Genetické algoritmy

Bc. Jiří Hrazdil
xhrazd06

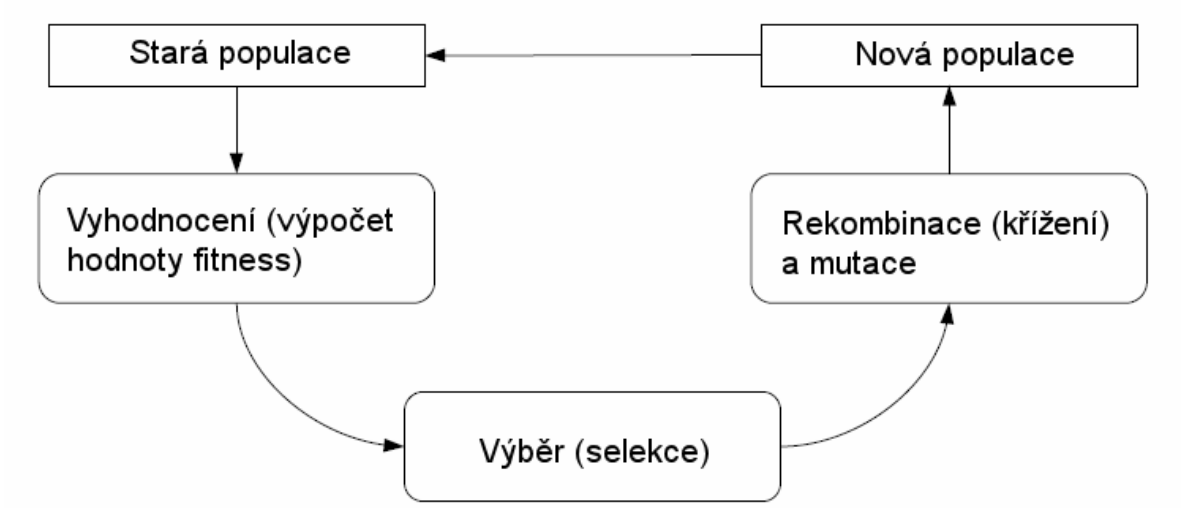
2008-04-13

1. Všeobecně

Mnoho optimalizačních problémů, objevujících se v oblasti plánování a směrování je kombinatorické povahy, tzn. řešení se skládá kombinováním a uspořádáním částí řešení. Při hledání optimálního řešení dojdeme k tomu, že počet řešení, které je třeba vyhodnotit, roste exponenciálně s velikostí problému. Například pro n prvků řešení existuje $n!$ různých posloupností.

V nedávné době došlo ke zvýšení obliby genetických algoritmů (GA) při řešení těchto optimalizačních problémů s využitím heuristik. Genetické algoritmy jsou jednoduché na implementaci. Navíc bylo pozorováno, že dokonce i základní, jednoduché verze genetických algoritmů jsou schopny dávat přijatelné výsledky i bez nutnosti náročného vyladění pro konkrétní problém. Jelikož GA pracují nad reprezentací (zakódovanou verzí) problému, je možné jednou naprogramovanou implementaci použít pro řešení mnoha problémů. První technický popis genetických algoritmů vydal Holland [4]. Obsáhlý popis, je uveden v publikacích Goldberg [3], Reeves [6] a Michalewicz [5].

GA pracují s tzv. populacemi skládajícími se z jedinců, kteří reprezentují možné řešení problému. Iterativním vytvářením nových populací vznikají jednotlivé generace. Na základě Darwinovy teorie se každý jedinec podílí na vlastnostech budoucí generace v závislosti na své kvalitě (v GA vyjádřené pomocí hodnoty fitness). Uvedené funkčnosti docílíme tak, že jedince vybereme náhodně s pravděpodobností odpovídající fitness hodnotě. Z vybraných jedinců vytvoříme novou generací pomocí dvou základních operací. První je křížení, kdy se vlastnosti dvou jedinců (rodičů) smíchají za vzniku jednoho nebo více potomků. Druhou operací je mutace, při které dojde k náhodné změně vlastností jedince. Schéma jedné iterace genetického algoritmu je uvedeno na následujícím obrázku.



Obvykle se genetický algoritmus provádí do té doby, než je splněna předem daná podmínka, např. maximální počet generací nebo překročení časového limitu.

