Co je člověk?

"Rád bych poukázal na to, že všechny pokusy
zodpovědět tuto otázku před rokem 1859 jsou bezcenné
a uděláme nejlépe, když je zcela pomineme."
zoolog G.G. Simpson

Genetický algoritmus

Teprve v druhé polovině devatenáctého století se dostalo světu myšlenky, která nabízí uspokojivou odpověď na otázku ohledně vývoje života. Charles Darwin v roce 1859 představil světu ucelenou a obhájitelnou teorii evoluce, která ukázala nový směr mnoha lidským oblastem bádání. Teorii, kde změna a vývoj stojí na chybách při replikaci a přírodním výběru, který zvýhodňuje vlastnosti jedinců, které jsou lépe přispůsobeni okolí a penalizuje ty, které jedince omezují. Odlišných vlastností vzniká díky chybám při replikaci spousta. Jen nepatrné množství z nichž je však výhodné. Nelze než strnout úžasem nad obrovským rozsahem pokusů a omylů, který příroda provádí už milióny let a v nichž platí jen jediné pravidlo, přežití zdatnějšího. Elegantní a krásná teorie, která evokuje odlišné nazírání problémů. Byl by hřích ji neuvyužít i ve výpočetní technice. Pro hlubší pochopení principů a dopadů Darwinovy teorie doporučuji knihu R. Dawkinse "Sobecký gen"[1] a pro získání přehledu o současném stavu pak knihu M. Rydleye "Genom"[2].

Technická reprezentace

Pro problémy řešené počítači se myšlenka evoluce transformovala do nástroje zvaného genetický algoritmus (dále GA), jehož cílem je počáteční populaci (20-100) různých nedostatečných řešení (jedinců) daného problému zlepšit. Každý jedinec je reprezentován svým chromozomem, který v sobě nese zakódované jedno z mnoha řešení problému. Jedince stávající populace ohodnotíme tzv. *fitness* funkcí, která je ukazatelem jeho kvality. Tuto funkci obecně nemusí být jednoduché nalézt a případně provést. Její předpis může být neměnný po celou dobu evoluce, nebo se může naopak měnit pro každou novou generaci. Přežití zdatnějšího se promítne tak, že jedinci s lepším ohodnocením budou rodiči více potomků. Kvůli hrozbě uváznutí v lokálním maximu však není žádoucí vybírat jen aktuálně nejlepší jedince a ty dál množit. Nové a kvalitní části řešení mohou vznikat postupně a nemusí být ze začátku výrazně přinosné. Teprve při vhodném spojení s jinými (při křížení) se projeví jejich hodnota. Užívá se několik metod pro určení množství potomků, které jedinec na základě svých aktuálních kvalit splodí, avšak všechny umožňují i množení méně úspěšných jedinců, biť v menším počtu. Princip rulety (*Roulete Wheel Selection*) přiřadí větší počet ruletových polí a tím i pravděpodobnost, že on bude vybrán pro rozmnožování, úspěšnějšímu jedinci. Každé točení vybere jednoho ze dvou (či více) rodičů. Pokud jsou rozdíly mezi kvalitou jedinců příliš výrazné, což by vedlo k příliš velkému zastoupení zrovna teď "skvělých" jedinců, lze sestavit jejich žebříček (*Rank Selection*) a lepšímu umístění dát sice opět více šancí na množení, ale již jen úměrně pořadí. Někdy může být škoda nechat zemřít všechny zástupce minulé generace a lze těm nejlepším umožnit přežití i v nové generaci v nezměněné podobě (*Elitism*). Vlastní vznik nového jedince probíhá ve dvou fázích. Nejdříve se vytvoří jeho chromozom a to rekombinací chromozomů rodičů. Křížení (*Crossover*) může být jednobodové, kdy všechny geny až do určené pozice budou pocházet od jednoho rodiče a zbytek od druhého, analogicky bude vypadat křížení vícebodové, v krajních případech se náhodně pro každý gen zvlášť určí jeho rodič nebo je potomek dokonalou kopií jednoho

