Paralelní gramatická evoluce pro optimalizaci elektronických obvodů

Pavel Ošmera

Ústav automatizace a informatiky, fakulta strojního inženýrství
Technická 2, 616 69 Brno
E-mail: osmera@fme.vutbr.cz

Abstrakt

Tento článek popisuje paralelní gramatickou evoluci (PGE), která je vhodná i pro optimalizaci elektronických obvodů nebo automaticky generovaných programů simulujících tyto obvody. Zvýšená rychlost výpočtu PGE je dosahována použitím algoritmu zpětného zpracování. Dále je testován vliv dvouúrovňové optimalizace kombinací diferenciální evoluce s gramatickou evolucí. Příspěvek navazuje na publikace v Kognice a umělý život I-V, zejména [5].

1 Úvod

Gramatická evoluce (GE) může být chápana jako typ genetického programování (GP) založeného na gramatice [6]. Genetické programování navržené Kozou bylo původně programováno v jazyce LISP. Pomocí gramatické evoluce můžeme vytvořit programy v libovolném jazyku, pokud použijeme Backus-Naurovu formu (BNF). V BNF autoři Backus a Naur definovali programovací jazyk ALGOL. Gramatiky BNF se skládají z terminálů, což jsou objekty, které se mohou vyskytovat v daném jazyku, např. +,-,sin,log atd. a neterminálů. Neterminály mohou být nahrazeny jedním nebo více terminály a neterminály. Neterminální symbol je každý symbol, který může být přepsán na jiný řetězec symbolů. Naopak terminální symbol nemůže být již přepsán. Hlavní výhodou GE oproti GP je dána možností generování víceřádkových funkcí v libovolném programovacím jazyku. Programy v GE nejsou zapsány přímo ve stromové struktuře jako je tomu u GP, ale pomocí lineárního genomu, což může být např. posloupnost celých čísel. Převod z genotypu do fenotypu se provádí pomocí operací modulo n, kde n je určeno maximálním smysluplným výběrem, např. daným počtem funkcí. Gramatikou budeme nazývat čtveřici $G = \{N,T,P,S\}$, kde

N je konečná množina neterminálních symbolů,
T je konečná množina terminálních symbolů, přičemž $N \cap T = 0$,
S je počáteční symbol, $S \in N$,
P je množina přepisovacích pravidel.

Chceme-li např. identifikovat jednoduchou funkci cos 2x v rozsahu $[0,2\pi]$ pomocí identické goniometrické funkce $1 - 2\sin^2(x)$, můžeme použít následující gramatiku [6]:

N = {expr,op,pre_op}
T = {sin,cos,log,+,-,*,/,X,1.0,(,)}
S = <expr>

a přepisovací pravidla P :

<expr>::= <expr> <op> <expr> (0)
nebo (<expr> <op> <expr>) (1)
nebo <pre_op> (<expr>) (2)
nebo <var> (3)

<op>::= + (0)
nebo - (1)
nebo / (2)
nebo * (3)

<pre_op>::= sin (0)
nebo cos (1)

<var>::= x (0)
nebo 1.0 (1)

Tento způsob zápisu pravidel se mírně liší od GP. První pravidlo má čtyři možnosti výběru, druhé tři, třetí dvě a čtvrté dvě. Neterminální symboly jsou v lomených závorkách <>. Začíná-li posloupnost genotypu např.: 220,240,220,203,101,53,202,……atd.,
pak aplikací prvního pravidla, které má čtyři možnosti, na počáteční symbol <expr> provedeme tak, že číslo výběru použité možnosti dostaneme operací modulo n, kde n = 4, neboť výběr začíná již možností (0) a např. 220 modulo 3 = 0. Počáteční symbol S je tedy nahrazen pomocí prvního pravidla, a to variantou (0): <expr> <op> <expr>. Nahrazování pak pokračuje aplikací pravidla vždy na neterminální symbol, který nejvíce vlevo, a to tak dlouho dokud nejsou všechny neterminální symboly nahrazeny terminálními symboly. Algoritmus končí např. výrazem: 1.0 – sin(x)*sin(x) – sin(x)*sin(x).

Paralelní genetické algoritmy (PEA - Paralel Evolutionary Algorithms) jsou výkonné stochastické

prohledávací strategie inspirované přírodou, dovolující řešit větší a složitější problémy, přičemž často vedou rychleji k řešení a současně konvergují i k lepším výsledkům. Používají se tři modely PEA: „farmářský“ model (farming model), migrační model a difusní model. Někdy se pro poslední dva typy PEA používá název distribuované genetické algoritmy (DEA). PEA pracuje s nezávislými podmnožinami populace, v nichž probíhá evoluce částečně izolovaně (dále jen podpopulace). Pojem paralelní nebo sekvenční se vztahuje k populačním strukturám, nikoli k hardwaru, na kterém jsou evoluční algoritmy implementovány.

