GATE 2.0
TOOLBOX PRO HEURISTICKOU OPTIMALIZACI

Radomil Matoušek

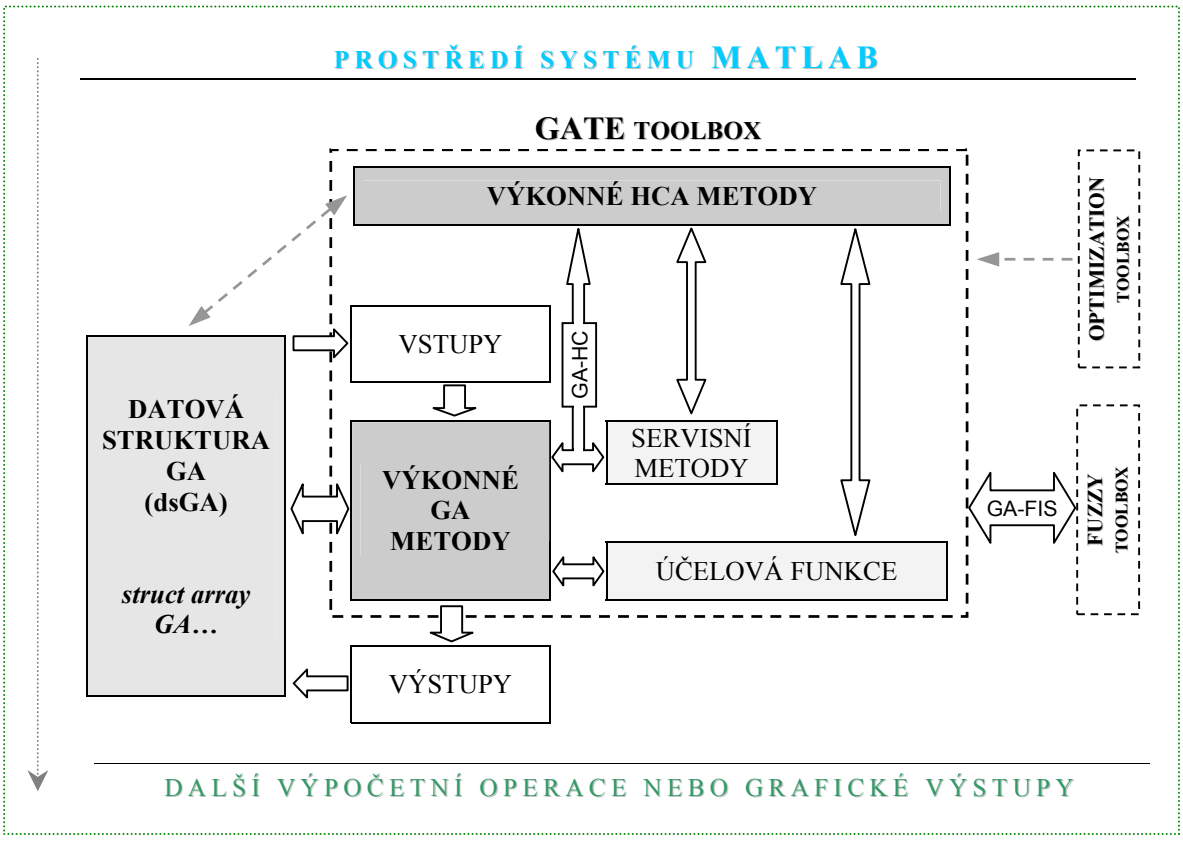
Vysoké učení technické v Brně, FSI, Ústav automatizace a informatiky

Abstrakt

Matematické optimalizační metody zahrnují široké spektrum algoritmů, jejichž základem jsou jak standardní matematické metody (simplexová metoda, gradientní metody, metoda větví a mezí aj.) tak metody heuristické, které využívají nejrůznějších principů. Prezentovaná sada funkcí souhrnně označená jako GATE 2.0 toolbox [1] je určena pro heuristickou optimalizaci, která patří do skupiny tzv. soft computing metod. Tato knihovna zahrnuje specifickou implementaci metod GA (Genetických algoritmů) a HCA (Horolezeckých algoritmů). Příspěvek prezentuje stručný popis knihovny, včetně demonstračních příkladů jejího užití. Nezabývá se komparací s Genetic Algorithm and Direct Search Tooolboxem firmy The Mathworks, případně jinými. V dalším textu jsou předpokládány jisté znalosti [1],[2] z oblasti GA a HCA.

