EVOLUČNÍ ADAPTACE PREDIKTORŮ V REÁLNÉM ČASE

Karel Slaný
Informační technologie, 2. ročník, prezenční studium
Vedoucí: Doc. Ing. Lukáš Sekanina, Ph.D.

Fakulta informačních technologií
Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno, Česká republika

slany@fit.vutbr.cz

Abstrakt Příspěvěk shrnuje teze disertační práce, která se zabývá konstrukcí adaptivního prediktoru časových řad na bázi genetického programování.

Klíčová slova evoluční adaptace, genetické programování, časové řady

1 Úvod

Evoluční algoritmy představují zajímavou alternativu ke klasickým návrhovým metodám. Dokáží generovat řešení problému, aniž by bylo nutné detailně znát jeho vnitřní strukturu. Tato řešení mnohdy svou kvalitou konkurují návrhům zkušeného člověka-návrháře. V některých případech je dokonce předčí [14].

Genetické programování [15], což je varianta evolučních algoritmů, se ukazuje jako efektivní metoda, kterou lze generovat funkční počítačové programy. Pomocí genetického programování je možné navrhnout jakoukoli turingovsky spočetnou funkci [26].

V současné době existuje mnoho aplikací evolučních algoritmů v mnoha inženýrských oborech [12]. Jednou z tradičních domén evolučních algoritmů je konstrukce efektivních systémů pro predikci [4].

Existuje několik způsobů, pomocí kterých lze evoluční algoritmy využít ke konstrukci prediktorů. Tím nejjednodušším je optimalizace parametrů již existujícího prediktoru, který je většinou dílem člověka-návrháře. Může se například jednat o optimalizaci koeficientů v rozhodovacích podmínkách programů nebo o nastavování vah neuronových sítí [2]. Složitějším přístupem, z hlediska složitosti problému, který musí evoluční algoritmus řešit, je návrh celého prediktoru. Zde je nejdůležitějším úkolem zvolit vhodnou reprezentaci kandidátního řešení.

V závislosti na tom, zda je evoluční algoritmus použit pro statická nebo dynamická trénovací data, lze hovořit o staticky vytvořeném prediktoru nebo o dynamicky adaptovaném prediktoru. Staticky vytvořené prediktory lze používat v případě, že se vlastnosti predikovaných dat v čase nemění. Pokud se mění, je nutné po čase znovu spustit evoluční proces a vygenerovat nový prediktor. V případě použití dynamického prostředí je prediktor adaptován na změny již v průběhu samotného evolučního procesu. Za předpokladu, že evoluční proces běží dostatečně rychle lze předpokládat, že se kandidátní řešení prediktoru stihnou adaptovat na nové prostředí.

2 Evoluce v reálném čase

V literatuře bylo prokázáno, že evoluční algoritmy dokáží generovat funkční řešení, která leží mimo rozsah možností člověka-návrháře [14], mohou provádět výpočet na nestandardních platformách [19,

11], mohou navrhovat funkční řešení pro extrémní podmínky [9], dokáží adaptovat stávající řešení na nový problém [13].

Evoluční algoritmy nejčastěji nacházejí uplatnění při řešení problémů, u nichž se fitness funkce v čase nemění. Popíšeme dva přístupy, kterými lze řešit problém proměnlivé fitness funkce.

První možností je snaha o maximální urychlení evolučního procesu. Evoluční algoritmus je spuštěn s nově vygenerovanou počáteční populací pokaždé, když je detekována změna prostředí. Ke změnám prostředí nesmí docházet v rychlém časovém sledu, protože systém musí mít dostatek času k vytvoření funkčního řešení.

Druhou možností je nerestartovat evoluční proces při změně prostředí, ale nechat populaci adaptovat na změněné prostředí [3]. Tyto systémy využívají evolučním procesem získaných informací. Tento přístup funguje pouze za předpokladu, že změna prostředí je určitého charakteru.

2.1 Funkce dynamického adaptivního systému

Dynamický evoluční adaptivní systém je většinou popisován jako systém s dynamickou fitness funkcí [22]. To znamená, že v každém generačním cyklu evolučního algoritmu může dojít ke změně fitness funkce a tudíž k rozdílnému ohodnocení jedince, jehož genotyp nebyl změněn.

Základní struktura dynamického evolučního systému zůstává přes různé možnosti nasazení podobná. Znalosti systému jsou reprezentovány množinou kandidátních řešení. Existuje zde evoluční jádro provádějící adaptaci množiny stávajících kandidátních řešení. Ohodnocování jednotlivých jedinců se provádí pomocí dynamické fitness funkce. Nejlepší řešení je pak předáváno do samostatné výkonné jednotky, která aplikuje dané řešení na vstupní data.

