Biologií inspirovaný vývin jako technika evolučního návrhu

Michal Bidlo

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 61266 Brno, Česká republika
bidlom@fit.vutbr.cz

Abstrakt

Biologický vývin (angl. development) mnohobuněčných organizmů - proces ontogeneze - poskytuje v posledních několika letech významnou inspiraci pro návrháře v oboru výpočetní inteligence (tzv. soft computing), zejména ve spojení s evolučními algoritmy. Cílem příspěvku je seznámit se základními principy ontogeneze a různými podobami developmentu inspirovanými tímto biologickým procesem, které byly v nedávné době úspěšně využity v oblasti evolučního návrhu elektronických systémů. Budou zmíněny nejznámější modifikace a rozšíření základních technik společně s novými směry výzkumu, které jsou na těchto principech založeny.

1 Úvod

Biologií inspirované algoritmy se v poslední době stále více uplatňují při řešení mnoha komplexních úloh v různých oblastech vědy. Evoluční návrh patří k relativně mladému a rozvíjejícímu se odvětví, které nám dosud umožnilo získat mnoho zajímavých výsledků. Jako příklad uveďme kreativní evoluční návrh nebo formy umělého života [1], číslicové obvody [2], neuronové sítě [3], analogové elektronické obvody nebo počítačové programy [4]. Přestože bylo aplikováno několik odlišných metod evolučních algoritmů, bylo možné takto získat pouze relativně jednoduchá řešení.

Problém škálovatelnosti patří pravděpodobně k nejvýznamnějším aspektům, které znesnadňují proces evolučního návrhu s využitím současných technologií. Se zvyšující se složitostí cílového systému (např. nárůst počtu vstupních a výstupních veličin nebo počtu komponent potřebných k jeho implementaci) vzrůstá délka chromozomu reprezentujícího kandidátní řešení během evolučního procesu, což má za následek zvětšení vyhledávacího prostoru a je tudíž velmi náročné navrhnout efektivní prohledávací (návrhový) algoritmus.

Dosud bylo vyvinuto několik odlišných přístupů snažících se problém škálovatelnosti překonat. Patří mezi ně *evoluce na úrovni funkčních bloků* (složitějších celků tvořících stavební kameny cílového systému -- viz např. [5]), *inkrementální evoluce* (viz např. Toressenova metoda *rozděl a panuj* [6]) a tzv. *development* (postupný *vývin* cílového systému podle určitých pravidel - inspirován biologickými principy ontogeneze, viz např. [7, 8, 9]).

Cílem tohoto příspěvku je poskytnout přehled principů nejpoužívanějších výpočetních modelů a technik aplikovaných pro development v oblasti evolučního návrhu spolu s možnými aplikacemi, zejména z oblasti návrhu číslicových obvodů.

2 Evoluční algoritmy a evoluční návrh

Evoluční algoritmy jsou stochastické vyhledávací systémy využívající principů Darwinovy evoluční teorie. Pod tímto označením však obvykle nalezneme celou řadu modifikací různých biologií inspirovaných technik, z nichž zřejmě nejznámější jsou genetické algoritmy [17], genetické programování [15], evoluční strategie [18, 19] a evoluční programování [20]. Tyto typy evolučních algoritmů se navzájem liší zejména ve způsobu reprezentace *kandidátních řešení* (potenciálních řešení daného problému) a jejich zpracování v průběhu umělého evolučního procesu. Kandidátní řešení (tzv. *fenotypy*) jsou zakódována do c*hromozomů* představujících prvky vyhledávacího prostoru algoritmu (tzv. *genotypy*). Chromozom se též obvykle nazývá *jedinec*. Podoba vyhledávacího prostoru závisí na řešeném problému, kterým může být například optimalizace parametrů číslicového filtru, optimalizace tvaru lopatky parní turbíny, návrh počítačových programů, kombinačních logických obvodů atd. Míra úspěšnosti kandidátního řešení je měřena pomocí tzv. *fitness funkce*, která pro daný problém vyhodnotí výsledek řešení s využitím informací obsažených v chromozomu. Chromozomy podstupují modifikace *genetickými operátory* (typicky křížení a mutace – inspirováno v biologii) zajišťujícími výměnu genetických informací mezi chromozomy a tvorbu nových kandidátních řešení (*potomků*). Evoluční proces sestává z posloupnosti generací, v pruběhu kterých jsou pro aplikace genetických operátorů upřednostňováni kvalitní jedinci, kteří mají větší šanci „přežít“ do dalších generací a vyprodukovat tak více kvalitnějších potomků. V důsledku tohoto procesu, reprezentujícího tzv. *selekční tlak*, dochází k postupnému zlepšování kvality hledaných řešení.

Evoluční algoritmy bývají v posledních letech hojně využívány k řešení náročných problémů v různých

oblastech. Jejich aplikace je možné rozdělit podle účelu využití evolučního algoritmu na (1) e*voluční optimalizace* a (2) *evoluční návrh*. (Tento příspěvek je zaměřen zejména na evoluční návrh.) Zásadní odlišnost mezi těmito přístupy je následující: Evoluční optimalizace využívá znalostí o již hotovém fungujícím systému, z něhož je extrahována množina parametrů ovlivňujících kvalitu jeho funkce. Evoluční algoritmus je v tomto případě využit k nalezení vhodné konfigurace těchto parametrů tak, aby daný systém fungoval co nejefektivněji, tj. evoluce je využita k optimalizaci daných vlastností již známého systému. Cílem evolučního návrhu je však vlastní *design* samotného systému. Na jeho počátku je známa pouze požadovaná funkce systému, množina komponent, ze kterých má být cílový systém sestaven, a pravidla jejich možného propojení. Evoluční algoritmus je v takovém případě využit k *návrhu* systému s využitím těchto znalostí. Do jisté míry lze oba přístupy kombinovat, např. v případě návrhu logických obvodů je možné klást důraz na řešení s menšími počty hradel, nižším zpožděním apod. Je zřejmé, že evoluční návrh je mnohem náročnější jak pro vývojáře (inženýra), který musí navrhnout vhodný evoluční algoritmus a reprezentaci kandidátních řešení, tak pro evoluční algoritmus samotný z důvodu enormního rozsahu vyhledávacího prostoru a jeho vlastností s ohledem na dostupné výpočetní prostředky. V mnoha případech tak má celý proces podobu ryze experimentální práce.

3 Biologický development

Ontogeneze (development) je biologický proces, během kterého dochází k vývinu zárodečné buňky (zygoty) ve vícebuněčný organismus. Základním principem developmentu je stavba organismu a sebeorganizace. Podstatnou částí ontogeneze je proces konstrukce, který je dán souhrou mezi geny, proteiny, buňkami a prostředím, které buňky obklopuje. Výsledkem tohoto procesu je vyspělý organismus. V průběhu vývoje zygoty můžeme pozorovat tyto základní fáze [21] (poznamenejme, že tyto dílčí procesy neprobíhají zcela odděleně, ale obvykle se překrývají):

- *Buněčné dělení*. Zárodečná buňka (tzv. zygota) podstoupí v této fázi rapidní dělení, jehož výsledkem je utvoření shluku buněk (tzv. blastula). Nedochází zde však k žádnému nárůstu objemu jednotlivých buněk.
- *Formování tvaru organismu*. V této fázi dochází k prostorové a časové organizaci buněk v embryu. V prvním stupni této fáze je vytvořen prostorový koncept organismu. Během druhého stupně jsou vytvořeny základní rysy těla.
- M*orfogeneze*. Buněčné migrace, dochází k přemísťování buněk, utváří se základ vnitřností. Tento proces se nazývá gastrulace, z blastuly se utvoří tzv. gastrula.
- *Buněčná diferenciace*. Jednotlivé buňky získají vlastní strukturu a funkci, vznikají odlišné typy buněk, např. nervové, svalové atd.
- *Růst*. Organismus nabývá na velikosti (rozmnožování buněk, nárůst buněčné hmoty).

