UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

BAKALÁŘSKÁ PRÁCE

Úloha o batohu a genetické algoritmy

Vedoucí bakalářské práce:
RNDr. Horymír Netuka, Ph.D.
Rok odevzdání: 2014

Vypracovala:
Lenka Hasalová
MAP, III. ročník

Prohlášení

Prohlašuji, že jsem vytvořila tuto bakalářskou práci samostatně za vedení RNDr. Horymíra Netuky, Ph.D. a že jsem v seznamu použité literatury uvedla všechny zdroje použité při zpracování práce.

V Olomouci dne 18. dubna 2014

Poděkování

Ráda bych na tomto místě poděkovala vedoucímu bakalářské práce RNDr. Horymíru Netukovi, Ph.D. za obětavou spolupráci i za čas, který mi věnoval při konzultacích. Dále si zaslouží poděkování zvláště moje rodina za trpělivost a podporu.

Obsah

1 Úvod

Cílem této práce je vyřešit úlohu o batohu pomocí genetických algoritmů.

První kapitola pojednává o úloze o batohu. Je zde popsán princip tohoto problému a jeho rozdělení na několik typů. Základním a nejčastěji se objevovaným typem je tzv. binární problém batohu (nebo též 0-1 Problém batohu), který lze zobecnit na typy méně známé, kam se řadí omezený problém batohu, mnohonásobný problém batohu, problém batohu s možností výběru nebo třeba problém součtu podmnožin.

Závěr této kapitoly náleží jiným způsobům řešení této úlohy (než jsou genetické algoritmy), jimiž jsou metoda větví a mezí, heuristika podle poměru cena/hmotnost a metoda hrubé síly.

Druhá kapitola je zaměřená na genetické algoritmy. Nejprve jsou zde vysvětleny základní pojmy, bez kterých se při sestrojení genetického algoritmu nelze obejít. V této části jsou uvedeny různé možnosti kódování informací (binární, permutační, kódování hodnotami a stromové). Blíže jsou zde popsány operátory genetických algoritmů, kterými jsou selekce (u ní jsou uvedeny tři typy, a to turnajovou metodu, váženou ruletu a poziční selekci), křížení a mutace, i to, jak algoritmus ukončit.

Konečná část této kapitoly patří ukázce dvou problému, které se dají vyřešit pomocí genetických algoritmů, a to problém obchodního cestujícího a problém N dam.

Následující kapitola se zabývá spojením informací získaných z prvních dvou částí, tedy způsobu, jak úlohu o batohu vyřešit pomocí genetických algoritmů. Je zde předvedeno, jak vhodně zvolit kódování, jak vytvořit počáteční populaci, na co si dát pozor při ohodnocení jedinců, jakým způsobem bude probíhat selekce, mutace a křížení a taky to, jak se proces zastaví.

Teorie uvedená v předešlých kapitolách je v závěru této práce demonstrována na konkrétních příkladech.

2 Úloha o batohu a její specifika

Problémy batohu se začly intenzivně zkoumat na konci padesátých let dvacátého století. Nejen díky jejich okamžitému využití v průmyslu a finančním managementu, ale také kvůli jejich rozsáhlému teoretickému využití například v celočíselném programování.

V této kapitole se seznámíme s úlohou o batohu a rozdělíme ji na několik typů, jako jsou například binární (nebo též jednoduchý) problém batohu, omezený a neomezený problém batohu, problém s možností výběru, mnohonásobný problém nebo problém součtu podmnožin.

Všechny tyto problémy patří mezi tzv. NP-úplné problémy, což znamená, že nebyl nalezen algritmus, který by jej vyřešil s nižší než exponenciální složitostí a předpokládá se, že takový algoritmus ani neexistuje. Přesto i v nejhorších případech exponenciální složitosti lze tento problém vyřešit během zlomku vteřiny. Tomuto minimálně překvapivému výsledku předcházely desetiletí výzkumu, který objasnil speciální konstrukční vlastnosti úlohy batohu, díky kterým lze tento problém poměrně snadno vyřešit.

Princip úlohy o batohu si můžeme ukázat na následujícím příkladu. Představme si zloděje, který vnikne do cizího objektu, kde je spousta více či méně drahých věcí. Ale zloděj má s sebou batoh, který má pouze omezenou nosnost, proto se snaží věci pobrat tak, aby jejich celková hodnota byla co největší, přičemž nemůže překročit nosnost batohu.

2.1 Typy úloh

Ve všech následujících problémech budeme mít k dispozici n předmětů a budeme předpokládat, že

$$p_j = \text{cena } j\text{-tého předmětu, } j = 1, ..., n$$
$$w_j = \text{hmotnost } j\text{-tého předmětu, } j = 1, ..., n$$
$$C = \text{celková kapacita batohu}$$

přičemž všechny koeficienty jsou celá nezáporná čísla.

2.1.1 Binární problém batohu (0-1 Knapsack problem)

Tento problém se považuje za nejdůležitější a je nejčastěji zkoumaný, zvláště z těchto tří důvodů:

1. Považuje se za nejjednodušší problém celočíselného lineárního programování.
2. Často se objevuje jako podproblém mnoha větších a složitějších problémů.
3. Poměrně snadno se implementuje do reálných problémů.

Pokud v literatuře najdeme pouze úlohu o batohu bez blíže specifikovaného typu, s největší pravděpodobností se bude jednat právě o tento případ.
V této úloze máme k dispozici n předmětů, ze kterých vybereme s předmětů tak, aby součet cen vybraných věcí byl co největší, ale zároveň aby jejich hmotnost nepřekročila kapacitu batohu C.

$$\max \sum_{j=1}^{n} p_j x_j$$
$$\text{za podmínky } \sum_{j=1}^{n} w_j x_j \leq C$$
$$x_j \in \{0, 1\}, j = 1, ..., n$$

přičemž x_j je rovno jedné, pokud j-tý předmět vybereme, a nule, jestliže jej nevybereme.

Binární problém batohu se ve většině případů objevuje jako maximalizační úloha, ale poměrně snadno se dá tranformovat na úlohu minimalizační, ta je potom ve tvaru:

$$\min \sum_{j=1}^{n} p_j y_j$$

$$\text{za podmínky } \sum_{j=1}^{n} w_j y_j \geq D$$

$$y_j \in \{0, 1\}, j = 1, ..., n$$

kde $y_j = 1 - x_j$ a $D = C - \sum_{j=1}^{n} w_j$. Dále nechť $zmax$ je hodnota řešení příslušné maximalizační úlohy, potom $zmin = \sum_{j=1}^{n} p_j - zmax$ je řešením naší minimalizační úlohy. Jinými slovy by se dalo říct, že maximalizujeme hodnotu věcí, které nejsou v batohu.

2.1.2 Omezený problém batohu (Bounded Knapsack problem)

Nechť máme omezené množství m_j položky typu j, dostaneme úlohu ve tvaru:

$$\max \sum_{j=1}^{n} p_j x_j$$

$$\text{za podmínky } \sum_{j=1}^{n} w_j x_j \leq C$$

$$x_j \in \{0, 1, ..., m_j\}, j = 1, ..., n,$$

kde m_j označuje maximální množství položky j, které můžeme vložit do batohu. Vlastně se jedná o zobecnění binárního problému batohu, ve kterém je $m_j = 1$ pro všechna $j = 1, ..., n$.

Neomezený problém batohu (Unbounded Knapsack problem) Speciálním případem omezeného problému batohu je úloha, ve které máme neomezené množství předmětů všech typů, tedy z předešlé úlohy $m_j = \infty$.