rodiče stejně jako k tomu dochází při nepohlavním rozmnožování. Ve druhé fázi se zohlední občasné přehmaty přírody při replikaci, kde ale slovo přehmat rozhodně nelze vnímat pejorativně, neboť jen díky němu jsme svědky (a plody) vývoje. S malou pravděpodobností (0,5-1%) se provede drobná změna vybraných genů v chromozomu. Charakter změny závisý na reprezentaci řešeného problému v chromozómu. Může se jednat o inverzi pro binární řetězce, přičtení či odečtení drobného čísla pro reálné hodnoty, výměnu podstromů při stromové reprezentaci nebo prohození pozic genů při reprezentaci permutací. Nově vzniklý jedinec se zařadí do nové populace a celý cyklus se opakuje znovu až do nalezení uspokojivého řešení nebo do dosažení stanoveného počtu iterací.
Podrobnější popis včetně ukázek řešení některých problémů a rad pro úspěšnou implementaci lze nalézt například na [3]. Obecně lze říci, že s použitím GA může docházet k efektivnějšímu procházení stavového prostoru všech řešení, které má předpoklady k vyhnutí se uváznutí v lokálních maximech a postupně zlepšuje své řešení. V polynomiálním čase lze tedy získat řešení, které nemusí být nutně nejlepší možné, ale bude pravděpodobně "dostatečně" dobré. Přesně v souladu s přirodou lze obdržet i řešení, které jsou naprosto originální a které je člověk se svým druhem logiky schopen pouze dodatečně pochopit a užasnout, nikoli vytvořit. Myslím že geniální lidské mozky se vyznačují právě tímto, takže bych se s trochou nadsázky odvažoval nazvat GA geniálními.
Pro technickou realizaci je velmi výhodné, že GA je už svou podstatou paralelní a proto jej lze snadno provádět na více strojích a to buď prostým rozdělením jedné populace na části nebo samostatným vývojem několika oddělených populací, mezi kterými lze případně umožnit občasnou výměnu genetického materiálu.
Spektrum problému, na které lze GA aplikovat sahá přes hledání řešení NP problémů [3], návrhu strukturu a počáteční nastavení vah neurových sítí, hledáním výhodných kompresních funkcí digitálních obrazů, návrhu zapojení číslicových a analogových obvodů, kde obvzláště v případě asynchronních obvodů dosahují vynikajicích výsledků, lze s nimi určovat nejvhodnější tvar a parametry mechanických součástek, samostatným odvětvím je pak užití při genetickém programování [4], v umělé inteligenci a nezastaví se ani před skládáním hudby [5] či malováním obrazů [6].

Návrh zapojení číslicových a analogových obvodů, vyvíjející se hardware

Pomalu se začíná vedle klasické von Neumanovi koncepce počítače objevovat nová, do značné míry opačná koncepce. Proti proudu instrukcí, které pracují s daty se postupně, a poněkud v pozadí zájmu programátorů, začíná prosazovat vize počítače řízeného proudem dat "procházejícím" předpřipravenou sadou instrukcí (*soft CPU*), které ho modifikují. Oproti klasickému přístupu má obrovskou výhodu v podobě eliminace časových prodlev při práci s pamětí, které jsou velkým trápením generací programátorů. Nutným požadavkem je ale možnost měnit modifikující sadu instrukci za běhu, rychle a neztrácet přitom výhodu rychlosti hardwarově implementovaných instrukcí. S pomocí FCPGA (*Field Programmable Gate Array,* např. značka Xilinx) je to možné, neboť nám umožnují ve velmi krátké době (řádově milisekundy) změnit kompletně zapojení čipu a to do podoby nám vyhovujicího zapojení instrukcí, které jsou následně prováděny v plné rychlosti hardwarového zapojení. Na čipu jsou umístěny jednotlivé buňky, které mohou realizovat základní logické operace jako OR, XOR a AND. Dále je přítomná přepisovatelná paměť, která obsahuje popis zapojení, dle kterého se zapojení realizuje.
Scénář použití může vypadat zhruba následovně: pomocí GA nalezneme výhodné zapojení sad modifikujících instrukcích, které postupně uvedou data z výchozí podoby do požadované. Tyto sady si zapamatujeme a při zpracování dat budeme tyto sady postupně nahrávat do FCPGA a vykonávat. Výsledkem bude rychlé a flexibilní řešení, ve kterém je ponechán

