 MUTLIKRITERIÁLNÍ OPTIMALIZACE

Budeme uvažovat, že chceme minimalizovat hodnoty účelových funkcí f_i. Máme-li dvě kandidátní řešení a a b, pak můžeme prohlásit, že řešení a dominuje řešení b právě tehdy, když platí
$f_i(a) \leq f_i(b)$, pro každé $i = 1,...,n$ a současně
$f_i(a) < f_i(b)$, pro alespoň jedno i.

Paretova fronta je množina kandidátních řešení, která nejsou dominována žádným jiným řešením. Při multikriteriální optimalizaci se snažíme získat řešení, která leží na Paretově frontě, případně co nejblíže k ní.

3 METODA

Pro generování číslicových obvodů je použito kartézské genetické programování v kombinaci s algoritmem NSGAII. Algoritmus NSGAII [2] je genetický algoritmus, který byl navržen pro multikriteriální optimalizaci. Mezi jeho hlavní přednosti patří nízká časová složitost $O(MN^2)$, kde M je počet kritérií a N je počet jedinců. Dále umí udržovat diverzitu populace, aniž by bylo nutné zadávat dodatečné parametry. Při vývoji postupujeme takto:

1. Vygenerujeme počáteční populaci pro algoritmus NSGAII. Pro vygenerování každého jedince spustíme jeden běh klasického kartézského genetického programování. Jako hodnotu fitness použijeme dosaženou funkčnost, tj. procento správně vypočtených výstupních hodnot.

2. Spustíme algoritmus NSGAII nad počáteční populací. Zde použijeme dosaženou funkčnost jako míru porušení omezení a další kritéria, jako zpoždění a počet funkčních jednotek jako kritéria optimalizace.

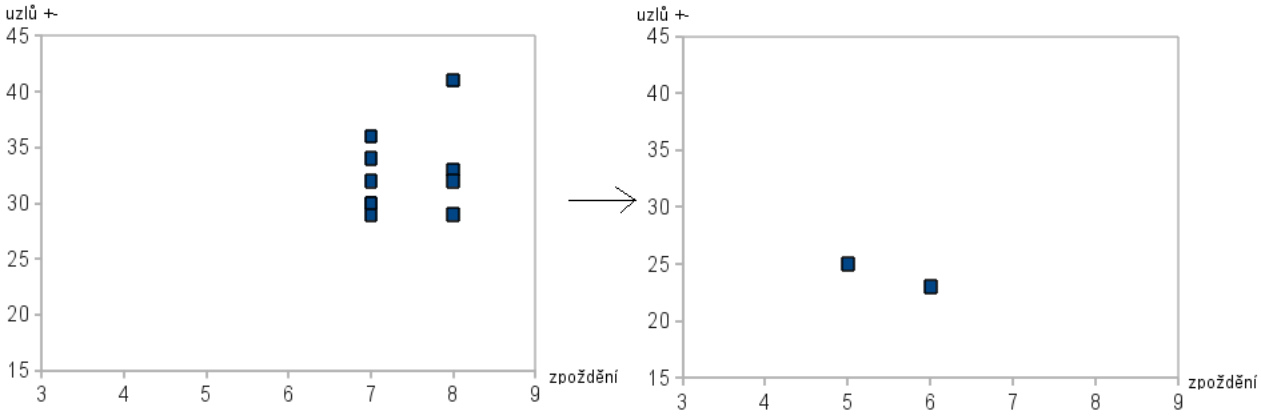
4 TESTOVÁNÍ A ZÁVĚR

Popsanou metodu lze použít pro návrh číslicových obvodů na úrovni hradel. Jako testovací úloha pro evoluci na této úrovni byl použit návrh binárních násobiček o 2x2 a 2x3 vstupech. Tyto násobičky byly optimalizovány na počet hradel a na zpoždění. Jako hradla byla použita and, or, xor a not. Parametr LBack byl vždy maximální. Pro nagenerování počáteční populace bylo použito CGP s populací $(1+4)$ a limitem 10^6 generací (krok 1). Každý z experimentů byl proveden dvacetkrát s populací 10 jedinců. Dosažené výsledky shrnuje tabulka 1. Tabulka uvadí počet generací NSGAII (krok 2).

typ	sloupců x řádků	gen.	prům. zpoždění	průměrně hradel	nejnižší zpoždění	nejméně hradel
2x2	6x6	0,25 M	3	7	3	7
3x2	6x10	2 M	3,45	13,5	3	13

Tabulka 1: Výsledky generování binárních násobiček.

