APLIKACE GA V PROSTŘEDÍ MATLAB

Radek Matoušek

Vysoké učení technické v Brně
Fakulta strojního inženýrství
Ústav automatizace a informatiky
Technická 2, 616 69 Brno, Česká Republika
matousek@uai.fme.vutbr.cz

Abstrakt: Genetické algoritmy – GA, představují heuristické optimalizační techniky založené na principech přirozené selekce a přírodní genetiky. Abstrahováním těchto přírodních principů vznikl počítačový algoritmus schopný řešit složité optimalizační problémy. Na základě svých operátorů je GA schopen poměrně účinně prohledávat prostor všech možných řešení. Iterační postup je prováděn bez znalosti gradientu dané funkce, což má samozřejmě svá pozitiva i negativa. Prezentovaný GA v prostředí MatLab, tak může představovat poměrně snadnou cestu pro řešení rozličných matematických i technických úloh.

Klíčová slova: Genetický algoritmus, počítačový model GA, m-funkce pro GA

1. ÚVOD

Abstrahováním přírodních biologických procesů byl navrhnut optimalizační mechanismus označovaný jako genetický algoritmus. Genetický algoritmus (GA) je adaptační metoda, která může být použita při řešení kombinačních a optimalizačních problémů. Z pohledu matematické optimalizace se řadí mezi tzv. heuristické metody. Vzhledem ke skutečnosti, že GA využívají náhodného procesu, je též možné označit GA za heuristicko-stochastickou metodu.

Základní principy genetických algoritmů položil John Holland (1969,1975) spolu se svými kolegy a studenty z Michiganské university. Pravý „boom“ genetických algoritmů však představovala až publikace D.E.Goldberga (1989), *Genetic Algorithms in Search,Optimization & Machine Learning*. Podrobný popis GA je za hranicí tohoto příspěvku a lze jej nalézt například v [1,2].

2. BIOLOGICKÁ PODSTATA GA

Podstata GA vychází z **genetického procesu** a **přirozené selekce** biologických organismů. Vzhledem k tomu, že GA jsou inspirovány přírodním biologickým vzorem, bude nyní vhodné objasnit některé základní pojmy z této oblasti.

Jak již bylo řečeno, postata GA vychází z genetického procesu a přirozené selekce biologických organismů. Opodstatněnost tohoto biologického optimalizačního algoritmu je zřejmá již při letmém pohledu do říše živočichů a rostlin. Zde můžeme nalézt nespočet druhů, které jsou velice dobře přizpůsobeny životu ve své přirozené lokalitě. Adaptace těchto organismů na specifické podmínky prostředí, byla způsobena právě oním biologickým mechanismem. Tento proces si klade za cíl vytvořit takové změny v organismu, které by zaručily jeho přežití a optimálnější fungování. Jako příklad zdárného vývoje si můžeme uvést některé biologické organismy:

- *člověk*, považujeme za nejdokonalejší známou strukturu mozku,
- *žralok*, vynikající hydrodynamický tvar těla, navigace,
- *masožravé rostliny*, nedostatek vhodných živin nahradily jiným způsobem,
- *atd.*

Nutno poznamenat, že ne vždy byl vývoj progresivní. Takové organismy potom postupně vymíraly, aby přenechaly místo jiným, ve vývojovém procesu zdárnějším organismům. Vzhledem k „časové náročnosti“ není možné dosáhnout těchto změn za jedno období života, tj. bráno v určité skupině čí populaci, za jednu generaci. Za první vědecké poznatky a závěry o uvedených biologických mechanismech jmenujme:

- Georg J. Mendel – kniha: *Law of Heredity,* zákony dědičnosti, genetické procesy
- Charles Darwin – kniha: *The Origin of Species*, přirozený výběr, selekce

Po mnoho generací se přírodní populace vyvíjí podle principů přirozeného výběru, kde „přežije jen silnější“. To znamená, že jedinec s lepšími vlastnostmi dominuje nad jedincem s vlastnostmi horšími a má tedy vyšší pravděpodobnost přežití.

