BINÁRNÍ KÓDOVÁNÍ A HC ALGORITMUS

Radomil Matoušek
Ústav automatizace a informatiky, FSI VUT Brno

Abstrakt

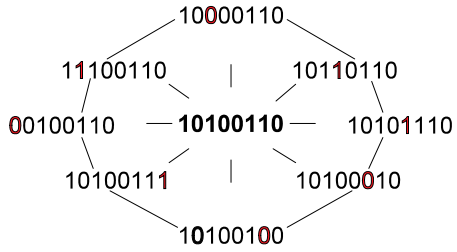
Anglický ekvivalent názvu *horolezecký algoritmus* je *hill climbing*, dále tedy *HC algoritmus*, nebo zkráceně *HCA*. Název v podstatě charakterizuje přístup metody k řešení optimalizačního problémů, tj. postup horolezce, který pokud se má čeho chytit, může „jít dál". V případě matematicky pojatého horolezeckého algoritmu se tedy jedná o postup vhodným směrem, kterým je samozřejmě „rádoby" extrém úlohy. Vhodnost směru postupu je určena na základě specifického prohledávání okolí aktuálního řešení. Volbě tohoto okolí a metodě kódování HCA, který byl implementován v prostředí Matlab, je věnován tento příspěvek.

1. Binárně reprezentovaný HC algoritmus

V kontextu optimalizačních metod můžeme HC algoritmus zařadit k metodám, které směr nejvhodnějšího postupu určují na základě prohledání svého okolí a určení nejstrmějšího spádu (*steepest descent*) či růstu. Z tohoto faktu vyplývá, že k výpočtu není potřeba gradient, ale pouze apriorní znalost hodnot účelové funkce. Prezentovaný HCA je založen na binární reprezentaci parametrů a jeho realizace odpovídá následujícímu postupu:

- Pro aktuální řešení (tj. i první aproximace) se v metrickém Hammingově prostoru $\mathbb{H}$ generuje pomocí konečného souboru transformací určité okolí.
- Poté se účelová funkce extremalizuje na tomto okolí, přičemž může být zahrnuto i původní (zdrojové) řešení.
- Získané řešení se použije jako základ pro generování nového okolí.
- Ukončovací kritérium může být voleno například na základě max. počtu iterací [3] nebo na základě neschopnosti algoritmu generovat lepší řešení [1].

Základním krokem algoritmu je vygenerovat okolí původního řešení na základě zvolené metodiky. Příkladem takovéhoto kroku je realizované okolí 8-mi bitového řetězce na následujícím obrázku.



Bitový řetězec uprostřed slouží jako základ uvažované transformace, tedy tzv. jádro, či střed. Okolí je vytvořeno vždy negací jednoho bitu. Bitové řetězce se ohodnotí, a nejvhodnější z nich postupuje do další iterace jako nové jádro.

Obr. 1: Generování okolí 8-mi bitového řetězce pomocí jednobitové inverze.

Na „kvalitě“ tohoto okolí pak bude bezprostředné záviset lokálnost či globálnost uvedeného heuristického postupu. Pro další popis metody uvažujme standardní problém optimalizace funkce $f(\mathbf{x})$ na diskrétní oblasti D. Equation Chapter (Next) Section 1

$$\min\left\{f(x)\,|\,x\in D\right\}\text{, resp. }\max\left\{f(x)\,|\,x\in D\right\} \tag{1}$$

Tedy hledání takového $\tilde{x}$, aby platilo:

$$\tilde{\mathbf{x}}=\arg\min_{\mathbf{x}\in D} f(\mathbf{x})\text{, resp. }\tilde{\mathbf{x}}=\arg\max_{\mathbf{x}\in D} f(\mathbf{x}). \tag{2}$$

Diskretizace oblasti $D\in\Re$ je dána relací $\Gamma:\{0,1\}^l\to D$, tedy zavedenou binární reprezentací **a** reálných proměnných **x**. Protože platí $\mathbf{x}=\Gamma(\mathbf{a})$, považujeme dále za optimální řešení úlohy (1) následující vztah (3).

$$\mathbf{a}_{opt}=\arg\min_{\mathbf{a}\in\{0,1\}^l} f\left(\Gamma(\mathbf{a})\right) \tag{3}$$

Nad touto binární reprezentací definujeme určitou *relaci sousednosti*, která pro každé přípustné $\mathbf{a}_{\text{jádro}}$ umožňuje stanovit „*okolí*“ $\mathbf{a}_{\text{okolí}}\in S(\mathbf{a}_{\text{jádro}})$. Volba transformační funkce S, bude determinovat chování a charakter HC algoritmu. Pokud bude mít stochastický charakter, můžeme mluvit o stochasticko-heuristickém HC algoritmu, v opačném případě o heuristickém HC algoritmu.