Dále si pomocí příkladu přiblížíme jednotlivé aspekty genetických algoritmů. Problémem je naplánování výroby, skládající se z jednotlivých úloh. Mějme n úloh, které mají být provedeny na jednom stroji. Každá úloha $j = 1, .., n$ má pevně stanovenu dobu zpracování d_j, přičemž proces zpracování úlohy není možno přerušit. Navíc nesmí být úloha j spuštěna dříve, než stanovuje parametr rd_j (release date) a měla by být dokončena dřív,dojde k dosažení parametru dd_j (due date). V případě pozdního dokončení je za každou jednotku zpoždění naúčtována pokuta c_j. Problém tedy spočívá v nalezení posloupnosti provádění úloh tak, aby byla minimalizována pokuta za zpoždění. V následující tabulce jsou uvedeny hodnoty veškerých parametrů.

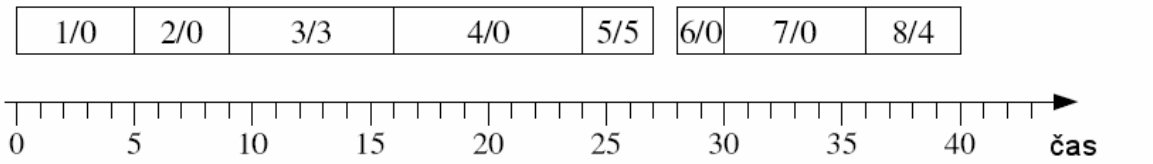
j	1	2	3	4	5	6	7	8
d_j	5	4	7	8	3	2	6	4
rd_j	0	2	4	16	18	28	25	28
dd_j	17	10	13	28	22	31	36	36
c_j	3	2	5	3	4	1	3	4

2. Populace a jedinci

Jak již bylo uvedeno dříve, populace se skládá z jedinců. Každý jedinec je uložen v podobě řetězce hodnot pevné délky, který je tvořen jednotlivými hodnotami kódujícími části příslušného řešení. Proces zjištění řešení z řetězce hodnot se nazývá *dekódování*. Jak délka řetězce, tak rozsah hodnot jednotlivých polí řetězce závisí na reprezentaci zvoleného problému.

Řešením našeho příkladu je posloupnost úloh, kterou označme S. Řetězec popisující řešení bude mít délku n, každý jeho znak může nabývat hodnot 1 až n (přičemž se každá hodnota musí v řetězci objevit právě jednou). Postup dekódování řetězce reprezentujícího řešení je následující. Postupně procházíme posloupnost S a spouštíme úlohy, přičemž úloha j je spuštěna v nejdivějším možné čase po dokončení předchozí úlohy ($s_j \geq rd_j$). Po dokončení všech úloh spočteme celkovou pokutu za zpoždění.

Uvažujme řetězec $S = \langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle$. Řešení odpovídající tomuto řetězci je zakresleno v následujícím Ganttově diagramu. Čísla v obdélnících vyjadřují číslo úlohy a zpoždění úlohy, šířka obdélníku odpovídá době zpracování úlohy.



Úloha číslo 1 může být spuštěna hned na počátku. Úloha 2 je naplánována na čas $s_2 = 5$, protože zpracování úlohy 1 nedovolí dřívější spuštění. Po ukončení úlohy je spuštěna úloha 3 v čase $s_3 = 9$, skončí tedy 3 časové jednotky po stanoveném okamžiku dokončení. Následují úlohy 4 a 5. Úloha číslo 6 nemůže být spuštěna dříve než v čase $s_6 = rd_6 = 28$. Na závěr jsou provedeny úlohy 7 a 8, přičemž druhá z nich končí v čase 40. Celková pokuta za zpoždění je $3 \cdot 5 + 5 \cdot 4 + 4 \cdot 4 = 51$.

Výše uvedenou reprezentaci jsme zvolili proto, že se používá ve velkém množství plánovacích problémů (viz [7]). Genetické algoritmy pracují obvykle nad binárními řetězci. Binární reprezentace není pro kombinatorické optimalizační úlohy vhodná.