Problém lze tedy zapsat:

$$\max \sum_{j=1}^{n} p_j x_j$$

$$\text{za podmínky } \sum_{j=1}^{n} w_j x_j \leq C$$

$$x_j \in \{0, 1, ..., \}, j = 1, ..., n,$$

2.1.3 Mnohonásobný problém batohu (Multiple Knapsack Problem)

V literatuře jej můžeme najít taky pod názvem problém batohu se všeobecnou horní hranicí (Knapsack problem with generalized upper bound constraints). Mnohonásobný problém batohu spočívá v tom, že máme umístit n předmětů do m batohů s (ne nutně různými) kapacitami C_i. Úloha je tedy ve tvaru:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij}$$

$$\text{za podmínek } \sum_{j=1}^{n} w_j x_j \leq C_i, i = 1, ..., n$$

$$\sum_{j=1}^{m} x_{ij} \leq 1$$

$$x_j \in \{0, 1\}, j = 1, ..., n, i = 1, ..., m,$$

tedy $x_{ij} = 1$ nám říká, že je j-tá položka obsažena v i-tém batohu, podmínka $\sum_{j=1}^{n} w_j x_j \leq C_i$ nám zajišťuje nepřekročení kapacity jednotlivých batohů a díky omezení $\sum_{j=1}^{m} x_{ij} \leq 1$ víme, že každá položka bude vybrána pouze jednou.

2.1.4 Problém batohu s možností výběru (Multiple-choise Knapsack problem)

I tento problému typ je jistým zobecněním binárního problému batohu, v tomuto případě každou položku vybíráme z k tříd, které označíme $N_i, i = 1, ..., k$ a z každé třídy si můžeme vzít právě jeden předmět.

Úlohu tedy formulujeme následovně:

$$\max \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij}$$

$$\text{za podmínek} \sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq C$$

$$\sum_{j \in N_i} x_{ij} = 1, i = 1, ..., k$$

$$x_{ij} \in \{0, 1\}, j \in N_i$$

Pokud je x_{ij} rovno jedné, znamená to, že byl j-tý předmět vybrán z i-té třídy. Podmínka $\sum_{j \in N_i} x_{ij} = 1, i = 1, ..., k$ nám zaručuje, že z každé třídy je vybrán právě jeden předmět.

2.1.5 Problém součtu podmnožin (Subset-sum problem)

Nyní opusťme zloděje a představme si člověka, který jede tábořit a chce si s sebou do batohu vzít co nejvíce užitečných věcí, ale už mu tolik nesejde na jejich hodnotě. Pro tento případ můžeme tedy předpokládat, že $p_j = w_j$ pro všechna $j = 1, ..., n$ a dostaneme úlohu:

$$\max \sum_{j=1}^{n} w_j x_j$$

$$\text{za podmínky} \sum_{j=1}^{n} w_j x_j \leq C$$

$$x_j \in \{0, 1\}, j = 1, ..., n$$

Už z názvu tedy plyne, že se na tento typ problému lze dívat jako na úlohu, ve které máme vybrat podmnožinu hmotností $w_1, ..., w_n$ takových, aby jejich součet byl co největší bez toho, aby překročil kapacitu C.

2.2 Metody řešení

Jak už jsme si řekli na začátku této kapitoly, úloha o batohu patří do třídy NP-úplných úloh, tedy neznáme jinou možnost přesného způsobu řešení tohoto

problému, než je (nejspíš úplné) vyčíslení prostoru řešení. Existují ovšem metody, které nám mohou mnoho času a úsilí ušetřit. Na některé z nich se teď podíváme.

2.2.1 Metoda větví a mezí (Branch and bound)

Mějme množinu všech možných řešení úlohy a předpokládejme, že tuto množinu můžeme seřadit do stromu, kde listy představují jednotlivá řešení, uzly symbolizují rozdělení těchto řešení do podmnožin podle nějaké společné vlastnosti. U každého uzlu stanovíme horní (a v případě minimalizační úlohy dolní) odhad hodnot. Tyto hodnoty pak mezi sebou porovnáváme, díky čemuž potom můžeme „odřezat nepotřebné větve“, což znamená, že rozdělíme-li množinu všech přípustných hodnot na dvě podmnožiny, a ta první má nižší horní odhad hodnot než ta druhá, tak první podmnožinu už při hledání hodnot nebudeme potřebovat a můžeme už dále prohledávat jen druhou.
Při řešení úlohy o batohu metodou větví a mezí budeme postupovat následovně (viz [1], str.353 - 355):

1. Nejprve si pro každý předmět stanovíme poměr jeho ceny a hmotnosti a označme jej r_j, tedy $r_j = \frac{p_j}{w_j}, j = 1, ..., n$.

2. Tyto hodnoty seřadíme do nerostoucí posloupnosti a přeindexujeme tak, aby první prvek v posloupnosti byl r_1.

3. Následujícím způsobem určíme horní odhad všech řešení obsažených ve vrcholu u

   (a) Nechť vrchol u obsahuje prvních l předmětů. Určíme součet cen a hmotností těchto položek: $p(u) = \sum_{j=1}^{l} p_j x_j$, $w(u) = \sum_{j=1}^{l} w_j x_j$.

   (b) Je-li $w(u) > C$, pak jsou všechna řešení v tomto uzlu nepřípustná. Pokud $w(u) = C$, potom jsme zcela naplnili batoh a dále již nepokračujeme. Pro $w(u) < C$ označme $t(u) = C - w(u)$ zbývající volné místo v batohu.

(c) Do batohu dále vkládáme předměty $l{+}1, l{+}2, ...$, dokud platí $\sum_{j=l+1}^{l+k} w_j < t(u)$.

(d) Předpokládejme, že $(l{+}k)$-tý předmět se jako poslední vejde do batohu celý a po jeho vložení nám zbude $t_k = t(u) - \sum_{j=l+1}^{l+k} w_j$ volného místa.

(e) Nechť předmět $(l + k + 1)$ lze rozdělit takovým způsobem, že se dá do batohu vložit taková jeho část, která ho maximálně doplní, a zároveň předpokládáme, že cena poměrově odpovídá vložené části.

(f) Tedy do batohu můžeme vložit už jenom (w_{l+k+1}/t_k)-tou část $(l{+}k{+}1)$-tého předmětu, která bude mít hodnotu $h_{l+k+1} = \frac{p_{l+k+1} w_{l+k+1}}{t_k}$.

(g) Horní odhad řešení, která se nechazejí v uzlu u je tedy ve tvaru $H(u) = p(u) + \sum_{j=l+1}^{l+k} w_j + h_{l+k+1}$.

4. Nalezením horních odhadů řešení jsme získali hodnoty, které mezi sebou můžeme porovnávat a tedy můžeme snadno určit, které podmnožiny řešení už nebudeme potřebovat a můžeme je "odřezat".

2.2.2 Heuristika podle poměru cena/hmotnost

1. Opět si nejprve pro každou položku určíme poměr ceny a hmotnosti a označíme jej r_j.

2. Poměry si seřadíme do nerostoucí posloupnosti a přeindexujeme je tak, že první prvek v posloupnosti bude r_1.

3. Potom postupně vkládáme předmět s poměrem r_1, předmět s poměrem r_2,... Takto pokračujeme, dokud se předměty vlezou do batohu.

4. Vrátíme se k původnímu indexování, čímž se dozvíme, které předměty byly vybrány.

5. Sečteme ceny věcí v batohu, a pokud je tento součet menší než cena nejdražší položky, pak je tato položka výsledkem.

2.2.3 Metoda hrubé síly (Broute force)

Metoda hrubé síly naší úlohu vyřeší tak, že nejprve vygeneruje množinu všech podmnožin předmětů, jejichž suma vah nepřekročí kapacitu batohu, a z těchto podmnožin vybere tu s největší cenou.