2.2 Aplikace

V dnešní době existuje několik aplikací, které využívají evoluční adaptivní systémy. V následujících podkapitolách jsou uvedeny vybrané příklady.

2.2.1 Evoluce hašovací funkce

Byl navržen systém pro dynamickou adaptaci hašovacích funkcí pro přístup do cache [7]. Hašovací funkce provádí mapování z 16-bitového adresového prostoru na 8-bitovou adresu paměti cache. Cílem je najít takovou hašovací funkci, která rovnoměrně pokrývá adresový prostor paměti cache v daném rozložení přístupů do paměti. Pro reprezentaci hašovací funkce bylo použito binární zakódovaní.

2.2.2 Evoluce obrazových filtrů

V literatuře je popsán adaptivní systém aplikovaný na rekonstrukci obrazové informace [22]. Systém je navržen tak, aby pracoval se vstupními daty přicházejícími jako poškozená obrazová informace. Tento systém pak vyfiltruje šum, který se v obrazu nachází. Systém je schopný rekonfigurace obrazového filtru za běhu pomocí blíže nespecifikovaného evolučního algoritmu.

2.2.3 Evoluce pravidel obchodování na mezinárodních trzích

Pro konstrukci a adaptaci pravidel mezinárodního obchodování ve FOREXu byl použit systém využívající evoluce v reálném čase [8]. FOREX je zkratka vycházející ze slov Foreign Exchange a znamená mezinárodní obchod. Tento systém využívá genetické programování k sestavování klasifikátorů v podově funkčních pravidel pro řízení obchodního agenta. Vytvořená pravidla jsou předávána systému

řídícímu tok financí. Tento systém se stará o samotné obchodování. Evoluční systém je částečně hybridní z pohledu běhu evolučního procesu. Evoluce totiž neběží na pozadí samotné predikce ani není inicializovaná změnou vstupních dat. Evoluční proces v tomto přístupu je inicializován v pravidelných časových intervalech. Nedochází k náhodné inicializaci populace, ale vychází se ze stávajícího stavu.

2.2.4 Evoluce prediktorů skoků

Pro predikování podmíněných skoků v procesoru byl navržen a odsimulován systém využívající evoluci v reálném čase pro adaptaci prediktorů [24, 23]. Jednotlivé prediktory jsou implementovány jako konečné automaty.

2.2.5 Řízení robotů

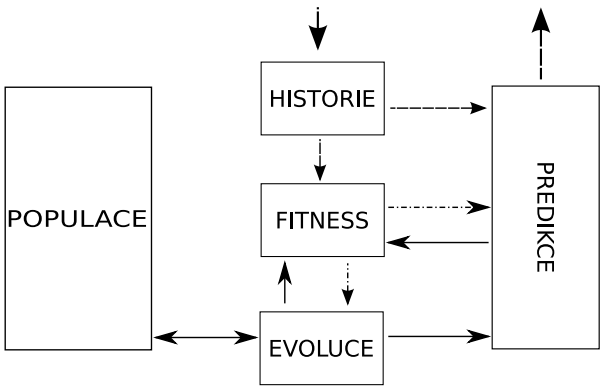
S pomocí genetického programování je možné reprezentovat znalosti systémů. V tomto případě bylo genetické programování použito pro reprezentaci znalostí o okolí v paměti malého robota. Robot získává informace pomocí senzorů. Cílem je vyhnout se překážkám v cestě. Evoluční algoritmus běžící v reálném čase provádí aktualizaci znalostí [20]. Bylo ukázáno, že evoluce v reálném čase dokáže řídit robota tak, aby nenarážel do překážek. Použitý evoluční algoritmus vykazuje zvýšení efektivity, pokud jsou trénovací data ze senzorů doplněna o jejich historii.

3 Disertační práce

Tato kapitola obsahuje stručný popis dosavadních činností v doktorském studiu a definuje předmět a cíle disertační práce.

3.1 Dynamický adaptivní systém

V rámci úvodních experimentů byl v programovacím jazyce C++ implementován systém využívající evoluční algoritmus běžící v reálném čase k řešení zvolené predikční úlohy. Systém byl navržen modulárně s ohledem na snadnou modifikaci jeho funkce. Je možná snadná změna všech jeho komponent bez nutnosti modifikace ostatního kódu.



Obrázek 1: Struktura systému prediktoru.