Principy ontogeneze bývají mnohdy využívány buď samostatně nebo v kombinaci s jinými biologií inspirovanými metodami. Jelikož je development ve skutečnosti velice komplikovaný proces, bývají jeho modely podstatně zjednodušeny, případně simulují pouze některé jeho části. V oblasti počítačového inženýrství je development primárně chápán jako mapovací proces z genotypu na fenotyp v evolučním algoritmu. Genotypy představují vhodně zakódovaná kandidátní řešení do chromozomů evolučního algoritmu a fenotypy reprezentují původní instance řešení daného problému, které jsou předmětem vyhodnocení pomocí fitness funkce. V případě aplikace modelu vývinu v kombinaci s evolučním algoritmem obsahuje genotyp *předpis* pro konstrukci cílového řešení (fenotypu).

Nádledující kapitoly popisují principy nejznámějších modelů vývinu v oblasti počítačového inženýrství, mezi které patří zejména celulární automaty a Lindenmayerovy systémy.

4 Celulární automaty

Celulární automaty (CA) původně zavedli Ulam a von Neumann ve čtyřicátých letech minulého století jako matematický model pro vyšetřování chování složitých systémů a sebereplikace [10]. Celulární automaty jsou dynamické systémy tvořené diskrétní soustavou *buněk*, z nichž každá se může nacházet v jednom *stavu* z konečné množiny stavů. Geometrické uspořádání buněk CA je specifikováno jeho *dimenzí*, která může být jednorozměrná, dvourozměrná, případně i vícerozměrná. Stavy buněk jsou aktualizovány synchronně v diskrétních časových krocích v závislosti na *lokálním přechodovém pravidle*, které určuje následující stav každé buňky v závislosti na aktuálním stavu této buňky a na stavech buněk v jejím *sousedství*. Celulární automat lze tedy plně charakterizovat geometrií buněčné mříže, tvarem sousedství, počtem stavů buňky a množinou lokálních přechodových pravidel [11].

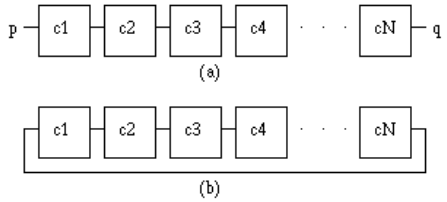
Do dnešní doby mají celulární automaty velké množství aplikací v různých oborech. Pomocí celulárních automatů byl popsán například růst krystalů, transport sněhu větrem s erozí a depozicí, difuze tepla a znečištění, a dále turbulentní proudění, tok lávy, šíření vln v heterogenním prostředí, dynamika mísitelných a nemísitelných tekutin a pórovitých látek, silniční transportní systémy a mnoho

biologických systémů od genetické úrovně počínaje až po systémy rostlin a ekologické interakce. Mohou být použity k modelování různých nástrojů, používaných v teorii formálních jazyků, jako Turingova stroje a jiných výpočetních systémů [12, 13]. Pravděpodobně nejznámější aplikací celulárních automatů je Conwayova hra Život (Game of Life) představující simulaci umělého života buněk v dvourozměrném prostoru.

V následujících podkapitolách uvedeme klasifikaci celulárních automatů z pohledu geometrie buněčné mříže a dále ve vztahu k charakteru množiny lokálních přechodových pravidel. Podrobně se zaměříme na jednorozměrné a dvourozměrné CA, jelikož právě tyto bývají používány nejčastěji.

4.1 Jednorozměrné celulární automaty

Jednorozměrný celulární automat (1D CA) je tvořen lineárním uspořádáním buněk (Obr. 1), v němž sousedství každé buňky sestává z r buněk bezprostředně obklopujících danou buňku z levé i pravé strany a této buňky samotné. Parametr r se nazývá *rádius*. V sousedství každé buňky v 1D CA se tedy nachází $2r+1$ buněk.

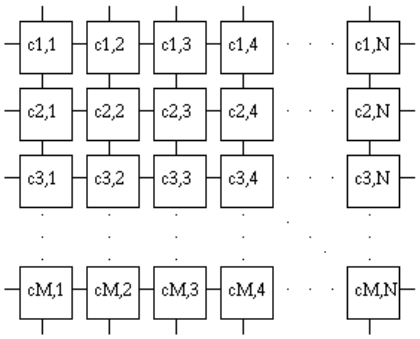


Obr. 1. Jednorozměrný CA: (a) konstantní okrajové podmínky, (b) cyklické okrajové podmínky

Uvažujeme-li konečný počet buněk celulárního automatu, je potřeba přesně vymezit sousedství pro dvě buňky nacházející se na okrajích celulární struktury, tzv. *okrajové podmínky* celulárního automatu. Mezi nejčastěji používané okrajové podmínky patří tzv. *konstantní* podmínky, kdy je pomyslným (chybějícím) sousedům buněk na okrajích CA přiřazen vybraný stav z konečné množiny stavů, který se v průběhu vývoje CA nemění (Obr. 1a). *Cyklické* okrajové podmínky představují další variantu vymezení sousedství buněk na okraji CA, které přiřazují levému, resp. pravému sousedovi nejlevější, resp. nejpravější buňky 1D CA hodnotu aktuálního stavu nejpravější, resp. nejlevější buňky CA (Obr. 1b).

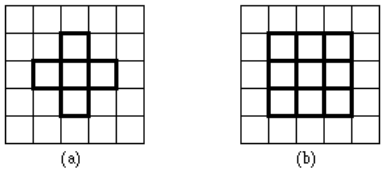
4.2 Dvourozměrné celulární automaty

Nejčastěji používaným vícerozměrným celulárním automatem je dvourozměrný CA, jehož buňky jsou uspořádány do tvaru mříže složené z M řádků a N sloupců (viz Obr. 2).



Obr. 2. Struktura dvourozměrného celulárního automatu

Ačkoliv je teoreticky možné uplatnit podobný princip definice sousedství buněk využívající rádius (v 2D prostoru), bývá obvykle specifikováno jedno z následujících typů sousedství, které v podstatě odpovídá parametru *r=1:* (1) *Von Neumannovo* sousedství, zahrnující čtyři bezprostřední sousedy dané buňky v horizontálních a vertikálních směrech a tuto buňku samotnou, viz Obr. 3a a (2) Moorovo sousedství, zahrnující osm bezprostředních sousedů v horizontálních, vertikálních a diagonálních směrech a danou buňku samotnou, viz Obr. 3b.



