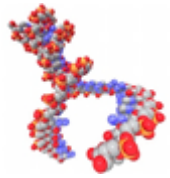
Portál pre odborné publikovanie ISSN 1338-0087

Genetické programování

Macháček Martin · Elektrotechnika

23.03.2011

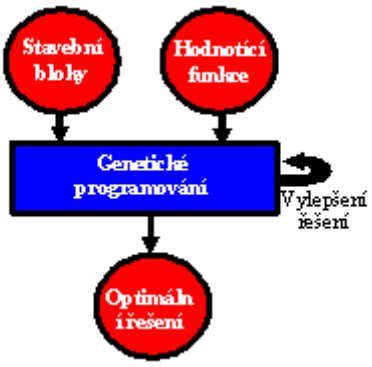


Evoluční výpočetní technicky (EVT) jsou numerické algoritmy, které vycházejí ze základních principů Darwinovy teorie evoluce, jejíž hlavní ideou je předávání rodičovského genomu novým potomkům a následné uvolnění životního prostoru těmto potomkům. Technologie EVT stojí a padá na existenci tzv. evolučních algoritmů (EA), které v podstatě tvoří většinu EVT. Mimo EA existují ještě další rozšíření, jako genetické programování, kterým se tato práce zabývá, evoluční hardware apod.[2]

Evoluční algoritmy představují netradiční přístup při hledání optimálního řešení složitých optimalizačních problémů, které nejsou řešitelné klasickými technikami. Evoluční algoritmy v současnosti patří mezi základní nástroje moderní informatiky v případech hledání řešení v extrémně složitých situacích, kdy použití standardních deterministických metod založených na technikách úplného prohledávání není možné. Ukazuje se, že evoluční metafora je velmi efektivním přístupem k řešení těchto složitých problémů a to zejména v případech, kdy nepotřebujeme optimální řešení problému, ale vystačíme si i s kvalitním suboptimálním řešením.[2], [3]

2. Genetické programování

Genetické programování (GP) je jedna z evolučních výpočetních technik (EVT), která slouží k řešení optimalizačních problémů pomocí počítačů, vycházející z principů přirozeného výběru podle Darwinovy evoluční teorie. Zjednodušený princip GP je takový, že stanovíme základní stavební bloky, ze kterých může být řešení složeno, a zároveň stanovíme analytickou metodu, která bude popisovat, jak dobře dané řešení zadanému problému vyhovuje. Zjednodušený postup je znázorněn na obrázku (Obr.1).



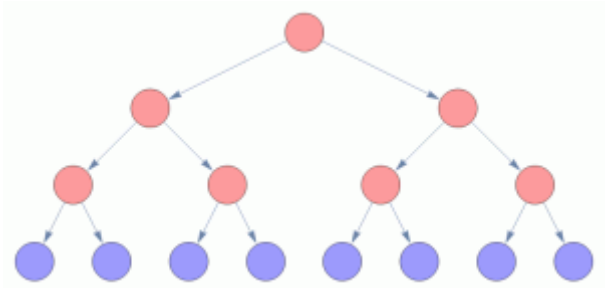
Obr. 1.:Schéma GP

Historie GP sáhá až do roku 1957, kdy se Friedberg snažil vytvořit samo-učící algoritmus. Program byl reprezentován jako sekvence instrukcí, jejichž kombinací měl vzniknout výsledný program. [9] Rozkvět GP však začíná teprve v 80. letech 20. století, kdy se objevují teoretické práce vědců Smitha a Cramera. Smith a Cramer ve svých vědeckých pracích o evolučních algoritmech definují genetické operace křížení a mutace a zabývají se reprezentací jedinců pomocí stromové struktury. [9]

Za zakladatele GP je nicméně považován britský vědec John R. Koza, který ve svých pracích položil teoretické základy GP a také definoval testovací příklady pro ověření funkčnosti GP. Jako příklad lze uvést problém umělého mravence nebo příklady z oblasti symbolické regrese. [8], [9] S rozvojem počítačů na konci 20. století a na začátku 21. století nachází GP uplatnění v mnoha oblastech, např. v medicíně, ekonomii a finančnictví, fyzice, astronomii, robotice, v oblasti komunikačních technologií, vojenských technologií nebo biochemii.