3 Genetické algoritmy

Genetický algoritmus je stochastická optimalizační metoda, jejíž princip je založen na Darwinově teorii o vývoji druhu. Umožňuje populaci mnoha jednotlivců vyvinout se pomocí předem stanovených pravidel selekce do stavu, který je optimální nebo alespoň dostatečně vyhovující (např. nalezení minima funkce).

Jako první genetické algoritmy popsal ve své práci "Adaptation in Natural and Artificial Systems"v roce 1975 americký vědec John Holland z Michiganské univerzity. Nicméně nápad s použitím evoluční teorie k optimalizaci přišel již o více než dekádu dříve od Ingo Rechenberga v jeho knize "Evolutions strategies"(1960). Další významnou osobností, která se zasloužila o popularizaci genetických algoritmů byl John Koza, který je díky své knize "Genetic Programming"(1992) považován za zakladatele genetického programování.

V této části si nejprve definujeme základní pojmy, které budeme využívat při aplikaci genetických algoritmů. Ukážeme si různé typy genetických operací selekce, mutace a křížení. Uvedeme si, jaké lze použít zastavovací kritéria i to, jak genetický algoritmus funguje. Na úplný závěr této kapitoly si předvedeme použití těchto algoritmů na problém obchodního cestujícího a na problém N dam.

3.1 Základní pojmy a procesy

Genetické algoritmy jsou iteračním optimalizačním procesem, který opakovaně využívá genetických operací jako je *selekce*, *mutace* a *křížení*, až do té doby, dokud není splněno nějaké zastavovací kritérium. V genetickém algoritmu jsou počáteční informace zakódovány v řetězci, který je analogií *chromozomu* v genetice. Tento řetězec je složen z jednotek, které jsou zase analogií *genů*. Množina chromozomů (nebo jedinců) tvoří *populaci*. Každý iterační krok, ve kterém vznikne nová populace, se nazývá *generace*. Číselnou hodnotu kvality jedince a jeho "vhodnost"k reprodukci nám dává tzv. *fitness funkce*.

3.1.1 Kódování

Kódování je jedním z prvních problémů, se kterým se setkáme, když se začneme zabývat genetickými algoritmy. Typ kódování volíme v závislosti na řešeném problému. V této podkapitole si několik druhů představíme.

Binární kódování Binární kódování je základním a jedním z nejčastěji využívaných typů kódování, zřejmě také proto, že se využívalo v první práci o genetických algoritmech.
V tomto kódování je každý chromozom zapsán pomocí řetězce nul a jedniček.

Příklad chromozomu:

$$(0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ )$$

Binární kódování nám umožňuje pracovat i s chromozomy s malým počtem genů. Nevýhodou však je, že toto kódování bývá někdy nepřirozené a musí provádět jisté opravy i po křížení a mutaci.

Permutační kódování Toto kódování se využívá v problémech, které řeší uspořádání (např. Problém obchodního cestujícího).
V tomto případě je chromozom reprezentován řetězcem, který tvoří nějakou posloupnost.

Příklad chromozomu:

(2 1 3 4 5 6 8 9 7)

I v problémech, které využívají permutační kódování, je třeba po křížení a mutaci provést určité korekce, abychom dosáhli kýženého výsledku (například posloupnost musí být reálná).

Kódování hodnotami Toto kódování využíváme v problémech s komplikovanějšími hodnotami, například by bylo obtížné použít binární kódování u problému, který popsán reálnými čísly.
U kódování hodnotami je chromozom představován řetězcem hodnot, které mohou být jakékoliv.

Příklady chromozomů:

1. (2.123 0.862 9.881 7.564 5.225 2.869 3.014 6.006 7.113)
2. (A V O L S H K N E)

Nedostatkem tohoto kódování je, že obvykle je nutné nalézt nějaké nové křížení specifické pro daný problém.

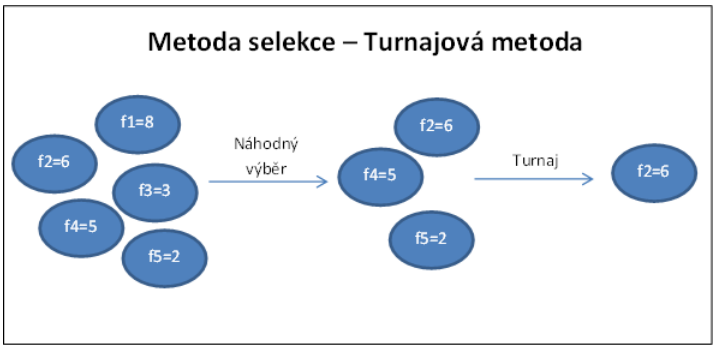
Stromové kódování Tento typ kódování je obvykle užíván v genetickém programování.
U stromového kódování je chromozom zapsán jako strom objektů, kterými mohou být například funkce nebo příkazy v nějakém programovacím jazyce.

3.1.2 Selekce

Úkolem selekce je rozhodnout, které chromozomy budou vybrány k reprodukci tak, aby jedinci nové generace měli co nejvyšší hodnotu fitness funkce. Operátor selekce zajistí, aby jedinci s vyšší fitness hodnotou měli větší šanci se dostat ke křížení a mutaci, ale zároveň aby i jedincům s nižší hodnotou zbyla nějaká,

i když menší, pravděpodobnost na vybrání. Různé druhy selekce nám dávájí různé možnosti, jak tuto pravděpodobnost vypočítat. Každý z typů selekce nám dává řešení založené na principu přežití nejsilnějšího. Je pravděpodobnější, že silnější jedinci se budou dále rozmnožovat a předávat svůj genetický materiál další generaci v podobě svých potomků. V následujícím textu si popíšeme tři druhy selekce, a to metodu turnaje, váženou ruletu a poziční selekci.

Metoda turnaje Tato metoda je oblíbená zejména díky své efektivitě a snadné implementaci. Metoda turnaje spočívá v tom, že z populace náhodně vybereme n jedinců, a ty mezi sebou necháme soutěžit. Vítězem turnaje je jedinec s nejvyšší hodnotou fitness funkce a ten se bude podílet na vytvoření jedinců populace v nové generaci. Počet jedinců, kteří soutěží v turnaji se často označuje jako *velikost turnaje* (v případě dvou jedinců hovoříme o *binárním turnaji*).
V této metodě mají všichni jedinci šanci být vybráni, což je výhodné zejména kvůli zachování rozmatitosti populace. Zachování genetické rozmanitosti ovšem může mít za následek snížení rychlosti konvergence k optimálnímu řešení. Velkou výhodou turnajové metody je nízká složitost.



Obrázek 1: Schéma metody turnaje (fi označuje fitness hodnotu i-tého jedince)

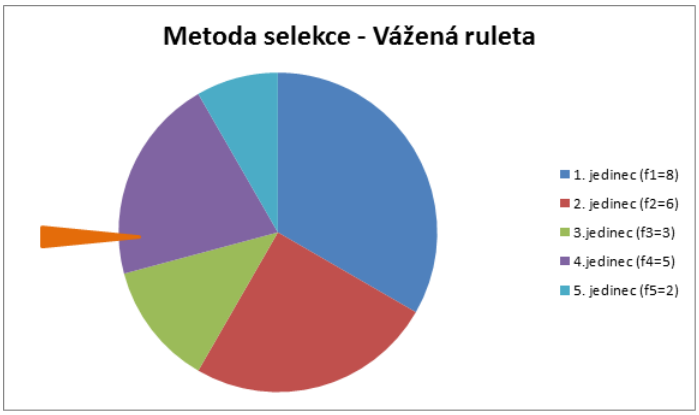
Metoda vážené rulety Jedinci jsou rozděleni podle pravděpodobnosti přímo úměrné hodnotě jejich fitness funkce. Rozdělení pravděpodobností rodičů si můžeme představit jako točící se ruletu, kde velikosti dílků poměrově odpovídají jejich

fitness hodnotě. Zřejmě tedy platí, že jedinec s vyšší hodnotou fitness funkce má větší pravděpodobdnost být vybrán, protože zabírá větší část rulety. Když ruletu roztočíme, tak při zastavení zobáček ukáže na jeden z dílků, s největší pravděpodobností na ten největší, ale stále ještě existuje šance, že ukáže i na jeden z menších dílků.
Nechť $f_1, ..., f_n$ jsou hodnoty fitness funkcí jedinců jedné populace, potom pravděpodobnost, že bude vybrán právě i-tý jedinec je rovna:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i} \text{ pro všechna } i = 1, ..., n.$$