Na obrázku 3.1 je znázorněna struktura predikčního systému. Přicházející data jsou ukládána do vyrovnávací paměti, která zaznamenává historii několika posledních hodnot. Znalosti systému jsou

reprezentovány pomocí populace kandidátních řešení. Celý systém řídí evoluční jádro, které aktualizuje stav populace. Fitness jednotka ohodnocuje jednotlivá kandidátní řešení podle momentálního stavu prostředí, tedy historie vstupních dat. Nejlépe ohodnocení jedinci jsou předáváni do predikční jednotky. V predikční jednotce jsou vstupní data použita k určení predikované hodnoty. V predikční jednotce se nachází několik vzájemně soutěžících prediktorů. Výsledná predikovaná hodnota je určena jako funkční hodnota výsledků všech prediktorů. Kvalita jednotlivých prediktorů v predikční jednotce je také posuzována pomocí fitness funkce. Pokud kvalita prediktoru v predikční jednotce klesne pod přípustnou mez, je prediktor vyřazen a jeho místo může zaujmout jiné řešení.

3.2 Aplikace systému

Funkčnost tohoto systému byla ve zjednodušené formě vyzkoušena v úloze predikce podmíněných skoků v procesoru [24, 23]. Jednalo se o predikování řady binárních hodnot. Systém využíval jako model predikce konečné automaty [27]. Zjednodušení systému spočívalo v redukci složitosti predikční jednotky. Ta obsahovala pouze jeden prediktor, který byl nahrazen v každém generačním cyklu. I přes svou jednoduchost dokáže tento systém predikovat v 80% správný směr větvení programu.

V disertační práci bych chtěl dynamický adaptivní systém použít pro realizaci evolučního prediktoru dat v mezinárodním obchodu. Pro reprezentaci prediktorů bude použita stromová reprezentace vycházející z genetického programování. Úlohu hledání funkčního prediktorů chci transformovat na multikriteriální optimalizační problém. Jednotlivá řešení by byla ohodnocena podle několika kriterií, aby bylo možné systémem objektivně posoudit kvalitu jednotlivých prediktorů.

3.3 Potenciální problémy navrženého přístupu

I přes nasazení evolučních algoritmů v širokém spektru optimalizačních a návrhových úloh a intenzivní výzkum v této oblasti existuje mnoho doposud neuspokojivě vyřešeních problémů. V této části si popíšeme ty z nich, které se dotýkají zvolené aplikace.

Volba reprezentace Problém vhodného zakódování genotypu a jeho následné mapování na fenotyp je jedním z důležitých parametrů použitého evolučního algoritmu. Nevhodná reprezentace genotypu, nevhodně navržená fitness funkce nebo nevhodné genetické operátory mohou znemožnit nalezení efektivního řešení [1].

Bloat Bloat je anglický termín vztahující se ke genetickému programování. Obvykle popisuje situaci, kdy velikost stromu kandidátního řešení začíná přerůstat únosnou mez [25]. V tomto případě nastává stav, kdy evoluční operátory již nedokáží efektivně řídit proces evoluce. Nedochází ke zlepšování fitness hodnoty a přesto dochází k dalšímu zvětšování stromu. Bloat představuje skutečný problém, protože výrazně zpomaluje evoluční proces.

Růst stromu lze omezit zavedením maximální hloubky stromu. Toto omezení hloubky bohužel představuje významné omezení prohledávaného prostoru řešení.

Diversifikace populace Všeobecně se pokládá za pravdivé, že podmínkou dobré adaptability je zachování diverzity populace jedinců. Diverzita se ve většině případů zachovává na základě porovnávání rozdílnosti jednotlivých řešení. Porovnání se provádí jako výpočet vzdáleností jednotlivých řešení v prostoru všech možných řešení, což představuje problém u nelineárních struktur, jako jsou stromy [17]. V literatuře je publikováno mnoho metrik, kterými lze určit vzájemnou podobnost dvou jedinců [10]. Nevýhodou těchto publikovaných metrik je jejich přílišná specializovanost na určitou aplikaci.

Robustnost Evolučně navržené řešení problému je obtížně analyzovatelné z hlediska jeho funkčnosti. Nalezené řešení může vykazovat požadované chování pro používaná data, ale o jeho obecných vlastnostech nelze z jeho struktury mnoho odvodit (aniž bychom řešení prověřili). Důvodem je ta skutečnost, že řešení vzniklo v procesu generuj-a-testuj, tudíž bez použití klasického návrhového postupu.

Dále zde existuje riziko přetrénování jedince, kdy výsledné nalezené řešení postrádá požadovanou funkčnost pro jiná než trénovací data [5, 6].