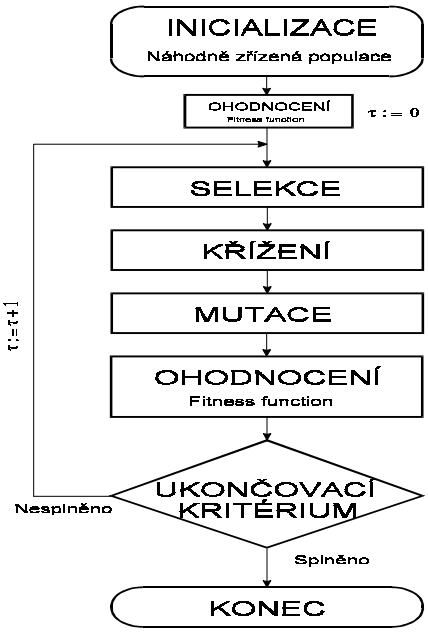
Procesem zvaným reprodukce je zajišťováno zachování druhu, přičemž dochází k přepisu tzv. dědičných informací. Při tomto procesu může dojít ke změnám genetické informace, čímž může vzniknout jedinec, jak s lepšími, tak s horšími vlastnostmi. Eliminace jedinců s horšími vlastnostmi je zajištěna právě principem přirozeného výběru. Po této selekci se obecně proces znovu opakuje, přičemž nemusí být nikdy ukončen.

3. POČÍTAČOVÝ MODEL GA

Napodobováním biologického procesu jsou genetické algoritmy schopny „vyvinout” řešení pro různé reálné problémy. Základní mechanismus GA je velmi jednoduchý a neobsahuje nic víc, než kopírování a výměnu části řetězců (genetických informací) s následným výběrem podle určitých kritérií. Při použití vhodné počítačové reprezentace tak dostáváme do rukou velmi mocný nástroj, vhodný k řešení mnoha problémů. GA mohou být potom aplikovány na ty problémy, kde běžné konvenční matematické metody selhávají nebo jsou z jiných důvodů nevýhodné.

GA představují matematický model biologického procesu, přenesený do prostředí číslicového počítače. GA pracuje paralelně s *N* řešeními daného problému, jednotlivá řešení jsou označována jako jedinci či chromozomy. K ohodnocení kvality každého řešení je užita ohodnocovací funkce, označovaná jako *fitness function*. Označíme-li prostor všech možných řešení daného problému Ω, můžeme říci že cílem GA je prohledat tento prostor Ω a najít optimální řešení. K tomuto cíli využívá GA třech základních operátorů, jak je znázorněno na obrázku 1. Jeden cyklus GA je označován jako generace, což je možno brát jako ekvivalent iterace u numerických metod. Popis jednotlivých částí algoritmu je následující:

- *INICIALIZACE:* Prvním krokem je inicializace všech ***N*** jedinců v populaci *P* a jejich ohodnocení. Inicializace může bít náhodná nebo heuristická.
- *SELEKCE – S:* Operátor eliminující nevhodná řešení.
- *KŘÍŽENÍ – C:* Dochází k výměně informací mezi jedinci a k přenosu informace do další generace.
- *MUTACE – M:* Operátor simuluje náhodné změny při přepisu genetické informace.



Obr. 1. Vývojový diagram pro GA

Aplikovaný GA představuje tzv. *generační model*, který je charakterizován tím, že nová generace *N* jedinců je celá výsledkem aplikování výběru (selekčního operátoru) jedinců z populace předchozí. Formální popis odpovídá následujícímu vztahu:

$$P_{\tau+1} = f\left(P_{\tau}, \xi\right), \qquad \xi \in \left(S, C, M\right) \tag{1}$$

kde: P…populace, množina možných řešení
ξ…množina GA operátorů
τ…číslo generace

Reprezentace a kódování informace: Aby bylo možné provést výše uvedený genetický algoritmus (obr. 1.), je samozřejmě nutné zvolit vhodnou reprezentaci informací a jejich kódování v chromozómu. V genetických algoritmech představuje chromozóm obvykle vektor bitů, tj. hodnot *0* nebo *1*. Pro většinu reálných inženýrských problémů však nelze přímo použít binární řetězcovou reprezentaci. Pro transformaci na celočíselný typ se používá nejčastěji přirozený binární nebo Grayův kód. Pro mnoho problémů nám však ani zakódování na uzavřený interval přirozených čísel (včetně nuly) nepostačuje. V tomto případě je třeba udělat další převod do oblasti reálných čísel[1]. Další možností pro kódování binárního řetězce je užití přímé reprezentace dle normy IEEE 754 – binární aritmetika s pohyblivou řádovou čárkou.

[1] Pojmem reálné číslo se zde rozumí reálný typ v rámci počítačové aritmetiky.