1. Úvod

Obecné členění GATE toolboxu, včetně základní datové vazby a návaznosti na případné další toolboxy, je znázorněno schématem (Obr. 1). Základem pro realizaci GA výpočtů je zavedení speciální datové struktury GA, dále též obecně označované *dsGA*. Tato struktura je typu *struct array* a obsahuje důležité informace sloužící jak k vytvoření *populace* (znázorněno blokem *vstup*), tak k následným GA operacím nad touto populací. Ve struktuře jsou kromě vybraných parametrů GA obsaženy též základní údaje o průběhu optimalizace a v případě potřeby další statistická data.



Obr. 1: Blokové schéma vazeb GATE toolboxu.

Blok označený jako *výkonné metody GA* představuje základní GA mechanismus realizovaný operátory *selekce*, *křížení* a *mutace*. V případě GATE toolboxu, můžeme do bloku výkonných metod zahrnout další postupy jako například *migraci*, *elitizmus*, *GA-HC, GA-FIS* aj. Jelikož základní vlastností GATE je práce s binárními řetězci, bylo vhodné vyvinout řadu specializovaných funkcí pro práci s nimi. Tyto funkce plus některé další pomocné jsou označeny jako *servisní metody*. Základním kritériem pro výpočty je pak sestavená *účelová funkce.* Tato funkce je užita i v případě samostatných výkonných HCA metod.

2. Datová struktura

Jak bylo uvedeno, je v případě nasazení GA v rámci GATE toolboxu, základním požadavkem vytvoření reprezentující datové struktury *dsGA*. Tuto strukturu je možné vytvořit jak v *command window*, tak implementovat do uživatelského skriptu nebo funkce, což je k dalšímu zpracování výhodnější. Struktura může být pochopitelně uložena do datového souboru a pro další potřebu následně načtena.

Přístup k jednotlivým entitám *dsGA* se provádí pomocí „tečkové notace“ jak je běžné u objektových struktur. Specifické vlastnosti *dsGA* vyplývají z dalšího kontextu (Obr. 2). Tučně zvýrazněné části datové struktury GA, tj. GA.* (např.: GA.funName, GA.funOpt, atd.) jsou povinné.

```
GA =

      funName: 'F6'                    } Informace vztahující se k objektivní funkci (fitness) a
       funOpt: 'min'                   } ke kritériu hledání optima.

       iParam: [-5 5]                  } Definiční interval optimalizovaných proměnných,
        iType: '[-]'                   } včetně typu definičního intervalu.

        nParam: 9                      }
     nBitParam: 10                     } Parametry nutné k inicializaci populace a switch
         nIndi: 50                     } určující metodu dekódování binárních vektorů.
         mCode: 'GC'                   }
         mInit: 'random'               }

      rndSeed: 13                      } Inicializační parametr generátoru náhodných čísel

       nGener: 100                     } Počítadlo generací, je inicializováno hodnotou 1.

          fit: [50x1 double]           }
         pool: [50x90 double]          } Základní datová struktura nad kterou pracuje GA, tj.
        param: [50x9 double]           } populace a příslušné ohodnocení jedinců.

       winFit: [100x1 double]          }
      winPool: [100x90 double]         } Datová struktura obsahující data z průběhu GA
     winParam: [100x9 double]          } optimalizace.

    flagElite: {'winner'  [10]}        }
     fitElite: 0                       } Datová struktura obsahující data nejúspěšnějšího
   paramElite: [0 0 0 0 0 0 0 0 0 0]   } (nejúspěšnějších) jedinců.
    poolElite: [1x90 double]           }

    migration: [1x1 struct]            } Datová struktura obsahující informace nutné při
                                       } aplikaci migračního modelu GA.
      meanFit: []                      } Datová struktura obsahující další statistická data
                                       } z průběhu optimalizace. Vytvořeno uživatelem.
```

Obr. 2: Příklad již inicializované datové struktury GA (dsGA) v iteračním kroku sté generace.

Ve výše uvedené datové struktuře *dsGA* je kořenový identifikátor pojmenován jako „*GA*“. Pro aplikaci GA metod může být tento identifikátor nazván libovolně. Identifikátory nižší úrovně a rovněž obsah daných proměnných, musí být zvoleny dle specifik uvedených v tabulce (Tab. 1), což je demonstrováno na příkladu výpisu datové struktury GA (Obr. 2). Pevně volené hodnoty některých proměnných mohou být označeny jako tzv. přepínače (switch), protože svým nastavením pevně určují činnost celého GA procesu.