prostor i paralelismu, neboť na jednom čipu FCPGA lze zároveň mít více sad instrukcí, tedy vlastně "víceprocesorový" systém.
Další výhodnou aplikací je oblast tzv. zapouzdřených (*embeded*) sytémů, poměrně levných a jednoúčelových zařízení. Jejich výhodnost spočívá v relativně nízké ceně, neboť nepotřebují všestranné výpočtové jednotky, ale stačí jim specializované obvody pro danou činnost. V případě, kdy se ale vnější podmínky mění a od zařízení je požadováno přizpůsobení se, jako například pro zařízení napojené na kamery na křižovatkách, jehož úkolem je identifikovat SPZ aut projíždějících na červenou, lze s pomocí FCPGA a GA vytvořit systém, který bude stále cenově výhodný. Problém přizpůsobení spočívá v různé kvalitě a nasvětlení obrazu z kamery, která závisí na denní době a na počasí. V zařízení probíhá trvalá evoluce, hodnotící funkce je dynamicky závislá na vnějších podmínkách a díky tomu lze stále udržovat populaci aktuálně přizpůsobenou vnějším podmínkám, která kóduje funkci filtru obrazu. Tento styl spolupráce hardwaru a GA se označuje termínem vyvíjející se obvody (*Evolvable Hardware, EHW*) a jedná se o slibně se rozšiřující oblast. Zřejmně se vývoj nebude ubírat k systémům plně tvořeným EHW, ale spíše se bude jednat o vybrané komponenty EHW začleněné do klasicky implementovaných systémů.

Při návrhu obvodů může docházet dokonce k situaci, kdy je na konkrétním čipu evolucí vyvinut dobře fungující obvod, který ale po přesném přepsání zapojení do jiného čipu stejného typu dává mírně horší (nebo lepší) výsledky. Je to způsobeno tím, že evoluce může využít fyzikální vlastnosti čipu (např. křemíku), které se mohou mírně lišit i pro čipy stejného typu, vliv mohou mít i okolní podmínky (teplota v místnosti).

Jak ukázal například A. Thomson, GA mohou dospět k velmi exotickým řešením, která jsou z klasického inženýrského pohledu naprosto nesrozumitelná. Thomson navrh s použitím evoluce obvod, který detekoval kmitočty 1kHz (log. 1 na výstupu) a 10kHz (log. 0 na výstupu). Zapojení se hledalo na ploše 10 x 10 prvků, evoluce však použila jen 32 prvků, výsledné zapojení bylo chaotické, pracovalo asynchronně a bylo více analogové než digitální. Evoluce probíhala na PC s jedinci o délce chromozomu 1800 bitů, poté byli jedinci přeneseni do FCPGA a otestování na kvalitu rozlišování vstupního signálu. Po 5000 generacích bylo vyvinuto zapojení, které fungovalo. Na evoluci nebyli kladeny žádné omezující podmínky. Při klasické implementaci by bylo potřeba nejméně 10x tolik logických členů, nebo alespoň přístup k synchronizačnímu signálu. Po přenesení do jiného čipu (stejného typu) vykazovalo řešení asi o 7% horší kvalitu, tedy i okolní podmínky měly vliv na výsledek řešení.

Další výraznou výhodou je odolnost takto vyvinutého zapojení je odolnost proti poruchám, protože je velká pravděpodobnost, že při poruše některého prvku se najde díky redundanci zapojení ze zbývajících prvků obvodu. Například pokročilá miniaturizace procesorů bude zanedlouho vyžadovat počítání s výskytem chyb na čipů na úrovni atomů, které budou natolik časté, že bude výhodnější nasadit prostředky, které umožní vytvořit požadované zapojení bez porušených prvků, než chybné čipy zahazovat.

Genetické programování [4]

Genetické programování (GP) je forma evolučních výpočtů, v nichž jedinci vyvíjející populace představují zápis počítačového programu. Typicky se používá stromová reprezentace odpovídající stromu, na který lze program rozložit. Oblíbeným implementačním jazykem je jazyk LISP a to právě díky nativní podpoře práce se stromy. Pro aplikaci GP je zapotřebí, aby uživatel definoval primitivní funkce (např. sin, cos ...) a terminály (např. a, 3 ...) Algoritmus pak evolučně prohledává prostor všech možných programů složených z těchto primitiv. Hodnotící funkcí je provedení vzniklého programu na množině trénovacích dat. Během křížení se přemisťuje náhodně vybraný podstrom jednoho rodiče na náhodně vybrané místo postromu druhého rodiče. Stejně jako u GA i zde záleží na vhodné reprezentaci problému. GP již bylo úspěšně použito k řešení velmi složitých úkolů, jako například návrh

obvodů elektronických filtrů, kdy byl vyvinut s pomocí GP program, který transformoval původní nepříliš dobrý, ale funkční obvod filtru na finální obvod. Zde byla použita populace o 640 000 jedincích (!) a finální obvod byl sestaven po 137 generacích.