4. IMPLEMENTACE GA DO PROSTŘEDÍ MATLAB

Pro realizaci GA v prostředí MatLab byla navržena sada funkcí. Vlastní proces programové realizace a nasazení GA je možné rozdělit na dvě základní části:

- Navržení příslušné ohodnocovací funkce - fitness funkce, a to ve formě funkce m-file.
- Implementace a začlenění funkcí GA ve vlastním programu. Tento proces zahrnuje nastavení parametrů a operátorů GA.

Poznámka: parametry funkcí uzavřené symboly: '⟨, ⟩' jsou doporučené, ale nepovinné.

Fitness funkce: Je realizována jako standardní MatLab funkce a uložena jako m-file.

Syntaxe: function fce=nazevFitnessFunkce(X);

fce… vektor fitness hodnot všech jedinců populace
X … matice proměnných v populaci, tj. hodnoty proměnných pro všechny jedince

Příklad:

```
% F6 DeJong's function
%   fce(x) = A*n + sum(x.^2-A*cos(2*pi*x)), A=10, x=x1,x2,...
%   -5.12 <= x1,x2,...,xn <=5.12; min(f)=(0,...,0)=0

function fce = f6(X);

if nargin < 1, error('Not enough input arguments.'); end

[r s]  = size(X);
A  = 10;

fce = A*s + sum((x.^2-A*cos(2*pi.*x)),2);
```

Funkce GA:

initga inicializace objektu GA a nastavení základních parametrů populace

Syntaxe: GA = initga(nParam,fitName,⟨nBitParam,nIndi,mInit,iRange,mCode⟩);

GA	GA objekt
nParam	počet proměnných v jednom řešení
fitName	název fitness funkce
nBitParam	počet bitů na jednu proměnnou
nIndi	počet jedinců v populaci, tj. počet možných řešení
mInit	metoda inicializace populace
iRange	rozsah hodnot proměnných
mCode	metoda kódování a dekódování binárního řetězce

setga nastavení parametrů GA operátorů

Syntaxe: GA = setga(GA,mSet,⟨select,pselect,cross,pcross,mutation,pmutation⟩);

GA	GA objekt		
mSet	metoda nastavení parametrů GA operátorů		
select	metoda selekce	pselect	parametr(y) selekce
cross	metoda křížení	pcross	parametr(y) křížení
mutation	metoda mutace	pmutation	parametr(y) mutace

fitness vypočte hodnoty fitness pomocí definované fitness funkce, vzhledem k zadaným hodnotám v metodě initga
select, cross, mutation základní operátory GA

Syntaxe: GA = fitness(GA); GA = select(GA); GA = cross(GA); GA = mutation(GA);

GA GA objekt

Aplikace GA:

Abstraktně:

```
GA = initga(… );
GA = setga(... );
GA = fitness(GA);

% cyklus GA
while ukončovací_kritérium
        GA = select(GA);
        GA = cross(GA);
        GA = mutation(GA);
        GA = fitness(GA);
end %while
```

Jako ukončovací kritérium je možné užít:
- počet generací
- přípustnou hodnotu fitness
- čas výpočtu

Příklad:

```
function GA = demoGA;
% testovani GA: minimalizace DeJong funkce F6

nG = 100;% max. pocet generaci 100

% parametry populace GA
nParam     = 2;
nBitParam  = 24;
nIndi      = 50;
mInit      = 'random';
iRange     = [-5.12 5.12];
fitName    = 'f6';
mCode      = 'gray'

GA = initga(nParam,nBitParam,nIndi,mInit,iRange,fitName,mCode);

GA  = setga(GA,'expert');  % expertní nastavení

% ------------------------------------------------------
% GA-LOOP START
while GA.nGener<nG

   GA = fitness(GA);

   % zobrazení čísla generace a nejlepšího řešení
   [valueMIN,position] = min(GA.fit);
    BCD = GC2BC  (GA.pool(position,:),GA.nBitParam);
    R   = BC2real(BCD,GA.nBitParam,GA.iRange);
    [GA.nGener,valueMIN,R];

    GA.nGener = GA.nGener+1;

   GA = select(GA); GA = cross(GA); GA = mutation(GA);

end
% GA-LOOP END
% ------------------------------------------------------
```