2.1 Reprezentace

Programy, výrazy nebo algoritmy, které v GP představují hledané řešení, bývají nejčastěji reprezentovány ve formě tzv. syntaktických stromů. Příklad syntaktického stromu je uveden na obrázku (Obr.2). [4]



Obr. 2.:Syntaktický strom

Syntaktický strom popisuje strukturu programu a skládá se z uzlů a listů. Listy představují koncové vrcholy stromu. [5]

2.2 Terminologie

2.2.1 Jedinec, populace

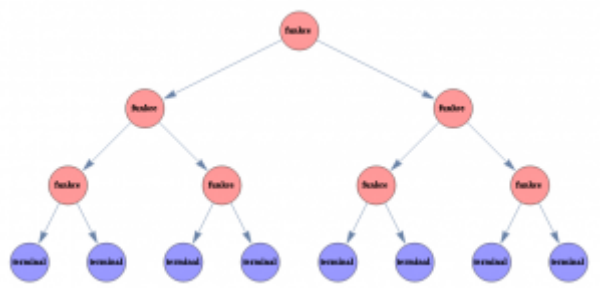
Stejně jako u ostatních EVT, tak i v případě GP se setkáváme s pojmy jedinec a populace. Jedinec představuje v GP jedno aktuální konkrétní řešení daného problému a populací se rozumí množina obsahující zvolený počet jedinců. [4]

2.2.2 Funkce, terminály

Každý jedinec má svou specifickou strukturu a je konstruován ze dvou množin symbolů. Z množiny terminálů $T = \{t_1, t_2, .., t_m\}$ a z množiny funkcí $F = \{f_1, f_2, .., f_n\}$. Každá konkrétní funkce $f_i \in F$ má specifický počet argumentů b_1,b_2,..,b_j. V závislosti na konkrétním řešeném problému mohou být funkcemi standardní aritmetické operace (sčítání, odčítání, násobení a dělení), standardní matematické funkce (sin, cos, tan, cotg, abs, apod.), logické operace (and, or, not, xor, apod.), podmíněné příkazy (if-then-else, apod.), iterativní operátory (do, while, apod.) a jiné. Terminály mohou být

proměnné, konstanty nebo funkce bez argumentů, které mají nějaký vedlejší efekt (MoveLeft, TurnRight, apod.). [6]

Při reprezentaci jedinců pomocí syntaktických stromů představují uzly funkce a listy reprezentují terminály. Počet hran vycházejících z každého uzlu představuje počet argumentů funkce. [5]

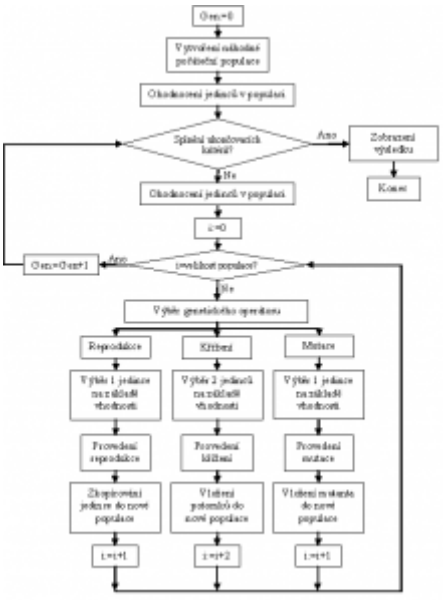


Obr. 3.:Reprezentace jedince

2.3 Algoritmus genetického programování

2.3.1 Schéma algoritmu genetického programování

Algoritmus GP se skládá z několika částí, které na sebe logicky navazují. Schéma algoritmu GP je znázorněno na obrázku (Obr.4).



Obr. 4.:Algoritmus GP

Samotný algoritmus začíná vygenerováním počáteční populace. Populace má předem definovanou velikost a je volena v závislosti na složitosti řešeného problému. Po vytvoření počáteční populace dojde k ohodnocení všech jedinců v populaci předem stanovenou analytickou metodou. Následuje kontrola splnění ukončovacích podmínek. Pokud tyto podmínky nejsou splněny, dochází k vytvoření nové generace jedinců. Nejprve se zcela náhodně vybere tzv. genetický operátor – reprodukce, mutace nebo křížení. Následně se tento genetický operátor aplikuje na jedince vybrané na základě jejich vhodnosti. Při reprodukci a mutaci je vybrán jen jeden jedinec, pro křížení je třeba mít jedince dva.