U GA je velmi důležité vybrat vhodné a efektivní kódování pro řešení problému, aby bylo možno jednoduše provádět jak dekódování řešení, tak i operace křížení a mutace. Výše uvedený problém můžeme reprezentovat například také řetězcem délky n, kde každá pozice v řetězci $j = 1, \ldots, n$ určuje prioritu úlohy j. Tuto reprezentaci lze dekódovat ve dvou krocích. Nejdříve získáme posloupnost úloh S tak, aby úlohy byly seřazené v pořadí neklesajících hodnot priorit. Následně vytvoříme plán provádění stejně jako při použití reprezentace pomocí posloupnosti úloh. Je zřejmé, že nutnost řazení podle priorit představuje zvýšení zátěže, pro které nalezneme opodstatnění pouze v případě, že se tím zefektivní provádění operací křížení a mutace. Uvažujeme-li reprezentaci, kdy pozice v řetězci udává počáteční čas s_j úlohy j, dojde k velkému zvýšení složitostí provádění křížení a mutace. Jak bylo uvedeno dříve, mutace způsobí náhodou změnu řetězce reprezentujícího jedince. Náhodná změna počátečního času způsobí nefunkčnost získaného řešení, neboť by došlo k vytvoření plánu, při kterém se mají dvě úlohy provést současně. Vyřešení tohoto problému při křížení jedinců je ještě složitější.

Při každé aplikaci GA je nutné vhodně stanovit velikost populace P, tzn. počet jedinců, ze kterých je složena jedna generace. Jestliže je tato hodnota příliš nízká, dojde k omezení prohledávaného stavového prostoru, protože se v každé iteraci kříží pouze malé množství jedinců, kteří jsou si s postupujícími generacemi čím dál podobnější. Naopak při příliš velké populaci dojde k tomu, že se křížení účastní i méně kvalitní jedinci, což vede ke zpomalení konvergence k optimálnímu řešení. Uvádí se, že nejvhodnější velikostí populace je sudé číslo z intervalu $P \in (50, 100)$ (viz [7]).

Před spuštěním genetického algoritmu je nutné vytvořit počáteční populaci. Obvykle se používá náhodné generování řetězců, avšak je možné také použít jednoduchou heuristiku, např. přístup založený na náhodných prioritních pravidlech (viz [2]), která nám zaručí kvalitní populaci.

3. Vyhodnocení a výběr jedinců

Jednotlivci se v nové generaci projeví s pravděpodobností odpovídající jejich hodnotě fitness. Vytvoříme soubor genů, který bude obsahovat P kopií jedinců, přičemž kvalitní jedinci mohou být v souboru obsaženi několikrát, nekvalitní jedinci nemusí být zastoupeni

ani jednou. Tento princip je stejný jako v biologické evoluci. Nejlepší jedinci by se měli na příští generaci podílet nejvíce, tj. jejich pozitivní vlastnosti by se měly co nejvíce přenést do potomků. Naopak jedinci s malou pravděpodobností výběru „vymřou“.

V nejjednodušším případě se hodnota fitness v_i pro jedince $i = 1, \ldots, P$ stanoví jako funkční hodnota f_i příslušejícího řešení. Při řešení maximalizačních úloh můžeme proces výběru jedinců rozdělit na dva následující kroky. Nejdříve vytvoříme ruletové kolo s P přihrádkami, jejichž velikost bude odpovídat hodnotě fitness. Celkovou hodnotu fitness spočteme jako $T = \sum_{i=1}^{P} v_i$. Následně každému jednotlivci přiřadíme pravděpodobnost výběru $p_i = v_i / T$ a kumulativní pravděpodobnost $q_i = \sum_{h=1}^{i} p_i$. Ve druhém kroku P-krát „roztočíme“ kolo rulety. Pokaždé do souboru genů zkopírujeme jedince vybraného podle následujícího schématu. Po vygenerování náhodného čísla $\beta \in [0, 1]$ je vybrán první jedinec, jestliže $\beta \leq q_1$, nebo i-tý jedinec, jestliže platí $q_{i-1} < \beta \leq q_i$.

V případě, kdy je úkolem minimalizace, musíme výše uvedený postup upravit. Jedním z řešení je stanovení čísla F, které převyšuje všechny funkční hodnoty možných řešení a položení hodnoty fitness $v_i = F - f_i$. Dalším problémem je relativní rozdíl hodnot fitness – např. hodnoty 20 a 40 se liší více než 1020 a 1040, ačkoliv je v obou případech rozdíl jen 20.

