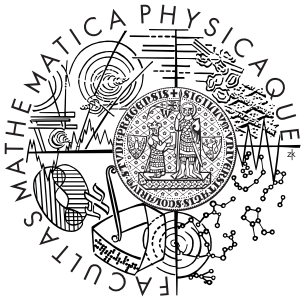
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jiří Krtek

Genetické algoritmy a jejich využití v optimalizaci

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce:	Prof. RNDr. Jaromír Antoch, CSc.
Studijní program:	Matematika
Studijní obor:	Pravděpodobnost a matematická statistika
Studijní plán:	Ekonometrie

2008

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce, Prof. RNDr. Jaromíru Antochovi, CSc., za veškeré připomínky k mé diplomové práci. Pomohly mi zejména při hledání směru, kterým bych se měl při vypracovávání této práce ubírat. Dále bych chtěl poděkovat svým rodičům za veškerou podporu při studiích, které se mi od nich dostalo.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 15.4.2008 Jiří Krtek

Obsah

Název práce: Genetické algoritmy a jejich využití v optimalizaci
Autor: Jiří Krtek
Katedra (ústav): Katedra pravděpodobnosti a matematické statistiky
Vedoucí diplomové práce: Prof. RNDr. Jaromír Antoch, CSc.
e-mail vedoucího: antoch@karlin.mff.cuni.cz

Abstrakt: V předložené práci se zabýváme odvětvím stochatických optimalizačních algoritmů, tzv. genetickými algoritmy. V první kapitole lze nalézt popis průběhu genetického algoritmu a hlavních operací určující směr prohledávání množiny přípustných řešení, tj. křížení a mutace. Nechybí modelový příklad, pomocí něhož čtenář všechny představené operace pochopí. Po části popisující různá vylepšení základního algoritmu, například Grayův kód, následuje nepříliš dlouhá kapitola věnovaná teorii genetických algoritmů. Ve třetí a zároveň poslední kapitole je nastolen skutečný optimalizační problém. K vyřešení tohoto problému jsme použili jednak teorii řízených Markovských řetězců pro modelování systému hromadné obsluhy, jednak genetické algoritmy k nalezení optimálního řešení. Optimální řešení jsme hledali i pomocí specializovaného algoritmu. Oba přístupy k hledání optima jsou v závěru této kapitoly zhodnoceny. Veškeré výpočty byly implementovány v jazyce Fortran.

Klíčová slova: Genetické algoritmy a jejich aplikace.

Title: Genetic algorithms and their utilization in optimization
Author: Jiří Krtek
Department: Department of Probability and Mathematical Statistics
Supervisor: Prof. RNDr. Jaromír Antoch, CSc.
Supervisor's e-mail address: antoch@karlin.mff.cuni.cz

Abstract: In the present work we deal with a branch of stochastic optimization algorithms, so called genetic algorithms. In the first chapter we can find description of a run of the genetic algorithm and the main operations which route searching of a feasible solution set, i.e. crossover and mutation. There is not absent a simple example, whereon reader can make sense of the presented operations. There is a short chapter devoted to theory of genetic algorithms which follows section describing various improvements of the basic algorithm, e.g. the Gray code. A real optimization problem is introduced in the third and also the last chapter. We have solved it using the theory of Markov decision processes for modeling a queuing system and by using genetic algorithms for finding optimum. We have also looked for optimum via a specialized algorithm. Both approaches are compared in the end of this chapter. All calculations have been implemented in the Fortran language.

Keywords: Genetic algorithms and their application.

Úvod

Genetický algoritmus (dále jen GA) je v současné době nejpoužívanějším evolučním algoritmem. Evoluční algoritmy patří mezi stochastické optimalizační algoritmy, které jsou inspirovány Darwinovou teorií evoluce. Ta byla publikována roku 1858 a zakládá se na jednoduché myšlence přirozeného výběru, že nejlépe přizpůsobení jedinci přežívají a rozmnožují se. Z jejich potomků se opět prosadí jen ti nejlépe přizpůsobení – nejsilnější atd. Tím by se měl daný druh neustále vyvíjet, zdokonalovat. Právě tuto myšlenku se snaží evoluční algoritmy přenést do světa matematiky.

Mějme nějakou funkci, již chceme optimalizovat na nějaké množině. Tato funkce není často moc „pěkná“, např. není hladká, nebo není ani spojitá. Množina, na níž danou funkci optimalizujeme, také nemusí patřit mezi „hezké“, např. může mít diskrétní charakter atd. V takových případech selhává analytický přístup. Může se použít nějaký vysoce sofistikovaný numerický postup nebo aplikovat něco z teorie optimalizace. Když všechny tyto přístupy selhávají, můžeme hledat optimum pomocí některého ze stochastických optimalizačních algoritmů, které jsou ceněny pro jejich rychlost a značnou robustnost. Můžeme je de facto použít skoro na každý optimalizační problém. Často sice nenajdou optimální řešení, ale v naprosté většině případů při jejich použití získáme velmi dobré suboptimální řešení. Samozřejmě lze stochastické optimalizační algoritmy kombinovat s ostatními zmíněnými přístupy, např. se můžeme pomocí GA přiblížit k optimu a následně zvolit numerický přístup.

Všechny evoluční algoritmy pracují s nějakou podmnožinou množiny všech prvků, přes kterou chceme danou funkci optimalizovat. Tato podmnožina se většinou nazývá populace a prvky se nazývají jedinci. Evoluční algoritmy postupují od populace k populaci s tím, že jedinci v nové populaci by měli být lepší (měli by poskytovat lepší řešení) než jedinci populace předešlé. Toto mají společné se specializovanými numerickými přístupy, které také s každou další iterací zlepšují dosažené řešení. Avšak zde je výběr jedinců nové populace, který probíhá na základě populace předchozí a různých pravidel řídících směr prohledávání množiny, značně ovlivněn náhodou.

Když se v přírodě rozmnožují dva jedinci, měli by se teoreticky spojit jejich nejsilnější chromozómy a vytvořit schopnějšího jedince, který bude mít silnější chromozómy (geny) než jeho rodiče. Pokud tomu tak není, jedinci mají tendenci vyhynout. Jak název napovídá, GA je značně ovlivněn genetikou. Jedince jeho populace můžeme také nazvat chromozómy. Pravidlo, podle kterého se vybírají chromozómy další populace (nebo také generace), je reprodukce zahrnující křížení a mutaci chromozómů generace předešlé. Křížením si chromozómy vymění geny, mutace mění gen daného chromozómu. Tyto operace se snaží

kopírovat své biologické předlohy, jsou sice značně zjednodušené, ale princip zůstává zachován.

GA se v současné době používá na NP–úplné a kombinatorické problémy nebo také k učení neuronových sítí.

Práce se zabývá problémem optimalizace funkcí na nějaké množině. Jelikož se bod, ve kterém leží maximum funkce f, shoduje s bodem, ve kterém leží minimum funkce $-f$, budeme se v celé práci zabývat pouze minimalizací.

V první kapitole si popíšeme průběh genetického algoritmu, operace reprodukce, křížení a mutace. Zavedeme si v ní pro GA určitou terminologii a rovněž si vysvětlíme pojmy související s GA. Nebude chybět ani modelový příklad. Další kapitola bude věnována teorii GA. Následovat bude kapitola, v níž si ukážeme aplikaci GA na praktický problém; v této části srovnáme výsledky dosažené pomocí GA s výsledky speciálně navrženého algoritmu. V závěru se pokusíme shrnout všechny naše poznatky a zhodnotit GA z různých úhlů pohledu.

V práci se objevují implementace různých částí GA v jazyku Fortran, u kterých nechybí komentáře a vysvětlivky. Případnému čtenáři to může značně zjednodušit chápání některých pojmů a operací.

Kapitola 1

Myšlenky genetického algoritmu

V této kapitole si představíme GA. Informace byly čerpány hlavně z knihy Kvasnička et al. (2000), dále pak z knih Goldberg (1989) a Reeves and Rowe (2002).