Hlavní výhodou vážené rulety je to, že dává šanci na vybrání všem jedincům populace (včetně těch s minimální fitness hodnotou). Díky tomu se zachovává v populaci genetická rozmanitost. Za nevýhodu lze zase považovat to, že silnější jedinci jsou hned od začátku upřednostňováni, což může zapříčinit zamrznutí v lokálně optimálním řešení nebo právě ztrátu diverzity. Představme si třeba, že se v populaci vyskytují jeden či dva dobří, ale ne nejlepší, jedinci a ostatní jedinci jsou slabí. Pak tito silní jedinci rychle ovládnou celou populaci a zabrání jí najít potenciálně lepší jedince. Silná dominance těchto jedinců způsobí velkou ztrátu rozmanitosti, což je ve finále velkou nevýhodou celého optimalizačního procesu. Na druhou stranu mají-li všichni jedinci v populaci velmi podobnou fitness hodnotu, pak je velmi obtížné pro populaci se zlepšovat, protože má každý jedinec skoro stejnou šanci být vybrán.

Poziční selekce Poziční selekce roztřídí jedince populace podle jejich fitness hodnoty a seřadí je. Každému chromozomu je přiřazena selekční pravděpodobnost, která je vypočítána z pořadí jedince. Tedy už nebere v úvahu přímo fitness hodnoty. A přímo úměrně těmto pravděpodobnostem je jedincům přiřazena velikost dílků na ruletě. Od předchozí metody se poziční selekce liší hlavně selekčním tlakem. Používá se tedy zejména v případech, kdy se hodnoty fitness funkce jednotlivých chromozomů významně liší. Tento typ selekce umí řešit problémy



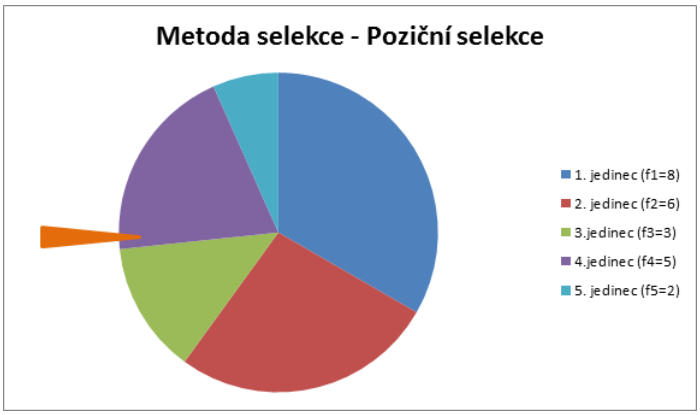
Obrázek 2: Schéma metody vážené rulety (fi označuje fitness hodnotu i-tého jedince)

jako jsou stagnace nebo předčasné zakotvení v lokálně nejlepším řešení. V tomto případě je pravděpodobnost výběru i-tého jedince dána vztahem:

$$p_i = \frac{r_i}{\sum_{i=1}^{n} r_i} \text{ pro všechna } i = 1, ..., n,$$

kde r_i udává pořadí jedince.

Výhodou této metody je, že dokáže řešit i případy, kdy má jeden chromozom velkou převahu nad ostatními. Ale vzhledem k tomu, že pravděpodobnosti výběrů jednotlivých chromozomů jsou podobnější než u metody rulety, tak je i celý proces pomalejší.



Obrázek 3: Schéma metody poziční selekce (fi označuje fitness hodnotu i-tého jedince)

3.1.3 Křížení a mutace

Křížení je operátor genetických algoritmů, který kombinuje dva jedince (*rodiče*) za účelem vytvoření nového jedince (*potomka*).
Četnost křížení nám udává tzv.*pravděpodobnost křížení*. Je-li tato pravděpodobnost rovna jedné, znamená to, že každý potomek nové populace vznikl křížením, pokud je pravděpodobnost rovna nule, pak je nová populace identická s tou původní.

Mutace je genetický operátor, který následuje po křížení. Obvykle probíhá na základě jediného jedince podle jeho fitness hodnoty. Funguje tak, že nejprve přečte jednu či více částí jedince, tato část je pak zmodifikována podle předem dané pravděpodobnosti nebo hodnoty fitness funkce. Bez mutace by chromozomy potomků byly omezeny pouze na na geny z počáteční populace. Mutace je schopná do populace přinést nový genetický materiál, jakož i změnit ten existující. S novými zmutovanými geny je genetický algoritmus schopen přijít na lepší řešení, než by bylo bez mutace možné.
Pravděpodobnost mutace nám říká, s jakou četností chromozom zmutuje. Jestliže je pravděpodobnost nulová, víme, že nezmutoval žádný chromozom. Naopak, pokud je tato pravděpodobnost rovna jedné, tak se změnily všechny chromozomy.

Křížení a mutace u binárního kódování

Křížení

1. **Jednobodové křížení**: Vybereme jeden křížící bod, všechny geny od začátku chromozomu po tento bod se zkopírují od prvního rodiče, část po křížícím bodě od druhého rodiče.

   (**1 0 1 0** | 1 1 0 0) + (1 1 1 0 | **0 0 1 0**) = (**1 0 1 0** | **0 0 1 0**)

2. **Dvoubodové křížení**: Tentokrát vybereme dva křížící body. Část chromozomu od začátku po první křížící bod se zkopíruje od prvního rodiče,

geny ležící mezi těmito body se vezmou od druhého rodiče a poslední část zase od prvního.

$$(\mathbf{0\ 1\ 0}\ |\ 1\ 0\ 1\ |\ \mathbf{1\ 1\ 1}) + (0\ 0\ 1\ |\ \mathbf{1\ 0\ 0}\ |\ 0\ 0\ 0) = (\mathbf{0\ 1\ 0}\ |\ \mathbf{1\ 0\ 0}\ |\ \mathbf{1\ 1\ 1})$$

3. **Rovnoměrné křížení**: Geny jsou náhodně kopírovány od obou rodičů. Operátor křížení rozhodne (s určitou pravděpodobností známou jako *mixing ratio*), který z rodičů předá jaké geny do chromozomu potomka.

$$(\ \mathbf{1\ 0}\ 1\ \mathbf{0\ 1\ 1}\ 1\ 0) + (\ 0\ 1\ \mathbf{1}\ 0\ 0\ 1\ \mathbf{0\ 0}\ ) = (\ \mathbf{1\ 0\ 1\ 0\ 1\ 1\ 0\ 0}\ )$$

4. **Aritmetické křížení**: Operátor křížení, který lineárně kombinuje rodiče a potomky vytváří podle vztahu:

$$P_1 = a * R_1 + (1 - a) * R_2$$
$$P_2 = a * R_2 + (1 - a) * R_1,$$

kde P_i označuje i-tého potomka, R_i i-tého rodiče a a je faktor náhodně vybraný před křížením.