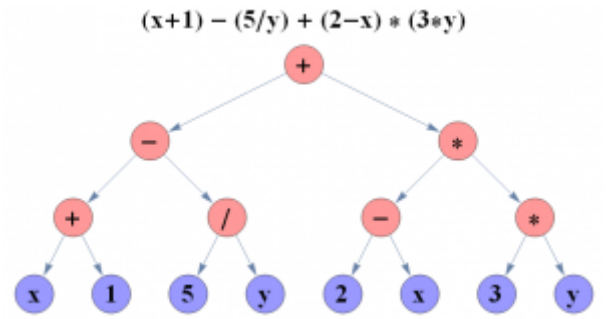
Aplikací genetických operátorů vzniknou potomci, kteří jsou vloženi do nové populace. Potomci jsou tímto postupem vytvářeni tak dlouho, dokud není velikost nové populace stejná jako velikost populace původní. U nově vzniklé populace dochází opět k ohodnocení všech jedinců a celý proces se opakuje do té doby, dokud nejsou splněny ukončovací podmínky. Ukončovací podmínky jsou dvojího typu. Jednak může dojít k ukončení evoluce, pokud došlo k vytvoření předem stanového počtu generací nebo v případě, že se v dané generaci vyskytuje optimální řešení nebo dostatečně kvalitní suboptimální řešení. [1], [3], [6]

2.4 Inicializace populace

V GP jsou jedinci v počáteční populaci vytvářeni náhodnou kombinací funkcí a terminálů z daných množin F a T. Zároveň se na začátku evoluce definuje maximální hloubka syntaktických stromů v počáteční populaci, čímž se rozumí počet hran vedoucích z kořene stromu k nejvzdálenějšímu uzlu. Ve většině případů se používá počáteční hloubka D_{max}=6. Pro generování počátečních existuje několik metod, z nichž nejpoužívanější jsou úplná metoda, rostoucí metoda a metoda půl na půl. [5]

2.4.1 Úplná metoda

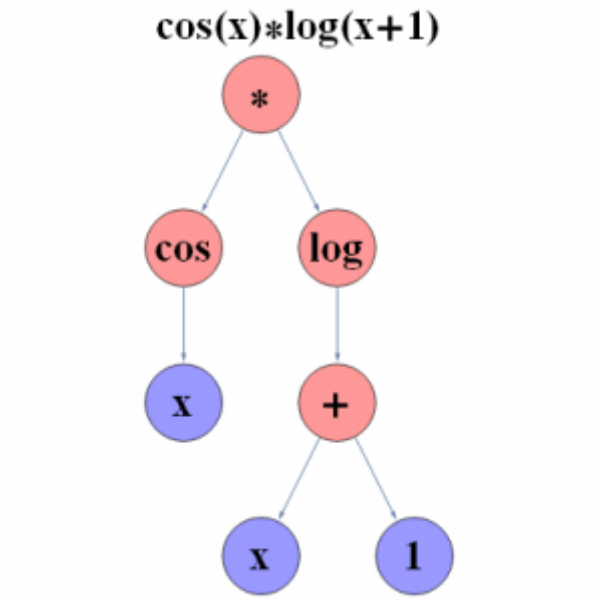
U této metody jsou vytvářeny stromy, kde se všechny listy nacházejí ve stejné hloubce. Neznamená to ovšem, že všechny počáteční stromy budou mít stejný počet uzlů nebo stejný tvar. Tvar syntaktického stromu je ovlivněn počtem argumentů funkcí na pozicích uzlů. Vytváření stromů probíhá tak, že se v maximální hloubce vybírají pouze terminály a v ostatních hloubkách se volí pouze funkce. [5]



Obr. 5.:Tvorba populace úplnou metodou

2.4.2 Růstová metoda

Při použití růstové metody vznikají rozmanitější syntaktické stromy, protože v každé vrstvě s výjimkou poslední vrstvy může být použita funkce nebo terminál. Jakmile je vybrán terminál, je příslušná větev stromu hotova bez ohledu, zda se dosáhlo požadované maximální povolené hloubky. [5]



Obr. 6.:Tvorba populace růstovou metodou

2.4.3 Lineární půl na půl metoda

V praxi se ovšem doporučuje používat metodu půl na půl, což znamená, že polovina jedinců v počáteční populaci je vytvořena úplnou metodou a druhá polovina je vytvořena rostoucí metodou. Zároveň by měly být rovnoměrně zastoupeny syntaktické stromy různých hloubek. Tedy, např. při velikosti populace N=100 a maximální hloubce D^{max}