Koncept dynamické dataptace Dynamická adaptace jedinců v populaci klade vysoké nároky na rychlost a efektivitu zvoleného evolučního algoritmu. Genetické programování využívá stovek jedinců v populaci. Pro ohodnocení jednoho jedince podle jednoho kriteria je zapotřebí jedné fitness funkce, pro více kritérií doba ohodnocení narůstá. Musí se tedy najít způsob, jak urychlit výpočet jednoho generačního cyklu vzhledem ke změnám v prostředí. Paralelizace evolučního algoritmu je částečným řešením tohoto problému. Výpočetní zátěž spojená s ohodnocením celé populace může být rozdělena mezi více výpočetních jednotek CPU. V posledních letech se začínají objevovat případy akcelerace genetického programování na GPU výkonných grafických karet [16, 21]. Dalším způsobem, kterým lze získat čas pro evoluci, je předzpracování vstupních dat do vhodnější reprezentace.

3.4 Cíle disertační práce

Cílem disertační práce je navrhnout a implementovatevoluční systém schopný adaptovat kandidátní řešení v populaci na v čase se měnící fitness funkci. Bylo zvoleno genetické programování jako podskupina evolučních algoritmů a řešený problém představuje konstrukce prediktoru vyvíjející se časové řady. Práci na tomto problému lze shrnout do následujících bodů.

1. Navrhnout obecnou strukturu systému umožňujícího adaptaci prediktorů v reálném čase s využitím evolučních algoritmů s ohledem na snadnou modifikovatelnost a přenositelnost.
2. Implementovat systém pro evoluční návrh prediktorů vývoje dat mezinárodního obchodu a na základě analýzy nalezených řešení identifikovat výhodné funkční elementy pro konstrukci kandidátního řešení.
3. Identifikované funkční elementy využít v systému evoluční adaptace prediktorů v reálném čase.
4. Vyhodnotit funkčnost tohoto systému v reálném provozu.
5. Na základě experimentů posoudit vliv rychlosti evoluce a multikriteriálního hodnocení prediktorů na výkon celého systému.

Výkonnost tohoto systému lze posuzovat vzhledem ke statickým prediktorům. Bude porovnána výkonnost genetického programování, lineárního genetického programování a kartézského genetického programování jako použité evoluční platformy. Zároveň lze porovnat výkonnost systému v závislosti na použité výpočetní architektuře (CPU, GPU) na které systém běží.

4 Závěr

V současné době se spráce na disertaci nachází ve fázi identifikace funkčních elementů systému. Výsledky ukazují, že využívání pouze genetického programování nevede k požadované přesnosti predikce. Ukazuje se, že využití některých prvků fraktální interpolace [28, 18] v genetickém programování vede ke zlepšení výsledků pro zvolený typ vstupních dat.

Dosahované výsledky jsou konzultovány s profesionálním obchodníkem FOREXu, který dodává trénovací a testovací data nutná pro vyhodnocení správné funkce systému. Systém je postupně navrhován tak, aby byl schopen reagovat na vstupní data, která jsou reprezentována na velmi nízké úrovni. Modularita systému slouží ke snadnější modifikaci jeho funkce a tím i snadnějšímu přechodu na jinou řešenou úlohu. V žádném případě se nesnaží jediný systém vyřešit všechny problémy popisované v 2.2.

Literatura

[1] Bentley, P. J.: *Representations Are More Important Than Algorithms: Why Evolution Needs Embryology*. In *Evolvable Systems: From Biology to Hardware ICES2000, keynote speech*, 2000.

[2] Branke, J.: *Evolutionary Algorithms in Neural Network Design and Training – A Review*. In *Proc. of the First Nordic Workshop on Genetic Algorithms and their Applications (1NWGA)*, editace J. T. Alander, 95-1, Vaasa, Finnland, 1995, s. 145–163.

[3] Branke, J.: *Evolutionary Approaches to Dynamic Optimization Problems*. In *Evolutionary Algorithms for Dynamic Optimization Problems*, editace J. Branke, T. Bäck, San Francisco, California, USA, 7 2001, s. 27–30.

[4] Bäck, T.: *Evolutionary Algorithms in Theory and Practise*. New York: Oxford Universty Press, 1996.

[5] Chongstitvatana, P.: *Using Perturbation to Improve Robustness of Solutions Generated by Genetic Programming for Robot Learning*. In *Journal of Circuits, Systems, and Computers*, ročník 9, 1999, s. 133–143.

[6] Chongstitvatana, P.: *Improving Robustness of Robot Programs Generated by Genetic Programming for Dynamic Environments*. In *IEEE Asia Pacific Conference on Circuits and Systems*, ročník 1, depart, s. 523–526.