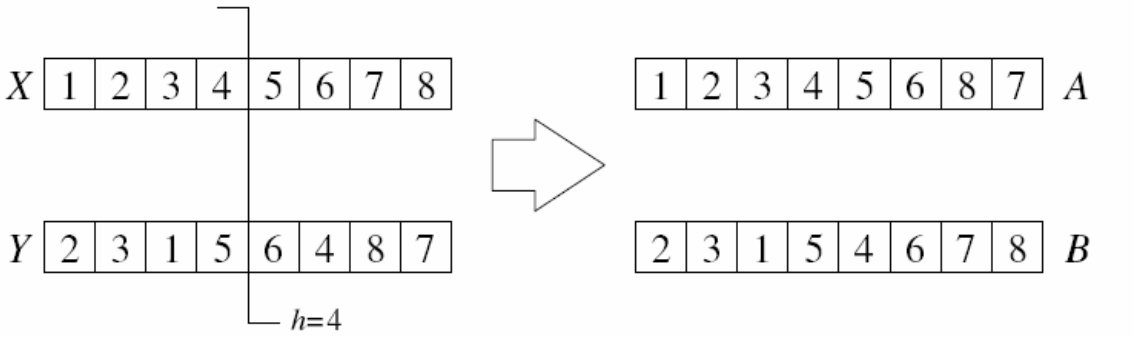
Proto se při implementaci procesu výběru jedinců používají další dvě metody. První je výběr podle pozice (ranking selection), při němž jsou jedinci seřazeni podle svých fitness hodnot v neklesající posloupnosti a každému je přiřazena hodnota r_i, označující jeho pořadí v posloupnosti. Pravděpodobnost výběru jednotlivce je pak $p_i = 2r_i / (P \cdot (P+1))$. Nejlepší jedinec je vybrán s pravděpodobností $p_i = 2 / (P + 1)$, což je skoro dvojnásobek mediánu, jehož pravděpodobnost je $p_i = 1 / P$. Jakmile vygenerujeme pravděpodobnosti p_i, je možné k výběru jedinců použít ruletového kola, popsaného výše.

Druhou možností je turnajový výběr (tournament selection). Množinu jedinců náhodně zamícháme a postupně z počátku vybíráme posloupnosti jedinců o délce L. V každé z takto vybraných posloupností najdeme jedince s nejlepší hodnotou fitness a zkopírujeme jej do souboru genů. Algoritmus končí v případě, že máme již vybráno P jedinců k reprodukci nebo vyčerpáme seznam jedinců. V takovém případě spustíme algoritmus od začátku.

Všechny výše uvedené metody, kromě turnajového výběru, nezaručují, že jsou pro reprodukci vybráni ti nejlepší jedinci, což vede k neefektivnosti hledání řešení optimalizačních úloh. Z tohoto důvodu se používá elitářství, kdy nejlepšího jedince z celé populace přímo zkopírujeme do souboru genů a postup výběru provedeme pouze (P-1)-krát. Obecně lze takto upřednostnit i více jedinců.

4. Rekombinace a mutace

Při rekombinaci vybíráme ze souboru genů dvojice jedinců náhodně nebo systematicky. Křížení jedinců provádíme s určitou pravděpodobností γ. Z toho plyne, že s pravděpodobností $(1 - \gamma)$ ke křížení nedojde a do nové populace jsou zkopírováni oba jedinci. Tento postup opakujeme do doby, než je v nové populaci P jedinců. V literatuře se uvádí různé hodnoty pravděpodobnosti γ, přičemž $\gamma < 0{,}6$ nejsou efektivní [7].



Popíšeme si nejjednodušší a nejpoužívanější typ křížení – jednobodové křížení. Obecně je definováno pro dva řetězce délky n. Pro každou dvojici jedinců X a Y náhodně vybereme bod křížení $h \in [1, n - 1]$. Prvního potomka získáme konkatenací prvních h znaků řetězce rodiče X s posledními $n - h$ znaky řetězce rodiče Y. Druhého potomka získáme přesně opačně. Tento postup však nefunguje obecně se všemi uvedenými reprezentacemi řešení. V našem příkladě by tato implementace křížení fungovala při reprezentaci založené na posloupnosti priorit, avšak selhala by při reprezentaci pomocí posloupnosti úloh, protože by se některé úlohy ve výsledku objevily dvakrát, zatímco jiné ani jednou.