Mutace : Jsou změněny vybrané geny.
Příklad (stav po křížení ⇒ stav po mutaci):

$$(\ 0\ 1\ 0\ 1\ \mathbf{1}\ 1\ 0\ 1\ ) \Rightarrow (\ 0\ 1\ 0\ 1\ \mathbf{0}\ 1\ 0\ 1\ )$$

Křížení a mutace při permutačním kódování

Křížení : Zvolíme jeden křížící bod. Geny které leží před tímto bodem zkopírujeme podle prvního rodiče. Dále si prohlédneme druhého rodiče a čísla, která ještě nejsou v potomkovi obsažena, zkopírujeme od něj. Existuje více možností, jak doplnit čísla po křížícím bodě.

$$(\ \mathbf{4\ 6\ 2\ 7\ 8}\ 1\ 9\ 5\ 3\ ) + (\ \mathbf{9\ 1}\ 8\ 2\ 7\ \mathbf{3}\ 6\ 4\ \mathbf{5}\ ) = (\ \mathbf{4\ 6\ 2\ 7\ 8\ 9\ 1\ 3\ 5}\ )$$

Mutace : Vybereme dvě čísla, a ta zaměníme.

Příklad (stav po křížení ⇒ stav po mutaci):

$$(\ 4\ \mathbf{6}\ 2\ 7\ 8\ 1\ \mathbf{9}\ 5\ 3\ ) \Rightarrow (\ 4\ \mathbf{9}\ 2\ 7\ 8\ 1\ \mathbf{6}\ 5\ 3\ )$$

Křížení a mutace při kódování hodnotami

Křížení : Je možno použít všechny typy křížení, které jsme si zmínili u binárního kódování.

Mutace : V případě, že jsou hodnotami reálná čísla, můžeme přičíst nebo odečíst malé číslo.

Příklad (stav po křížení ⇒ stav po mutaci):

(2.123 **4.225** 9.286 **7.442** 2.333) ⇒ (2.123 **4.218** 9.286 **7.436** 2.333)

Křížení a mutace při stromovém kódování

Křížení : U obou rodičů je vybrán jeden křížící bod, rodiče jsou v tomto bodě rozděleni a potomek vznikne tak, že se vymění část pod křížícím bodem.

Mutace : Změní se vybrané uzly.

3.1.4 Zastavovací kritéria

Tato kritéria nám říkají, kdy už je pro nás výsledek přijatelný a proces může být ukončen. Existuje více možností ukončení genetického algoritmu, mezi ty nejznámější patří například:

1. **Počet iterací**

   V tomto případě je algoritmus u konce po počtu iterací (nebo v našem případě generací), který si předem zadáme.

2. **Hodnota fitness funkce**

   U maximalizačních úloh si zadáme, jak velká hodnota už nám "bude stačit", u těch minimalizačních zase, která hodnota je dostatečně malá.

3. **Počet iterací beze změn**

   Při použití tohoto kritéria se proces zastaví, pokud v posledních k generacích nedochází ke změnám ohodnocení jedinců populace (k si předem zvolíme).

3.2 Popis genetického algoritmu

V předchozích podkapitolách jsme si vysvětlili základní pojmy a operace genetických algoritmů, tak už nám nic nebrání v tom, se podívat, jak genetický algoritmus funguje.

1. V prvním kroku si definujeme fitness funkci a její proměnné, nastavíme si operace genetických algoritmů (velikost populace, metoda selekce, pravděpodobnost křížení a mutace) a stanovíme si zastavovací kritérium.

2. Potom si náhodně vygenerujeme počáteční populaci.

3. Každého jedince zhodnotíme pomocí fitness funkce.

4. Vygenerujeme si populaci potomků pomocí operací genetického algoritmu (selekce, křížení, mutace)

5. Každého jedince z populace potomků ohodnotíme fitness funkcí.

6. Rozhodneme, kteří jedinci budou patřit do nové generace. V tomto kroku jsou jedinci z populace rodičů nahrazeni novou populací, jejíž jedinci pocházejí z populace potomků a/nebo z populace rodičů.

7. Pokud je v tuto chvíli splněno zastavovací kritérium, proces je u konce. V opačném případě se opakuje od kroku 4.

3.3 Aplikace genetických algoritmů

Genetické algoritmy se dají užít na celou škálu problému ze všech různých odvětví. Ať už je to predikce chování akciových trhů, sestavení a vylepšení plánů výroby nebo třeba řešení problému přisávání koberců na vysavač. My si v této podkapitole ukážeme řešení dvou konkrétních problémů pomocí genetických algoritmů, a to problém obchodního cestujícího a problém N dam.

3.3.1 Problém obchodního cestujícího

Principem tohoto problému je, že obchodník musí projet N měst s tím, že každé z nich může navštívit pouze jednou a na konci své cesty se musí vrátit do města, ze kterého vyjel.
Nalezení optimálního řešení už u 30 měst by nám trvalo dobu v řádu stovek biliónů let, avšak pomocí genetických algoritmů jsme schopni nalézt řešení velmi blízké tomu optimálnímu pro 100 měst za méně než minutu. postupovat budeme následovně:

1. Nejprve si vytvoříme množinu cest takových, že při každé z nich projedeme všechna města a počáteční a koncové body těchto cest si budou rovny. Tato množina bude naší počáteční populací.

2. Potom některou z metod selekce vybereme z počáteční populace dva vhodné jedince (rodiče) a křížením vytvoříme dvě nové cesty (potomky).

3. Abychom zabránili tomu, že budou všichni jedinci nové generace vypadat stejně, aplikujeme na ně mutaci.

4. Těmito dvěma novými (a už i kratšími) cestami nahradíme dvě delší cesty v počáteční generaci.

5. Body 2.-4. opakujeme, dokud není splněno zastavovací kritérium.

3.3.2 Problém N dam

Jedná se o klasický kombinatorický problém, kdy máme umístit N dam na šachovnici NxN tak, aby se vzájemně neohrožovaly. Zřejmě tedy musí být každá z dam v jiném řádku a v jiném sloupci šachovnice. Všechna řešení tohoto problému tedy mohou být reprezentována pomocí N-tice $(q_1, ..., q_N)$, která je permutací N-tice $(1, ..., N)$. Pozice čísla v této N-tici odpovídá sloupci, ve kterém se královna nachází, zatímco hodnota představuje její řádek. Pomocí této reprezentace, řešíme problém tak, že chceme eliminovat "diagonální konflikty", protože ty řádkové a sloupcové jsme už na začátku vyloučili.
Postup je následující:

1. Nejprve si vygenerujeme řetězce N řádkových pozic pro každý sloupec. Tyto pozice reprezentují rozmístění dam na šachovnici. Tímto jsme vytvořili počáteční populaci N jedinců.

2. Každého jedince ohodnotíme pomocí fitness funkce a některou z metod selekce vybereme dva vhodné jedince, kteří půjdou do křížení, čímž nám vznikne nový jedinec.

3. S malou pravděpodobností na tohoto jedince aplikujeme mutaci.

4. Kroky 2. a 3. provádíme, dokud nejsou obměněny všechny řetězce původní populace.

5. Opakujeme kroky 2.-4. do chvíle, než získáme výsledný řetězec, který bude odpovídat správnému rozmístění dam.

4 Aplikace genetických algortimů na úlohu o batohu

V této kapitole si ukážeme, jak lze pomocí genetických algoritmů řešit úlohu batohu. Nejprve si zvolíme hmotnosti a ceny předmětů, počet genů jedince (počet předmětů), kapacitu batohu, počet jedinců v generaci, počet generací (neboli počet iterací, po kterých se proces ukončí) a pravděpodobnosti křížení a mutace.

