Soft-Hardware

Programy inspirované biologií – hardware s vlastnostmi software

LUKÁŠ SEKANINA
VLADIMÍR DRÁBEK

V informatice a výpočetní technice je dnes velmi populární *soft computing.*[1] Jde o soubor výpočetních metodologií, které se zrodily z klasické umělé inteligence. Od svého mateřského oboru se liší hlavně ve využívání biologické inteligence. Zatímco klasická inteligence, zabývající se programy schopnými soutěžit s člověkem, využívá spíš elementy, které v biologické inteligenci nejsou klíčové (heuristiku, reprezentaci znalostí či symbolické zpracování), soft computing čerpá z modelování přírodních procesů. Inspiruje se jimi na úrovni fylogeneze (evoluční algoritmy), ontogeneze (celulární automaty) i epigeneze (umělé neuronové sítě).

Soft computing zahrnuje řadu přístupů,[2] které lze všelijak kombinovat. Například evoluční algoritmy ve spojení s automaty inspirovanými buňkou vedly ke vzniku nového výpočetního paradigmatu – celulárního programování. Přístupy soft computing používáme především proto, že v určité kategorii úloh jimi dokážeme za stejný čas spočítat více než konvenčními technikami. Jestliže je simulace počítačového modelu na konvenčním procesoru pomalá, můžeme navrhnout *specializovaný hardware* (např. nový procesor). Je to podobné jako třeba grafický procesor pro urychlení operací na grafické kartě v osobním počítači.

Návrh hardware je (skoro) stejný jako návrh software

Návrh specializovaného procesoru se příliš neliší od návrhu běžného programu. V programovacím jazyce pro popis obvodů se navrhne a simuluje zapojení číslicového obvodu (tj. našeho procesoru), a z tohoto popisu se v jiném programu vytvoří *konfigurační informace*, která se nahraje do univerzálního *rekonfigurovatelného* (programovatelného) *obvodu*. Pokud potřebujeme jiný specializovaný obvod, změníme pouze konfigurační informaci programovatelného obvodu (samozřejmě ji musíme mít připravenou předem). Konfigurační informace určuje, jak se elementy obvodu propojí a naprogramují jednotlivé elementy (zda bude element pracovat jako logický součin nebo logický součet a kam se mají připojit jeho vstupy i výstupy) a také jak se to celé připojí k vstupům a výstupům obvodu. K tomu účelu se používají *programovatelná hradlová pole*, která mají dostatečnou kapacitu elementů (řádově miliony) na to, aby takový procesor mohl vzniknout. Přeprogramování hradlových polí trvá několik milisekund, jednotlivé skupiny elementů lze přeprogramovat za pár mikrosekund, někdy i rychleji. Konfigurační informace je uložena v paměti typu RAM a je zajímavé, že některé obvody umožní nahrát i náhodnou konfiguraci, aniž se zničí čip. Tímto způsobem lze některé výpočty urychlit (například výpočet umělé neuronové sítě se už nesimuluje programově, ale řeší se obvodově).

Hardware čerpající z biologie

Od počátku devadesátých let (a hlavně kolem r. 1996) vedly rekonfigurovatelné obvody ve spojení se soft computing k formování *hardwaru inspirovaného biologií*. Zasloužili se o to zejména T. Higuchi, A. Thompson, H. de Garis a D. Mange. Rekonfigurovatelný obvod přestává být pouhým prostředkem k urychlení výpočtu a stává se podstatou věci. Nejvýrazněji je to patrné na *vyvíjejících se obvodech*, kde je *fyzické* zapojení obvodu subjektem evoluce, je inspirováno fylogenezí. C*elulární systémy* jsou zase inspirovány ontogenezí. Celý obvod je chápán jako vícebuněčný organizmus. Počáteční obvod se vytváří z *mateřské buňky* a od počátku nese ve své struktuře celý genetický program *buněčného dělení* a *buněčné specializace* i s užitečnými vrozenými rysy, jako je *sebeoprava* či *sebereplikace.*

Celulární automat, jehož přechodové funkce jsou evoluční subjekty, lze ještě kombinovat s umělou neuronovou sítí. Klasifikaci hardware inspirovaného biologií zavádí *POE hardware* (POE – zkratka pro filogenezi, ontogenezi a epigenezi). Citujme autory modelu: *Jestliže se díváme do budoucnosti a trochu při tom sníme, můžeme si představit skutečné nanosystémy (bioware), které budou vybaveny evolučními, sebereprodukčními, sebeopravujícími a nervovými schopnostmi. Takové systémy potom povedou ke vzniku nových druhů, které budou existovat spolu s živými bytostmi založenými na uhlíku. Tím je nejspíš vymezen náš cíl.*