1.1 Základní pojmy genetického algoritmu

Mějme funkci

$$f : \{0,1\}^k \to \mathbb{R},\ k\ \in \mathbb{N}, \tag{1.1}$$

kterou chceme minimalizovat, tj. hledáme

$$\boldsymbol{\alpha}_{opt} = \underset{\boldsymbol{\alpha}\in\{0,1\}^k}{\arg\min}\, f(\boldsymbol{\alpha}). \tag{1.2}$$

Na tomto značně zjednodušeném problému si ukážeme, jak funguje genetický algoritmus.

Jak již bylo řečeno, genetický algoritmus je inspirován Darwinovou teorií evoluce, a proto názvosloví genetického algoritmu odpovídá různým biologickým termínům. Nyní si zavedeme a vysvětlíme několik nezbytných pojmů genetického algoritmu.

Prvek

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in \{0,1\}^k$$

nazveme jedincem nebo také chromozómem. Je to nějaký prvek prostoru $\{0,1\}^k$, na kterém minimalizujeme funkci f. Jednotlivé složky tohoto vektoru budeme v souladu s biologickou terminologií nazývat geny.

Množina

$$\mathcal{P} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_n\} \subset \{0,1\}^k$$

se nazývá populace jedinců (chromozómů). Její mohutnost je $n = |\mathcal{P}| \in \mathbb{N}$.

Funkce f reprezentuje prostředí, ve kterém žijí jedinci populace. Jedinec $\boldsymbol{\alpha}$ reprezentuje genotyp organismu a jeho hodnota účelové funkce $f(\boldsymbol{\alpha})$ reprezentuje fenotyp organismu. Mírou síly chromozómu je právě jeho funkční hodnota. Funkci f chceme minimalizovat, a

tak čím menší je hodnota $f(\boldsymbol{\alpha})$, tím je jedinec $\boldsymbol{\alpha}$ silnější. To nás vede k zavedení nové funkce $F : \{0,1\}^k \to \mathbb{R}^+$, tzv. *fitness*, jež hodnotí právě sílu jedince. Tato funkce musí splňovat následující podmínku:

$$\forall \boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathcal{P} : f(\boldsymbol{\alpha}_1) \leq f(\boldsymbol{\alpha}_2) \Rightarrow F(\boldsymbol{\alpha}_1) \geq F(\boldsymbol{\alpha}_2).$$

Ještě si zavedeme tzv. normalizovanou fitness F' a to následujícím vztahem:

$$F'(\boldsymbol{\alpha}) = \frac{F(\boldsymbol{\alpha})}{\sum_{\boldsymbol{\alpha}' \in \mathcal{P}} F(\boldsymbol{\alpha}')}.$$

Snadno nahlédneme, že pro *normalizovanou fitness* navíc platí:

1. $\sum_{\boldsymbol{\alpha} \in \mathcal{P}} F'(\boldsymbol{\alpha}) = 1.$
2. $0 < F'(\boldsymbol{\alpha}) < 1, \ \forall \boldsymbol{\alpha} \in \mathcal{P}.$

Vydělíme-li fitness jedince součtem všech fitness dané populace, dostaneme pravděpodobnost přežití daného jedince, tj. šanci jedince na to, že se reprodukuje. Normalizovaná fitness tedy plně odpovídá výběru jedince k reprodukci.

Možností jak volit fitness je celá řada. My si zde uvedeme dva způsoby, které jsou uvedené v Kvasnička et al. (2000). První způsob přiřazuje chromozómu s minimální hodnotou účelové funkce v rámci populace předem zvolenou maximální hodnotu fitness ($= F_{max}$) a naopak chromozómu s maximální hodnotou účelové funkce v rámci populace přiřadí předem zvolenou minimální hodnotu fitness ($= F_{min}$). Nejsilnějšímu jedinci tedy přiřadíme největší hodnotu fitness, čímž získá největší šanci reprodukovat se, předat své geny další generaci. Naopak nejslabšímu jedinci přiřadíme nejmenší hodnotu fitness, takže má minimální šanci stát se rodičem další generace. Hodnoty fitness pro ostatní chromozómy v populaci jsou lineárně interpolovány mezi těmito dvěma hraničními hodnotami, tj.

$$F(\boldsymbol{\alpha}) = \frac{F_{max} - F_{min}}{f_{min} - f_{max}} f(\boldsymbol{\alpha}) + \frac{f_{min} F_{min} - f_{max} F_{max}}{f_{min} - f_{max}},$$

kde

$$f_{min} = \min_{\boldsymbol{\alpha} \in \mathcal{P}} f(\boldsymbol{\alpha}), \quad f_{max} = \max_{\boldsymbol{\alpha} \in P} f(\boldsymbol{\alpha}).$$

Obvykle se volí $F_{max} = 1$ a $F_{min} = \varepsilon$, kde ε je malé kladné reálné číslo, např. $\varepsilon = 0{,}01$. F_{min} se nevolí rovno 0 proto, aby i ti nejhorší jedinci měli alespoň nějakou šanci, že budou vybráni jako rodiče další generace.

Druhý způsob, stejně jako ten první, chromozómu s minimální, respektive maximální hodnotou účelové funkce v rámci populace přiřadí maximální, respektive minimální hodnotu fitness. Hodnoty fitness ostatních chromozómů jsou ovšem v tomto případě rozmístěny s pevným krokem mezi těmito dvěma hraničními hodnotami podle hodnoty jejich účelové funkce. Nechť $Q = \{q_1, \ldots, q_n\}$ je taková permutace chromozómů populace $\mathcal{P}$, že platí

$$f(\boldsymbol{\alpha}_{q_1}) \leq f(\boldsymbol{\alpha}_{q_2}) \leq \cdots \leq f(\boldsymbol{\alpha}_{q_n}).$$

Potom fitness chromozómů populace $\mathcal{P}$ je dána vzorcem

$$F(\boldsymbol{\alpha}_{q_i}) = \frac{1}{1-n}[(1-\varepsilon)i + \varepsilon - n], \qquad i = 1, 2, \ldots, n. \tag{1.3}$$

V tomto případě se však může stát, že dva chromozómy se stejnou hodnotou účelové funkce budou mít rozdílné hodnoty fitness. Možností, jak se s tímto problémem vypořádat, je více.

Nabízí se přidat zcela přirozenou podmínku, která v případě shody hodnot účelové funkce přiřadí chromozómům stejnou hodnotu fitness, tj.

$$f(\boldsymbol{\alpha}_{q_{i-1}}) = f(\boldsymbol{\alpha}_{q_i}) \Rightarrow F(\boldsymbol{\alpha}_{q_i}) = F(\boldsymbol{\alpha}_{q_{i-1}}), \qquad i = 2, 3, \ldots, n. \tag{1.4}$$

Další možností je například přiřadit chromozómům se stejnou hodnotou účelové funkce pořadí náhodně. Rovnají-li se hodnoty účelové funkce chromozómů

$$\boldsymbol{\alpha}_{q_1}, \boldsymbol{\alpha}_{q_2}, \ldots, \boldsymbol{\alpha}_{q_l},$$

generujeme permutaci π čísel z množiny $\{1, 2, \ldots, l\}$, podle níž chromozómy seřadíme.

1.2 Průběh genetického algoritmu

Nyní si ukážeme, jak vlastně genetický algoritmus funguje. Mějme funkci $f : \{0,1\}^k \to \mathbb{R}$, $k \in \mathbb{N}$, kterou chceme minimalizovat, tj. hledáme $\boldsymbol{\alpha}_{opt}$, které splňuje (1.2).