Migrací rozumíme smísení určité populace s jinou populací. Jejím následkem je vnášení (import, tok) cizích alel do genového fondu dané populace. Tím se v něm mění frekvence alel. Migrace může být jednosměrná nebo vzájemná (u sousedících populací), jednorázová, periodická nebo trvalá. Kombinovaným účinkem migrace, selekce a genového posunu může v populaci nastat genetická rovnováha se stálým rozdělením genotypových četností.

Jednotlivé migrační modely se liší strukturou propojení jednotlivých podpopulací. Struktura migračního modelu může být s centralizovaným, hierarchickým (stromovým), kruhovým, kruhovým centralizovaným, maticovým (mřížkovým) a toroidním uspořádáním. Každý hierarchický (vícestupňový, víceúrovňový) systém je složen z elementů jednodušších (podsystémů). Tyto podsystémy mohou být ovšem tvořeny z dalších podsystémů nižšího řádu, tyto z dalších podsystémů ještě nižšího řádu atd. Jednotlivé organizační stupně nazýváme též hladiny (úrovně) systému. Vztahy uvnitř hierarchického systému můžeme rozdělit na horizontální a vertikální. Horizontální vztahy jsou vztahy mezi podsystémy ležícími na stejné organizační úrovni, vztahy vertikální pak vztahy mezi podsystémy na různých organizačních úrovních. Definování různého charakteru vztahů na různých organizačních hladinách má zásadní význam. Se zvyšující se vertikální organizovaností se objevují (emergují) nové vztahy, které na nižších organizačních hladinách neexistovaly, tzn. objevují se i kvalitativně nové vlastnosti, které nemají podsystémy.

2 Použitá gramatika

Gramatiku používáme stejně jako v původním návrhu BNF (O’Neill,, Ryan,2003), kde gramatika je reprezentována jako *{N,T,P,S}*, kde N je soubor terminálů, T je soubor neterminálů. P je množina produkčních pravidel, S se je startovací výraz.

N = {expr, fnc}

T = {sin, cos, +, -, /, *, X, 1, 2, 3, 4, 5, 6, 7, 8, 9}

S = <expr>

množina 4 produkčních pravidel:

1. <expr> := <fnc><expr>
   <fnc><expr><expr>
   <fnc><num><expr>
   <var>
2. <fnc> := sin
   cos
   +
   *
   -
   U-
3. <var> := X
4. <num> := 0,1,2,3,4,5,6,7,8,9

Jednotlivá pravidla mají počet možných výběrů:

č.pravidla	počet možností
1	4
2	4
3	1
4	10

Místo symbolů <pre_op> a <op>, které jsou používané v původním návrhu používáme jednu společnou množinu pro funkce s aritou n. Dále byla vypuštěna pravidla obsahující závorky, jsou nahrazené stromovou reprezentací funkce. Byla přidána množina <num> obsahující číslice a pravidlo <fnc><num><expr>, které slouží ke generování výrazů, v nichž jeden operand je číslo. Pravidlo <fnc><num><expr> je odvozené od pravidla <fnc><expr><expr>, generuje však jednodušší výrazy, kde jeden argument funkce je číslo např. +(4,x), což je ekvivalentní zápisu (4 + x). Stejného efektu je možné dosáhnout, pokud se jeden ze členů pravidla <fnc><expr><expr> rozbalí na <var> a poté na číslo, které by bylo v sadě proměnných uložené. Důvodem pro zavedení pravidla je zjednodušení generování jednoduchých (jednociferných) čísel a umožnění generování složitějších čísel.

Pokud bude generování funkcí probíhat bez omezení, tak je prohledávaný prostor nekonečně veliký a současně i počet řešení je nekonečný. Např. hledání funkce cos(2x), může mít řešení cos(x+x); cos(x+x+1-1); cos(x+x+x-x); cos(x+x+0+0+0...). Proto je nutné omezit velikost funkce. Dalším omezením je, že každý neterminál i terminál má jako vlastnost počet možných použití (počet výskytů v těle jedince). Pro omezení velikost funkce jsou významné pouze prvky sady <expr>, tedy produkční pravidla. Ostatní objekty se mohou nastavit pokud do systému chceme vnést další znalosti o očekávaném výsledku.Všechny sady <expr>, <fnc>, <var>, <num> jsou implementované jako virtuální seznamy. Po každém použití prvku sady se sníží zbývající počet možných použití. V okamžiku, kdy je tento počet roven 0 je prvek odebrán ze sady. To znamená, že dojde ke snížení počtu pravidel. Také to znamená, ze hodnota

genu může mít jiný význam v různé úrovni vytváření těla jedince.
Např.:

pravidlo	zbývající počet	celkový počet možností
A	0	2
B	3	5
C	3	6
D	3	3

před omezením:
8 mod 4 = 0 => A
8 mod 3 = 1 => C
po omezení:
9 mod 4 = 1 => B
9 mod 3 = 0 => B

Tyto změny jsou však deterministické a nevnášejí do systému chaos. Přesto je vhodnější seznam pravidel seřazený sestupně podle počtu použití.

pravidlo	zbývající počet	celkový počet možností
C	3	6
B	3	5
D	3	3
A	0	2

před omezením:
8 mod 4 = 0 => A
8 mod 3 = 1 => B
po omezení:
9 mod 4 = 1 => B
9 mod 3 = 0 => C

Tedy stejný výsledek operace modulo kóduje stejné pravidlo. Je sice i nadále možné, že dojde nejprve k vyčerpání pravidla C a pak teprve k vyčerpáních ostatních a tím opět ke změně významů operace modulo, nicméně nastává to zřídka.