4.1 Kódování jedinců

Při přípravě řešení problému batohu pomocí genetických algoritmů si nejprve musíme rozmyslet, jakým způsobem budeme kódovat jedince, v tomto případě kódujeme obsah batohu.
Pro řešení binárního problému batohu je výhodné si zvolit binární kódování, kde batoh bude řetězec jedniček a nul, kde jednotlivé položky budou rovny jedné, pokud v batohu obsaženy budou a nule, pokud se v batohu nenachází. Délka řetězce tedy odpovídá počtu položek, které můžeme vložit do batohu.

4.2 Počáteční populace

V naší úloze vygenerování počáteční populace znamená to, že si náhodně vygenerujeme D jedinců, tj. D různých obsahů batohu, s n složkami tak, že součet hmotností předmětů v každém z těchto obsahů nesmí překročit nosnost batohu C. Tedy získáme D binárních vektorů $\vec{x_1}, \ldots, \vec{x_D}$ takových, že

$$\vec{x_1} = (x_{11}, ..., x_{1n})$$
$$\ldots$$
$$\vec{x_D} = (x_{D1}, \ldots, x_{Dn}),$$

které zároveň musí splňovat

$$\sum_{j=1}^{n} w_j x_{ij} \leq C \text{ pro všechna } i = 1, \ldots, D.$$

4.3 Ohodnocení jedince

Jak už jsme si popsali v druhé kapitole, kvalitu jedince udává jeho hodnota fitness funkce. Pro problém batohu je fitness funkce jedince x ve tvaru:

$$f(x) = \sum_{i=1}^{n} p_i x_i,$$

kde p_i je cena i-té položky a hodnota x_i nám říká, zda se i-tá položka nachází v řetězci (v obsahu) x.

Ovšem i přesto, že jsme si při přípravě úlohy nastavili omezení, díky kterému součet hmostností jedinců počáteční populace nemůže překročit kapacitu batohu, se může stát, že nově vzniklí jedinci po křížení a mutaci tuto podmínku splňovat nebudou. Abychom takové řetězce vyřadili z množiny přípustných řešení, musí přijít na řadu jisté korekce. Dva způsoby si teď představíme.

4.3.1 Užití penalizační funkce

Nejprve si definujeme speciální funkci (*penalizační*), která zapříčiní jisté znehodnocení jedince. Tato funkce je kladná pro ty řetězce, které nesplňují podmínku nepřekročení kapacity. Pro řetězce, které patří do prostoru přípustných řešení, je nulová.

Kvalitu jedince tedy určíme pomocí fitness funkce v následujícím tvaru:

$$f(x) = (\sum_{i=1}^{n} p_i x_i) - Pen(x),$$

kde $Pen(x)$ je penalizační funkce.

Existuje více způsobů, jak tuto funkci zvolit, například:

1. $Pen(x) = \log_2(1 + r(\sum_{i=1}^{n} w_i x_i - C))$
2. $Pen(x) = r(\sum_{i=1}^{n} w_i x_i - C)$
3. $Pen(x) = (r(\sum_{i=1}^{n} w_i x_i - C))^2$

kde $r = \max_{i=1,\ldots,n}\{p_i/w_i\}$.

4.3.2 Užití opravného algoritmu

Opravný algortimus funguje tak, že nepřípustný řetězec x přemění na přípustný řetězec x' tím, že z něj bude postupně odebírat předměty (tzn. měnit jedničky na nuly), dokud se všechny zbylé věci v řetězci do batohu nevejdou. Pak se podle vztahu

$$f(x') = \sum_{i=1}^{n} p_i x'_i$$

spočítá kvalita nového jedince.
Odebírat předměty můžeme více různými způsoby, nejobvyklejšími z nich jsou:

1. **Náhodné vybírání** věcí, které odebereme
2. **Hladový princip**, který spočívá v tom, že si ke každému předmětu vytvoříme poměr cena/hmotnost, tyto poměry seřadíme do neklesající posloupnosti a pak odebíráme předměty od začátku této posloupnosti.

4.4 Selekce, křížení a mutace

Selekce

Turnajová metoda Z populace si náhodně vybereme k různých obsahů batohů a porovnáme jejich ceny. Vybereme obsah s nejvyšší cenou a turnaj provedeme ještě jednou bez tohoto vítězného obsahu. Získáme tak dva jedince s největšími fitness hodnotami, které pošleme ke křížení a k mutaci.

Vážená ruleta Určíme si ceny všech obsahů v populaci a sečteme je. Součet těchto hodnot (S) nám dá obvod rulety. Pro každý obsah si stanovíme jeho krajní body na ruletě. Jinými slovy i-tý obsah náleží intervalu $[c_{i-1}, c_i]$, $i = 1, ..., D$, kde $c_0 = 0$, $c_D = S$ (D udává počet jedinců v populaci) a $c_i = c_{i-1} + f_i$ (f_i je fitness funkce i-tého jedince), $i = 1, ..., D-1$. Náhodně vybereme číslo z intervalu $[0, S]$ a podle toho, do kterého intervalu toto číslo padne, vybereme jedince příslušného tomuto intervalu. Opět proces zopakujeme dvakrát, abychom získali dva rodiče.

Je zřejmé, že obsahy batohu s vyšší cenou budou mít tyto intervaly delší, a tudíž budou mít taky větší šanci být vybráni.

Křížení Pro tento problém si zvolíme jednobodové křížení pro binární kódování, se kterým jsme se seznámili v kapitole 3.1.3. Křížící bod určíme tak, že náhodně vygenerujeme číslo kb z intervalu $[0, n-1]$, kde n je počet předmětů, které máme k dispozici. Potomka pak tvoří prvních kb genů od prvního rodiče a zbylých $n-kb$ genů od rodiče druhého.
To, že jsme si jako pravděpodobnost křížení zvolili číslo 0.95, znamená, že 95% z nové generace vznikne křížením a zbylých 5% ze zkopíruje od předešlé generace.

Mutace Mutace nám zabrání, abychom "zamrzli" na lokálně nejlepším výsledku. Pravděpodobnost mutace $\frac{1}{n}$ nám říká, že zmutuje $(100\frac{1}{n})\%$ chromozomů. V úloze o batohu to znamená, že se s touto pravděpodobností změní obsah batohu neboli že se přidá či odebere předmět.

4.5 Nahrazení jedinců v populaci

U nově vzniklých jedinců si ověříme, zda splňují podmínku nepřekročení kapacity batohu a ohodnotíme si je fitness funkcí - čili zjistíme jejich cenu.
Dále si v původní populaci najdeme jedince, kteří mají nejmenší fitness hodnoty a tyto hodnoty porovnáme s cenami potomků. Pokud jsou hodnoty potomků větší než nejmenší hodnoty jedinců, pak jedince s těmito minimálními hodnotami v populaci nahradíme novými jedinci, které jsme získali při křížení a mutaci.

4.6 Konec procesu

Jako zastavovací kritérium použijeme počet generací, který jsme si přiblížili v kapitole 3.1.4. Při počátečním zadávání parametrů si určíme, kolikrát chceme, aby celý tento proces proběhl. Po posledním iteračním kroku je tedy proces u konce a my získáme výsledek, jimž je obsah batohu s nejvyšší cenou v poslední populaci.

5 Příklady

Příklad 5.1. *Na stole leží 5 předmětů, jejichž hmotnosti (v kg) a ceny (v Kč) jsou uvedeny v tabulce níže, a zloděj se má rozhodnout, které z nich ukrade. Chce tedy, aby jejich celková hodnota byla maximální, ale zároveň se mu musí vejít do batohu, který má nosnost 8 kg.*

Předměty	1	2	3	4	5
Hmotnosti	1	2	2	4	3
Ceny	2	5	4	7	5

Nejprve zkusí vybrat 3 různé obsahy batohu, které by nepřekročily jeho kapacitu (vytvoří si počáteční populaci se třemi jedinci) a určí si jejich hodnotu (jednička na i-tém místě znamená, že si v batohu je obsažen i-tý předmět).

$$x_1 = (00110) \text{ a } f_1 = 11$$
$$x_2 = (11001) \text{ a } f_2 = 12$$
$$x_3 = (10010) \text{ a } f_3 = 9$$

1. Vybere si 2 jedince (obsahy) s nejvyšší hodnotou a ty pošle ke křížení a mutaci. Obsahy s největší cenou jsou x_1 a x_2.