Vyvíjející se obvody

Lze je charakterizovat jako *technologii umožňující vytvořit vyvíjející se systém, který je schopen autonomně a dynamicky měnit své fyzické zapojení v závislosti na změně prostředí, v němž je použit.* Evoluční algoritmus (např. genetický) pracuje s populací bitových řetězců, které představují různá zapojení obvodu (např. jako konfigurační informace programovatelného obvodu) a je použit pro konstrukci zapojení. Nová populace je vygenerována s využitím operátorů inspirovaných biologií (mutacemi, křížením a selekcí), které jsou definovány nad těmito zapojeními. Kvalitu řešení určuje *fitness funkce* (říká

Ing. Lukáš Sekanina (*1976) vystudoval Fakultu elektrotechniky a informatiky VUT v Brně. V Ústavu informatiky a výpočetní techniky téže fakulty se jako postgraduální student zabývá využitím biologických principů pro návrh hardware.

Doc. Ing. Vladimír Drábek (*1946) vystudoval Fakultu elektrotechnickou VUT v Brně. Na této fakultě působí v Ústavu informatiky a výpočetní techniky. Přednáší disciplíny: výstavba počítačů, systémy odolné proti poruchám, diagnostika, grafické a multimediální procesory. Jeho diplomanti a doktorandi se zabývají návrhem speciálních operačních obvodů vjazyce VHDL.

1) O *soft computing* informoval již P. Hájek (viz Vesmír 79, 683, 2000/12). Ani my v tomto článku nechceme tento termín překládat.

2) Mám na mysli např. *fuzzy počítání* (o něm viz již citovaný článek P. Hájka), *umělý* (resp. uměle vzniklý) *život* (artificial life), *rojové výpočty* (swarm intelligence), *těžení z dat* či – podle T. Hrušky – *datokopectví* (data mining).

POČÍTAČOVÁ SIMULACE A VÝPOČET OBVODU NENÍ TOTÉŽ

Genetické operace a funkce vhodnosti se nejdříve uskutečňovaly programově, dnes však existuje řada implementací (podprogramů) s genetickými operacemi v hardware, dokonce i s výpočtem funkce vhodnosti (fitness). Důsledná hardwarová implementace je typická pro vyvíjející se obvody pracující v proměnném prostředí. Pro evoluční návrh se často používají simulátory obvodů.

■ Omezená evoluce. **Prostor návrhu se dá zmenšit například tím, že zakážeme zpětné vazby v obvodech na bitové úrovni reprezentace zapojení. Pro zapojení navržená evolucí potom nevzniká rozpor mezi funkcí obvodu, která se ověří programovou simulací, a funkcí obvodu, která se ověří fyzickou implementací.**

■ Neomezená evoluce**. Vede často k *exotickým zapojení*, která jsou z pozice tradičního návrhu nepochopitelná. Předpokládejme, že jsme prostřednictvím evoluce a s využitím simulátoru obvodů nalezli obvod reprezentovaný konfigurací *C* a hodnotou fitness *f0*. Jestliže konfiguraci *C* nahrajeme do rekonfigurovatelného obvodu *RC1*, získáme po ohodnocení fitness *f1* , a když ji nahrajeme do rekonfigurovatelného obvodu *RC2*, získáme fitness *f2*. Obecně hodnoty fitness *f0*, *f1* a *f2* mohou být různé (ačkoliv jde o stejnou konfiguraci). Evoluce totiž při návrhu může využít i fyzikální vlastnosti čipu (např. křemíku), které nejsou totožné pro všechny čipy programovatelných hradlových polí stejného typu (výrobce garantuje interval těchto hodnot) a také podle okolních podmínek (např. teploty v místnosti).**

Ve spojení s evolučními algoritmy tedy rekonfigurovatelné obvody přinášejí kromě urychlení výpočtu i něco mnohem význačnějšího: překvapující řešení, která nelze získat žádným jiným způsobem.

se jí též *funkce vhodnosti*). Programovatelný obvod se nakonfiguruje pomocí navrženého řešení a vzniklé zapojení se ohodnotí. Jestliže má evoluce nalézt například zapojení sčítačky (sčítací jednotky), musíme tuto funkci zkonstruovat tak, aby jejím výsledkem byla větší hodnota pro *lépe* sčítající sčítačku a menší hodnota pro *hůře* sčítající sčítačku. (*Hůře* znamená, že taková sčítačka pracuje správně jen pro některé vstupy.)