Implementace funkce s aritou n

Seznam <fnc> se od ostatních seznamů liší tím, že je virtuální a dvojrozměrný. První rozměr zde představuje počet operandů fce. Po vybrání počtu operandů vznikne seznam, ze kterého se vybere stejným způsobem jako z ostatních. Objekt TFnc je potomkem objektu TEvalObj rozšířený o seznam zdrojových objektů Sources: TList.

Generování závorek ve funkcích

Je vyřešeno pomocí stromové reprezentace funkce. Pravidla se závorkami byla odstraněna a také terminální symboly „(“ a „)“ byly odstraněny ze sady terminálů. Závorky se do textového výpisu funkce vkládají až při vlastním výpisu.

3 Paralelní gramatická evoluce

Paralelní gramatická evoluce PGE vychází z gramatického evolučního algoritmu popsaného v [6]. Algoritmus je rozšířen o paralelní populační systém, který rozlišuje pohlaví. Každá podpopulace s rozdílným pohlavím používá jiný typ selekce a mutace. Dále obsahuje zpětnou vazbu fenotypu na genotyp.

Chromozom představuje řetězec celých nezáporných čísel, v počáteční populaci náhodně vygenerovaných. Hodnoty genů slouží k rozhodování, který prvek (terminál/neterminál) vybrat při generování těla jedince. První aplikované pravidlo má sadu čtyř možností, proto pří výběru pravidla je vybraná možnost určena: hodnota_genu mod 4. Při výběru proměnné má sada proměnných pouze jeden prvek, výsledek mod 1 je vždy 0, vybere se tedy vždy první prvek (proměnná X). Gen se přečte vždy, i pokud není potřeba udělat žádné rozhodnutí (vybírá se z jednoprvkového seznamu). Toto vnáší do chromozomu jedince redundanci, kdy některé hodnoty genů mohou být libovolné.

Zpracování gramatiky

Zpracování produkčních pravidel probíhá v opačném pořadí, odpředu dozadu. Např. pravidlo <fnc><expr1><expr2> se zpracovává jako <expr2><expr1><fnc>. Zásadní rozdíl mezi neterminály <fnc> a <expr> spočívá v tom, že symbol <fnc> se rozbalí na právě jeden terminální symbol. Naproti tomu neterminál <expr> může obecně představovat libovolný počet členů funkce na různých úrovních stromu výsledného fenotypu. Je-li fenotyp realizovaný pomocí stromu objektů, pak objekt, který vznikne překladem neterminálu <fnc> je zodpovědný za zpracování částí fenotypu, které vzniknou zpracováním <expr1> a <expr2>. Pravidlo lze tedy znázornit také jako strom

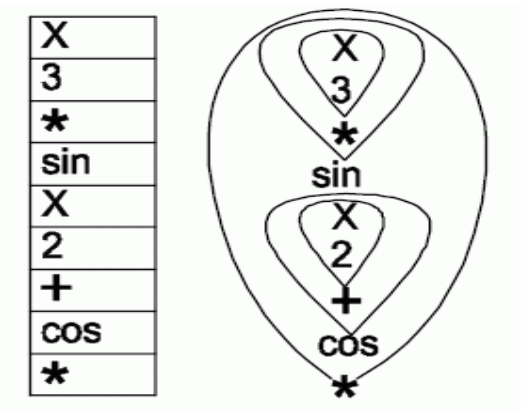
```
        <fnc>
       ../  \
<expr1>  <expr2>
```

Pro výběr pravidla (výběr typu stromu) je potřeba právě jeden gen, pro jeho zpracování je potřeba n-genů a pro výběr terminálního symbolu funkce je potřeba opět jeden gen. Pozn.: při zpracování pravidla v pořadí <fnc><expr1><expr2> je kořen zpracovávaný jako druhý a poslední zpracovávaný výraz je jeden z listů podstromu.

Pozn: konstrukce algoritmu umožňuje velmi jednoduché přídání libovolné fce. Např. přidaní fce tří proměnných vyžaduje pouze přidaní nové funkce do

množiny funkcí, a přidání pravidla <expr> <expr> <expr> <fnc> nebo <num> <num> <num> <num> nebo jejich kombinací. Výběr funkcí z množiny je implementován pomocí virtuálního dvojrozměrného pole, tzn., že všechny funkce se vkládají do jednoho seznamu, ale jejich výběr probíhá tak, jakoby byly oddělené seznamy funkcí stejného počtu proměnných.