Návrh struktury a rozložení vah neuronových sítí

Volba vhodné struktury neuronových sítí je obecně obtížný proces, neboť je třeba vhodně zvolit počet neuronů a jejich propojení které povede k uspokojivému řešení po přijatelnému počtu učících cyklů. Velký počet neuronů výrazně zvyšuje dobu potřebnou k natrénování sítě a příliš malý nemusí k řešení postačovat. Dalším problémem je vhodné počáteční nastavení vah synapsí, které se většinou volí jako náhodná malá čísla. Pokud jsou však již od počátku nastaveny na "rozumné" hodnoty, může proces učení výrazně rychleji konvergovat k požadovanému řešení. GA se proto pokusí nalézt výchozí hodnoty tak, aby výstupem sítě byly alespoň částečně rozumné hodnoty, což může být výpočetně méně náročné než proces učení sítě vedoucí k témuž. Lze předpokládat, že pokud nalezená struktura sítě alespoň částečně funguje, její učení povede k zlepšení výsledků.

Kompresní funkce digitálních obrazů a obrazových filtrů

Dobrá kompresní funkce se vyznačuje výhodným poměrem mezi výslednou kvalitou a velikostí komprimovaného obrazu. GA lze užít pro nalezení efektivních kompresních funkcí pro konkrétní části obrazu. Obraz se rozdělí do malých oblastí (např. 8x8 bodů) a pro tyto oblasti se evolučně hledá zápis funkce, která popisuje rozložení bodů a přitom její zápis je úspornější než původní formát obrazu. Důležitým parametrem je i doba potřebná k nalezení zmíněných funkcí, po urychlení lze užít implementaci s pomocí FCPGA, kde jsou jedinci reprezentující funkce hardwarově implementováni a jejich kvalitu lze rychle hodnotit. Lze též vyvinout obrazové filtry pro určitý typ šumu a následně je využít pro získávání informací ze zašuměných obrazů, jako například při rozeznávání dopravních značek snímaných kamerou umístěnou na voze za různých světelných a pozičních podmínek.

Skládání hudby a kreslení obrazů (Evolutionary Art) [5]

Příroda kolem nás vytváří nádherné scenérie a ač to nedělá záměrně, vede si v tom nadmíru dobře. Další oblastí užití GA, která je určena pro potěchu oka, je evolučně vyvíjená hudba a kreslené obrazy. Při skládání hudby se vlastně jedná o speciální aplikaci genetického programování, kdy terminály jsou jednotlivé noty, pomlky a funkcemi pak křížky, béčka či takt. Evolucí se vyvíjí program, který je interpretován jako notový zápis. Hodnotitelem je pak člověk, který určuje poslechem libost dané melodie a funguje tak jako *fitness* funkce pro danou populaci melodií.

Obdobnou roli plní člověk u evolučně vyvíjených obrazů, generování rostlin, kdy jsou v chromozomu jedince obsaženy parametry užitých fraktálů, velikosti , rychlosti a směru růstu listů, barvy a jiných vlastností výsledné rostliny. Při tvorbě animací se přidává gen, který kóduje čas zobrazení výsledného obrazu. Nádherné galerie s popisy metod křížení a mutace lze nalézt na stránkách jednoho z duchovních otců tohoto směru, Karla Simse [6].

Petr Švenda
e-mail: xsvenda@fi.muni.cz
http://www.fi.muni.cz/~xsvenda

Použitá literatura, odkazy:
[1] Dawkins, R. : Sobecký gen, Mladá Fronta, Praha 1998, ISBN 80-204-0730-8
[2] Rydley, M. : Genom, Portál, Praha 2001, ISBN 80-7178-507-5
[3] Obitko, M. : Genetic algorithm http://cs.felk.cvut.cz/~xobitko/ga/main.html
[4] Genetic Programing http://www.genetic-programming.com/johnkoza.html
[5] Evolutionary Computation http://www.red3d.com/cwr/evolve.html
[6] Sims, K. : Evolutionary Art http://www.genarts.com/karl/

Klíčová slova: Genetic Algorithm, Evolution, Genetic Programing, Evolvable Hardware, Evolvable Art, Field Programmable Gate Array