Na začátku náhodně generujeme populaci $\mathcal{P}_0$ n chromozómů z $\{0,1\}^k$, tedy

$$\mathcal{P}_0 = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_n\} \subset \{0,1\}^k.$$

Co se rozumí pod pojmem náhodně generovaná populace? Tím míníme, že každý gen G každého chromozómu z $\mathcal{P}_0$ byl vybrán z alternativního rozdělení s $P[G = 1] = P[G = 0] = 1/2$. Označme

$$\boldsymbol{\alpha}_i = (\alpha_1^i, \alpha_2^i, \ldots, \alpha_k^i), \qquad i = 1, 2, \ldots, n.$$

Potom pro chromozómy z $\mathcal{P}_0$ tedy platí

$$\mathcal{L}(\alpha_j^i) = Alt(1/2), \qquad \forall i = 1, 2, \ldots, n \qquad \forall j = 1, 2, \ldots, k.$$

Všechny chromozómy populace $\mathcal{P}_0$ ohodnotíme pomocí normalizované fitness. Chromozóm z $\mathcal{P}_0$ je vybrán jako rodičovský chromozóm další generace s pravděpodobností rovnající se jeho normalizované fitness. Takto náhodně vybereme vždy dva chromozómy z $\mathcal{P}_0$, na které aplikujeme reprodukční operátor, tj. zkřížíme je mezi sebou a následně je ještě zmutujeme. Proces reprodukce uskutečníme s pravděpodobností P_{repro} blízké 1, např. $P_{repro} = 0{,}9$. Pokud k reprodukci nedojde, zůstávají chromozómy stejné. Operace křížení a mutace vysvětlíme níže. Takto postupujeme, dokud nemáme m nových chromozómů, kde $m \in \mathbb{N}, m \geq n$ je pevně zvolené sudé přirozené číslo. Sudé se m volí proto, že chromozómy vybíráme jako rodiče další generace po dvojicích (kvůli křížení). Z m nových chromozómů vybereme n s nejmenšími hodnotami účelové funkce, které vytvoří novou populaci $\mathcal{P}_1$.

Chromozómy populace $\mathcal{P}_1$ ohodnotíme normalizovanou fitness atd. Celý postup opakujeme, dokud není splněna nějaká ukončovací podmínka.

Touto ukončovací podmínkou může být např. maximální počet generací, které GA projde. Další možností je zastavit algoritmus, pokud jsou všichni jedinci populace stejní.

Slabí jedinci jsou ohodnoceni malou fitness, a tak se dále moc neprosadí, neboť se reprodukují (jsou vybrány jako rodiče další generace) jen s malou pravděpodobností. Na základě pravděpodobnosti výběru k reprodukci bychom tedy měli postupně napříč generacemi získávat stále silnější jedince.

Fortranovský kód výběru rodičovských chromozómů

Na tomto místě si ukážeme jednoduchý kód, který lze použít k výběru dvou jedinců jako rodičů další generace, tj. k operacím křížení a mutace.

```
function vyber_rodice(populace,psti)
 integer,intent(in)::populace(n,k) !populace, ze které vybíráme rodiče
 real,intent(in):: psti(n) !vektor nasčítaných normalizovaných fitness
 integer:: vyber_rodice(2,k) !výstup funkce
 integer:: i,j !pomocné proměnné
 real:: rnd01 !pomocná reálná proměnná

 do i=1,2
  call random_number(rnd01)
  !dosadí do rnd01 pseudonáhodné číslo z intervalu [0,1)
  if(rnd01 .lt. psti(1)) then
   vyber_rodice(i,:) = populace(1,:)
  else
   j=2
   do
    if ((psti(j-1) .lt. rnd01) .and. (rnd01 .lt. psti(j))) then
     vyber_rodice(i,:) = populace(j,:)
     exit
    end if
    ! .lt. znamená "<", z anglického "lower than"
    j = j + 1
   end do
  end if
 end do
end function vyber_rodice
```

Jednotlivé chromozómy jsou ve dvourozměrném poli `populace(n,k)` uspořádány samozřejmě do řádků. Znak „!“ v jazyku Fortran říká překladači, že do konce řádku bude následovat komentář. Klíčové spojení `intent(in)` znamená, že daná proměnná je pouze

vstupní, po skončení této funkce se s ní tedy dále nepracuje. Pole `psti(n)` vznikne z normalizované fitness chromozómů dané populace následujícím způsobem:

$$\begin{aligned} &\texttt{psti(1)} = F^{'}(1) \\ &\texttt{psti(i)} = \texttt{psti(i - 1)} + F^{'}(i) \qquad i = 2, 3, \ldots, n \end{aligned}$$

1.2.1 Křížení chromozómů

Operace křížení chromozómů napodobuje svou biologickou předlohu. Když v biologii zkříží-me dva jedince, měl by vzniknout silnější jedinec, neboť se při přenosu genetické informace prosadí nejsilnější geny. Zde slouží křížení jako heuristika k nalezení nového, lepšího řešení. Křížením v GA nemusí vždy vzniknout jen silnější jedinci, to ale uvidíme později.

Zde je na místě si vysvětlit, co se myslí pod pojmem heuristika. Je to postup používaný k nalezení řešení daného problému, který sice nezaručuje nalezení optimálního řešení a ani nezaručuje, že toto řešení najde v krátkém čase, ale obvykle rychle najde dostatečně přesné (dobré) řešení.

Zaveďme operátor křížení O_{cross} (značení je z anglického *crossover*). Operací křížení vzniknou ze dvou náhodně vybraných chromozómů dva nové chromozómy

$$(\boldsymbol{\alpha}^{'}, \boldsymbol{\beta}^{'}) = O_{cross}(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Ke křížení obvykle dochází s pevně zvolenou pravděpodobností P_{cross}. Pokud ke křížení nedojde, ponecháme dané rodičovské chromozómy nezměněné, tj. $(\boldsymbol{\alpha}^{'}, \boldsymbol{\beta}^{'}) = (\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Možností jak definovat operátor křížení je opět celá řada. Základním typem je tzv. jednobodové křížení, u kterého náhodně vybereme bod křížení $p \in \{1, 2, \ldots, k-1\}$. Do bodu křížení do nových chromozómů zkopírujeme obsah starých, za tímto bodem zkopírujeme do $\boldsymbol{\alpha}^{'}$ obsah $\boldsymbol{\beta}$ a do $\boldsymbol{\beta}^{'}$ obsah $\boldsymbol{\alpha}$, tj.

$$\alpha_i^{'} = \alpha_i, \ \beta_i^{'} = \beta_i, \qquad i = 1, 2, \ldots, p \tag{1.5}$$

$$\alpha_i^{'} = \beta_i, \ \beta_i^{'} = \alpha_i, \qquad i = p+1, p+2, \ldots, k. \tag{1.6}$$

$$\begin{matrix} (\alpha_1, \ldots, \alpha_p | \alpha_{p+1}, \ldots, \alpha_k) \\ (\beta_1, \ldots, \beta_p | \beta_{p+1}, \ldots, \beta_k) \end{matrix} \quad \rightarrow \quad \begin{matrix} (\alpha_1, \ldots, \alpha_p, \beta_{p+1}, \ldots, \beta_k) \\ (\beta_1, \ldots, \beta_p, \alpha_{p+1}, \ldots, \alpha_k) \end{matrix}$$

Obrázek 1.1: Grafické znázornění jednobodového křížení.

Zmíníme například ještě tzv. dvoubodové křížení, při kterém si chromozómy vymění informaci mezi dvěma náhodně zvolenými body křížení. Tento typ nebudeme blíže popisovat. Jeho grafické znázornění můžeme vidět na obrázku 1.2.

$$\begin{matrix}(\alpha_1,\ldots,\alpha_p|\alpha_{p+1},\ldots,\alpha_q|\alpha_{q+1},\ldots,\alpha_k) & & (\alpha_1,\ldots,\alpha_p,\beta_{p+1},\ldots,\beta_q,\alpha_{q+1},\ldots,\alpha_k)\\ & \rightarrow & \\ (\beta_1,\ldots,\beta_p|\beta_{p+1},\ldots,\beta_q|\beta_{q+1},\ldots,\beta_k) & & (\beta_1,\ldots,\beta_p,\alpha_{p+1},\ldots,\alpha_q,\beta_{q+1},\ldots,\beta_k)\end{matrix}$$

Obrázek 1.2: Grafické znázornění dvoubodového křížení s body křížení p a q.