[7] Damiani, E., Tettamanzi, A., Liberali, V.: *On-line Evolution of FPGA-based Circuits: A Case Study on Hash Functions*. In *The First NASA/DoD Workshop on Evolvable Hardware*, editace A. Stoica, J. Lohn, D. Keymeulen, Pasadena, California: IEEE Computer Society, 19-21 1999, ISBN 0-7695-0256-3, s. 26–33.

[8] Dempster, M. A. H., Jones, C. M.: *A real-time adaptive trading system using genetic programming*. In *Quantitative Finance*, ročník 1, institute of Physics Publishing, 2001, s. 397– 413.

[9] Garvie, M.: *Reliable Electronics through Artificial Evolution*. Dizertační práce, University of Sussex, 2005.

[10] Gustafson, S. M.: *An Analysis of Diversity in Genetic Programming*. Dizertační práce, University of Nottingham, 2004.

[11] Harding, S., Miller, J. F.: *Evolution In Materio : Evolving Logic Gates in Liquid Crystal*. *Journal of Unconventional Computing*, ročník 3, č. 4, 2007: s. 243–257.

[12] Haupt, R. L., Haupt, S. E.: *Practical Genetic Algorithms*. Wiley-Interscience, 2004.

[13] Holland, J.: *Adaptation in Natural and Artificial Systems*. Cambridge: MIT Press, 1992.

[14] Hornby, G. S., Globus, A., Linden, D. S., aj.: *Automated Antenna Design with Evolutionary Algorithms*. In *Proceedings of 2006 AIAA Space Conference*, 2006.

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

[16] Langdon, W. B., Banzhaf, W.: *A SIMD interpreter for Genetic Programming on GPU Graphics Cards*. In *EuroGP*, *LNCS*, ročník 4971, editace M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcazar, I. De Falco, A. Della Cioppa, E. Tarantino, Naples: Springer, 26-28 Březen 2008, ISBN 978-3-540-78670-2, s. 73–85.

[17] Langdon, W. B., Poli, R.: *Foundations of Genetic Programming*. Berlin Heidelberg: Springer-Verlag, 2002.

[18] Mazel, D. S., Hayes, M. H.: *Using iterated function systems to model discrete sequences*. *IEEE Transactions on Signal Processin*, ročník 40, 1992: s. 1724 – 1734.

[19] Miller, J. F., Downing, K.: *Evolution in Materio: Looking Beyond the Silicon Box*. In *The 2002 NASA/DoD Conference on Evolvable Hardware*, editace A. Stoica, J. Lohn, R. Katz, D. Keymeulen, R. S. Zebulum, Alexandria, Virginia, USA, 2002, s. 167–176.

[20] Nordin, P., Banzhaf, W., Brameier, M.: *Evolution of a world model for a miniature robot using genetic programming*. *Robotics and Autonomous Systems*, ročník 25, č. 1-2, 1998: s. 105–116.

[21] Robilliard, D., Marion-Poty, V., Fonlupt, C.: *Population Parallel GP on the G80 GPU*. In *EuroGP*, *LNCS*, ročník 4971, editace M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcazar, I. De Falco, A. Della Cioppa, E. Tarantino, Naples: Springer, 26-28 Březen 2008, ISBN 978-3-540-78670-2, s. 98–109.

[22] Sekanina, L.: *Evolvable Components: From Theory to Hardware*. Berlin Heidelberg: Springer-Verlag, 2004.

[23] Slaný, K.: *Branch Predictor On-line Evolution*. In *2008 Genetic and Evolutionary Computational Conference GECCO*, Association for Computing Machinery, 2008, ISBN 978-1-60558-131-6, s. 1643–1648.

[24] Slaný, K., Dvořák, V.: *Evolutionary Designed Branch Predictors*. In *13th International Conference on Soft Computing*, Faculty of Mechanical Engineering BUT, 2007, ISBN 978-80-214-3473-8, s. 18–23.

[25] Soule, T.: *Code Growth in Genetic Programming*. Dizertační práce, University of Idaho, 1998.

[26] Teller, A.: *Turing Completeness in the Language of Genetic Programming with Indexed Memory*. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, ročník 1, Orlando, Florida, USA: IEEE Press, 27-29 1994, s. 136–141.

[27] Yeh, T.-Y., Patt, Y. N.: *Two-Level Adaptive Training Branch Prediction*. In *International Symposium on Microarchitecture*, 1991, s. 51–61.

[28] Zhou, Y., Yip, P. C., Leung, H.: *On the efficient prediction of fractal signals*. *IEEE Transactions on Signal Processing*, ročník 45, 1997: s. 1865–1868.