Realizovaná Matlab implementace HC algoritmu má následující specifika:

- První aproximace řešení je algoritmem volena náhodně.
- Následující aproximace jsou získány pomocí specifické (specifických) bitových transformací. Tyto transformace již mají deterministickou povahu.
- Není připuštěno zhoršení hodnot účelové funkce a iterační cyklus algoritmu je zastaven pokud není danou transformací nalezeno lepší řešení.

```
% binární HC algoritmus (minimalizace)

a_opt = náhodně vygenerovaný či zvolený binární vektor

repeat
    a_jádro = a_opt
    ∀a_okolí = S(a_jádro)                 % předem zvolená transformační funkce
    a_opt = arg min_{a∈S(a_jádro)} f(Γ(a))   % výběr nejlepšího řešení
until f(Γ(a_opt)) ≥ f(Γ(a_jádro))         % test ukončení
```

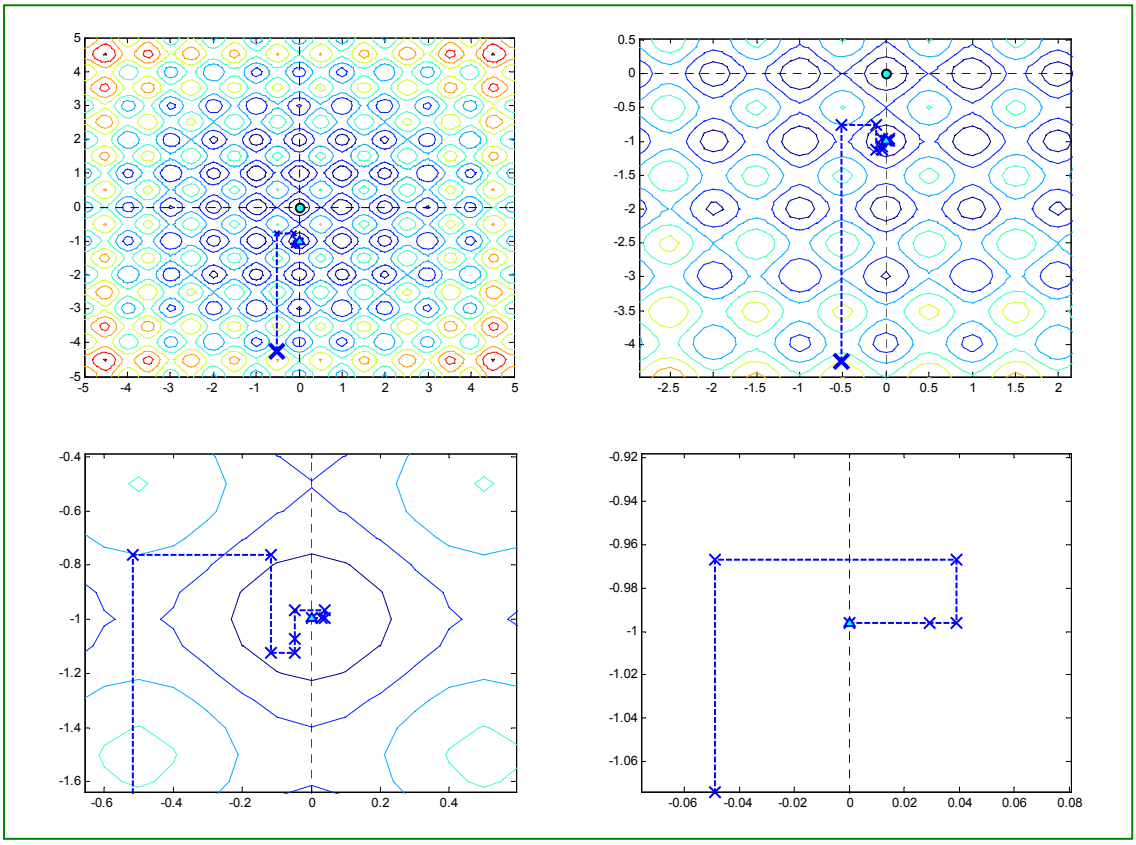
Obr. 2: Pseudo-kód: HC algoritmus s testem kvality řešení.