POŽADAVEK **EXISTENCE, SWITCH**	**PROMĚNNÁ DATOVÉ STRUKTURY GA** (nadřazený identifikátor GA je vynechán)	**MOŽNÉ HODNOTY** nebo **DATOVÝ TYP**	**POPIS**
☑☐	`funName`	`char array`	Obsahuje název objektivní funkce, tato funkce je sestavena uživatelem a je po této stránce základem pro další užití GATE toolboxu.
☑☑	`funOpt`	`'min'` `'max'`	Udává kritérium optimalizačního problému. Hodnota `'min'` reprezentuje minimalizační problém, `'max'` reprezentuje maximalizační problém.
☑☐	`iParam`	`double array`	Nastavuje definiční interval optimalizovaných parametrů objektivní funkce. Pokud je pro více parametrů použit jeden rozsah, např. `iParam` = [r s], je tento automaticky vztažen na všechny parametry. Obecně je `iParam` = [r_1 s_1; r_2 s_2; ...; r_m s_m]
☐☑	`iType`	`'[-]'` `'[0)'`	Určuje typ užitého definičního intervalu: `'[-]'` dekódování na uzavřený interval [`iParam`], `'[0)'` dekódování na interval [`iParam`). *Pozn.:* pro základní GA metody je `'[-]'` přednastaveno.
☑☐	`nParam`	`double`	Udává počet reálných optimalizovaných parametrů. Hodnota je určena definicí objektivní funkce.
☑☐	`nBitParam`	`double` `double array`	Udává počet bitů použitých pro kódování daného reálného optimalizovaného parametru. *Pozn.:* Standardně je užito ekvivalentního počtu bitů pro každý parametr. V případě potřeby je možno užít spec. servisních metod, které dovolují pracovat s volitelnou hodnotou, zvlášť pro každý parametr [nBP_1 nBP_2 … nBP_m].
☑☐	`nIndi`	`double`	Udává počet jedinců v populaci.
☑☑	`mCode`	`'BC'` `'GC'` `'NT'`	Určuje způsob dekódováni binárního řetězce (jedince): `'BC'` přímé binární dekódování, `'GC'` Grayovo binární dekódování, `'NT'` přímé užití binární reprezentace..
☑☑	`mInit`	`'random'` `'zeros'` `'ones'` `'middle'` `'ortho'` `'testTO'`	Určuje způsob inicializace jedinců v metodě `initPool`: `'random'` náhodně generovaný binární vektor, `'zeros'` nulový binární vektor, `'ones'` jednotkový binární vektor, `'middle'` binární vektor dle středu intervalu `iParam`, `'ortho'` binární vektory rovnoměrně rozděleny, `'testTO'` speciální nastavení pro test *TakeOver*.

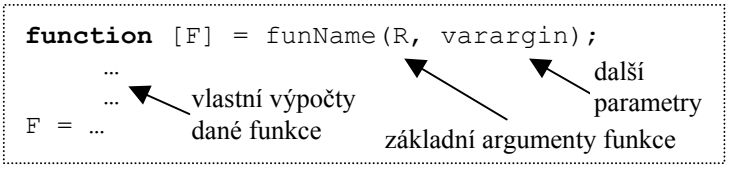
Tab. 1: Základní specifikace datové struktury GA (dsGA), povinná inicializační část.

Jiné proměnné, jako například počet bitů reprezentujících jeden reálný parametr, jsou pochopitelně při zachování jistých pravidel volbou uživatele. Závěrem této kapitoly uveďme, že další rozšířeni *dsGA* je volbou uživatele a daného účelu.

3. Účelová funkce

Úspěšné a efektivní sestavení účelové funkce pro danou optimalizační úlohu, je základem k dalšímu využití implementovaných heuristických metod. Koncepce a zásady tvorby objektivní M-funkce jsou obdobné jako pro tvorbu jakékoliv M-funkce (obdoba účelových funkcí Optimization toolboxu). Pokud pomineme matematickou stránku problému formulace účelové funkce, kterou nyní berme za apriorně splněnou, stojíme v podstatě před jedinou technickou otázkou, a to jak naprogramovat danou funkci, z hlediska rychlostního výkonu, dostatečně efektivně. V rámci prostředí Matlab je jednoznačným řešením a obecným doporučením vytvořit, pokud je to možné, tzv. *vektorizovaný kód*.

GATE metody předávají datovou strukturu populace `pool` jako binární matici ve formátu `[1:nIndi,1:nParam*nBitParam]` k vyhodnocení výkonné metodě `fitness`. Tato metoda volá uživatelem definovanou účelovou funkci, například `funName` a předává ji matici parametrů `R` optimalizační úlohy ve formátu `[1:nIndi,1:nParam]`. Účelová funkce provede výpočet funkčních hodnot v závislosti na `R` a vrací tyto hodnoty jako vektor `F` ve formátu `[1:nIndi,1]` metodě `fitness`, ta je uloží do pole `fit`.



Obr. 3: Obecný formát objektivní funkce (M-funkce).