Pro testování evoluce na úrovni funkčních bloků byl použit návrh násobiček s vícenásobnými konstantními koeficienty (MCM). Funkční bloky realizují funkce součet, rozdíl a bitový posuv o 1 - 16 bitů. Je garantováno, že v případě použití těchto operací stačí otestovat násobičku pouze na jednu hodnotu vstupu [3]. Pokud pro tuto hodnotu vstupu dává správný výsledek, pak dává správný výsledek pro všechny hodnoty vstupů. Při testování byly použity stejné úlohy jako v [3]. Byly tedy vytvářeny násobičky se 3, 5, 10 a 20 výstupy. LBack byl vždy maximální. Optimalizováno bylo zpoždění a počet uzlů s operací sčítání a odečítání. Zpoždění je počet funkčních bloků nejdelší cesty mezi vstupy a výstupy. Pro generování počáteční populace bylo použito CGP se strategií $(1+4)$ a limitem 10^7 generací. Výsledky a srovnání s jinými metodami je uvedeno v tabulce 2. Pro otestování navrhované metody bylo použito dvacet běhů s populací deseti jedinců. Počet generací NSGAII je uveden v tabulce. Obrázek 1 zachycuje jeden z průběhů evoluce MCM s deseti výstupy.



Obrázek 1: Graf vlevo ukazuje situaci po nagenerování počáteční populace (krok 1), graf napravo pak situaci po aplikaci NSGAII (krok 2).

Nastavení		Průměrný výsledek		Nejlepší výsledek	
sloupců x řádků	maximálně generací	zpoždění	uzly +-	zpoždění	uzly +-
3 konstanty: 2925, 23111, 13781					
Heuristiky [4]				8	8
5x6 dle [3]	20 M	-	14	5	9
6x6 dle [3]	20 M	-	14	6	8
7x4 dle [3]	40 M	-	13	7	8
7x7	**20 M**	**5,15**	**11,85**	**5**	**9**
5 konstant: 83, 221, 71, 387, 13					
Heuristiky [4]				5	6
4x6 dle [3]	20 M	-	10	4	7
5x6 dle [3]	20 M	-	11	5	6
6x6 dle [3]	20 M	-	11	6	6
6x6	**50 M**	**4**	**7,15**	**4**	**6**
10 konstant: 117, 1123, 743, 221, 1069, 7605, 987, 16689, 3033, 29					
Heuristiky [4]				8	14
10x4 dle [3]	40 M	-	23	7	15
7x6 dle [3]	20 M	-	23	6	17
9x4 dle [3]	40 M	-	22	9	17
10x6	**20 M**	**5,65**	**21,45**	**5**	**19**
20 konstant: 1, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71					
Heuristiky [4]				4	19
4x10 dle [3]	40 M	-	23	4	19
5x10 dle [3]	40 M	-	23	4	19
6x5 dle [3]	40 M	-	21	5	19
6x10	**20 M**	**4**	**20,1**	**4**	**20**

Tabulka 2: Výsledky testů generování MCM násobiček ve srovnání s jinými metodami. Výsledky metody, která byla představena v tomto článku, jsou označeny tučně.

Hlavní předností metody jsou oproti [3] lepší průměrné parametry obvodů. Ve srovnání s [4] pak bylo například vylepšeno zpoždění u násobiček s 3, 5 a 10 koeficienty.

PODĚKOVÁNÍ

Tato práce vznikla v rámci projektu FIT-S-11-1 a MSM 21630528.

REFERENCE

[1] Miller, J., F., Thomson, P.: Cartesian Genetic Programming. Proceedings of the Third European Conference on Genetic Programming (EuroGP2000). LNCS. Vol. 1802, 2000., p. 121-132

[2] Deb, K., Pratap, A., Agarwal, A.,Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. VOL. 6 NO. 2, 2002., p. 182 - 197

[3] Vašíček, Z., Žádník, M., Sekanina L., Tobola J.: On Evolutionary Synthesis of Linear Transforms in FPGA. In Proc. of the Conf. on Evolvable Systems: From Biology to Hardware, LNCS 5216, Springer Verlag, 2008, p. 141-152

[4] Voronenko, Y., Puschel, M.: Multiplierless multiple constant multiplication. ACM Transactions on Algorithms 3(2) (2007) 1-28