Obr. 3. Typy sousedství dvourozměrného CA: (a) Von Neumannovo, (b) Moorovo

Analogicky k 1D CA jsou nejčastěji definovány konstantní okrajové podmínky (přiřazení vybraných stavů pomyslným chybějícím sousedům buněk na hranici celulární struktury), případně cyklické (protilehlé buňky na obvodu CA jsou propojeny a 2D CA tak získává podobu toroidu

4.3 Uniformní a neuniformní CA

Celulární automaty libovolné dimenze je možné klasifikovat podle charakteru množiny lokálních přechodových pravidel. V základním principu CA obsahuje každá buňka vlastní lokální přechodové pravidlo (odtud jeho název – přechodové pravidlo specifikuje chování buňky v závislosti na *lokálních* podmínkách této buňky, t.j. na stavech buněk v jejím blízkém okolí. Přestože lokální přechodová pravidla mohou být sama o sobě velice jednoduchá (vyjádřena např. základními logickými funkce AND, OR, XOR

apod.), celulární automat může vykazovat velmi složité *globální* chování.

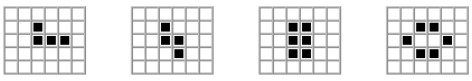
V obecném případě obsahují různé buňky CA různá lokální přechodová pravidla. Takový celulární automat se nazývá *neuniformní*, případně *hybridní*. Jsou-li lokální přechodová pravidla všech buněk totožná, jedná se o speciální případ CA, tzv. *uniformní* celulární automat. Mimo tyto třídy automatů je možné vyčlenit tzv. *kvazi-uniformní* CA, ve kterých převažují buňky obsahující identické přechodové pravidlo a zbývající buňky obsahují konečný počet přechodových pravidel od dominantního pravidla odišných.

4.4 Aplikace celulárních automatů

Pravděpodobně nejznámější aplikaci celulárních automatů představuje tzv. **Hra Life (The Game of Life)** objevená Johnem Conwayem koncem 60. let minulého století. Původní myšlenka této aplikace byla publikována v [14]. Jendá se o simulaci života buněk s využitím uniformního 2D celulárního automatu, který využívá Moorovo buněčné sousedství. Lokální přechodové pravidlo určující chování každé buňky je následující:

1) Živá buňka přežívá, jsou-li v jejím okolí 2 nebo 3 živé buňky.
2) Živá buňka hyne následkem přemnožení, vyskytují-li se v jejím okolí 4 živé buňky a více, případně v důsledku osamocení, je-li v jejím okolí pouze jedna nebo žádná živá buňka.
3) Obklopují-li prázdnou buňku právě 3 živé buňky, zrodí se na tomto místě nová živá buňka.

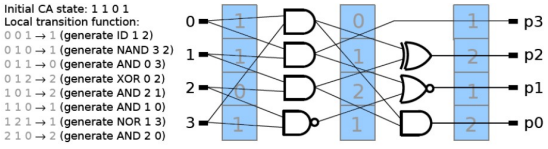
Hra Life je typickou ukázkou komplexního globálního chování celulárního automatu na základě jednoduchých pravidel aplikovaných lokálně na každou jeho buňku. Příklad vývinu celulárního automatu simulující hru Life znázorňuje Obr. 4.



Obr. 4. Příklad vývinu celulárního automatu implementujícího Conwayovu hru Life

Jako druhý přiklad aplikace celulárního automatu bude uveden systém pro **evoluční návrh kombinačních logických obvodů na úrovni hradel.** Základ systému tvoří jednorozměrný uniformní celulární automat využívající sousedství o parametru *r=1* a nulové okrajové podmínky. Pro potřeby návrhu číslicových obvodů byl tento základní model CA rozšířen o tzv. *generativní schopnost buněk* po každém jeho vývojovém kroku. S každou kombinací stavů buněk v definovaném sousedství je asociován *symbol* tvaru [*F*, *a*, *b*], který je interpretován jako logické hradlo reprezentující stavební blok obvodu. *F* označuje funkci hradla (např. AND, OR , NOT apod.) a znaky *a*, *b* představují indexy primárních vstupů obvodu, případně výstupů hradel vygenerovaných v předchozím kroku CA, na které budou připojeny vstupy daného hradla. Lokální přechodové pravidlo je možné specifikovat tabulkou s řádky ve tvaru (s_{i-1}, s_i, s_{i+1}, s' [*F*, *a*, *b*]) pro všechny možné kombinace stavů buněk v sousedství s_{i-1}, s_i, s_{i+1}, kde s_{i-1} označuje aktuální stav levého souseda *i*-té buňky, s_{i+1} představuje aktuální stav pravého souseda, s_i reprezentuje aktuální stav vyšetřované buňky a s' udává následující stav této buňky pro danou kombinaci stavů jejích sousedů. V každém kroku vývoje CA generuje každá jednotlivá jeho buňka logické hradlo specifikované symbolem [*F*, *a*, *b*] v závislosti na kombinaci stavů s_{i-1}, s_i, s_{i+1} uplatněné pro určení následujícího stavu s' dané buňky. V každém kroku CA je tedy vygenerován jeden stupeň obvodu. Vstupy hradel generovaných v prvním kroku CA (tj. vstupy prvního stupně obvodu) jsou připojeny na primární vstupy obvodu, vstupy hradel dalšího stupně tvoří výstupy hradel stupně předchozího. Zpětná vazba je nepřípustná.

Pro nalezení počátečního stavu celulárního automatu a přechodového pravidla buněk byl využit genetický algoritmus. Počet buněk CA byl v každém experimentu shodný s počtem primárních vstupů cílového obvodu. Počet vývojových kroků CA potřebný ke konstrukci daného obvodu a počet stavů byl zjištěn experimentálně. Příklad CA pro konstrukci násobičky 2x2 bity spolu s vývojem tímto CA a výsledným obvodem ukazuje Obr. 5. Kombinační násobičky bývají považovány za třídu obvodů, pro kterou je evoluční návrh na nízké úrovni (typicky na úrovni zakladních logických hradel) velmi obtížný. To potvrdil i tento přístup, v němž se nepodařilo nalézt větší instanci řešení. V případě jiných tříd obvodů však byly experimenty mnohem úspěšnější. Přínosem jsou např. 2-bitové úplné sčítačky, 6-bitové řadicí sítě, 7-bitové mediánové a až 14-bitové paritní obvody [31].



Obr. 5. Příklad vývinu celulárního automatu nalezeného genetickým algoritmem realizujícího návrh kombinačních obvodů na úrovni hradel

5 Přepisovací systémy, gramatiky

Původní myšlenku využití gramatik pro modelování biologického vývinu představil v roce 1968 Aristid Lindenmayer [22], který zavedl originální přepisovací systémy, v dnešní době nazývané Lindenmayerovy

systémy nebo L-systémy. Inspirovány biologickým vývinem, jehož procesy probíhají paralelně, jsou L-systémy ze své podstaty paralelní přepisovací systémy. Základními prvky L-systémů v původní podobě jsou buňky reprezentované symboly (tzv. *abeceda*) a množina přepisovacích pravidel tvaru $a \rightarrow X$, kde a představuje jeden symbol (buňku) a X řetězec symbolů konečné délky. Vyvíjející se objekt je reprezentován řetězcem symbolů (počáteční řetězec se nazývá *axiom*), jehož vývin probíhá paralelně (v jednom kroku jsou přepsány všechny symboly řetězce) s využitím množiny přepisovacích pravidel. Postupem času bylo publikováno množství rozšíření původních L-systémů (tabulkově řízené, parametrické atd.), které zvyšují generativní sílu L-systémů a umožňují tak tvorbu složitějších modelů nejen v oblasti biologie, viz např. [35].