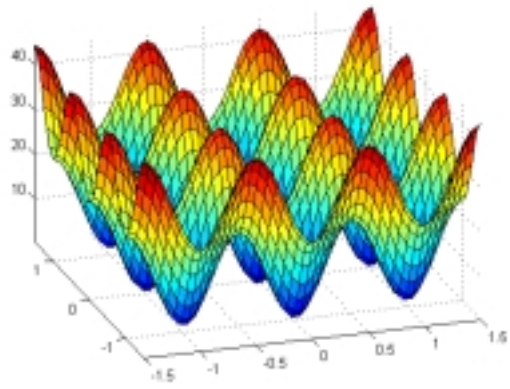
Pomocné funkce: GC2BC…universální funkce pro převod Grayova kódu do binárního
BC2real…universální funkce pro převod binárního kódu na typ real

Syntaxe:
```
binární_číslo = GC2BC(matice_bitů, počet_bitů_na_parametr);
pseudorealné_číslo = BC2real(matice_binárních_čísel, …
                    počet_bitů_na_parametr, rozsah_pseudoreálné_proměnné)
```

Uvedený příklad ukazuje implementaci GA pro nalezení minimální hodnoty multimodální funkce označované jako F6 a dané následujícím vztahem:

$$f_6(\vec{x}) = n \cdot A + \sum_{i=1}^{n} \left(x_i^2 - A \cdot \cos(2\pi \cdot x_i) \right), \quad x_i \in [-5.12, 5.12]; \qquad \min \; f_6(\vec{x}) = 0 \tag{2}$$

Náročnost úlohy nejlépe demonstruje zobrazení funkce F6 pro dvě proměnné dle obrázku 2. Představa komplikovanosti optimalizační úlohy, například pro osm proměnných dle vztahu (2) při použití klasických gradientních metod je zřejmá.



Obr. 2.: F6 – Rastigrinova funkce

5. ZÁVĚR

Na závěr bych chtěl poukázat na základní, a v mnoha případech zřejmé, výhody a nevýhody implementace genetických algoritmů do prostředí MatLab. Tyto úvahy vyplývají jak z technických možností tohoto programového prostředí, tak i z praktických zkušeností, např. dle [3,4]. Programová realizace byla provedena ve verzi R11, přičemž obecné závěry je možné použít i pro verze nižší. Srovnání bude provedeno na základě porovnání a možné implementace v jiném programovém prostředí (C/C++, Pascal, Assembler, …).

Výhody:

- Snadná implementace GA do vlastního programu
- Jednoduchá definice fitness funkce
- Jednoduchá aplikace matematických funkcí MatLabu
- Možnost využití specializovaných toolboxů
- Využití knihovny grafických funkcí pro snadnou vizualizaci
- Vyspělý ladící režim (obecně vlastnost překladače)

Nevýhody:

- Malá rychlost provádění výpočtu vyplývající z:
  - interpretového režimu
  - absence přímých HW orientovaných bitových operací
- Nevhodné datové struktury pro realizaci GA

Obecně je možné říci, že výhody aplikace GA v prostředí MatLab jsou značné, obzvláště ve fázi vývoje algoritmů a při řešení méně náročných úloh. Pro praktické a rozsáhlejší aplikace GA, obzvláště v reálném čase, by bylo vhodné dané funkce zkompilovat, případně užít jiný programový modul (C/C++,Pascal), a to buď samostatně či ve spojení s MatLabem, např. pomocí DDE.

6. PODĚKVÁNÍ

Děkuji všem bezejmenným recenzentům za případné připomínky či dotazy. Uvedené GA funkce pro prostředí MatLab je možné obdržet na výše uvedené e-mailové adrese. Referát byl zpracován za podpory grantu MŠMT # J22/98: 261100009 a J22/98 260000013.

LITERATURA:

[1] Goldberg, D. E. (1989) "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley

[2] Matoušek, R. (1999) "*Využití vybraných metod UI pro optimalizaci regresního modelu s vazební podmínkou*", Teze disertační práce, ÚAI VUT-FSI Brno, Czech Republic, (In Czech)

[3] Matoušek, R., Ošmera, P. (1999) "*A Fuzzy Setting of GA Parameters*", p.154-160, Proceeding of 7th Zittau Fuzzy Colloquium 1999, Germany

[4] Matoušek, R., Ošmera, P., Roupec, J. (2000) "*GA with Fuzzy Inference System*", p.646-652, Proceedings of the 2000 Congress on Evolutionary Computation - CEC00, La Jolla, California, USA