Při klasickém zpracovaní v pořadí <fnc> <expr> <expr> popsaném v [6] nelze jednoduše najít stromovou strukturu z genotypu, protože části <expr> <expr> spotřebovávají předem neznámý počet genů a poslední zpracovávaný neterminál je (z hlediska struktury) nevýznamný list podstromu. Navržený způsob kódování stromové struktury je zobrazen na obr.1. Opačný směr kódování stromové struktury používají zpětné algoritmy PGE, které pomocí vnitřní informace (začátku bloku BB a konce bloku EB), kterou se doplní genotyp, umožňují efektivní křížení a cílenou vyšší
pravděpodobnost mutace parametrů fenotypu a nižší pravděpodobnost mutace struktury fenotypu.



Obr.1 Opačný směr kódování stromové struktury

Existuje i složitější způsob zpracování gramatiky, který ještě více používá přírodní způsob kódování DNA. Algoritmus (chemické genetické programování) si sám přidává nová potřebná produkční pravidla. Velice efektivní kódování se dá dosáhnout uzavřením genotypu do kruhové struktury. Jednotlivé geny se pak mohou číst několikrát, přičemž jejich význam je jiný, což je odvoze z kontextu kódu. Tímto způsobem jsou kódovány např. viry.

Promítání fenotypu do genotypu

Při aplikaci tohoto obráceného systému kódování začíná pak každé zpracování podstromu fenotypu výběrem pravidla a končí kořenem tohoto podstromu (v našem případě vlastní funkce). Z pohledu genotypu je pak gen použitý pro výběr pravidla následovaný *n* geny s různou interpretací (použitými pro zpracování pravidla), které jsou však následovány genem určujícím terminální symbol <fnc>. Tedy gen určující pravidlo je párový s genem určujícím kořen podstromu pravidla. Tyto geny lze při překladu chromozomu na tělo jedince označovat. Nejjednodušší značení může vypadat takto:

BB - Begin Block
IB - Inside Block
EB - End Block

Značky EB,BB jsou párové, vymezují v chromozomu úseky, které se mohou vnořovat, nemohou se však křížit (jako závorky). Značka IB párová není, ale je vždy obsažená uvnitř nějakého páru. Nalezení příslušného EB genu k zadanému IB genu lze tedy provést pomocí jednoduchého zásobníku (LIFO).

Část chromozomu vymezená jedním párem BB-EB genů pak kóduje jistý podstrom fenotypu. Takovýto blok chromozomu je zcela samostatný a lze ho vyměnit s jiným blokem. Všechny BB geny (a pouze BB geny) určují strukturu těla jedince. EB a IB geny určují terminály, které se objeví ve výsledném fenotypu. BB geny určují výsledný strom jedince (počet podstromů, větví a listů) jsou tedy strukturální, jejich změna ovlivní výběr pravidla a tedy interpretaci ostatních genů. Změna (např. mutace) BB genu způsobí změnu ve výběru pravidla (výsledků operace modulo) mnoha ostatních genů.

Toto označování bloků zavádí zpětnou vazbu fenotypu na genotyp a přitom nesnižuje obecnost algoritmu. Nezávisí totiž na použitých terminálech ani pravidlech, ale pouze na stromové struktuře. Díky tomuto systému je možné podstatné zvýšení výkonu algoritmu nedestruktivním křížením a mutací.

Křížení

Při použití gramatiky závisí výsledný fenotyp jednak na hodnotě genu a jednak na jeho kontextu. Při křížení v náhodném bodě obvykle dochází ke změně kontextu a tedy i ke změně fenotypu. Křížení je tedy destruktivní, neboť nové části chromozómu kódují jiný fenotyp než v původním jedinci. Pokud se však využije označených bloků v chromozomu křížení není destruktivní.

Křížení probíhá jako výměna částí chromozomů dvou jedinců, kde každá část (blok) začíná BB genem a končí odpovídajícím EB genem. BB a EB geny jsou párové značky, proto k výměně dochází vždy v sudém počtu bodů. Výběr BB genů je náhodný a je z něj vyloučen první gen. První gen je párový s posledním genem a uzavírá tedy celé tělo jedince, výměna takovýchto bloků by znamenala pouze záměnu obou jedinců. Křížit lze dokonce i stejné chromozomy.

Tento způsob křížení pracuje podobně jako přímé křížení podstromů funkcí, nicméně stále se vyměňují

pouze části chromozomu, tedy genotyp a fenotyp zůstává oddělený. Po křížení je tedy vzniklý jedinec kombinací podstromů obou rodičů, ale známe jeho genotyp, který tuto strukturu může kdykoliv vytvořit. Lze tedy i zjistit pravidla, která jsou pro vytvoření jedince použitá. Kdyby se křížily přímo podstromy, byla by na zjištění použitých pravidel potřeba zpětná analýza stromu. Další výhodou tohoto způsobu křížení je, že k vlastnímu křížení není nutné znát fenotyp ani s ním nijak manipulovat (kombinování stromových struktur vs. výměna prvků jednorozměrného pole). Tato znalost struktury kódování umožňuje velice efektivně provádět křížení (viz obr.2). U prvního rodiče se náhodně vybere gen s označením BB (začátek podbloku). K němu automaticky přísluší daný gen EB (konec podbloku). Vybraný podstrom definovaný geny BB a EB (16 a 27) se vymění s podblokem druhého rodiče, který byl vybrán stejným způsobem (definován geny 38,8,78,22,45). Jednotlivé podbloky jsou podstromy fenotypu.