5.1 Aplikace L-systémů

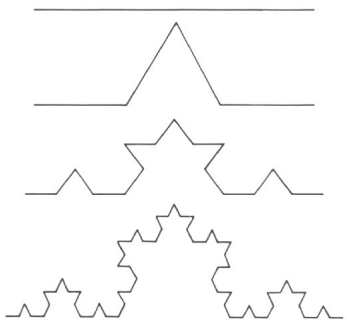
Asi nejznámější oblastí s mnoha aplikacemi využívajícími L-systémy je počítačová grafika a fraktály. Typickou ukázkou je **fraktál Kochové křivka** (Obr. 6).

fáze 0: axiom *F*

fáze 1: *F+F - - F+F*

fáze 2: *F+F - - F+F + F+F - - F+F - - F+F - - F+F + F+F - - F+F*

fáze 3, atd.



Obr. 6. Část fraktálu Kochové křivky vytvořená L-systémem s axiomem *F* a pravidlem $F \rightarrow F+F - - F+F$

Abeceda tohoto fraktálu obsahuje 3 symboly: F, +, -, jejichž interpretace je následující:

- *F*: vykresli úsečku,
- +: změň směr pro vykreslování následující úsečky o 60° proti směru hodinových ručiček,
- -: změň směr pro vykreslování následující úsečky o 60° ve směru hodinových ručiček.

Postupnou aplikací přepisovacího pravidla $F \rightarrow$ F+F - - F+F jsou paralelně nahrazovány všechny symboly F v řetězci, jehož grafická interpretace odpovídá fraktálu na Obr. 6. Poznamenejme, že tento přístup k vykreslování objektů založený na posuvech kreslicího bodu, postupném vykreslování úseček a změnách směru vykreslování bývá nazýván *želví grafika*.

Ortega a kol. publikovali metodu evolučního **návrhu L-systémů pro tvorbu fraktálních křivek o předem specifikované fraktální dimenzi** [23]. Evoluční algoritmus byl využit k nalezení vhodných pravidel L-systému, jehož symboly jsou interpretovány jako povely k vykreslování fraktálu ve stylu želví grafiky. Kvalita řešení je vyhodnocována jako rozdíl požadované fraktální dimenze a hodnoty fraktální dimenze vypočtené na základě pravidel nalezených evolucí. Rozdíl fraktální dimenze a dimenze topologické v vyjadřuje členitost fraktální křivky zkonstruované navrženým L-systémem. Topologická dimenze odpovídá počtu nezávislých proměnných potřebných k jednoznačnému určení všech bodů objektu - v případě geometricky hladkých křivek je tato dimenze rovna 1. Je-li fraktální dimenze ostře větší než dimenze topologická, je daný objekt velmi členitý. Poznamenejme, že nalezení pravidel pro konstrukci fraktální křivky o specifikované fraktální dimenzi není v mnoha případech triviální problém. Fraktály je možné v současné době využít v mnoha oblastech, například ve fyzice, chemii, astronomii, geologii, v oblasti zpracování obrazu a dalších.

Systém pro **návrh struktur číslicových obvodů s využitím modelu založeného na L-systémech** byl prezentován v [33]. Jako cílová platforma pro implementaci vytvořených obvodů bylo zvoleno programovatelné hradlové pole. Programovatelné elementy implementují homogenní strukturu 16×16 buněk, jejímž základním prvkem je tzv. *Sblock* [8], který sestává z programovatelného 5-vstupového funkčního generátoru, paměťového prvku a prostředků propojujících každou buňku s jejími čtyřmi bezprostředními sousedy celulární struktury. Kombinační logika funkčního generátoru je programována na základě výstupů všech možných 32 kombinací binárních hodnot na jeho vstupech. Ignorováním některých vstupů je možné realizovat logické funkce o méně vstupech, včetně jednovstupových propojovacích elementů pro přenos signálů ze sousedních buněk (Sbloků) propojených s danou buňkou. Konfigurací několika buněk jako přenosových elementů lze tedy docílit i přenos informace mezi bezprostředně nesousedními buňkami. Axiom L-systému představuje náhodně vygenerovanou konfiguraci (binární řetězec 32 bitů) Sbloku uprostřed celulární struktury, jehož vývoj probíhá pomocí přepisovacích pravidel navrhovaných evolučním algoritmem. Pro experimenty jsou uvažovány 2 typy přepisovacích pravidel: modifikující a růstové. Modifikující pravidla mají stejný počet symbolů na levé i pravé straně a umožňují změnu konfigurace funkčních generátorů bez rozšíření struktury obvodu. Růstová pravidla jsou určena k vytváření nových Sbloků na volných pozicích celulární struktury. Binární řetězec vyvíjený L-systémem představuje konfigurace funkčních generátorů jednotlivých Sbloků. Přepisovací pravidla jsou aplikována deterministicky na odpovídající pozice tohoto řetězce, na kterých se levá strana některého pravidla shoduje s příslušným podřetězcem. Tento podřetězec je

po aplikaci pravidla nahrazen řetězcem pravé strany pravidla, což umoňuje změnu konfigurace Sbloků a růst struktury obvodu v rámci celulární architektury. Cílem experimentů provedených v [33] byl evoluční návrh pravidel L-systému pro vývin struktury jednovstupových propojovacích elementů z náhodně vygenerované konfigurace Sbloku uprostřed celulární struktury (funkce obvodu nebyla blíže specifikována).

6 Algoritmy symbolických instrukcí

Celulární automaty a přepisovací systémy představují dva odlišné přístupy k modelování některých biologických procesů. Přestože jsou tyto techniky podloženy rigorózním matematickým aparátem a postupem času stále více pronikají do různých (zejména technických) oborů jako vhodné prostředky pro studium a řešení náročných problémů, nemusí být zcela vyhovující pro aplikace ve všech oblastech a jejich reprezentace, struktura a podstata funkce mohou představovat omezení pro některé typy úloh. S ohledem na současné požadavky v aplikacích evolučního návrhu je často vhodné realizovat specifický systém, založený na podstatě a charakteru konkrétního problému, jehož struktura, prvky a pravidla jsou přizpůsobeny konkrétním rysům řešené úlohy.

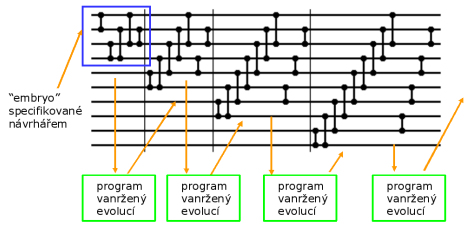
V roce 1992 představil John R. Koza originální metodu založenou na principech přírodního výběru zvanou genetické programování [15], která představuje v současné době jednu z nejznámějších a nejpoužívanějších evolučních technik pro řešení problémů z mnoha oblastí [16]. Základní myšlenka genetického programování představuje využití evoluce k automatické tvorbě počítačových programů. Ačkoliv se princip genetického programování liší od ostatních běžně používaných evolučních technik (zejména z pohledu reprezentace kandidátních řešení a genetických operátorů), klíčová myšlenka automatického programování může být využita prakticky v libovolném evolučním algoritmu. Na základě tohoto principu je možné specifokovat další model vývinu pro potřeby evolučního návrhu, který nazveme *algoritmus symbolických instrukcí*. Pro účely tohoto příspěvku budeme tento přístup chápat jako evoluční návrh programu tvořeného aplikačně specifickými instrukcemi, který představuje předpis pro řešení (případně konstrukci) určitého řešení.