Fortranovský kód jednobodového křížení

```
function krizeni(rodice)
 integer,intent(in):: rodice(2,k)
 integer:: krizeni(2,k)
 real:: rnd01
 integer:: p !pomocná proměnné - bod křížení

 krizeni = rodice
 call random_number(rnd01)
 p = 1 + int((k-1)*rnd01)
 !funkce int(x) vrácí dolní celou část čísla x
 krizeni(1,(p+1):k)=rodice(2,(p+1):k)
 krizeni(2,(p+1):k)=rodice(1,(p+1):k)
end function krizeni
```

1.2.2 Mutace chromozómů

Mutace vnáší stejně jako v přírodě do celého procesu evoluce něco nového. Zde slouží zejména k tomu, aby se algoritmus dostal z lokálních minim. Mutací jednoho či více genů chromozómu můžeme získat jedince, který se v populaci vůbec nenachází. Jeho následným zkřížením s ostatními pak může GA dostat zcela nový směr.

Zaveďme operátor mutace O_{mut}. Jeho aplikací na chromozóm $\boldsymbol{\alpha}$ vznikne nový chromozóm $\boldsymbol{\alpha}'$

$$\boldsymbol{\alpha}' = O_{mut}(\boldsymbol{\alpha}).$$

Nyní si ukážeme základní, a také nejrozšířenější typ mutace. Mějme pevně zvolenou pravděpodobnost mutace P_{mut}, jež udává pravděpodobnost mutace genu. Postupně generujeme k náhodných čísel $r_1,\ldots,r_k$ z intervalu $[0,1)$. Pokud je r_i menší než P_{mut}, změníme i–tý gen. V opačném případě ponecháme gen nezměněný. Pro $i=1,2,\ldots,k$ tedy platí

$$\alpha_i' = \begin{cases} 1-\alpha_i, & r_i < P_{mut} \\ \alpha_i, & r_i \geq P_{mut} \end{cases} \tag{1.7}$$

Fortranovský kód mutace

```
function mutace(chromozom)
 integer,intent(in):: chromozom(k)
```

```
 integer:: mutace(k)
 real:: rnd01
 integer:: i

 mutace = chromozom
 do i=1,k
  call random_number(rnd01)
  if (rnd01 .lt. P_mut) then
   mutace(i) = 1 - mutace(i)
  end if
 end do
end function mutace
```

Tento kód je implementací základního typu mutace. Proměnná `P_mut` plně odpovídá již popsané pravděpodobnosti mutace P_{mut}, v programu by mohla vystupovat např. jako globální konstanta.

1.2.3 Operace reprodukce

Operace reprodukce vznikne pouhým složením operací křížení a mutace. Reprodukujeme-li dva vybrané chromozómy, získáme dva nové. Zavedeme si pro ni podobné značení jako pro křížení a mutaci

$$(\boldsymbol{\alpha}', \boldsymbol{\beta}') = O_{repro}(\boldsymbol{\alpha}, \boldsymbol{\beta})$$

Fortranovský kód reprodukce

```
function reprodukce(rodice)
 integer,intent(in):: rodice(2,k)
 integer:: pomoc(2,k) !pomocné pole
 integer:: reprodukce(2,k)

 pomoc = krizeni(rodice)
 reprodukce(1,:) = mutace(pomoc(1,:))
 reprodukce(2,:) = mutace(pomoc(2,:))
end function reprodukce
```

Fortranovský kód genetického algoritmu

Nyní si ukážeme implementaci genetického algoritmu. Předpokládejme, že:

1. Máme definovanou nějakou funkci f, viz (1.1), a umíme ji vyčíslit $\forall \boldsymbol{\alpha} \in \{0,1\}^k$.
2. Máme implementovanou funkci, která vygeneruje počáteční populaci, nazvěme ji `generuj_populaci()`.

3. Máme implementovanou funkci, jejímž vstupem bude populace chromozómů a výstupem bude pole nasčítaných hodnot normalizované fitness

$$\left(F^{'}(\boldsymbol{\alpha}_1), F^{'}(\boldsymbol{\alpha}_1) + F^{'}(\boldsymbol{\alpha}_2), \ldots, \sum_{i=1}^{n} F^{'}(\boldsymbol{\alpha}_i) \right),$$

   tuto funkci nazveme `spocti_psti(populace)`.

4. Máme implementovanou funkci, která z populace m nových chromozómů vybere n chromozómů s nejmenšími hodnotami účelové funkce f. Takto vybrané chromozómy se stanou výstupem této funkce, již nazveme `vyber_nejlepsi(populace)`.

```
function geneticky_alg(max_pocet_generaci)
 integer,intent(in):: max_pocet_generaci
 integer:: geneticky_alg(n,k)
 integer:: i,j !pomocné proměnné
 integer:: P(n,k), Q(m,k) !populace a pomocná populace
 integer:: pomocne_chrom(2,k)
 real:: rnd
 real,parameter:: P_repro = 0.9
 real,parameter:: P_mut = 0.05

 P = generuj_populaci()
 do i=1,max_pocet_generaci !počítadlo generací
  prsti = spocti_psti(P)
  do j=1,m/2
   pomocne_chrom = vyber_rodice(P,prsti)
   call random_number(rnd)
   if (rnd .lt. P_repro) then
    Q((2*j-1):2*j,:) = reprodukce(pomocne_chrom)
   else
    Q((2*j-1):2*j,:) = pomocne_chrom
   end if
  end do
  P = vyber_nejlepsi(Q)
 end do
 geneticky_alg = P
end function geneticky_alg
```

Vstup funkce je počet generací, které má GA projít. Výstupem je poslední dosažená populace chromozómů.

1.2.4 Modelový příklad

Mějme funkci $f(x) = x^2 - 32x + 250$, jíž chceme minimalizovat na množině $D = \{0, 1, \ldots, 31\}$. Naší úlohou je tedy nalézt

$$x_{opt} = \arg\min_{x \in D} f(x).$$

Tento příklad samozřejmě nemusíme vůbec řešit pomocí GA, neboť každý hned vidí, že funkce f nabývá minima na D pro $x_{opt} = 16$. Avšak tím, že si na tomto příkladu ukážeme přechod od počáteční generace (populace $\mathcal{P}_0$) k prvním potomkům (populace $\mathcal{P}_1$), dojde k osvětlení průběhu GA.

Použijeme strategii šíření `čárka(4,8)`, což znamená, že populace $\mathcal{P}$ čítá 4 chromozómy a populace $\mathcal{Q}$, ze které se vybírá následující generace, obsahuje 8 chromozómů. Strategie šíření genetického algoritmu vysvětlíme později. Fitness budeme počítat podle vzorce (1.3) s úpravou (1.4), $F_{max} = 1$ a $F_{min} = \varepsilon = 0{,}05$. Použijeme jednobodové křížení (1.5) a mutaci pro tentokrát vynecháme.

Nejdříve musíme převést prvky množiny D do binárního kódu. K tomu nám bude stačit pětimístný binární kód:

$$\begin{aligned} 0 &= (00000) \\ 1 &= (00001) \\ 2 &= (00010) \\ \vdots &\quad\ \vdots \\ 31 &= (11111) \end{aligned}$$

Vygenerujeme náhodně počáteční populaci

$$P_0 = \{(11100), (00110), (11010), (01001)\}.$$

Spočítáme hodnoty účelové funkce pro všechny chromozómy z $\mathcal{P}_0$ a následně ještě jejich fitness, kterou znormalizujeme. Všechny tyto operace shrnuje tabulka 1.1. Vysvětlení některých kolonek tabulky:

- x udává celočíselnou hodnotu kódovanou chromozómem
- "kolikrát vybrán" udává, kolikrát byl daný chromozóm vybrán na základě normalizované fitness k reprodukci

číslo chromozómu	$\mathcal{P}_0$	x	$f(x)$	$F(x)$	$F'(x)$	kolikrát vybrán
1	(11100)	28	138	0,05	0,02	0
2	(00110)	6	94	0,683	0,285	2
3	(11010)	26	94	0,683	0,285	1
4	(01001)	9	43	1	0,41	5

Tabulka 1.1: Vyhodnocení populace $\mathcal{P}_0$.