Realizovaný HC algoritmus je tedy stochastický pouze volbou své první iterace a další průběh je pevně určen zvolenou transformací. Iterační výpočet je ukončen v okamžiku, kdy nelze pomocí užité transformace dosáhnout lepšího řešení.

```
# 1: 0111001011 0111001011 F6_min=  48.1301  x1:-0.5176 x2:-4.2480
# 2: 0111001011 0111001011 F6_min=  30.0515  x1:-0.5176 x2:-0.7617
# 3: 0111110100 0111110100 F6_min=  12.4488  x1:-0.1172 x2:-0.7617
# 4: 0111110100 0111110100 F6_min=   6.7081  x1:-0.1172 x2:-1.1230
# 5: 0111111011 0111111011 F6_min=   4.5732  x1:-0.0488 x2:-1.1230
# 6: 0111111011 0111111011 F6_min=   2.6910  x1:-0.0488 x2:-1.0742
# 7: 0111111011 0111111011 F6_min=   1.6208  x1:-0.0488 x2:-0.9668
# 8: 1000000100 1000000100 F6_min=   1.4527  x1: 0.0391 x2:-0.9668
# 9: 1000000100 1000000100 F6_min=   1.2964  x1: 0.0391 x2:-0.9961
#10: 1000000011 1000000011 F6_min=   1.1650  x1: 0.0293 x2:-0.9961
#11: 1000000000 1000000000 F6_min=   0.9952  x1: 0.0000 x2:-0.9961
>>
```

Obr. 3: Funkční optimalizace algoritmem HCA-HC1 - úloha F6, výpis z průběhu výpočtu.

Ukázka realizovaného výpočtu funkční minimalizace je demonstrována výpisem (Obr. 3). Použitá metoda transformace *S* odpovídá naznačenému postupu dle (Obr. 1). Čísla s prefixem „#“ označují číslo iterace výpočtu, binární hodnoty odpovídají příslušným $\mathbf{a}_{opt}$ vektorům, dále následuje funkční hodnota jako kritérium optimality a příslušné dekódované reálné parametry, ze kterých je tato hodnota vypočtena. Pro rychlejší a názornější popis úlohy označené jako F6 je příklad doplněn grafickým znázorněním průběhu optimalizace (Obr. 4.). V příkladu je užito dekódování pomocí Grayova binárního kódu. Zvolená diskretizace prostoru odpovídající 10 bitové reprezentaci reálných parametrů odpovídá hodnotě možné chyby $\varepsilon = 9.8\text{E-}03$.



Obr. 4: Funkční optimalizace algoritmem HCA-HC1 - úloha F6, zobrazení průběhu výpočtu.

Jakékoliv, v literatuře prezentované varianty HC algoritmu, jsou v podstatě založeny na odlišné metodě transformace *S*, tedy způsobu definice relace sousednosti. K významným variantám můžeme zařadit:

- **HC algoritmus s mutací**
  Tento algoritmus využívá k tvorbě okolí, tedy vektorů $\mathbf{a}_{\text{okolí}}$, stochastický *operátor mutace* popsaný v předchozím odstavci. Pomocí tohoto operátoru transformuje binární vektor $\mathbf{a}_{\text{jádro}}$ na množinu vektorů $\mathbf{a}_{\text{okolí}}$. Kardinalita této množiny je z oboru $\mathbb{N}$ a je předem daná.
- **HC algoritmus s učením** [5]
  Tato modifikace HC algoritmu se snaží pomocí specifického mechanismu navrhnout nové okolí vektoru $\mathbf{a}_{\text{jádro}}$ tak, aby pravděpodobnost vzniku vektoru $\mathbf{a}_{\text{okolí}}$ odpovídala jistým pravidlům. Mechanismus využívá tzv. pravděpodobnostní vektor, který determinuje pravděpodobnosti stavů jednotlivých bitů generovaných vektorů okolí $\mathbf{a}_{\text{okolí}}$. Vhodné „nastavení" tohoto pravděpodobnostního vektoru je realizováno pomocí Hebbova pravidla učení.
- **Tabu search (zakázané prohledávání)** [6]
  Glover navrhl modifikaci HC algoritmu, která využívá množiny přípustných transformací a tzv. *krátkodobou paměť*. Pomocí této paměti jsou zaznamenány po jistý čas inverzní transformace transformací, které vedly k nalezení nejlepšího lokálního řešení. Tyto inverzní transformace jsou při generování nového okolí vektoru $\mathbf{a}_{\text{jádro}}$ zakázány, tedy *tabu*. Rozšíření metody zakázaného prohledávání využívá dalšího mechanizmu, který je označen jak tzv. *dlouhodobá paměť*. Ta v podstatě slouží k penalizaci těch transformací, které se v průběhu optimalizace využívaly příliš často.