Mutace

Mutaci lze rozdělit na mutaci strukturálních genů. Mutace jednoho strukturálního genu ovlivní ostatní geny, její míra by proto měla být velmi malá (0%-4%). Míra strukturální mutace může být dokonce i nulová, protože nové struktury lze vytvářet progresivním křížením. Rozlišení mutací umožňuje nastavení poměrně vysoké mutace nestrukturálních genů a tedy i rychlejší prohledávání. Přitom nehrozí poškození již nalezené struktury. Pokud jsme schopni určit, že struktura jedince je správná, lze nastavit strukturální mutaci na 0% a mutovat pouze terminální symboly. Např. z funkce sin(2 + x) + cos(3 * x) lze pouze pomocí mutace vygenerovat funkci cos(5 - x) * sin(1 * x).

Populační model

Systém používá tři populace uspořádané do následující struktury, ve které se rozlišuje pohlaví jedinců:

Master
/ \
Female Male

Tento systém se jeví jako stabilnější, než systém ve kterém jsou všechny 3 populace stejné.

Samičí populace

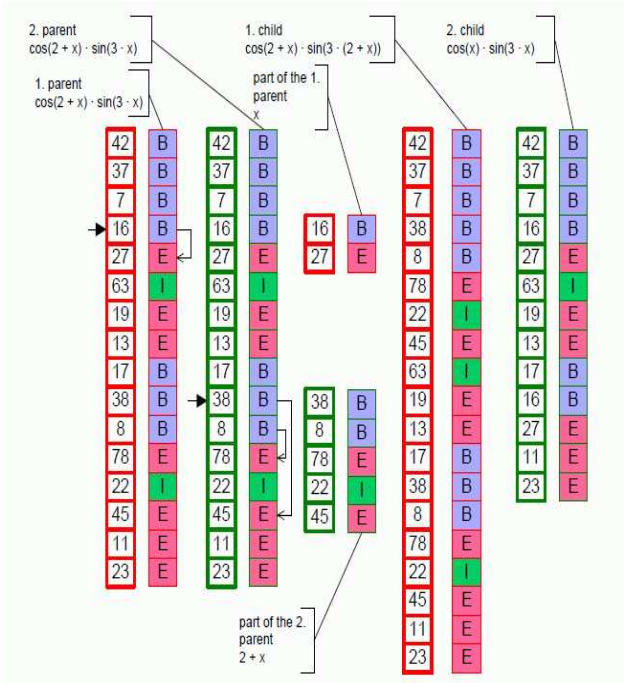
Při zařazování nového jedince do populace se kontroluje jestli již v populaci neexistuje stejný nebo podobný jedinec. Pokud existuje, pak není nový jedinec do populace zařazen. V samičí populaci se tedy vyskytuje každý genotyp pouze jedenkrát a podobné genotypy také pouze jedenkrát. Takto se v populaci udržuje vysoká míra rozmanitosti, jedinci v populaci tedy vůbec nepodléhají mutaci. Vyřazování jedinců probíhá na základě dvou kritérií, nejprve podle věku (délka pobytu v systému, nikoli v populaci), potom podle fitness, tj. omezení na maximální dovolenou velikost populace. Výběr rodičů probíhá metodou tournamentu.

Samčí populace

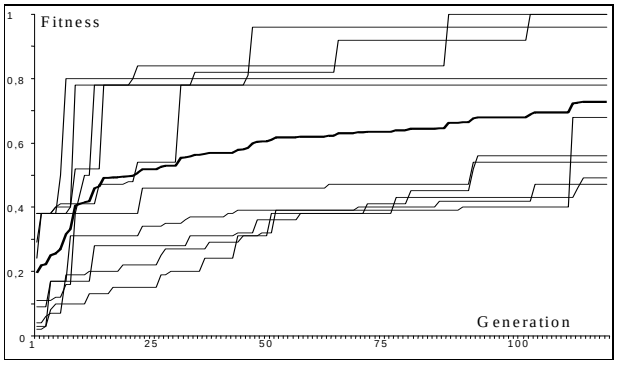
Není kontrolováno přidávání nových jedinců, populace proto podléhá mutaci. Míra mutace může být vysoká (až 30%) pokud je mutace strukturálních genů nízká (asi 1%). Pro několik nejlepších jedinců jsou mutace nedestruktivní. Pokud má být jedinec mutován, zařadí se do populace jeho klon. Při stagnaci řešení se mutace lineárně zvětšuje s počtem cyklů, po které se nezlepšilo řešení. Nejhorší jedinci v populaci mají pravděpodobnost výběru velmi malou (0,02), ale vždy nenulovou.