Jak je zřejmé z výše uvedeného popisu, pro specifický problém je možné zvolit libovolnou vhodnou množinu instrukcí a zavést libovolnou jejich interpretaci a pravidla vykonávání takto tvořeného programu v závislosti na konkrétních rysech dané aplikace.

6.1 Aplikace vývinu symbolických instrukcí

Úspěšná metoda zaožená na aplikaci algoritmů symbolických instrukcí pro **evoluční návrh generických řadicích sítí** byla představena v [17]. Řadicí síť je kombinační obvod s N vstupy a N výstupy realizující uspořádání libovolné vstupní sekvence hodnot do neklesající posloupnosti. Základním stavebním blokem řadicí sítě je tzv. komparátor obsahující dva vstupy a dva výstupy, který porovnává vstupní hodnoty a v případě, že tyto nejsou uspořádány v požadované relaci, provede na výstupech jejich záměnu. Řadicí síť je tvořena konečnou posloupností komparátorů, jejichž vstupy jsou propojeny s primárními vstupy sítě nebo s výstupy jiných komparátorů. Vhodné propojení a uspořádání komparátorů je klíčové pro (1) optimalizaci počtu komparátorů a tedy i ceny řadicí sítě a (2) pro optimalizaci zpoždění sítě, které má zásadní vliv na rychlost řazení.

Systém pro návrh generických řadicích sítí prezentovaný v [17] je tvořen jednoduchými instrukcemi typu COPY, respektive MODIFY, které provádí kopírování určité posloupnosti již existujících komparátorů řadicí sítě, respektive modifikaci zapojení komparátorů. Na počátku je návrhářem specifikováno tzv. *embryo*, které představuje triviální instanci řadicí sítě. Cílem návrhového procesu je nalezení vhodného programu tvořeného uvedenými instrukcemi, jehož aplikací na embryo vznikce větší funkční řadicí síť, která je použita jako vstup stejného programu pro vytvoření další instance řadicí sítě atd. Řadicí síť může tedy „růst“ teoreticky nekonečně, prostřednictvím iterativní aplikace tohoto programu. K nalezení vhodné posloupnosti instrukcí tvořících program byl využit genetický algoritmus. Příklad takto navržené řadicí sítě, která je schopna genericky „růst“ ukazuje Obr. 7. Poznamenejme, že výsledné řadicí sítě vykazují lepší vlastnosti z pohledu počtu komparátorů a zpoždění v porovnání s konvenčním řešením [17]. Algoritmus konstrukce řadicích sítí zobrazený na Obr. 7 byl formálně dokázán jako obecný [32].



Obr. 7. Příklad vývinu generických řadicích sítí pomocí programu navrženého evolucí s využitím algoritmu symbolických instrukcí

7 Interakce a zavedení prostředí

V předchozích kapitolách byly popsány různé modely biologií inspirovaného vývinu založené na základních matematických technikách. Jednalo se zejména o celulární automaty, přepisovací systémy (gramatiky) a obecné algoritmy tvořené posloupností symbolických instrukcí. Vzorové aplikace těchto modelů ukázaly možnost jejich uplatnění v oblasti evolučního návrhu a schopnost řešit vybrané inženýrské problémy. Klasifikace systémů na základě použitého bázového modelu však není jediným kritériem pohledu na možné přístupy k realizaci evolučního návrhu. V této části bude zmíněn nezanedbatelný aspekt *interakce*, který může v mnoha případech podstatně ovlivnit schopnosti daného modelu. Pro potřeby tohoto článku budeme interakci chápat jako jistou formu vzájemného působení, případně výměnu informací mezi různými částmi vyvíjejícího se systému.

Nejjednodušší případ představují systémy bez interakcí, v nichž proces vývinu obvykle probíhá postupným „sestavováním“ stavebních bloků na základě pravidel vývojového modelu bez dalšího vlivu těchto komponent na strukturu či funkci cílového objektu. Koza vytvořil systém pro návrh elektronických obvodů pomocí genetického programování s vývinem (Developmental Genetic Programming) [16]. Na základě povelů zakódovaných ve stromové struktuře probíhá modifikace topologie embrya a vkládání elektronických součástek na příslušné pozice vyvíjeného obvodu. Jiný příklad systému bez interakce představuje vývin řadicích sítí prezentovaný v podkapitole 6.1 [9]. Ačkoliv vývojový systém založený na celulárním automatu prezentovaný v podkapitole 4.4 [31] ve své podstatě využívá mezibuněčné interakce celulárního automatu, z pohledu návrhu obvodů je zařazen do kategorie systémů bez interakce, jelikož celulární automat je v tomto případě použit pouze jako prostředek realizující algoritmus konstrukce obvodu, jehož komponenty (logická hradla generovaná buňkami CA) proces vývinu nikterak neovlivňují.

7.1 Vnitřní interakce v průběhu vývinu

V průběhu formování a vývinu organismů v biologii, kromě procesů probíhajících uvnitř buněk, dochází běžně také k mezibuněčné interakci. Tento fenomén poskytuje inspiraci modelům simulujícím biologické procesy, které jsou využívány k řešení úloh v oblastech umělých evolučních a jiných, biologií inspirovaných technik. Jelikož je tento proces extrémně komplikovaný, není obvykle v současné době možné využít zcela všechny jeho principy a vlastnosti. Proto je nutné do umělých, biologií inspirovaných modelů, zavádět podstatná zjednodušení, případně simulovat pouze některé partie původních biologických principů. Typickým příkladem mezibuněčné interakce biologií inspirovaného modelu je proces vývinu celulárního automatu (sousední buňky ovlivňují následující stav vyšetřované buňky) nebo interakce mezi symboly řízených L-systémů a gramatik. Je zřejmé, že způsob realizace interakce závisí na konkrétním modelu a aplikaci. Není tedy možné uvádět obecný princip vhodný pro libovolný model. Jelikož tato interakce probíhá uvnitř vyvíjejícího se systému mezi jednotlivými jeho prvky, nazveme ji *vnitřní* (*interní*) interakcí.

7.2 Vnější interakce – vývin v prostředí

Živé organismy v biologii nejsou obvykle zcela izolovány od *prostředí*, v němž vývin probíhá. Proto kromě interakcí probíhajících v organismech interně mezi buňkami hraje během vývinu nezanedbatelnou roli také vliv prostředí na vyvíjející se organismus, který je svou interakcí s ním schopen se do jisté míry *přizpůsobovat* (*adaptovat*) různým podmínkám (např. teplota, tlak, světlo atd.). Jelikož prostředí není součástí organismu v pravém slova smyslu (ovlivňuje organismus zvenku – externě), můžeme tuto interakci organismu nazvat *vnější* interakcí, případně *vývin v prostředí*. V umělých modelech biologického vývinu je prostředí obvykle interpretováno jako dodatečná (externí) informace vedle genetické informace obsažené v chromozomu evolučního algoritmu. Zatímco genetické informace představují primární předpis průběhu vývinu, prostředí je chápáno jako externí řízení, které může být uplatněno jak na úrovni vývinu samotného, tak na úrovni struktury vyvíjejícího se objektu. Podobně jako v případě vnitřní interakce, reprezentace prostředí a způsob interakce s tímto prostředím, je aplikačně specifická záležitost.