2. Použije jednobodové křížení, kde si náhodně jako křížící bod vybere trojku. To znamená, že první nový obsah bude mít první 3 předměty z x_1 a zbytek z x_2. Druhý nový obsah to bude mít naopak. Tedy

$$x'_1 = (00101)$$
$$x'_2 = (11010)$$

3. Mutace - zkusí přidat jeden předmět a ohodnotí si obsahy:

$$x''_1 = (01101) \text{ a } f''_1 = 13$$
$$x''_2 = (11010) \text{ a } f''_2 = 14$$

4. Najde si v počáteční populaci obsahů ty s nejmenší hodnotou a porovná je s novými obsahy. Pokud nové obsahy budou hodnotnější, nahradí ty původní těmi novými.
   V počáteční populaci mají nejnižší hodnoty obsahy x_1 a x_3.

$$f_3 \leq f_1'' \text{ - v populaci tedy vymění } x_3 \text{ za } x_1''$$
$$f_1 \leq f_2'' \text{ - v populaci tedy vymění } x_1 \text{ za } x_2''$$

Zlodějova nová populace je tedy ve tvaru:

$$x_1 = (11010) \text{ a } f_1 = 14$$
$$x_2 = (11001) \text{ a } f_2 = 12$$
$$x_3 = (01101) \text{ a } f_3 = 13.$$

Protože je pro zloděje lup za 14 Kč pořád málo, rozhodne se, že celý postup ještě jednou zopakuje, aby tak získal vyšší cenu.

1. Z nové populace si vybere dva obsahy s nejvyššími hodnotami, v tomto případě se jedná o x_1 a x_3, na které aplikuje křížení a mutaci.

2. Tentokrát si jako křížící bod zvolí dvojku. Potomci tedy budou vypadat:

$$x_1' = (11101)$$
$$x_3' = (01010)$$

3. Protože mutace probíhá vždy jen s malou pravděpodobností, v tomto případě se neprojeví. A tedy

$$x_1'' = x_1' \text{ a } f_1'' = 16$$
$$x_3'' = x_3' \text{ a } f_3'' = 12$$

4. V současné populaci mají nejmenší cenu obsahy x_2 a x_3. Teď tedy zloděj porovná jejich ceny s cenami nových obsahů.

$$f_2 \leq f_1'' \text{ - v populaci vymění } x_2 \text{ za } x_1''$$
$$f_3 \geq f_3'' \text{ - v populaci zůstane } x_3.$$

Populace je tedy ve tvaru:

$$x_1 = (11010) \text{ a } f_1 = 14$$
$$x_2 = (11101) \text{ a } f_2 = 16$$
$$x_3 = (01101) \text{ a } f_3 = 13$$

Závěr Pro zloděje už je kořist za 16 Kč dostatečná, a proto si do batohu vloží 1., 2., 3., a 5. předmět.

Poznámka 5.1. *Následující příklady řešíme pomocí programů* `gaknaproul.m` *(pro selekci metodou vážené rulety) a* `gaknaptour.m` *(pro selekci turnajovou metodou) na přiloženém CD.*

Příklad 5.2. *Máme k dispozici 15 předmětů, jejichž hmotnosti (v kg) a ceny (v Kč) jsou zapsány v tabulce. Tyto předměty chceme umístit do batohu, který má nosnost 100 kg tak, abychom tuto nosnost nepřekročili a zároveň, aby celková cena batohu byla co nejvyšší.*

Předměty	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hmotnosti	28	5	25	30	45	34	45	2	13	22	20	3	27	30	4
Ceny	4	29	19	33	17	44	17	37	41	8	4	4	38	33	39

Řešení Podíváme se, jak budou vypadat řešení, když budeme hýbat s pravděpodobností křížení, mutace a velikostí populace a pokud použijeme jako selekci metodu turnaje nebo váženou ruletu.

1. Zadáme si pravděpodobnost křížení $p_k = 0.95$ a pravděpodobnost mutace $p_m = 0$.

Velikost populace (počet jedinců v populaci) nastavíme na 20.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	184	2,4,8,10,14,15	189	1,6,8,9,13
5	184	2,4,8,10,14,15	189	1,6,8,9,13
10	184	2,4,8,10,14,15	189	1,6,8,9,13
20	184	2,4,8,10,14,15	201	2,7,8,9,13,15
30	184	2,4,8,10,14,15	232	2,6,8,9,12,13,15
40	184	2,4,8,10,14,15	232	2,6,8,9,12,13,15
50	184	2,4,8,10,14,15	232	2,6,8,9,12,13,15
75	184	2,4,8,10,14,15	232	2,6,8,9,12,13,15
100	184	2,4,8,10,14,15	232	2,6,8,9,12,13,15

Velikost populace si nastavíme na 50.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	202	2,3,4,8,9,11,15	190	2,6,8,10,14,15
5	202	2,3,4,8,9,11,15	190	2,6,8,10,14,15
10	202	2,3,4,8,9,11,15	190	2,6,8,10,14,15
20	202	2,3,4,8,9,11,15	198	6,8,9,12,14,15
30	202	2,3,4,8,9,11,15	198	6,8,9,12,14,15
40	202	2,3,4,8,9,11,15	198	6,8,9,12,14,15
50	202	2,3,4,8,9,11,15	221	2,8,9,12,13,14,15
75	202	2,3,4,8,9,11,15	232	2,6,8,9,12,13,15
100	202	2,3,4,8,9,11,15	232	2,6,8,9,12,13,15

2. Pravděpodobnost křížení $p_k = 0.99$, pravěpodobnost mutace $p_m = \frac{1}{n}$

Velikost populace - 20.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	183	2,4,8,10,12,14,15	159	1,2,9,11,12,13,15
5	183	2,4,8,10,12,14,15	176	7,8,9,12,13,15
10	183	2,4,8,10,12,14,15	188	1,2,8,9,13,15
20	183	2,4,8,10,12,14,15	198	1,2,6,8,9,12,15
30	184	2,9,11,13,14,15	221	2,4,8,9,12,13,15
40	184	2,9,11,13,14,15	221	2,4,8,9,12,13,15
50	184	2,9,11,13,14,15	221	2,4,8,9,12,13,15
75	192	2,8,9,11,12,13,15	221	2,4,8,9,12,13,15
100	192	2,8,9,11,12,13,15	221	2,4,8,9,12,13,15

Velikost populace - 50.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	195	6,8,12,13,14,15	188	2,8,10,12,13,14,15
5	195	6,8,12,13,14,15	188	2,8,10,12,13,14,15
10	195	6,8,12,13,14,15	188	2,8,10,12,13,14,15
20	195	6,8,12,13,14,15	188	2,8,10,12,13,14,15
30	195	6,8,12,13,14,15	188	2,8,10,12,13,14,15
40	195	6,8,12,13,14,15	198	4,6,8,9,12,15
50	195	6,8,12,13,14,15	201	2,3,4,6,8,15
75	199	2,3,8,12,13,14,15	221	2,8,9,12,13,14,15
100	221	2,8,9,12,13,14,15	221	2,8,9,12,13,14,15

3. Pravděpodobnost křížení $p_k = 0.7$, pravděpodobnost mutace $p_m = 0.05$.

Velikost populace - 20.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	188	2,8,9,11,13,15	161	4,8,9,10,12,13
5	188	2,8,9,11,13,15	161	4,8,9,10,12,13
10	192	8,9,11,13,14,15	161	4,8,9,10,12,13
20	198	2,8,9,10,11,13,15	188	2,8,9,12,13,15
30	203	6,8,9,11,13,15	196	2,8,9,10,12,13,15
40	203	6,8,9,11,13,15	196	2,8,9,10,12,13,15
50	203	6,8,9,11,13,15	205	2,5,8,9,12,13,15
75	203	6,8,9,11,13,15	205	2,5,8,9,12,13,15
100	203	6,8,9,11,13,15	205	2,5,8,9,12,13,15

Velikost populace - 50.