Nyní přistoupíme k reprodukci. Spárujeme chromozómy podle pořadí jejich výběru. Pro každý pár generujeme náhodné číslo z intervalu $[0, 1]$. Pokud je toto číslo menší než P_{repro}, bude se daný pár reprodukovat, v opačném případě ponecháme chromozómy nezměněné. V případě reprodukce daného páru chromozómy jen zkřížíme; operaci mutace záměrně vynecháme, protože je jednoduchá na pochopení a znesnadnila by nám chápání křížení. Mutaci lze z algoritmu vynechat například položením $P_{mut} = 0$. Vzhledem k nepřítomnosti mutace můžeme vynechat i testování toho, zda-li dojde ke křížení. Položíme tedy $P_{cross} = 1$. Po operaci reprodukce shrnuté v tabulce 1.2, ve které je u rodičovských chromozómů naznačen bod dělení, vybereme z nových chromozómů do další generace čtyři s nejmenší hodnotou účelové funkce. Které chromozómy z populace Q půjdou dále do populace P_1, vidíme v tabulce 1.3. Výběrem a reprodukcí populace $\mathcal{P}_0$ jsme získali

$$P_1 = \{(01001), (01001), (01001), (01110)\}.$$

Celý postup bychom opakovali do té doby, než by byla splněna nějaká ukončovací podmínka.

č. chromozómu	chromozóm	v páru s	reprodukce	bod křížení	nové chromozómy
1	(010|01)	2	ano	3	(01001)
2	(010|01)	1	ano	3	(01001)
3	(01001)	4	ne	–	(01001)
4	(11010)	3	ne	–	(11010)
5	(0011|0)	6	ano	4	(00111)
6	(0100|1)	5	ano	4	(01000)
7	(01|001)	8	ano	2	(01110)
8	(00|110)	7	ano	2	(00001)

Tabulka 1.2: Vytvoření pomocné populace $\mathcal{Q}$.

č. chromozómu	$\mathcal{Q}$	x	$f(x)$	do $\mathcal{P}_1$?
1	(01001)	9	43	ano
2	(01001)	9	43	ano
3	(01001)	9	43	ano
4	(11010)	26	94	ne
5	(00111)	7	75	ne
6	(01000)	8	58	ne
7	(01110)	14	-2	ano
8	(00001)	1	219	ne

Tabulka 1.3: Vyhodnocení pomocné populace $\mathcal{Q}$.

1.3 Ukončovací podmínka

Už jsme si ukázali jak iterovat v GA, tedy jak postupovat od jedné generace k další. Ještě jsme si však neřekli, kdy GA ukončit. Samozřejmě nás hned napadne nejběžnější ukončovací podmínka podobných algoritmů a to zastavit GA, pokud projde určitý předem stanovený počet generací, tedy dosáhneme-li určitého počtu iterací.

Při této ukončovací podmínce musíme zvolit za maximální počet generací dostatečně velké číslo, abychom si byli jisti, že nedojde k ukončení GA příliš brzy. Dost často se však stává, že se celá populace naplní jedním typem chromozómu. GA tedy dosáhne minima ať už lokálního nebo globálního, ze kterého se posléze nedokáže dostat mutací, natož pak křížením. Lepší řešení je v takovém případě už téměř nedosažitelné, pokud existuje. Algoritmus však běží stále dál, dokud nedosáhne stanoveného počtu generací. Tím se výpočet značně a zbytečně prodlužuje. Kvůli tomuto neduhu se vymýšlejí jiné podmínky, které GA ukončí například tehdy, pokud jsou všechny chromozómy v populaci stejné.

Představíme si o něco rafinovanější ukončovací podmínku, která bude nicméně právě uvedené ukončovací podmínce dosti podobná. Označme nejsilnější chromozóm v populaci $\widehat{\boldsymbol{\alpha}}$. Nechť $0 < w(\widehat{\boldsymbol{\alpha}}) \leq 1$ je podíl chrommozómů v populaci, které jsou rovny $\widehat{\boldsymbol{\alpha}}$. Potom GA zastavíme, pokud $w(\widehat{\boldsymbol{\alpha}}) > w_{krit}$, kde w_{krit} je předem zvolené reálné číslo z intervalu $(0, 1)$. Obvykle je voleno $0{,}8 \leq w_{krit} \leq 0{,}95$; udává se jím, jak velkou část populace stačí aktuálně nejsilnějšímu chromozómu obsadit, aby tím ukončil GA. Takto bude GA ukončen v ten moment, kdy už je málo pravděpodobné, že bude nalezeno ještě lepší řešení.

Poslední ukončovací podmínka, kterou si zde uvedeme, bude o trochu složitější. Budeme jako obvykle pracovat s populací $\mathcal{P} = \{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_n\}$, kde $\boldsymbol{\alpha}_i = (\alpha_1^i, \alpha_2^i, \ldots, \alpha_k^i)$. Zavedeme si vektor $\mathbf{w} = (w_1, w_2, \ldots, w_k)$ následujícím způsobem

$$\mathbf{w} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\alpha}_i,$$

takže w_j vlastně vyjadřuje průměrnou hodnotu j–tého znaku (genu) chromozómů populace $\mathcal{P}$. Pro vektor $\mathbf{w}$ si zavedeme ještě tzv. parametr uspořádání

$$\chi(\mathbf{w}) = \frac{4}{k} \sum_{j=1}^{k} (w_j - 0{,}5)^2.$$

Tento parametr je v prvních krocích GA blízký nule, protože se v populaci vyskytuje mnoho různých chromozómů, a tak $w_j \approx 0{,}5$. Není těžké ukázat, že $\chi(\mathbf{w}) \leq 1$. S postupující evolucí se w_j přibližuje buď 0, nebo 1, a tak se $\chi(\mathbf{w})$ blíží jedné. GA ukončíme, jakmile platí $\chi(\mathbf{w}) > \chi_{krit}$, kde χ_{krit} opět volíme blízké jedné. K ukončení algoritmu dojde, když se většina populace shoduje ve většině genů, takže ze stávající populace vygenerujeme lepší řešení jen s velmi malou pravděpodobností.

1.4 Adaptace na různé problémy

Zatím jsme si ukázali, jak lze pomocí GA minimalizovat funkci f na množině $\{0,1\}^k$, viz (1.1). V modelovém příkladu z části 1.2.4 jsme zakódovali množinu $D = \{0,1,\ldots,31\}$ pomocí standartního binárního kódu, a tím jsme zadaný problém převedli na problém (1.1).

Zaměřme se nejprve na jednorozměrný diskrétní případ množiny přípustných řešení. Jediné, s čím se musíme vypořádat při převádění daného problému na problém (1.1), je právě kódování množiny přípustných řešení. Podobně, jako jsme postupovali v příkladu z části 1.2.4, se postupuje ve všech případech, kdy máme minimalizovat funkci na diskrétní množině. Seřadíme tedy prvky množiny přípustných řešení do posloupnosti a pomocí chromozómů kódujeme indexy prvků této posloupnosti.

Ve vícerozměrném případě množiny přípustných řešení můžeme například všechny prvky této množiny seřadit podle nějakého logického klíče do posloupnosti, čímž daný problém převedeme na jednorozměrný.

Mějme například množinu $D = \{0,1,\ldots,31\} \times \{0,1\}$, na níž máme minimalizovat nějakou funkci $f : D \to \mathbb{R}$. Seřadíme uspořádané dvojice z množiny D způsobem

$$(0,0) < (1,0) < \cdots < (31,0) < (0,1) < (1,1) < \cdots < (31,1).$$

Následně můžeme prvky této posloupnosti indexovat pomocí čísel z množiny $\{0,1,\ldots,63\}$, jejíž prvky již jednoduše převedeme na standartní šestimístný binární kód.