V posledních několika letech byl však princip prostředí také využit ve zcela odlišném smyslu než výše uvedené externí řízení vývinu a to k ovlivnění samotné funkce elektronických obvodů. Vznikl tak nový obor nazvaný *polymorfní elektronika* [24]. V principu jde o zavedení vlivu určité veličiny do obvodu, která, v závislosti na její hodnotě, je schopna měnit funkci obvodu, avšak bez změny topologie (struktury) obvodu. Takovou veličinou může být například teplota, napájecí napětí, externí napětí atd. Hovoříme o tzv. polymorfních (multifunkčních) obvodech. V oblasti číslicových obvodů bylo vyvinuto několik tzv. *polymorfních hradel*, která se stala základními stavebními bloky polymorfních číslicových obvodů. Polymorfní hradlo reprezentuje multifunkční komponentu, jejíž funkce závisí na hodnotě externí veličiny interpretované jako prostředí. Například hradlo AND/OR pracuje při teplotě 27°C jako AND a při 125°C jako OR, funkce hradla AND/OR/XOR závisí na hodnotách externího napětí 3,3/0,0/1,5V [24]. Jelikož v současné době zatím neexistuje ucelená metodologie

návrhu polymorfních obvodů, jsou zde využívány zejména evoluční techniky.

7.3 Příklady aplikací systémů s interakcemi

Implementace interakcí na různých úrovních do umělých modelů inspirovaných principy biologického vývinu je jedním z cílů současného výzkumu v oboru biologií inspirovaných systémů. V následujících odstavcích jsou uvedeny příklady aplikací vývojových modelů z oblasti evolučního návrhu číslicových systémů využívajících různé druhy interakcí ke studiu vybraných vlastností, případně k dosažení požadovaného chování systému.

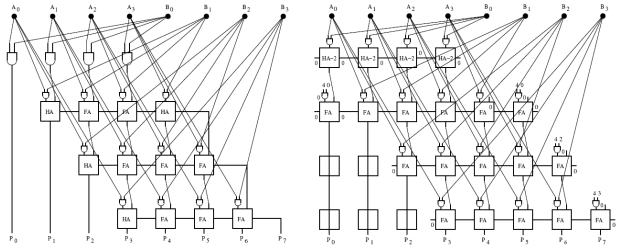
Gordon a Bentley [7] vytvořili model inspirovaný zejména procesem odlišování buněk (známého z biologie) pro **evoluční návrh číslicových obvodů s využitím programovatelného hradlového pole**. Jejich systém sestává z celulární struktury ve tvaru mříže, která je tvořena programovatelnými bloky hradlového pole (Configurable Logic Block – CLB). Každý CLB reprezentuje „buňku“, která se skládá z funkčního generátoru (programovaného evolučním algoritmem a vývojovým modelem), generátoru proteinů, detektoru proteinů, vstupů a výstupů. Proteiny jsou modelovány množinou jednoduchých symbolů. Detekce proteinů probíhá pouze ve smyslu přítomen/nepřítomen. Evoluční algoritmus je využit k nalezení množiny pravidel, které sestávají z podmínkové části a příkazové části. Podmínková část pravidla specifikuje proteiny, které musí být v dané buňce přítomny, případně ty, které nesmí být přítomny, pro vykonání příkazové části pravidla. Příkazová část buďto generuje určitý protein nebo modifikuje strukturu vyvíjeného obvodu realizovaného funkčními generátory uvnitř CLB. Pravidla vývojového systému jsou vykonávána podle sestavy proteinů detekovaných danou buňkou v jejím okolí, tj. na generátorech proteinů sousedních buněk. Každá buňka ovlivňuje chování sousedních buněk prostřednictvím proteinů, dochází tedy k mezibuněčné interakci. Jelikož tato interakce probíhá mezi komponentami uvnitř systému, jedná se o *vnitřní* (*interní*) interakci. Experimenty byly prováděny na návrhu dvoubitových sčítaček s využitím výše zmíněného modelu [7].

Podobný model, určený pro **návrh sekvenčních obvodů s využitím FPGA**, uvedl Tufte v [26]. Tento systém je založen na principu vývinu neuniformního 2D celulárního automatu, jehož buňky, které jsou mapovány přímo na strukturu FPGA. Každá buňka sestává z jendotky realizující vývin, Sblocku [8] implementujícího funkční generátor, jehož konfigurace udává *typ* buňky, a paměťový prvek, a chromozomu obsahujícího pravidla pro konstrukci (vývin) obvodu. Pravidla vývinu jsou navrhována evolučním algoritmem. Každé pravidlo sestává z podmínkové a příkazové části. Podmínková část specifikuje typy buněk v sousedství dané buňky požadované k provedení příkazové části. Příkazová část může buďto změnit typ dané buňky nebo vykonat tzv. růstový příkaz, který umožňuje rozšiřování struktury obvodu. Interakce zde probíhá při rozhodování podmínkových částí pravidel na základě typů sousedních buněk, jedná se tedy o *vnitřní* interakci. Cílem experimentu byl vývin sekvenčních obvodů realizujících inkrementální čítač buněk s výstupem v logické 1 během série hodinových cyklů obvodu.

M. Bidlo představil systém pro **vývin generických kombinačních násobiček,** založený na modelu symbolických instrukcích [27]. Cílem je evolučním algoritmem nalézt množinu nezávislých *programů*, které sdílí množinu interních proměnných vývojového systému a parametr specifikující bitovou šířku vyvíjené násobičky. Instrukční soubor obsahuje operace přiřazení proměnné, inkrementace a dekrementace proměnné, jednoduchý cyklus a instrukci pro generování stavebních bloků obvodu. Návrh je inspirován strukturou konvenčních kombinačních násobiček, jejichž některé části se liší od struktury zbylé části obvodu a tvoří tak nepravidelné struktury obvodu (první úroveň AND hradel a druhá úroveň sčítaček s AND hradly, viz Obr. 8a). Tyto odlišné části a zbývající dvě úrovně obvodu je možné vytvořit třemi různými algoritmy parametrizovanými bitovou šířkou operandu (4 bity). Z tohoto důvodu byl do systému zaveden *vliv prostředí* reprezentujícího externí informaci pro řízení vývinu. Hodnoty na jednotlivých pozicích prostředí určují, který program z hledané množiny programů se má provést pro vytvoření určité části násobičky. Obvodu na Obr. 8A odpovídá posloupnost aplikace těchto algoritmů specifikovaná prostředím (0, 1, 2, 2), tj. V prvním kroku je aplikován program 0, dále program 1 a následně dvakrát program 2. Stejný tvar prostředí byl použit i pro evoluční návrh, jehož výsledek ukazuje Obr. 8b. Podoba prostředí je určena návrhářem na počátku procesu vývinu. Je zřejmé, že prostředí není interní částí vývojového modelu – interní části tvoří programy tvořené instrukcemi, proměnné a stavební bloky obvodu. Jedná se tedy o *vnější* (*externí*) interakci vývojového modelu s prostředím. Optimalizované výsledky tohoto systému odpovídají struktuře konvenční násobičky (Obr. 8a), lze tedy hovořit o automatickém znovuobjevení tohoto principu. Poznamenejme, že metoda prezentovaná v [27] představuje první případ evolučního návrhu generických násobiček s využitím modelu vývinu.