Generace	Ruleta	Vybrané předměty	Turnaj	Vybraný obsah
1	195	2,6,8,10,13,15	217	2,8,9,13,14,15
5	195	2,6,8,10,13,15	217	2,8,9,13,14,15
10	195	2,6,8,10,13,15	217	2,8,9,13,14,15
20	195	2,6,8,10,13,15	217	2,8,9,13,14,15
30	195	2,6,8,10,13,15	217	2,8,9,13,14,15
40	195	2,6,8,10,13,15	228	2,6,8,9,13,15
50	198	2,3,4,8,9,15	228	2,6,8,9,13,15
75	198	2,3,4,8,9,15	232	2,6,8,9,12,13,15
100	223	2,6,8,9,14,15	232	2,6,8,9,12,13,15

Závěr: Nejvyšší nalezená hodnota batohu je 232 Kč, přičemž jsme vybrali 2., 6., 8., 9., 12., 13., a 15. předmět.

Příklad 5.3. *V tomto příkladu budeme řešit problém součtu podmnožin. V tabulce máme dány hmotnosti 14 předmětů, které máme umístit do batohu. Chceme maximalizovat hmotnost batohu bez toho, aby překročila nosnost, která je 91 kg. Velikost populace si nastavíme na 20.*

Předměty	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Hmotnosti	41	34	21	20	8	7	7	4	3	3	12	11	8	8

Řešení Problém součtu podmnožin je klasický binární problém batohu, ve kterém pro každý předmět platí, že je hmotnost rovna ceně. Opět se podíváme, jak budou vypadat řešení při změně pravděpodobnosti křížení a mutace a při různých typech selekce a kdy dosáhneme úplného naplnění batohu.

1. Pravděpodobnost křížení $p_k = 0.7$, pravděpodobnost mutace $p_m = 0.05$

   Turnajová metoda

Generace	Hmotnost	Vybrané předměty
1	88	1,3,5,7,9,13
5	90	2,3,5,7,11,14
10	90	2,3,5,7,11,14
25	91	2,3,5,6,10,11
50	91	2,3,5,6,10,11
100	91	2,3,5,6,10,11

   Úplného naplnění batohu jsme dosáhli v 15. iteračním kroku.

   Vážená ruleta

Generace	Hmotnost	Vybrané předměty
1	90	1,4,5,7,9,10,14
5	90	1,4,5,7,9,10,14
10	91	1,4,5,9,12,14
25	91	1,4,5,9,12,14
50	91	1,4,5,9,12,14
100	91	1,4,5,9,12,14

   Batoh jsme zcela naplnili v sedmém iteračním kroku.

2. Pravděpodobnost křížení $p_k = 0.99$, pravděpodobnost mutace $p_m = 1/n$

   Turnajová metoda

Generace	Hmotnost	Vybrané předměty
1	88	1,3,7,10,13,14
5	88	1,3,7,10,13,14
10	91	2,4,6,10,12,13,14
25	91	2,4,6,10,12,13,14
50	91	2,4,6,10,12,13,14
100	91	2,4,6,10,12,13,14

   Batoh byl naplněn v osmém iteračním kroku.

   Vážená ruleta

Generace	Hmotnost	Vybrané předměty
1	86	2,4,6,7,8,9,10,14
5	90	1,4,5,7,9,12
10	90	1,4,5,7,9,12
25	91	1,4,5,7,9,11
50	91	1,4,5,7,9,11
100	91	1,4,5,7,9,11

   Naplnění batohu jsme dosáhli ve 13. kroku.

3. Pravděpodobnost křížení $p_k = 0.95$, pravděpodobnost mutace $p_m = 0$

   Turnajová metoda

Generace	Hmotnost	Vybrané předměty
1	89	1,3,8,9,11,14
5	89	1,3,8,9,11,14
10	90	2,3,5,8,9,11,14
25	91	2,3,5,11,13,14
50	91	2,3,5,11,13,14
100	91	2,3,5,11,13,14

   Batoh jsme naplnili v 17.kroku.

Vážená ruleta

Generace	Hmotnost	Vybrané předměty
1	90	1,4,6,7,10,11
5	90	1,4,6,7,10,11
10	90	1,4,6,7,10,11
25	91	4,5,6,7,8,9,10,11,12,13,14
50	91	4,5,6,7,8,9,10,11,12,13,14
100	91	4,5,6,7,8,9,10,11,12,13,14

Batoh byl úplně naplněn ve 14. iteračním kroku.

Závěr Jak můžeme vidět v jednotlivých tabulkách, existuje mnoho způsobů, jak maximálně naplnit batoh.

Poznámka 5.2. *Řešení tohoto problému lze velmi dobře využít například u společností rozvážejících zboží nebo třeba stěhovacích firem. V obou případech se totiž snaží maximálně naplnit své auto, aby zvládli odvézt co nejvíc věcí zároveň, ale nechtějí auto přetížit.*

Literatura

[1] VANÍČEK, Jiří, Martin PAPÍK a Robert PERGL. Teoretické základy informatiky. 1. vyd. Praha: Kernberg Publishing s.r.o., 2007. ISBN 978-80-903962-4-1

[2] Spears, William M., and Vic Anand. A study of crossover operators in genetic programming. Springer Berlin Heidelberg, 1991.

[3] KUMAR, Rakesh a JYOTISHREE. Blending Roulette Wheel Selection. International Journal of Machine Learning and Computing. 2012, č. 4, s. 365-370. DOI: 10.7763/IJMLC.2012.V2.146. Dostupné z: http://www.ijmlc.org/show-32-116-1.html

[4] WAN, WEN, and JEFFREY B. BIRCH. "An Improved Genetic Algorithm Using a Directional Search."

[5] PISINGER, David. Algorithms for Knapsack Problems. Copenhagen, Denmark, 1995. PhD. thesis. University of Copenhagen, Universitetsparken 1.

[6] MARTELLO, Silvano a Paolo TOTH. Knapsack problems: algorithms and computer implementations. New York: J. Wiley, 1990, xii, 296 p. ISBN 04-719-2420-2.

[7] Hristakeva, Maya, and Dipti Shrestha. "Different Approaches to Solve the 0/1 Knapsack Problem."Retrieved November 3 (2004): 2012.

[8] HAUPT, Randy L a Sue Ellen HAUPT. Practical genetic algorithms. 2nd ed. Hoboken: John Wiley, 2004, 253 s. ISBN 04-714-5565-2.

[9] RAZALI, Noraini Mohd a GERAGHTY, John. Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. Proceedings of The World Congress on Engineering. 2011, s. 1134-1139.

[10] OBITKO Marek. Introduction to Genetic Algorithms. VUT Brno, 1998. Dostupné z: http://www.obitko.com/tutorials/genetic-algorithms/

[11] PETERKA, Ivo. Genetické algoritmy. Praha, 1999. Diplomová práce. Matematicko-fyzikální fakulta Univerzity Karlovy

[12] MIČEK, David. Genetické algoritmy. Brno, 2009. Diplomová práce. Fakulta elektrotechniky a komunikačních technologií, VUT Brno.