Master populace

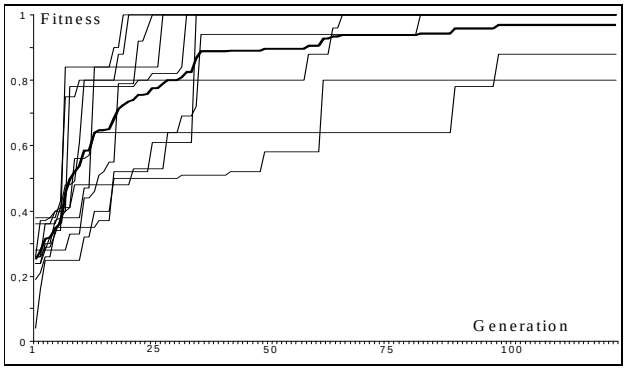
Master populace je populace nadřazená samčí a samičí podpopulaci. Ze samčí a samičí podpopulace dostává v pravidelných intervalech nejlepší řešení. Zároveň provádí vlastní optimalizaci, pro křížení jsou oba rodiče vybíráni pouze z master populace systémem tournamentu. Používá stejný systém mutací jako populace samčí, včetně dynamických mutací. Vyřazování jedinců však probíhá pouze podle fitness (omezení na maximální povolenou velikost populace). Tato populace tedy slouží zároveň jako archiv nejlepších řešení celého systému.



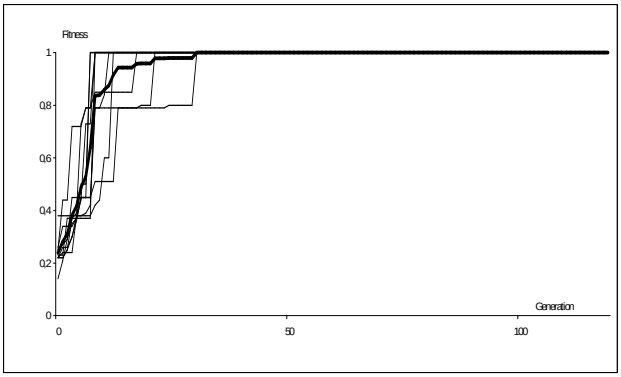
Obr.2 Princip křížení s využitím označení pro podstromy mezi B (BB) a E (EB)



Obr.3 PGE s dopředným algoritmem



Obr.4 PGE s zpětným algoritmem (bez dvouúrovňové hierarchické struktury)

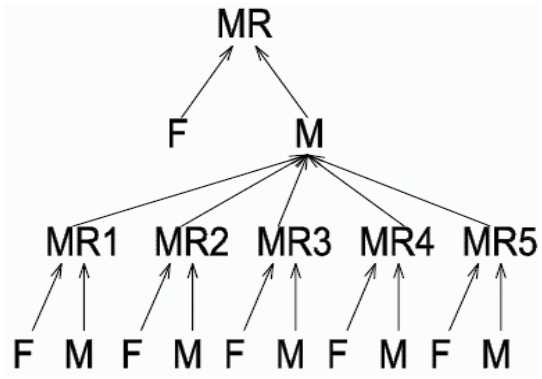


Obr.5 Vliv paralelní hierarchické struktury zobrazené na Obr.6 na rychlost výpočtu danou počtem generací pro PGE se zpětným algoritmem

Hodnotící funkce (fitness)

Okolo hledané funkce, definované funkčními hodnotami je toleranční pásmo dané šířky. Fitness jedince je pak poměr počtu vyhovujících bodů a všech kontrolních bodů. Použitý počet kontrolních bodů byl 100. Vyhovující bod je takový, ve kterém hodnota generované funkce leží v tolerančním pásmu okolo hledané funkce. Tento způsob hodnocení vytváří silný selekční tlak, algoritmus proto obvykle velmi rychle nalezne přibližné řešení.

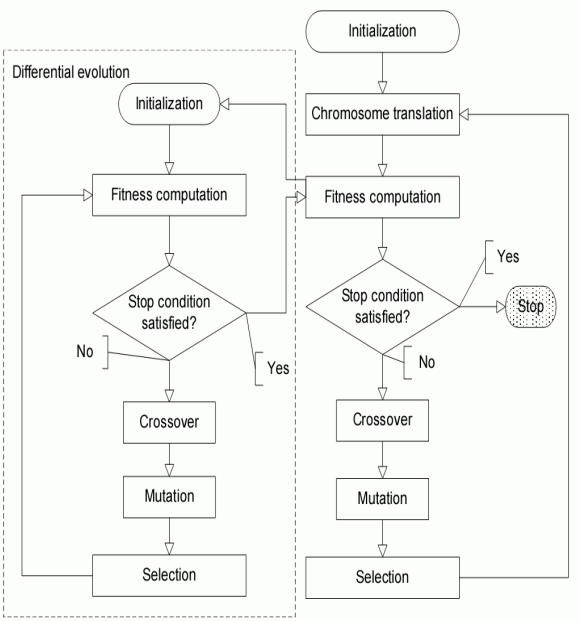
Díky použití progresivního křížení s využitím informace BB a EB a rozlišení mutací u samčí populace lze řešit složitější funkce v kratším čase. Dále použití paralelní struktury GE s rozlišeným pohlavím je konvergence rychlejší a méně závislá na opakování algoritmu. Na obr.4 je vývoj funkce fitness Y v závislosti na počtu generací x pro hledanou funkci: sin(2+x)*cos(3+x). Dosažení hodnoty jedna znamená, že vytvořená funkce leží uvnitř všech tolerančních oblastí zadaného průběhu neznámé funkce.