Využití prostředí k řízení funkce obvodu (návrh polymorfních obvodů) představuje v současné době populární a perspektivní oblast výzkumu. Původní myšlenka **návrhu polymorfních obvodů na úrovni tranzistorů pomocí evolučních technik** byla představena v [25]. Sekanina publikoval metodu

evolučního návrhu polymorfních obvodů na úrovni hradel [28]. K reprezentaci obvodu a jeho evoluci byla využita metoda kartézského genetického programování [29]. Polymorfní hradla zde byla simulována softwarově, přepínání funkce obvodu bylo realizováno na základě hodnot pomocné proměnné (0/1). Poznamenejme však, že tyto aplikace nevyužívaly modelů biologického vývinu. Kandidátní řešení zakódovaná v chromozomech evolučního algoritmu byla mapována přímo na cílové obvody. Metoda **návrhu generických polymorfních obvodů** s využitím modelu vývinu založeného na symbolických instrukcích byla představena v [30]. Model vývinu je ve své podstatě totožný s modelem uvedeným v [9], stavební bloky jsou však tvořeny dvojicemi vybraných polymorfních hradel (AND/OR, OR/AND, ID/NOT, NOT/ID) v kombinaci s hradly konvenčními, které sdílejí vždy dva vstupy (např. dvojice hradel AND, OR se vstupními signály a, b dává výstupy uvedených logických funkcí nad těmito vstupy, tj. $a \cdot b$, $a + b$). Pomocí této metody byly navrženy programy pro konstrukci libovolně velkých polymorfních řadicích sítí, které řadí vzestupně/sestupně v závislosti na hodnotě proměnné prostředí (0/1) a paritních obvodů, jejichž výstupem je sudá/lichá parita v závislosti na prostředí. Experimenty v [9] však ukázaly na extrémní obtížnost návrhu polymorfních obvodů s využitím umělého vývinu, což je pravděpodobně způsobeno topologickým omezením obvodu z důvodu požadovaného multifunkčního chování.



(a) (b)

Obr. 8. Kombinační násobička 4×4 bity: (a) konvenční řešení, (b) struktura vyvinutá programy nalezenými evolucí, která je po optimalizaci komponent s nulovými vstupy totožná s konvenční násobičkou

Je zřejmé, že v případě samotných polymorfních obvodů nelze hovořit o externí interakci v pravém slova smyslu v souvislosti s jejich vývinem. Jedná se spíše o biologií inspirovaný přístup s cílem *adaptace* elektronických obvodů na různé fyzikální podmínky, což je však důležitá vlastnost z pohledu budoucího výzkumu elektroniky, v souvislosti s nanotechnologiemi apod.

8 Diskuze

Z obecného pohledu byly ve stručnosti zmíněny zejména systémy evolučního návrhu obvodů na různých úrovních abstrakce (úroveň tranzistorů [16, 25], základních logických hradel klasických a polymorfních [28, 31], vyšších funkčních bloků – komparátory, dvojice hradel, sčítačky apod. [9, 27, 30] a dále bloků navrhnutých za účelem realizace vývinu na rekonfigurovatelné platformě [7, 26, 33]). S využitím této klasifikace můžeme na základě popisů použitých modelů a dosažených výsledků provést následující zhodnocení.

V případě návrhu na úrovni tranzistorů nebyl využit žádný model uplatňující interakce v průběhu vývinu, případně bylo použito přímé mapování chromozomů na cílový obvod. Je zřejmé, že návrh obvodů na takto nízké úrovni abstrakce bude vyžadovat komplexní vývojový algoritmus pro získání funkčních řešení složitých instancí obvodů, který je pomocí evolučních technik extrémně náročné nalézt s využitím současných technologií z důvodu enormní velikosti vyhledávacího prostoru. Lze však očekávat, že by byla evoluce schopna objevit vysoce inovativní řešení, jelikož nízká úroveň abstrakce představuje méně informací vložených do systému a tudíž menší omezení evoluce v prozkoumávání vyhledávacího prostoru. Příkladem mohou být polymorfní hradla [25], pro která dosud neexistuje ucelená konvenční metodologie návrhu.

Experimenty provedené při reprezentaci obvodů na úrovni hradel umožnily návrh složitějších řešení v porovnání s nižší úrovní abstrakce jak v případě použití přímého mapování chromozomů na obvody [28], tak při aplikaci modelu vývinu [31]. Škálovatelnost obou přístupů je však velmi omezená, zvláště u obvodů s komplikovanou logickou strukturou – např. násobičky.

Zvýšení úrovně abstrakce spolu s aplikací modelu vývinu založeného na symbolických instrukcích umožnilo konstrukci generických (libovolně velkých) obvodů různých tříd. V případě návrhu řadicích sítí s využitím dvojic hradel AND, OR (tj. komparátorů) jako stavebních bloků byl objeven inovativní algoritmus konstrukce řadicích sítí o sudém počtu vstupů v porovnání s konvenční metodou stejného typu (např. s algoritmem přímého vkládání [34]). Poprvé byl představen experiment úspěšného evolučního návrhu generických násobiček s využitím modelu vývinu uplatňujícího interakci s prostředím k externímu řízení procesu vývinu. [27]. V tomto případě však zatím nebyla pozorována žádná inovace oproti konvenčním násobičkám, což může být zapříčiněno vyšší úrovní abstrakce využívající kombinace základních AND hradel a sčítaček.

Využití programovatelných hradlových polí pro vývin obvodových struktur zřejmě představuje vhodnou platformu pro studium mezibuněčných interakcí [7, 26]. Jelikož obvykle bývá pouze jedna struktura buňky vytvořená návrhářem s ohledem na zvolenou

rekonfigurovatelnou platformu společná pro všechny stavební bloky, obsahující nejčastěji funkční generátor a paměťový prvek, představují techniky mezibuněčné interakce vhodný způsob řízení vývinu obvodu přímo v rámci rekonfigurovatelné architektury. To však může být omezující pro návrh některých tříd reálných obvodů z důvodu omezení daných vlastnostmi a strukturou hradlového pole a možnosti vývinu obvodů fyzicky přímo technickými prostředky (tedy v hardwaru). Výsledky experimentů tohoto typu tak v mnoha případech poskytují pouze velmi jednoduché obvody specifikované funkce (např. dvoubitové sčítačky [7]) nebo rozsáhlejší obvodové struktury, jejichž funkcionalita je omezena na řešení jednoduchých umělých problémů vhodných k demonstraci funkčnosti daného vývojového modelu (např. inkrementální čítač počtu buněk s výstupy v logické 1 v průběhu funkce obvodu [26]).

Jak je patrné z výsledků uvedených experimentů, umělé modely vývinu poskytují schopnosti návrhu reálných obvodů. V některých případech se podařilo objevit algoritmy a obvodové struktury, které byly ještě donedávna neznámé (viz např. [9]). Techniky inspirované biologickými principy implementované v umělých modelech poskytují užitečné mechanismy pro vyšetřování různých vlastností a chování těchto systémů. Zdá se však, že z praktického hlediska není striktní přejímání principů biologie nejvhodnější pro řešení reálných (v našem případě obvykle technických) problémů. Volba vhodných technik s ohledem na reprezentaci cílových řešení je tak pravděpodobně silně aplikačně specifickou záležitostí. Otázkou však zůstává zejména to, jak navrhnout optimální model pro řešení konkrétního problému a zda jsou současné technologie schopny poskytnout vhodnou platformu pro efektivní realizaci takového systému.