V rámci tvorby GATE toolboxu bylo vytvořeno několik tzv. testovacích funkcí pro problémy funkční optimalizace. Tyto funkce mají z technického hlediska optimální vektorizovanou strukturu. Další popis i příklady tvorby vektorizovaného kódu je možné nalézt v příslušných referenčních příručkách produktu Matlab®.

```
% TEST FUNCTION: F6
% ------------------------
% (c) R.Matousek 1996,2004

function [fce] = f6(x);

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

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

$F_6(\mathbf{x}) = A \cdot n + \sum_{i=1}^{n} \left(x_i^2 - A \cdot \cos(2\pi x_i^2) \right)$, kde $A = 10$ a n je dimenze prostoru řešení $\mathbb{E}^n$

Optimalizační úloha: $\min \left\{ F_6(\mathbf{x}) \mid x_i \in [-5.12, 5.12], i \in \mathbb{N} \right\}$

Řešení $\min F_6(0, \ldots, 0) = 0$

Obr. 4: Příklad vektorizované účelové funkce (M-funkce) pro optimalizační problém F6.

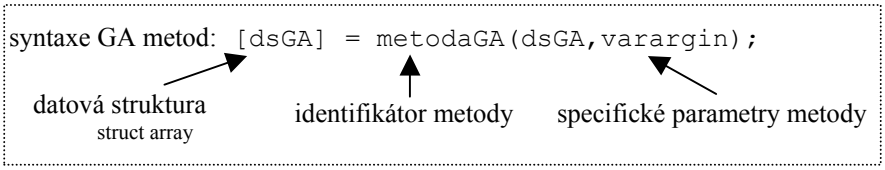
4. Výkonné metody

Pojem *výkonné metody* byl zaveden jako zastřešující termín pro skupinu metod reprezentujících základní i rozšířené GA operátory a metody HCA. Do této skupiny jsou v GATE toolboxu zařazeny metody:

• select	• initPool	• elite	• HCA5
• cross	• fitness	• migration	• HCA6
• mutation	• stopGA	• mutationHC	• HCA7

Tab. 2: Výkonné metody GATE toolboxu.

Základní koncepcí při užití GA metod, tedy volání příslušných M-funkcí, resp. P-funkcí, je užití *dsGA* a daných parametrů metody. Výkonná metoda nad touto předanou strukturou vykoná příslušné operace a modifikovanou strukturu opět vrátí.



Obr. 5: Obecná syntaxe GA metod GATE toolboxu.

Metoda:	**fitness**		
Popis:	Metoda na základě objektivní funkce (fitness) ohodnotí populaci GA.		
Syntaxe:	[dsGA] = fitness(dsGA,varargin)		
parmetr	hodnoty	popis	
varargin	*struct array*	Variantní proměnná Matlabu. Tento parametr je metodou využíván v případě předávání doplňkových informací nutných při volání uživatelem definované objektivní funkce.	
dsGA input♣		dsGA output♦	a priory a servisní metody♥
.pool .funOpt .nBitParam	.iParam .iType .mCode	.fit .param	.winFit .winParam .winPool	initPool	GC2BC BC2real GC2BCx BC2realx
Metoda:	**select**		
Popis:	Selekční operátor GA.		
Syntaxe:	[dsGA] = select(dsGA,mSelect,pSelect)		
parmetr	hodnoty	popis	
mSelect	'tournament'	Toutnament selekce (selekce založená na „souboji“ dvou a více jedinců, tedy potenciálních řešení).	
	'tournamentE'	ToutnamentE selekce (tournament selekce garantující „souboj“ každého jedince).	
	'truncation'	Truncation selekce (tzv. zkrácený výběr).	
pSelect	*double*	V případě *tournament* a *tournamentE selekce* specifikuje velikost tournamentu v rozsahu [2,3,…,nIndi]. Pro případ *truncation selekce* může být zadán i jako procentuální podíl (dělený stem) výběrové části populace v rozsahu [0, 1).	
dsGA input♣		dsGA output♦	a priory a servisní metody♥