Dalším možným způsobem je kódovat každý rozměr zvlášť a výsledné chromozómy pro jednotlivé složky (rozměry) seřadit do jednoho chromozómu. Zvolíme-li druhý přístup, můžeme křížení chromozómů rozdělit na několik dílčích křížení. Jednotlivá dílčí křížení aplikujeme na „podchromozómy“ odpovídající jednotlivým složkám (rozměrům).

Vysvětleme si poslední zmiňovaný přístup na jednoduchém příkladu. Nechť

$$D = \{a_{11}, a_{12}, \ldots, a_{1m_1}\} \times \{a_{21}, a_{22}, \ldots, a_{2m_2}\} \times \cdots \times \{a_{q1}, a_{q2}, \ldots, a_{qm_q}\}$$

je nějaká diskrétní množina. Jak kódovat

$$\mathbf{x} = (x_1, x_2, \ldots, x_q) \in D,$$

kde $x_i \in \{a_{i1}, a_{i2}, \ldots, a_{im_i}\}$, $i = 1,2,\ldots,q$? Každou složku x_i vektoru $\mathbf{x}$ zakódujeme binárním řetězcem $\boldsymbol{\alpha}_i$ délky k_i a následně seřadíme všechny dílčí řetězce $\boldsymbol{\alpha}_i$ do jednoho binárního řetězce

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1 | \boldsymbol{\alpha}_2 | \ldots | \boldsymbol{\alpha}_q),$$

kde znak | symbolizuje oddělení jednotlivých „podchromozómů“.

Při křížení chromozómů potom můžeme aplikovat jednobodové křížení na každý „podchromozóm“ $\boldsymbol{\alpha}_i$ zvlášť.

Nechť je množina přípustných řešení jednorozměrná a spojitá, například uzavřený interval. Zvolíme měřítko podle požadované přesnosti a z množiny přípustných řešení vybereme

posloupnost bodů, která začíná levým krajem intervalu a ve které mají sousední body vzdálenost danou zvoleným měřítkem. Tím množinu přípustných řešení zdiskretizujeme, problém tedy převedeme na případ jednorozměrné diskrétní množiny.

Ve vícerozměrném případě spojité množiny přípustných řešení postupujeme naprosto analogicky. Zvolíme vhodnou ortogonální mřížku bodů z množiny přípustných řešení. Vzdálenost sousedních bodů v daném rozměru opět volíme na základě požadované přesnosti. Tím množinu zdiskretizujeme a postupujeme stejně jako v případě vícerozměrné diskrétní množiny.

Nalezením minima na ortogonální mřížce se většinou dostaneme blízko skutečnému minimu dané funkce na množině nezdiskretizované. Můžeme tak například kombinovat GA s nějakými dalšími specializovanými algoritmy. Pomocí GA se k minimu přiblížíme a jeho okolí pak prohledáme pomocí specializovaného algoritmu. To se většinou dělá kvůli úspoře času, neboť GA jsou poměrně rychlé.

1.5 Strategie šíření

V modelovém příkladu jsme si v krátkosti představili strategii šíření `čárka(n,m)`, při které se z n chromozómů populace $\mathcal{P}_t$ náhodně vybere dohromady $m \geq n$ rodičů další generace. Ti se reprodukují a z reprodukovaných chromozómů (populace $\mathcal{Q}$) se vybere n nejlepších chromozómů (s nejnižší hodnotou účelové funkce), které utvoří novou populaci $\mathcal{P}_{t+1}$.

Druhou možnou strategií šíření je strategie `plus(n+m)`. Ta probíhá skoro stejně až na jednu dosti podstatnou odlišnost. Nová populace $\mathcal{P}_{t+1}$, tedy n nejlepších jedinců, se nevybírá jen z populace $\mathcal{Q}$ (z nových jedinců), ale z $\mathcal{P}_t \bigcup \mathcal{Q}$. Tím je zaručeno, že populace chromozómů $\mathcal{P}$ bude s každou další generací lepší, nebo při nejhorším stejná.

Strategie `plus(n+m)` konverguje rychleji než strategie `čárka(n,m)`, což je způsobeno tzv. elitismem projevujícím se ve strategii `plus`. Nejlepší dosud nalezené řešení nelze v algoritmech vybavených elitismem vynechat na základě náhody (v GA reprezentované výběrem rodičovských chromozómů). Pouze jej můžeme nahradit ještě lepším řešením.

Mějme strategii `plus(n+m)`. Potom nejlepší řešení z populace $\mathcal{P}_t$, označme ho $\boldsymbol{\alpha}_{opt}^t$, nepostoupí do populace $\mathcal{P}_{t+1}$ jen v případě, že z nově reprodukovaných m chromozómů populace $\mathcal{Q}$ bude alespoň n lepších než $\boldsymbol{\alpha}_{opt}^t$. Jinak to není možné.

Rychlost konvergence lze ovlivnit i volbou n a m. Chceme-li dosáhnout rychlé lokální konvergence, zvolíme malé n, např. strategii `(5,100)`. Jestliže naopak hledáme globální minimum, pak zvolíme vyšší n, např. strategii `(15,100)`.

Největší šanci se reprodukovat mají nejsilnější jedinci, mají rovněž velkou šanci dostat se do další generace bez jakékoli změny. Při malém n a velkém m je vysoká pravděpodobnost, že se v přechodné populaci $\mathcal{Q}$ objeví více případů nejsilnějšího chromozómu. Ten potom rychle zaplní další generaci populace $\mathcal{P}$. Pokud není ukončovací podmínkou počet generací, dojde následně k ukončení algoritmu. V opačném případě ovšem nemá algoritmus skoro žádnou šanci najít nějaký nový směr prohledávání množiny přípustných řešení, neboť se

v celé populaci nachází jen nejsilnější chromozóm a jeho kopie.

Nejvíce chromozómů navštívíme, jestliže použijeme strategii šíření `čárka(n,n)` s velkým n. Nejsilnější chromozóm neopanuje celou populaci $\mathcal{P}$ tak snadno.

1.6 Inverzní operátor a Grayovo kódování

V této části uvedeme některá úskalí standartního kódování prohlédávané množiny (použitém v modelovém příkladu z podsekce 1.2.4), popíšeme však i různé způsoby řešení tohoto problému.

Ze všeho nejdříve si zadefinujeme Hammingovu vzdálenost. Hammingova vzdálenost ρ_H chromozómů $\boldsymbol{\alpha}$ a $\boldsymbol{\beta}$ se rovná počtu genů, v nichž se chromozómy liší. Formálně

$$\rho_H(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{k} |\alpha_i - \beta_i|$$

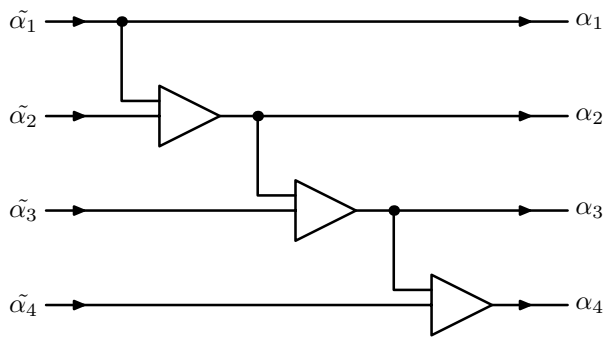
Nyní si ukážeme, v čem tkví úskalí standartního kódování. Představme si situaci, kdy kýžené globální minimum leží v bodě, jenž je kódován chromozómem $\boldsymbol{\alpha}_{opt} = (0100\ldots0)$. Chromozóm $\boldsymbol{\beta} = (0011\ldots1)$ reprezentuje ve standartním kódování souseda $\boldsymbol{\alpha}_{opt}$. Avšak abychom z chromozómu $\boldsymbol{\beta}$ dostali pomocí mutace chromozóm $\boldsymbol{\alpha}_{opt}$, museli bychom změnit všechny geny $\boldsymbol{\beta}$ kromě prvního. To je dáno velkou Hammingovou vzdáleností těchto dvou chromozómů, proto se tomuto jevu také říká Hammingova bariéra. Pravděpodobnost mutace P_{mut} je obvykle velmi malá, a tak je prakticky nemožné z $\boldsymbol{\beta}$ získat $\boldsymbol{\alpha}_{opt}$. Představme si dále, že populace $\mathcal{P}$, s níž pracujeme, obsahuje pouze chromozómy typu $(00111**\cdots*)$, kde $*$ může nabývat znaku 0 i 1. Potom křížením chromozómů z populace $\mathcal{P}$ se k $\boldsymbol{\alpha}_{opt}$ nedobereme a pomocí mutace jen s takřka mizivou pravděpodobností. Jak tedy získat globální minimum? Možností, jak vyřešit tento problém, je jako obvykle několik.