2. Transformační funkce

Navržené a realizované HC algoritmy schématicky popsané dle (Obr. 2) jsou založeny na libovolné, ale pevně dané množině transformací příslušných binárních vektorů [2].

$$\mathbf{a} \in \{0,1\}^n, \quad \text{pro } n \in \mathbb{N} \quad (n\text{...délka binárního vektoru}) \tag{4}$$

Množinu navržených transformací označme *H* a jednotlivé transformace jako *t*.

$$H = \{t_0, t_1, ..., t_n\} \tag{5}$$

Užitím transformace $t \in H$ realizujeme zobrazení binárního vektoru $\mathbf{a}_{jádro}$ na množinu $A_{\text{okolí}}$ binárních vektorů $\mathbf{a}_{\text{okolí}}$, dále též označovanou jako matici $\mathbf{A}_{okolí}$.

$$A_{okolí} = \{\mathbf{a}_1, \ldots, \mathbf{a}_c\}, \quad \mathbf{A}_{okolí} = \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_c \end{pmatrix}, \quad \text{pro } c \in \mathbb{N} \tag{6}$$

$$t : \mathbf{a}_{jádro} \to A_{okolí}, \quad \text{tedy } t : \{0,1\}^n \to \left(\{0,1\}^n\right)^c \tag{7}$$

Kardinalita c množiny $A_{okolí}$ je dána zvolenou transformací t a délkou n binárního vektoru $\mathbf{a}_{\text{jádro}}$.

$$c_k\left(t_k, n\right) = \left|A_{okolí}\right| = \binom{n}{k}, \quad \text{pro } k \in \{0,1,\ldots,n\}, \tag{8}$$

kde index k označuje příslušnost ke konkrétnímu prvku z množiny H dle (5).

K realizaci množiny transformací H je zaveden systém matic **M**. O **M** matici příslušející dané transformaci t_k budeme hovořit jako o **M** matici k-tého řádu a označíme ji $\mathbf{M}_k$, tento řád není totožný s řádem čtvercové matice.

$$t_k, n \Leftrightarrow \mathbf{M}_k, \quad \text{pro } k \in \{0,1,\ldots,n\} \tag{9}$$

Definice: M matice řádu k , zkráceně $\mathbf{M}_k$, je taková matice, jejíž řádky reprezentují všechny body Hammingova metrického prostoru $\mathbb{H}^n$, se vzdálenostmi k od počátku (tj. nulového vektoru délky n) ve smyslu Hammingovi metriky ρ_H.

□

Schéma možné konstrukce **M** matic je následující:

$$\mathbf{M}_0 = \begin{pmatrix} 0_{1,1} & 0_{1,2} & \cdots & 0_{1,n} \end{pmatrix}$$

$$\mathbf{M}_1 = \begin{pmatrix} 1_{1,1} & 0_{1,2} & \cdots & 0_{1,n} \\ 0_{2,1} & 1_{2,2} & & \\ \vdots & & \ddots & \\ 0_{c_1,1} & & & 1_{c_1,n} \end{pmatrix}$$

$$\mathbf{M}_2 = \begin{pmatrix} 1_{1,1} & 1_{1,2} & 0_{1,3} & \cdots & 0_{1,n} \\ 1_{2,1} & 0_{2,2} & 1_{2,3} & & 0_{2,n} \\ \vdots & & & & \\ & & & \ddots & \\ 0_{c_2,1} & & & 1_{c_2,n-1} & 1_{c_2,n} \end{pmatrix}$$

$$\vdots$$

$$\mathbf{M}_{n-1} = \begin{pmatrix} 0_{1,1} & 1_{1,2} & \cdots & 1_{1,n} \\ 1_{2,1} & 0_{2,2} & & \\ \vdots & & \ddots & \\ 1_{c_{n-1},1} & & & 0_{c_{n-1},n} \end{pmatrix} \quad . \tag{10}$$

$$\mathbf{M}_n = \begin{pmatrix} 1_{1,1} & 1_{1,2} & \cdots & 1_{1,n} \end{pmatrix}$$

Pro výpočet matice $\mathbf{A}_{okolí}$ (6) připomeňme, že realizované operace jsou provedeny v lineárním prostoru nad specifickým tělesem $Z_2 = \{0,1\}$.