Obr.6 Hierarchická struktura PGE s šesti PC s rozlišeným pohlavím (F - samičí, M - samčí)

Výsledky symbolické regrese pomocí PGE s třemi různými algoritmy vždy pro 10 běhů algoritmu jsou na obr.3 až obr.5. Tučně je vždy zobrazen průměrný algoritmus. Jak je patrné z obrázků nejrychleji konverguje PGE s hierarchickou strukturou s rozlišeným pohlavím podle obr.5. Současně má i nejmenší rozptyl mezi jednotlivými běhy algoritmu. Zpětný algoritmus PGE je rychlejší než původní algoritmus popsaný v [6] a [7]. Jednotlivé typy algoritmů lze kombinovat a vytvořit trak např. dvouúrovňový PGE zobrazený na obr.7, který je navíc rozšířen o použití diferenciální evoluce (DE). Algoritmus využívá výhod obou metod. PGE lépe najde optimální strukturu a DE rychleji najde nastavení parametrů pro danou strukturu. Díky použití progresivního křížení s využitím informace ve sloupci G na obr.2 a rozlišení mutací u samčí populace lze řešit složitější funkce v kratším čase. Dále použití paralelní struktury GE s rozlišeným pohlavím je konvergence rychlejší a méně závislá na opakování algoritmu [9]. K křížení s rozlišeným pohlavím jsou zapotřebí dvě podpopulace (samčí a samičí). Příroda našla optimum ve

dvou paralelně fungujících podpopulacích. Cílem je rychlá konvegence a současně dobrá adaptace. Toho se právě dosáhlo rozdělením na dvě paralelní struktury. V každé ale probíhá jiný způsob selekce. Velice názorný je příklad harému. Mezi samci probíhá boj kdo bude vlastníkem harému, což může být jen ten z nějakého hlediska nejlepší (řeší se tak problém konvergence) a naopak do samčí podpopulace se přidává pouze samice s jiným genotypen (nebo fenotypem). Vítěz boje samců se pak zkříží se všemi samicemi. Samice dodávají tak do systému rozmanitost a zprostředkovaně přes potomky dobrou adaptivní schopnost.



Obr.7 Dvouúrovňová PGE kombinovaná s diferenciální evolučním algoritmem

Tím, že se oddělilo řešení problému konvergence a adaptace do dvou oddělených populací s jinými typy selekce, tento paralelní systém se vyznačuje současně dobrou konvergencí i dobrými adaptivními vlastnostmi. Pokud se vše to spojí pouze do jedné populace systém může např. dobře konvergovat ke globálnímu řešení ale nemusí být schopen sledovat dynamické změny prostředí a dostatečně rychle se na ně adaptovat. To se projeví jako měnící se fitness.

4 Výsledky testování pro logickou funkci XOR

Pro dvě logické proměnné a, b; < 0, 1 > je výstupní hodnota c pro logickou funkci XOR popsána následujícími trojicemi (a, b, c):
P = {(0, 0, 0); (0, 1, 1); (1, 0, 1); (1, 1, 0)}.

Trénovací množinou byla pravdivostní tabulka logické funkce XOR. Funkce XOR (+) může být vyjádřena pomocí logických funkcí OR (_), AND (^), NOT (¬):
a + b = (a ^ ¬b) _ (¬a ^ b) = (a _ b) ^ (¬a _ ¬b) = (a _ b) ^ ¬(a ^ b)

Výsledky automatického generování programu pomocí PGE jsou následující:

1a) po 30 generacích

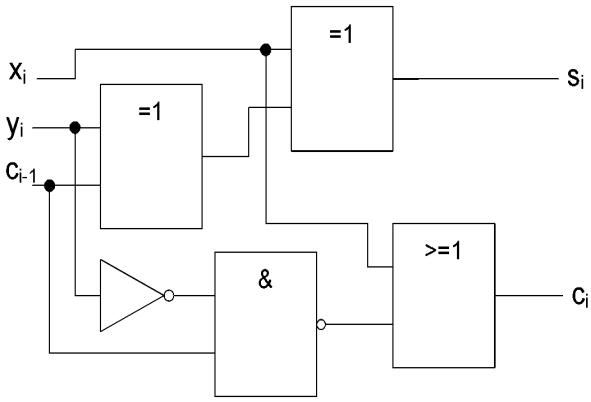
```
function xxor($a,$b) {
$result = "no_value";
$result = ($result) |
((((~(~(~(~(~($result)))))) | (($a) & (($a) & (~($b))))) &
        ($a))
| ((~($a)) & ($b)));
return $result;
```