Můžeme zvýšit pravděpodobnost mutace, ale toto řešení nedoporučujeme. Směr prohledávání množiny, na které minimalizujeme, se pak příliš znáhodní a GA se bude podobat slepému prohledávání, čímž ztratí svou myšlenku i sílu.

1.6.1 Inverzní operátor

Další možností je přidat do GA inverzní operátor O_{inv}, který bude součástí reprodukčního operátoru. K inverzi chromozómu dochází s pravděpodobností P_{inv}, jež je velmi malá, obvykle $P_{inv} \leq 0{,}01$. Pakliže k inverzi dojde, generujeme bod inverze podobně jako bod křížení. Bod inverze a je tedy náhodné celé číslo z intervalu $[1, k-1]$. Až do tohoto bodu včetně je nový chromozóm stejný jako původní, za tímto bodem jsou všechny znaky nového chromozómu komplementární ke znakům původního chromozómu. Nechť $\boldsymbol{\alpha}' = O_{inv}(\boldsymbol{\alpha})$ a bod inverze je a, pak

$$\alpha_i' = \begin{cases} \alpha_i, & i = 1, 2, \ldots, a \\ 1 - \alpha_i, & i = a+1, a+2, \ldots, k \end{cases}$$



Obrázek 1.3: Grafické znázornění převodu Grayova kódování na standartní binární kód.

1.6.2 Grayův kód

Elegantní možností, jak obejít problém Hammingovy bariéry, je použít Grayovo kódování binárních řetězců. V Grayově kódu mají totiž chromozómy, které reprezentují sousední hodnoty, vždy Hammingovu vzdálenost rovnu 1.

Grayův kód vzniká transformací standartního binárního kódu pomocí algebraické operace `xor`, kterou budeme značit $\oplus$. Operace `xor` je definovaná jako $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$ a $1 \oplus 1 = 0$. Výsledek operace "xor" lze také interpretovat jako zbytek součtu binárních čísel po dělení dvěmi, tedy $x \oplus y = (x + y) \mod 2$.

Ukažme si, jak převádět standartní binární kód na Grayův a naopak. Nechť $\boldsymbol{\alpha}$ je chromozóm kódovaný standartně a $\tilde{\boldsymbol{\alpha}}$ je odpovídající chromozóm převedený do Grayova kódu. Nejdříve si uvedeme vztahy, s jejichž pomocí lze ze standartního kódu dostat kód Grayův:

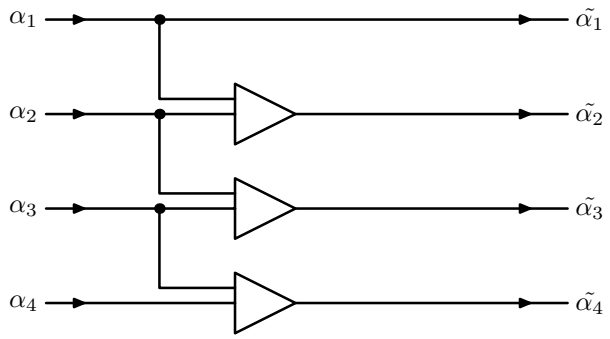
$$\begin{aligned}
\tilde{\alpha_1} &= \alpha_1 \\
\tilde{\alpha_2} &= \alpha_1 \oplus \alpha_2 \\
\tilde{\alpha_3} &= \alpha_2 \oplus \alpha_3 \\
&\vdots \quad\quad \vdots \\
\tilde{\alpha_k} &= \alpha_{k-1} \oplus \alpha_k
\end{aligned}$$

Právě popsaný převod je znázorněn na obrázku 1.3, v němž prázdný trojúhelník reprezentuje operaci `xor`.

Následující souhrn popisuje, jak z Grayova kódu dostat standartní kódování. Graficky je znázorněn na obrázku 1.4; prázdný trojúhelník opět reprezentuje operaci `xor`.

$$\begin{aligned}
\alpha_1 &= \tilde{\alpha_1} \\
\alpha_2 &= \tilde{\alpha_1} \oplus \tilde{\alpha_2} = \alpha_1 \oplus \tilde{\alpha_2} \\
\alpha_3 &= \tilde{\alpha_1} \oplus \tilde{\alpha_2} \oplus \tilde{\alpha_3} = \alpha_2 \oplus \tilde{\alpha_3} \\
&\vdots \quad\quad \vdots \\
\alpha_k &= \tilde{\alpha_1} \oplus \tilde{\alpha_2} \oplus \cdots \oplus \tilde{\alpha_k} = \alpha_{k-1} \oplus \tilde{\alpha_k}
\end{aligned}$$

Vraťme se k našemu modelovému příkladu a kódování, které jsme v něm použili. Převedeme si několik hodnot do Grayova kódu.



Obrázek 1.4: Grafické znázornění převodu standartního kódování binárních řetězců na Grayovo.

x	standartní kódování	Grayův kód
0	(00000)	(00000)
1	(00001)	(00001)
2	(00010)	(00011)
3	(00011)	(00010)
⋮	⋮	⋮
14	(01110)	(01001)
15	(01111)	(01000)
16	(10000)	(11000)
⋮	⋮	⋮
30	(11110)	(10001)
31	(11111)	(10000)

Tabulka 1.4: Převádění binárních řetězců do Grayova kódu.

Ještě si ukážeme, jak kódovat reálnou proměnnou z intervalu $[0, 1]$. Ke každému $x \in [0, 1]$ se dokážeme přiblížit s libovolnou přesností pomocí celých záporných mocnin čísla 2 následujícím způsobem

$$\begin{aligned} x &\approx \alpha_1 \tfrac{1}{2} + \alpha_2 \tfrac{1}{2^2} + \alpha_3 \tfrac{1}{2^3} + \cdots \\ \alpha_i &\in \{0, 1\}, \qquad i = 1, 2, \ldots \end{aligned}$$

Položíme $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots)$, čímž získáme standartní binární kód x. Např. čísla z intervalu $[\frac{1}{2}, \frac{9}{16})$ jsou kódovány pomocí řetězců $(1000\ldots)$, kdežto interval $[\frac{7}{16}, \frac{1}{2})$ je kódován pomocí řetězců $(0111\ldots)$. Pokud se blížíme zleva, tj. od řetězců ve tvaru $(0111\ldots)$, k číslu $\frac{1}{2}$, jež je kódováno pomocí řetězce (10000), narážíme znovu na problém Hammingovy bariéry.

Pomocí Grayova kódování ovšem můžeme zajistit, aby nebyly intervaly $[0, \frac{1}{16})$, $[\frac{1}{16}, \frac{1}{8})$ atd. v binárním kódování příliš vzdálené. To uděláme tím, že první čtyři znaky řetězce převedeme do Grayova kódu a zbylé ponecháme na svém místě. Takto můžeme volit různá dělení intervalu $[0, 1]$, přičemž počet podintervalů volíme vždy rovný mocnině čísla 2. S rostoucím počtem těchto podintervalů klesá Hammingova vzdálenost řetězců kódujících sousední hodnoty.

interval	standartní kódování	Grayův kód
$[0, \frac{1}{16})$	$(0000\ldots)$	$(0000\ldots)$
$[\frac{1}{16}, \frac{1}{8})$	$(0001\ldots)$	$(0001\ldots)$
$[\frac{1}{8}, \frac{3}{16})$	$(0010\ldots)$	$(0011\ldots)$
$\vdots$	$\vdots$	$\vdots$
$[\frac{7}{8}, \frac{15}{16})$	$(1110\ldots)$	$(1001\ldots)$
$[\frac{15}{16}, 1)$	$(1111\ldots)$	$(1000\ldots)$

Tabulka 1.5: Ukázka převodu standartního kódování intervalů do Grayova kódu.