.pool .fit .funOpt	.nIndi .nParam .nBitParam	.pool .fit	fitness
♣*Povinné vstupní parametry.*		♦*Realizované výstupní parametry a proměnné s „ukončenou“ aktuálností.*	♥*Servisní funkce užité metodou.*

Tab. 3: Příklad specifikace (syntaxe) výkonných metod GA.

Výkonné metody ze skupiny HC algoritmů jsou implementovány variantami označenými HCA5, HCA6 a HCA7. Posledně jmenovaná je využívána jako interní metoda GA, metody cílené mutace, tedy `mutationHC` (tzv. hybridní varianta GA-HC). Vstup HCAx funkce je datová struktura obdobná *dsGA* plus parametr `mView` určující formát zobrazení výsledků. Výstupem HCAx funkcí je pole, reprezentující záznam z průběhu realizované heuristické optimalizace. Záznam obsahuje informace o nejlepším řešení v rámci každého iteračního kroku, konkrétně: hodnotu účelové funkce, hodnotu reálných optimalizovaných parametrů a příslušný binární vektor. Příklad syntaxe HCA5 a HCA7 metod je uveden následující tabulkou.

Metoda:	**HCA5**	
Popis:	Realizuje HC algoritmus s využitím zvolené/zvolených binárních transformací.	
Syntaxe:	`[vecA] = HCA5(vecB|nBitParam,iParam,iType,mCode,…` `mHC,mView,funOpt,funName,varargin)`	
parametr	Hodnoty	
`mHC`	`'HC1'`	Generuje okolí se vzdáleností $\rho_H = 1$ od původního řešení.
	`'HC2'`	Generuje okolí se vzdáleností $\rho_H = 2$ od původního řešení.
	`'HC12'`	Generuje okolí se vzdálenostmi $\rho_H = 1$ a $\rho_H = 2$ od původního řešení.
	`'HC1n'`	Generuje okolí se vzdálenostmi $\rho_H = 1$ a $\rho_H = (n-1)$ od původního řešení.
Metoda:	**HCA7**	
Popis:	Realizuje HCA v rámci podřetězce bitového řetězce; základ metodyGAHC.	

Syntaxe:	`[vecS fit R B] = HCA7(vecB,kPos,kSize, ...`
	`nBitParam,iParam,iType,mCode, ...`
	`mHC,mView,funOpt,funName,varargin)`	

parmetr	hodnoty	
`mHC`	*dle HCA5*	dle *HCA5*
`kPos`	*1,2,...,n-s*	Souřadnice jádra v bitovém řetězci, hodnota *n* odpovídá délce řetězce.
`kSize`	*1,2,...,s*	Velikost HC jádra.

Formát `[vecA]=[fit, R, B]`; počet řádků = počet iterací		servisní metody	
`fit`	vektor funkčních hodnot nejlepších řešení v daných iteracích.	`BC2GC`	`BC2GCx`
`R`	matice dekódovaných hodnot nejlepších řešení v daných iteracích.	`GC2BC`	`GC2BCx`
`B`	matice binárních hodnot nejlepších řešení v daných iteracích.	`BC2real`	`BC2realx`

Tab. 4: Základní specifikace (syntaxe) výkonných metod HCA5 a HCA7.

5. Servisní metody

Pojmem *servisní metody* jsou v GATE toolboxu označovány funkce, které jsou buď nezbytné pro činnost tzv. *výkonných metod*, nebo mají další určení v rámci celku. Jejich součástí jsou i vytvořené testovací účelové funkce (např. F6). Servisní funkce byly navrženy tak, aby jejich použití, v rámci své domény, bylo universální a nebylo vázáno jen na GATE toolbox. Následující tabulka 5 stručně uvádí metody určené ke kódování resp. dekódování optimalizovaných parametrů.