9 Závěr

V příspěvku byly shrnuty principy základních modelů inspirovaných biologickým vývinem, které se v současné době nejčastěji používají v oblasti evolučního návrhu. Byly představeny tři přístupy k realizaci modelů vývinu založené na (1) celulárních automatech, (2) Lindenmayerových systémech a (3) algoritmech symbolických instrukcí. Ačkoliv celulární automaty a L-systémy byly původně vytvořeny za účelem modelování biologických procesů, postupem času našly uplatnění v řadě jiných oborů včetně počítačového inženýrství. Zaměření příspěvku a výběr zmíněných vývojových technik a aplikací byl zvolen s ohledem na autorovu současnou oblast výzkumu, kterou je evoluční návrh výpočetních systémů využívající development. Jak je patrné z podkapitol popisujících aplikace, všechny zmíněné modely prokázaly schopnost uplatnění v evolučním návrhu číslicových obvodů, případně v experimentech vyšetřujících určité vlastnosti cílové platformy.

Ačkoliv výsledky dosažené těmito modely nejsou v mnoha případech zdaleka optimální, představují významný přínos této problematice z důvodu možnosti nalezení nových algoritmů, jejichž vlastnosti a výsledky mohou být inspirací pro další zdokonalování vývojových systémů. Tyto skutečnosti indikují potřebu dalšího výzkumu v oblasti biologií inspirovaných modelů s využitím dosavadních výsledků a poznatků, který bude v našem případě zaměřen zejména na vývoj generických obvodových struktur a studium jejich vlastností s ohledem na spolehlivost, bezpečnost, adaptabilitu a další aktuální aspekty současné doby.

Poděkování

Výzkum byl proveden za podpory Ministerstva školství, mládeže a tělovýchovy České republiky v rámci projektu č. 0021630528, Fondu rozvoje vysokých škol MŠMT v rámci projektu č. 2472/2007/G1 a Grantové agentury České republiky v rámci projektu č. 102/05/H050.

Literatura

[1] P. J. Bentley (ed.): *Evolutionary Design by Computers,* Morgan Kaufmann Publisher, San Francisco, 1999

[2] J. F. Miller, D. Job: *Principles in the evolutionary design of digital circuits – part I*, Genetic Programming and Evolvable Machines, 1(1), pp 8–35, 2000

[3] X. Yao: *Evolving artificial neural networks*, Proceedings of the IEEE, 87(9), pp 1423–1447, 1999

[4] J. R. Koza et al: *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann Publisher, San Francisco, 1999

[5] M. Murakawa et al: *Evolvable Hardware at Function Level*, In: Proc. of the Parallel Problem Solving from Nature IV, LNCS vol. 1141, pp 62–71, Springer–Verlag Berlin Heidelberg New York, 1996

[6] J. Toressen: *A scalable approach to evolvable hardware*, Genetic Programming and Evolvable Machines, 3(3), pp 259–282, 2002

[7] T. G. W. Gordon, P. J. Bentley: *Towards development in evolvable hardware*, In: Proc. of the 2002 NASA/DoD Conference on Evolvable Hardware, pp 241–250, IEEE Computer Society Press, 2002

[8] P. Haddow, G. Tufte: *Bridging the genotype–phenotype mapping for digital FPGAs*, In: Proc. Of the 3rd NASA/DoDWorkshop on Evolvable

Hardware, pp 109–115, IEEE Computer Society Press, 2001

[9] L. Sekanina, M. Bidlo: *Evolutionary Design of Arbitrarily Large Sorting Networks Using Development*, Genetic Programming and Evolvable Machines 6(3), pp 319–347, 2005

[10] J. von Neumann: *The Theory of Self-Reproducing Automata* (edited by A. W. Burks), University of Illinois Press, 1966

[11] V. Drábek: *Vlastnosti a použití binárních celulárních automatů* [Habilitační práce], UIVT FEI VUT v Brně, 1997

[12] S. Wolfram: A New Kind of Science, Wolfram Media, 2002

[13] M. Sipper: *Evolution of Parallel Cellular Machines*, Lecture Notes in Computer Science vol. 1194, Springer, Berlin Heidelberg New York,1997

[14] M. Gardner (ed.): MATHEMATICAL GAMES: *The fantastic combinations of John Conway's new solitaire game "life"*, Scientific American, č. 223, říjen 1970, s. 120-123 (http://ddi.cs.uni-potsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm, březen 2007)

[15] J. R. Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992

[16] J. R. Koza, F. H. Bennett III., D. Andre, M. A. Keane: *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann pub., 1999

[17] J. H. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975

[18] I. Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme und Prinzipien der biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973

[19] H.-P. Schwefel: *Numerical optimazaion of computer models* , Wiley, Chichester, 1981

[20] L. J. Fogel, A. J. Owens, M. J. Walsh: *Artificial Intelligence through Simulated Evolution*, John Wiley, 1966

[21] S. Kumar, P. J. Bentley (eds.): *On Growth, Form and Computers*, Elsevier Academic Press, 2003

[22] A. Lindenmayer: *Mathematical models for cellular interaction in development*, *parts I and II*, Journal of Theoretical Biology 18, s. 280–315, 1968

[23] A. Ortega, A. A. Dalhoum, M. Alfonseca: Grammatical evolution to design fractal curves with a given dimension, IBM Journal of Research and Development 47(4), s. 483–493, 2003

[24] A. Stoica, R. S. Zebulum, D. Keymeulen: *Polymorphic Electronics*, Proc. of the 4th International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes In Computer Science vol. 2210, s. 291–302

[25] A. Stoica, R. S. Zebulum, D. Keymeulen, J. Lohn: *On Polymorphic Circuits and Their Design Using Evolutionary Algorithms*, In Proc. of IASTED International Conference on Applied Informatics (AI2002), Innsbruck, Austria 2002

[26] G. Tufte: *Cellular Development: A Search for Functionality*, In: Proc. of the 2006 Congress on Evolutionary Computation, s. 2669–2676, IEEE CIS, 2006

[27] M. Bidlo: *Evolutionary Development of Generic Multipliers: Initial Results*, In: Proc. of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems, 2007 (přijato k publikaci)

[28] L. Sekanina: *Evolutionary Design of Gate-Level Polymorphic Digital Circuits*, Lecture Notes in Computer Science, roč. 2005, č. 3449, Springer, Berlin, 2005

[29] J. F. Miller, P. Thomson: *Cartesian Genetic Programming*, In: Genetic Programming, Proceedings of EuroGP'2000, s. 121–132, Springer, Berlin, 2000

[30] M. Bidlo, L. Sekanina: *Providing Information from the Environment for Growing Electronic Circuits Through Polymorphic Gates*, In: Proc. of Genetic and Evolutionary Computation Conference - Workshops 2005, s. 242–248, ACM, New York, 2005

[31] M. Bidlo: *Generování grafů celulárními automaty* [*Technická zpráva*], Ústav počítačových systémů, FIT VUT v Brně, 2005

[32] M. Bidlo, R. Bidlo, L. Sekanina: *Designing a Novel General Sorting Network Constructor Using Artificial Evolution*, In: TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY, roč. 15, s. 85–90, Enformatika, Barcelona, 2006

[33] P. C. Haddow, G. Tufte, P. Van Remortel: *Shrinking the genotype: L-systems for EHW?*, Proc. of the 4th International Conference on Evolvable Systems: From Biology to Hardware, Lenture Notes in Computer Science vol. 2210, s. 128–139, Springer, 2001

[34] D. E. Knuth: *Art of Computer Programming, Volume 3: Sorting and Searching*, Addison-Wesley, 1998

[35] G. Rozenberg, A. Salomaa (eds.): *Handbook of Formal Languages, Vol. 3: Beyond Words*, Springer-Verlag, Berlin Heidelberg, 1997