Z tohoto důvodu se pro reprezentace založené na posloupnostech používá jiný přístup, který vidíme na obrázku. Náhodně vybereme bod křížení, prvních h znaků rodiče X zkopírujeme do prvního potomka A, a následně se zbývajících $n - h$ znaků naplní tak, že vezmeme všechny zbývající hodnoty z rodiče Y, které nejsou v potomkovi A přítomny, a to v pořadí podle jejich stoupajícího umístění v rodiči Y. Obdobně vytvoříme druhého potomka B. Povšimněte si, že v našem příkladu mají oba potomci lepší hodnotu fitness než jejich rodiče. Celková zpoždění pro rodiče X a Y jsou 51 a 93, zatímco pro potomky A a B dostaneme 47 a 29.

Kromě rekombinace se na určitém počtu jedinců provede mutace, jež má za cíl rozrůznit populaci, aby nedocházelo k prohledávání stále stejných jedinců. Proto je v GA nutné stanovit pravděpodobnost mutace *d*. Obvykle se hodnota *d* vybere buď jako malé číslo, např. $d = 0{,}01$, nebo se použije $d = 1 / n$, neboť tato hodnota se osvědčila při řešení mnoha problémů [7]. Obecně se mutace provádí náhodnou změnou obsahu jednoho znaku řetězce. V našem případě by to mohlo vést k nekorektnímu výsledku, proto použijeme jednu ze dvou dalších možností. Tou první je mutace pomocí výměny, kdy náhodně vybereme dvě pozice v řetězci a jejich hodnoty prohodíme. V našem případě můžeme zvolit např. pozice 3 a 6 jedince *A*, čímž nám po mutaci vznikne $A' = \langle 1, 2, 6, 4, 5, 3, 8, 7 \rangle$. Mutace pomocí posunu je druhou možností a spočívá v náhodném výběru pozice v řetězci a posunutí odpovídající hodnoty o náhodný počet míst vlevo nebo vpravo. Například po výběru pozice 6 potomka *B* a posunutí o tři pozice vlevo dostáváme $B' = \langle 2, 3, 6, 1, 5, 4, 7, 8 \rangle$.

5. Závěr

Předchozí odstavce shrnuly základní principy použití genetických algoritmů pro řešení kombinatorických optimalizačních problémů. Také jsme ukázali, že při řešení konkrétních problémů je možno použít různých postupů, a to jak pro reprezentaci řešení, tak i pro výběr jednotlivců, ale i různé strategie pro rekombinaci a mutaci. Naštěstí jsou i jednoduché implementace genetických algoritmů schopné uspokojivě řešit mnoho problémů. Nicméně je třeba poznamenat, že většina výpočetních experimentů, uvedených v literatuře, řeší problémy malého a středního rozsahu. V oblasti plánování se obvykle nevyskytují problémy rozsáhlejší než sto úloh. Pro *n* úloh může existovat *n*! různých posloupností, takže se prohledávaný prostor se zvětšujícím počtem řešení rapidně zvětšuje, a může být nutno vyhodnotit mnoho generací, než se dostaneme k optimálním řešením.

Navíc může být v závislosti na řešeném problému obtížné vhodně nastavit omezení, která nám odstraní řešení nevedoucí k cíli. U problémů řešících plánování výroby bývají omezení závislá na posloupnosti jednotlivých úloh nebo časových intervalů, během kterých mohou být jednotlivé úlohy provedeny. K řešení takovýchto komplikací můžeme použít více různých technik. Jednou možností je zavedení pokuty do hodnoticí funkce, která nevhodným řešením přiřazuje nízké hodnoty fitness. Více informací na toto téma je možno nalézt v literatuře [7].

6. Literatura

[1] DOWSLAND, K. A. Genetic algorithms - A tool for OR?. In *Journal of the Operational Research Society*. Vol. 47, 1996. s. 550-561.

[2] DREXL, A. Scheduling of project networks by job assignment. In *Management Science*. Vol. 37, 1991. s. 1590-1602.

[3] GOLDBERG, David E., et al. *Genetic algorithms in search, optimization, and machine learning*. Reading, Massachusetts : Addison-Wesley Longman Publishing Co., Inc., 1989. 372 s.

[4] HOLLAND, John H. *Adaptation in natural and artificial intelligence*. University of Michigan Press, 1975. 206 s.

[5] MICHALEWICZ, Z., et al. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edition. Berlin : Springer, 2000. 387 s.

[6] REEVES, C. R., et al. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993. 320 s.

[7] REEVES, C. R. Genetic algorithms for the operations researcher. In *INFORMS Journal on Computing*. Vol. 9, 1997. s. 231-250.