K uvedené tabulce poznamenejme, že existují varianty popsaných metod, které jsou odlišeny jednoznakovým postfixem „`x`“. Tyto varianty jsou schopny dekódovat binární řetězce s rozdílnými délkami. Parametr `nBitParam` potom není skalár, ale vektor těchto délek. K dalším servisním metodám patří: `makeHD2mask`, `makeHDxRmask`, `makeCmask`, `testHD`, `testHD3`, `testED`, `swapbits`, `view2D`, `view3D`, `poolView` aj.

SKUPINA METOD PRO KONVERZI ČÍSEL	
Metoda:	**BC2GC**
Popis:	Převádí matici binárních vektorů z přímého binárního kódu do Grayova kódu.
Syntaxe:	`[GC] = BC2GC(BC,nBitParam)`
Metoda:	**GC2BC**
Popis:	Převádí matici binárních vektorů z Grayova kódu do přímého binárního kódu.
Syntaxe:	`[BC] = GC2BC(GC,nBitParam)`
Metoda:	**BC2real**
Popis:	Převádí matici binárních vektorů do matice reprezentovaných reálných hodnot.
Syntaxe:	`[R] = BC2real(BC,nBitParam,<iEP>,<iType>)`
Metoda:	**real2BC**
Popis:	Převádí matici reprezentovaných reálných hodnot na matici binárních vektorů.
Syntaxe:	`[GC] = real2BC(BC,nBitParam,<iEP>,<iType>)`
Pozn.:	*Zápis syntaxe s užitými znaky „<>" odpovídá nepovinným parametrům.* *V rámci výkonných GA i HCA metod jsou však tyto požadovány.*

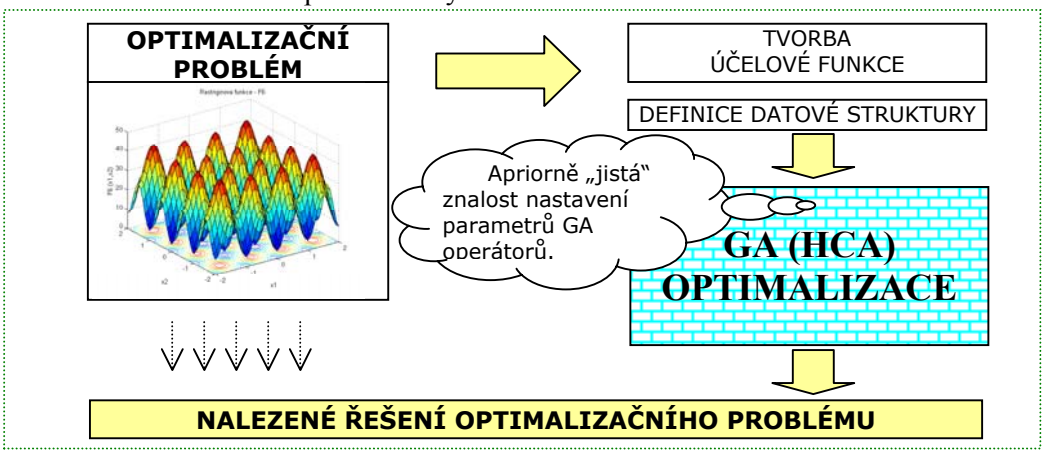
parametr	popis
`BC`	Matice binárních vektorů přímého binárního kódu. Tyto vektory reprezentují, v případě GA nebo HC algoritmu, optimalizační problém.
`GC`	Matice binárních vektorů Grayova binárního kódu. Tyto vektory reprezentují, v případě GA nebo HC algoritmu, optimalizační problém.
`real`	Matice reálných hodnot. Tyto hodnoty reprezentují, v případě GA nebo HC algoritmu, dekódované reálné parametry optimalizačního problému.
`nBitParam`	Parametr udávající počet bitů, které jsou užity ke kódování reálných parametrů.
`iEP`	Nepovinný parametr udávající mezní hodnoty definičního intervalu daného parametru. Formát: [r s], r…dolní mez, s…horní mez.
`iType`	Nepovinný parametr udávající typ definičního intervalu, přípustné jsou volby: `'[-]'` odpovídá uzavřenému intervalu [r s] `'[0)'` odpovídá polootevřenému intervalu [r s)

Tab. 5: Servisní metody určené ke kódování, resp. dekódování optimalizovaných parametrů.

6. Aplikace GATE toolboxu

Znalosti nutné k úspěšné aplikaci GATE toolboxu:

- Základní znalost Matlabu, a schopnost tvorby M-funkcí.
- Základní znalost užitých metod GA a HCA.
- Znalost řešené problematiky.



Obr. 6: Princip funkční optimalizace pomocí GATE toolboxu.

V dalším bude demonstrována koncepce návrhu základní varianty GA a některých rozšíření (GA-HC, „elitářská“ strategie), vč. samostatného HC algoritmu. Koncepce bude schematizována základními funkčními fragmenty M-kódu. Princip jednotlivých kroků výpočtu bude vycházet z postupu dle obrázku (Obr. 6). Pro ukázková řešení bude zvolena optimalizace (minimalizace) již popsané (Obr. 4) testovací funkce F6.