Jediná funkce vhodnosti je málo

Kdybychom definovali jen *jedinou* funkci vhodnosti, neměl by se náš přístup (dle některých autorů) nazývat *vyvíjející se obvody*, ale *evoluční návrh obvodů*. Ten nám bude stačit, jestliže budeme hledat pouze jedno zapojení – například zmíněnou sčítačku. Výsledek evoluce – tedy zapojení sčítačky, které by mělo mít lepší parametry než doposud nejlepší známá sčítačka navržená tradičními technikami – se potom může začít používat ve všech aplikacích. Termín *vyvíjející se obvody* by měl být používán jen v případě, že použijeme *vícero* funkcí vhodnosti, které budou představovat dynamicky se měnící prostředí (evoluce nemá předem daný cíl) a že celý systém bude zahrnut v hardware. V takovém případě jde o *současnou* evoluci zapojení obvodu a prostředí. Evoluci nelze nikdy trvale ukončit, protože dosavadní kvalitní řešení nemusí být dost kvalitní v novém prostředí.

Jako příklad uveďme *kompresi obrazu* (balení). Vyvíjející se obvod pracuje jako prediktor obrazových elementů (pixelů) v bloku obrazu předem neznámé velikosti. Pokud je predikce úspěšná, může se blok postupně zvětšovat, evoluce tento prediktor neustále vylepšuje. V jistém okamžiku ale klesne kvalita komprese bloku pod povolenou úroveň a evoluce se musí zastavit. Konfigurace nalezeného prediktoru pak (společně s dalšími údaji) představuje blok v zabaleném souboru. Stejně se pokračuje se zbytkem obrazu. Jednotlivé bloky, respektive jejich konkrétní datové obsahy, pak lze chápat jako různá prostředí pro vyvíjející se prediktor.

Výhody vyvíjejících se obvodů

■ Evoluce umožňuje navrhnout obvody přesahující možnosti konvenčního návrhu. Odstraňuje omezení, která jsou součástí tradičního návrhu (nepoužívá klasické návrhové techniky založené na dekompozici a minimalizaci), odstraňuje i předsudky, které vnášel sám návrhář.

■ Umělá evoluce najde (díky uzpůsobení hardware) dostatečné řešení rychleji než konvenční metoda, anebo poskytne alespoň nějaké řešení tam, kde konvenční metoda z časových důvodů neposkytne žádné.

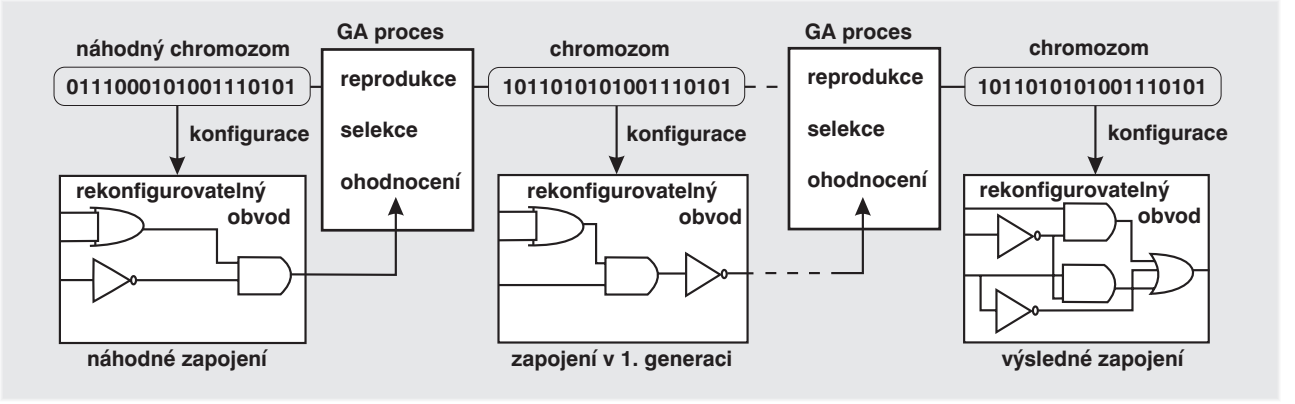
■ Tato technika má praktické výsledky, protože existuje řada aplikací, a dokonce i několik komerčních čipů založených na vyvíjejících se obvodech (např. pro kompresi obrazu či pro řízení protézy horní končetiny).