1b) po 58 generacích

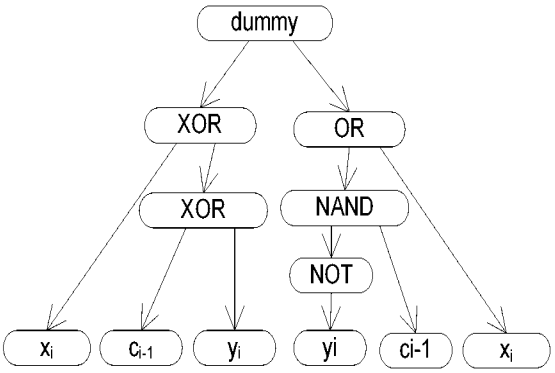
```
function xxor($a,$b) {
$result = "no_value";
$result = ($result) | (((~$result | ($a & ($a & ~$b))) & $a)
| (~$a & $b));
return $result;
```

5 Optimalizaci elektronických obvodů

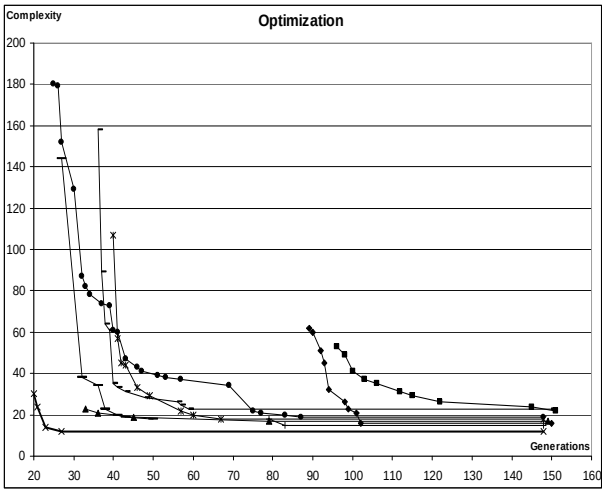
Paralelní GE byla též testována na problému optimalizace obvodu sčítačky. Cílem bylo nalezení nejjednodušší a nejrychlejší struktury.



Obr.8 Zapojení sčítačky



Obr.9 Stromová reprezentace problému na obr.8



Obr.9 Průběh optimalizace sčítačky s nejlepším řešením zobrazeným na obr.8

Jak je vidět z obr.9 nejdříve se nalezlo funkční řešení (kolem generace 80 na obr.9) a potom se minimalizoval počet nutných prvků obvodu sčítačky.

6 Závěr

Doporučené nastavení parametrů PGE:
- malý počet migrantů (1 až 3 do každé sousední podpopulace), kombinovaný se střední délkou epochy (přibližně 5 až 10% z celkového počtu generací), vede na základě zkušeností k nejlepším výsledkům,
- proměnná délka epochy, závislá na vlastnostech podpopulací v průběhu evoluce, dává o něco lepší výsledky než optimalizovaná pevná délka epochy, ale za cenu snížení časové efektivnosti (prodloužení výpočtu),
- omezení výběru migrantů pouze na lepší jedince navede k celkovému zlepšení chování algoritmu, naopak výběr pouze lepších jedinců vede k předčasné stagnaci,
- při dostatečné velikosti podpopulace vede paralelní vývoj většího počtu podpopulací ke kvalitativně lepším výsledkům, a to i při pevném celkovém počtu generací. Složitost problému a minimální velikost podpopulace musí být brána v úvahu při dělení populace na menší podpopulace.

Implementace PGE má lepší vlastnosti než sekvenční GE. Migrační a difusní modely řeší problémy vždy stejně dobře jako sekvenční GE, ale v mnoha případech najdou i lepší řešení.

Poděkování: tato práce byla podpořena výzkumným záměrem MSM21630529.

Literatura

[1] Ošmera, P., Roupec, J., Matoušek, R.: Energie, entropie a evoluce živé hmoty, sborník konference Kognice a umělý život I, Smolenice, Slovensko (2001) 203-223

[2] Ošmera, P.: Adaptace složitých systémů, sborník česko-slovenské konference Konice umělý život II, Mílovy – Česká republika, (2002) 163 – 174

[3] Ošmera, P.:Paralelní evoluce s hierarchickým uspořádánímpie sborník konference Kognice a umělý život III, Stará Lesná, Slovensko (2003) 115-124

[4] Ošmera P.: Evoluce cestou bolesti a slasti, sborník konference Kognice a umělý život IV, ediční středisko FPF SU Opava, (2004) 415-424

[5] Ošmera P.: Paralelní gramatická evoluce, sborník konference Kognice a umělý život V, ediční středisko FPF SU Opava, (2005) 471-481

[6] O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language Kluwer Academic Publishers 2003.

[7] O'Neill, M., Brabazon, A., Adley C.: The Automatic Generation of Programs for Classification Problems with Grammatical Swarm, Proceedings of CEC 2004, Portland, Oregon (2004) 104 – 110

[8] Piaseczny, W., Suzuki. H., Sawai, H.: Chemical Genetic Programming – Evolution of Amino Acid Rewriting Rules Used for Genotype-Phenotype Translation, Proceedings of CEC 2004, Portland, Oregon (2004) 1639 - 1646.

[9] Zelinka I, Oplatková Z., Šeda M., Ošmera P, Včelař F.: Evoluční výpočetní techniky - principy a aplikace, BEN, 2009