```
function [GA]=demoGA

       funName: 'F6'
        funOpt: 'min'

        iParam: [-5 5]
         iType: '[0)'

        nParam: 5
     nBitParam: 10
         nIndi: 50
         mCode: 'GC'
         mInit: 'random'
```

```
GA = initPool(GA);
GA = fitness(GA);

while GA.nGener < 200
       GA.nGener=GA.nGener+1;
       GA = select(GA,'tournamentE',4);
       GA = cross(GA,'pcross',[0.8 3]);
       GA = mutation(GA,'bitmut',0.02);
       GA = fitness(GA);
end
```

omezení max. počtu generací

Programová realizace GA optimalizace pro řešení optimalizačního problému F6, s využitím toolboxu GATE. Postup odpovídá schématu z obrázku (Obr. 6). Účelová funkce definující optimalizační úlohu odpovídá návrhu dle obrázku (Obr. 4).

Obr. 7: Princip konstrukce GA pomocí metod toolboxu GATE.

```
function [GA]=demoGA

       funName: 'F6'
        funOpt: 'min'

        iParam: [-5 5]
         iType: '[0)'

        nParam: 5
     nBitParam: 10
         nIndi: 50
         mCode: 'GC'
         mInit: 'random'
```

```
GA = initPool(GA);
GA = fitness(GA);
GA = elite(GA,'save',3)

while GA.nGener < 200
       GA.nGener=GA.nGener+1;
       GA = select(GA,'tournamentE',4);
       GA = cross(GA,'pcross',[0.8 3]);
       GA = mutation(GA,'bitmut',0.02);
       GA = fitness(GA);

       GA = elite(GA,'clone',3,'refine');
       GA = elite(GA,'save',3);
end
```

uschování elity (3 nejlepší řešení)

návrat elity do populace

Programová realizace GA optimalizace s aplikováním elitismu pro řešení optimalizačního problému F6, s využitím toolboxu GATE. Postup odpovídá schématu z obrázku (Obr. 6). Objektivní funkce definující optimalizační úlohu odpovídá návrhu dle obrázku (Obr. 4).

Obr. 8: Princip konstrukce GA aplikující elitářskou strategii pomocí metod toolboxu GATE.

```
>> [result]=HCA5(2,12,[-5 5],'[0)','GC','HC12','real','min','F6');
# 1: F6_min=  48.1199  x1: 0.5664 x2: 3.6621
# 2: F6_min=  17.5172  x1: 0.0562 x2: 1.3354
# 3: F6_min=   1.7693  x1: 0.0562 x2: 1.0229
# 4: F6_min=   1.0941  x1: 0.0195 x2: 1.0059
# 5: F6_min=   0.9886  x1: 0.0195 x2: 0.0684
# 6: F6_min=   0.0685  x1: 0.0171 x2: 0.0073
# 7: F6_min=   0.0000  x1: 0.0000 x2: 0.0000
```

Průběh optimalizace, kdy po sedmi iteracích byl nalezen extrém. Výsledek včetně iteračních kroků je uložen v proměnné `result`.

Programová realizace HCA optimalizace s využitím metody HCA5. Objektivní funkce definující optimalizační úlohu odpovídá návrhu dle obrázku (Obr. 4).

Obr. 9: Princip aplikace HCA pomocí metody toolboxu GATE.

Následující obrázek (Obr. 10) ukazuje snadnou aplikaci hybridního GA-HC algoritmu, přičemž interně užitá metoda metody `mutationHC` je HCA7. Princip HCA7 je demonstrován obrázkem (Obr. 11).

```
function [GA]=demoGA

       funName: 'F6'
        funOpt: 'min'

        iParam: [-5 5]
         iType: '[0)'

        nParam: 5
     nBitParam: 10
         nIndi: 50
         mCode: 'GC'
         mInit: 'random'
```

```
GA = initPool(GA);
GA = fitness(GA);

while GA.nGener < 200
      GA.nGener=GA.nGener+1;
      GA = select(GA,'tournamentE',4);
      GA = cross(GA,'pcross',[0.8 3]);
      GA = mutation(GA,'bitmut',0.02);


      GA = mutationHC(GA,'HC12',5,'rand',6);
      GA = fitness(GA);
end
```

metoda HCA-HC12

Programová realizace GA optimalizace s aplikováním HC mutace pro řešení optimalizačního problému F6, s využitím toolboxu GATE. Postup odpovídá schématu z obrázku (Obr. 6). Objektivní funkce definující optimalizační úlohu odpovídá návrhu dle obrázku (Obr. 4).

Obr. 10: Princip konstrukce GAHC pomocí metod toolboxu GATE.

```
% vecS … optimalizovane HC jadro
% fit … hodnota objektivnifunkce, R … vektor realnych parametru

>> [vecS fit R vecA]= ...
HCA7([0 1 1 1 1 1 1 0],3,4,1,[0 1],'[-]','BC','HC1','binary','min','Fbit1');
```

HC jádro | souřadnice polohy | šířka jádra | interní testovací funkce

```
# 1: 0 1 1 1 1 1 1 0 Fbit1_min=   6.0000
# 2: 0 1 0 1 1 1 1 0 Fbit1_min=   5.0000
# 3: 0 1 0 0 1 1 1 0 Fbit1_min=   4.0000
# 4: 0 1 0 0 0 1 1 0 Fbit1_min=   3.0000
# 5: 0 1 0 0 0 0 1 0 Fbit1_min=   2.0000
```

výpis optimalizačního procesu metody HCA7

HC jádra jednotlivých iteračních kroků

```
>> vecS

vecS =

     0     1     0     0     0     0     1     0


>> HCA7([0 0 0 0 0 0 0 0 0 0],1,5,1,[0 1],'[-]', ...
                                    'GC','HC12','binary','max','Fbit1');