■ Umělá evoluce vnáší do systému „vrozenou" odolnost proti poruchám. Při poruše některého prvku je velmi pravděpodobné, že se (díky redundanci programovatelných elementů obvodu) podaří najít zapojení ze zbylých prvků.

■ Návrhy využívající tuto techniku jsou uživatelsky příjemné – uživatel definuje, *co* se má udělat, a ne *jak* se to má udělat.

V čem je problém?

■ **Ve škálovatelnosti**. Pro velký počet elementů v programovatelném obvodu je výpočet zatím časově nezvládnutelný. Evoluce zaznamenala úspěch jen pro poměrně jednoduchá zapojení. *Hlavní překážkou, kterou je nutno překonat při škálování evolučního*



návrhu číslicových obvodů je (dle V. Vassileva a J. F. Millera) *jsou obtíže s definováním stavebních bloků tak, aby byly vhodné pro evoluci. Zjistilo se, že velké sčítačky mohou být evolucí navrhovány efektivně jen tehdy, jestliže se sčítačky menší velikosti využijí jako stavební bloky. To ale neplatí o násobičkách. Možná že násobička je základní jednotka, kterou nelze rozložit na menší bloky tak, aby byl výsledek evoluce tím nejlepším možným návrhem. Je také pravděpodobné, že existují další aritmetické a logické funkce, které neumožňují, aby byl nejlepší možný návrh rozložen do subobvodů.*

■ **Ve schopnosti generalizovat**. Pro logické obvody se funkce vhodnosti obvykle počítá jako procento správně určených výstupů pro všechny vstupy. Pro malou šířku vstupu je doba, po kterou se zapojení hodnotí, přijatelná, ale přidáním jednoho vstupu se velikost tabulky zdvojnásobí (kombinační logické sítě). Evoluce obvykle nenajde správný kombinační obvod, pokud nemá k dispozici všechny možné vstupy.

■ **V objemnosti**. Pokud není obvod navržený evolucí triviální, můžeme ho chápat pouze jako „černou skříňku". Vykazuje sice požadované chování, ale o jeho dalších vlastnostech nevíme téměř nic. Zapojení nebylo připraveno tradiční inženýrskou metodou, ale propojováním, které se řídí zákonitostmi, jež neznáme . Správnou činnost zapojení můžeme ovlivnit například tím, že využijeme materiálové vlastnosti konkrétního čipu. Jestliže ale výsledný obvod není „proměřen", nelze garantovat jeho činnost v jiném prostředí než v tom, v němž evoluce probíhala.

Soft-Hardvare

Vyvíjející se obvody (jako disciplína) potvrdily oprávněnost své existence. Nejenže již existují komerčně dostupné čipy založené na této technologii, ale evoluce umožnila najít i zcela unikátní a elegantní zapojení elementárních obvodových prvků. Návrh aplikací vyvíjejících se obvodů je časově i intelektuálně náročný, vyžaduje zkušenosti s navrhováním programů, obvodů a evolučních algoritmů. Je založen na experimentování.

Důsledkem toho všeho je změkčení hardware. Zůstává tradiční rychlost a paralelní výpočet, ale obvody můžeme chápat jako programy, které se přizpůsobují požadavkům prostředí, v němž pracují. Můžeme je mazat, kopírovat, doučovat... Jsme teprve na začátku, tudíž nezbývá než souhlasit s P. Hájkem: *Dnes ještě patrně nelze odhadnout, jak to se soft computing* (a – dodejme – ani s hardware inspirovaným biologí) *dopadne.* Nicméně návrháři hardware odkryli v tomto nesmírně vzrušujícím odvětví další potenciál. Výsledky bádání nemusí být použity jen v inženýrské a návrhářské praxi, mohou také zpětně ovlivnit studium a vývoj zdroje své inspirace – biologie. □

K DALŠÍMU ČTENÍ

Sekanina L., Drábek V.: Evolvable hardware – evoluce na čipu, Elektrorevue – internetový časopis FEI VUT Brno 1999/5, http://www.elektrorevue.cz

Sekanina L.: Vyvíjející se komponenta. Zpráva o subjektu, cílech a stavu rozpracovanosti disertační práce k rigorozní zkoušce, VUT Brno, Ústav informatiky a výpočetní techniky, 2000 (internetová prezentace projektu je na URL http://www.fee.vutbr.cz/~sekanina/ehw/index.html)

Vassilev V., Miller J. F.: Scalability Problems of Digital Circuit Evolution. In: A. Lohn et al.: The Second NASA/Dod workshop on Evolvable Hardware, iEEE computer Society, Los Alamitos 2000