K výpočtu matice $\mathbf{A}_{okolí}$ je dále třeba zavést operaci realizující tzv. *replikaci vektoru* $\mathbf{a}_{jádro}$. Tato operace vytvoří matici $\mathbf{A}_{jádro}$ obsahující po řádcích identické kopie binárního vektoru $\mathbf{a}_{jádro}$. Počet řádků této matice odpovídá počtu řádků příslušné **M** matice a tedy kardinalitě c cílové množiny $A_{okolí}$ dle (8).

$$\mathbf{A}_{jádro} = \begin{pmatrix} \mathbf{a}_{1,jádro} \\ \vdots \\ \mathbf{a}_{c,jádro} \end{pmatrix} \tag{11}$$

Nyní pomocí (9), respektive (10) může být realizována příslušná transformace t (7).

$$t_k : \ \mathbf{A}_{okolí} = \mathbf{A}_{jádro} \oplus \mathbf{M}_k, \quad \text{pro } t_k \in H \text{ a } k = 0,1,...,n \tag{12}$$

O transformaci t_k lze říci, že generuje úplnou množinu vektorů, které jsou ve smyslu metriky ρ_{H} ($\rho_H(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$) vzdáleny od počátku o hodnotu k.

$$t_k \Rightarrow \rho_H(\mathbf{a}_{jádro}, \mathbf{a}_{okolí}) = k, \quad \text{pro } \forall \mathbf{a}_{okolí} \in A_{okolí} \tag{13}$$

Zobecnění vztahu (12) pro libovolný, ale pevně daný výběr prvků z množiny H, je zřejmé. Množinu vybraných, a dále pro HC algoritmus (Obr. 2) pevně daných transformací označíme H_v.

$$H_v \subseteq H \tag{14}$$

Množina H_v vzájemně jednoznačně určuje transformaci S, která je sjednocením transformací z této množiny.

$$H_v \Leftrightarrow S, \quad \text{a tedy} \quad S : \mathbf{A}_{OKOLÍ} = \begin{pmatrix} \mathbf{A}_{jádro,k_1} \oplus \mathbf{M}_{k_1} \\ \vdots \\ \mathbf{A}_{jádro,k_v} \oplus \mathbf{M}_{k_v} \end{pmatrix}, \tag{15}$$

kde $k_i \in I$ a I je indexová množina vybraných prvků z množiny H definovaná výběrem H_v.

3. Realizované transformace a závěr

Předchozí kapitola uvedla navrhovanou třídu binárních transformací H (5) a demonstrovala matematický aparát pro jejich realizaci (12), respektive (15). Vzhledem ke kombinatorické náročnosti byly pro praktickou realizaci vybrány transformace t_0, t_1 a t_2, respektive zbývající, ovšem modifikované transformace z množiny H.

K další práci bylo pro výběr transformací H_v, respektive transformaci S, zavedeno následující zjednodušující označení:

- transformace označené HC1,HC2 a HC12 jsou z množiny $H_v = \{t_0, t_1, t_2\}$,
- transformace označené HCxR jsou výběrem H_v z H, přičemž kardinalita příslušných množin $A_{okolí,k_i}$ je redukovatelná.

Pro aplikaci HC algoritmů musí být zvážena povaha úlohy. Implementované a teoreticky popsané [2] varianty samozřejmě disponují možností volby způsobu binárního dekódování i rozsahem binární reprezentace. Diskutované algoritmy byly realizovány v prostředí Matlab a jsou pro odzkoušení volně k dispozici po dotazu na e-mail autora tohoto článku.

Reference

[1] Matoušek, R., *Hill Climbing and 0/1 Knapsack Problem*, 7th Internation Conference MENDEL 2001, Brno, Czech Republic, 2001, ISBN 80-214-2135-5.

[2] Matoušek, R., *Vybrané metody umělé inteligence – implementace a aplikace*, Ph.D. práce v oboru Technická kybernetika, VUT Brno, Brno, Czech Republic, 2004.

[3] Kvasnička, V., Beňušková, Ľ., Pospíchal, J., Farkaš, I, Tiňo, P., Kráľ, A.: *Úvod do teórie neuronových sietí*. Vydavateľstvo IRIS, Bratislava, 1997, ISBN 80-88778-30-1.

[4] Kvasnička, J., Pospíchal, J.: *Evoluční algoritmy, Tabu search*, Computer word 95, Praha, 1995.

[5] Kvasnička, V., Pospíchal, J., Pelikán, M.: *Hill Climbing with Learning*. Proceedings of MENDEL'95, 1st International Conference on Genetic Algorithms, Brno, 1995, pp. 65-70.

[6] Glover, F.: *Tabu Search – Part I.* ORSA Journal of Operations Research 1., 1998.

Radomil Matoušek, matousek@fme.vutbr.cz
ODBORNÝ ASISTENT ODBORU INFORMATIKY
ÚSTAV AUTOMATIZACE A INFORMATIKY, FSI VUT V BRNĚ
TECHNICKÁ 2896/2, 616 69 BRNO