# 1: 0 0 0 0 0 0 0 0 0 0 Fbit1_max=   0.0000
# 2: 1 1 0 0 0 0 0 0 0 0 Fbit1_max=   2.0000
# 3: 1 1 1 1 0 0 0 0 0 0 Fbit1_max=   4.0000
# 4: 1 1 1 1 1 0 0 0 0 0 Fbit1_max=   5.0000
```

výsledný vektor po provedení optimalizace HC jádra

Obr. 11: Ukázka optimalizace části bitového řetězce metodou HCA7.

V každém z algoritmu HCA5,HCA6 a HCA7 je „zabudována“ interní testovací funkce nazvaná `Fbit1`.

Tato účelová funkce může sloužit jak k demonstraci funkce algoritmu, tak k ověření časové relace algoritmu se specifickým nastavením parametrů a vzhledem k použitému hardware.

$$F_{bit1}(\mathbf{a}) = \sum_{i=1}^{n}(a_i) \qquad \text{Optimalizační úloha:} \quad \operatorname{opt}\left\{F_{bit1}(\mathbf{a}) \mid a_i \in \{0,1\}, i \in \mathbb{N}\right\}$$

$$\text{Řešení:} \quad \min F_{bit1}(0_1,\ldots,0_n) = 0$$

$$\max F_{bit1}(0_1,\ldots,0_n) = n$$

Obr. 12: Interní testovací funkce F_{bit1} realizovaných HC algoritmů a příklad optimalizační úlohy.

7. Závěr a aplikační poznámky

Před vlastní aplikací realizovaných metod je třeba si připomenout jistá fakta:

- Koncepce GATE toolboxu je založena na práci s binární reprezentací. Tato reprezentace nutně znamená diskretizaci prostoru řešení. Při návrhu algoritmu, respektive binárních reprezentantů reálných řešení, je kromě jiného třeba zvážit přesnost řešení a kombinatorický rozsah daného problému.
- U GA běžně používaná míra optimality označovaná jako fitness, je pro prezentované implementace algoritmů nahrazena přímo účelovou funkcí. Toto řešení je z hlediska „manipulace“ s optimalizační úlohou mnohem výhodnější, přičemž umožňuje přímou spolupráci metod HCA i případných dalších.
- Tzv. reálná čísla realizovaná na současných typech počítačů ve skutečnosti reálná nejsou, neb jsou reprezentovaná binárními řetězci konečné délky.
- Jakékoliv, i již vyřčené pojmy týkající se realizace „náhody“, jsou v počítačové aplikaci vždy nahrazeny tzv. „pseudo-náhodou“. Ta je realizována pomocí generátoru náhodných čísel. V případě GATE toolboxu je využíván generátor náhodných čísel (Matlab) s rovnoměrným rozdělením a teoretickou periodou opakování 2^1492.

Koncepce GATE toolboxu:

- ✓ Optimalizační nástroj pracující s binární, resp. dekódovanou celočíselnou nebo reálnou reprezentací problému.
- ✓ Jednoduchost aplikace a možnost využití mnoha dalších funkcí Matlabu (například využití optimalizačního toolboxu).
- ✓ Modularita návrhu s prostorem pro vlastní koncepci či modifikaci.
- ✓ Zahrnutí vyspělých metod GA a metod HC algoritmů s možností dalšího rozšíření.

Tento příspěvek představil sadu funkcí, označenou jako GATE toolbox, určenou pro tzv. heuristickou optimalizaci pomocí GA a HCA metod. Příspěvek nemohl a nechtěl substituovat „referenční“ příručku. Uvedení GATE verze 2.0 je datováno [1] před uvedením profesionálního Genetic Algorithm and Direct Search Tooolboxu firmy The Mathworks. Jakékoliv výkonnostní srovnání s touto knihovnou nebylo provedeno. Vhodným atributem pro minimálně „odzkoušení“ GATE toolboxu je fakt, že jej autor pro akademické užití poskytuje zdarma prostřednictvím url odkazu [3].

Reference

[1] Matoušek, R.: *Vybrané metody umělé inteligence – implementace a aplikace*. Disertační práce, FSI VUT v Brně, 2004

[2] Kvasnička, V., Pospíchal, J., Tiňo, P.: *Evolučné algoritmy*. STU, Bratislava, 2000, ISBN 80-227-1377-5.

[3] URL: http://www.uai.fme.vutbr.cz/~matousek, odkaz [GATE]

Radomil Matoušek
Ústav automatizace a informatiky, FSI VUT v Brně, Technická 2, 616 69 Brno, CZ
matousek@fme.vutbr.cz, tel. +420 4114 2298