Grayovo kódování binárních řetězců má tu nevýhodu, že při jeho použití musíme při každém vyhodnocování účelové funkce v daném chromozómu nejdříve chromozóm převést do standartního kódování. Toto je operace navíc, jež stojí procesor počítače určitý výpočetní čas. Jak bylo řečeno, provádí se tato operace pokaždé, když chceme vyhodnotit účelovou funkci v bodě, jenž je kódován chromozómem. Vzhledem k tomu, že k této operaci dochází v GA velice často, může čas strávený převáděním kódů sehrát velkou roli v celkovém výpočetním čase. Proto si musíme vždy důkladně rozmyslet, jestli chromozómy kódovat standartně či pomocí Grayova kódu.

Historie Grayova kódu

Grayův kód se často používá právě v GA kvůli odstranění Hammingovy bariéry, ale jak a proč vznikl?

Grayův kód vznikl dříve než genetické nebo evoluční algoritmy. Jmenuje se po vědci z Bellových laboratoří, Franku Grayovi, který ho použil ve svém patentu z roku 1947 a pojmenoval ho zrcadlový binární kód. Do té doby kód neměl žádné jméno. Jméno zrcadlový dostal proto, že vzniká pomocí posloupnosti zrcadlových operací aplikovaných na standartní binární kód. Pojmenování po Grayovi kód dostal kvůli tomu, že ho pod tímto názvem začali později používat ostatní lidé, tedy vědci a inženýři.

Franke Gray si nechal roku 1953 patentovat kód a výbojku, díky níž dokázal převést analogový signál na zrcadlový binární kód. To se s úspěchem použilo například při přenosu signálu v barevných televizích. Právě úspěch tohoto patentu způsobil, že Grayovo jméno se spojilo i s kódem.

Zrcadlový kód však používal v telegrafii již od roku 1878 francouzský inženýr Émile Baudot, který za svoji práci dokonce obdržel Řád cti. I v dnešní době dochází ke znovuobjevování zrcadlového binárního kódu. Například v roce 1998 pánové Klaschka J. a Mola F. vymysleli rovněž zrcadlový binární kód, jenž použili ve své práci. Nevěděli, že patent na něj je na světě už 45 let a že vymyšlen byl před 120 lety.

Grayův kód byl používán, aby nedocházelo k rušivým výstupům z elektromechanických přepínačů. Dnes se využívá hlavně k opravě chyb v digitální komunikaci.

Kód byl samozřejmě využívaný právě kvůli tomu, že dva řetězce v Grayově kódu, které reprezentují sousední hodnoty, mají vždy Hammingovu vzdálenost 1. Používáme-li například čtyřmístný binární čítač, pak při přechodu od hodnoty $7 = (0111)$ k hodnotě $8 = (1000)$ musíme vyměnit všechny čtyři bity řetězce. V praxi nelze změnit všechny čtyři bity ve stejný okamžik, a tak se tomu tak děje postupně po velmi krátkých časových úsecích. V jednom okamžiku vždy změníme jeden bit. Pokud ovšem dochází ke čtení hodnoty čítače blízko po takové změně hodnoty, může se stát, že se přečte jakákoli hodnota od $0 = (0000)$ do $15 = (1111)$.

Informace o historii Grayova kódu a jeho opakovaném objevování lze nalézt například v Klaschka (2004).

V závěru této části si uvedeme fortranovský kód převodu Grayova kódování binárních řetězců na standartní. Tento převod je v implementacích GA mnohem používanější než převod opačným směrem.

Fortranovský kód převodu Grayova kódování na standartní

```
function preved(gray)
 integer,intent(in):: gray(k)
 integer:: preved(k)
 integer:: i, pomoc

 preved(1) = gray(1)
 do i=2,k
  pomoc = preved(i-1) + gray(i)
  preved(i) = modulo(pomoc, 2)
  !funkce modulo(n,p) vrací zbytek po dělení
  !celého čísla n celým číslem p
  !výsledek má stejné znaménko jako p
 end do
end function preved
```

Kapitola 2

Teorie

V této kapitole si povíme něco málo o teorii GA. Představíme si koncept schémat (nebo také šablon) nacházející se prakticky v každé knize, jež pojednává o teorii GA. V závěru této kapitoly se dozvíme, za jakých podmínek najde GA optimální řešení s jednotkovou pravděpodobností.

Stále pracujeme s prostorem chromozómů $\{0,1\}^k$. Nejdříve si řekneme, co vlastně pojem schéma v GA znamená. *Schéma* je stejně jako chromozóm řetězec délky k, jehož geny ovšem mohou kromě hodnot 0 a 1 nabývat také volného znaku $*$, tedy schéma $\boldsymbol{\sigma} \in \{0,1,*\}^k$. Znak $*$ může nabývat jak hodnoty 0, tak 1. Je to vlastně takový univerzální symbol, řekli bychom žolík. Dále řekneme, že chromozóm $\boldsymbol{\alpha} \in \{0,1\}^k$ je příkladem schématu $\boldsymbol{\sigma}$, značení $\boldsymbol{\alpha} \in \boldsymbol{\sigma}$, právě tehdy, když platí následující podmínka:

$$\forall i \in \{1,2,\ldots,k\} : \sigma_i \in \{0,1\} \Rightarrow \alpha_i = \sigma_i.$$

Příkladem schématu $\boldsymbol{\sigma} = (*01**1101**)$ může být např. chromozóm $\boldsymbol{\alpha} = (10100110110)$.

Definujeme množinu $I(\boldsymbol{\sigma})$ všech příkladů schématu $\boldsymbol{\sigma}$ vztahem

$$I(\boldsymbol{\sigma}) = \{\boldsymbol{\alpha} : \boldsymbol{\alpha} \in \boldsymbol{\sigma}\}.$$

Pro naše modelové schéma $\boldsymbol{\sigma} = (*01**1101**)$ obsahuje množina $I(\boldsymbol{\sigma})$ celkem 32 chromozómů.

$$I(\boldsymbol{\sigma}) = \{(00100110100), (00100110101), \ldots, (10111110111)\}$$

Obsahuje-li schéma $\boldsymbol{\sigma}$ l volných znaků, $0 \leq l \leq k$, pak počet prvků množiny $I(\boldsymbol{\sigma})$ je roven 2^l.

Dále definujeme délku schématu $\delta(\boldsymbol{\sigma})$

$$\delta(\boldsymbol{\sigma}) = \max\{i : \sigma_i \in \{0,1\}\} - \min\{i : \sigma_i \in \{0,1\}\}.$$

Délka schématu $\boldsymbol{\sigma}$ je největší vzdálenost mezi dvěma znaky, které nejsou volné. Délka našeho modelového schématu $\boldsymbol{\sigma} = (*01**1101**)$ je 7, $\delta(\boldsymbol{\sigma}) = 7$.

Řád schématu $o(\boldsymbol{\sigma})$ je počet znaků schématu $\boldsymbol{\sigma}$ rovných 0 nebo 1, tedy

$$o(\boldsymbol{\sigma}) = |\{i : \sigma_i \in \{0, 1\}\}|.$$

Řád našeho modelového schématu je 6.

Implicit parallelism

Jestliže pracujeme s populací o n chromozómech, pak GA projde během každé generace $\mathcal{O}(n^3)$ schémat. Na tomto místě si naznačíme důkaz tohoto Hollandova tvrzení, kterému se v angličtině říká „implicit parallelism“. Budeme postupovat podobně jako v Goldberg (1989).

Stále pracujeme s populací čítající n chromozómů délky k. Z výpočtu vyřadíme schémata, jejichž pravděpodobnost přežití je menší nebo rovna P_{prez}. Náš výpočet se tedy omezí jen na schémata, jejichž pravděpodobnost zániku P_{zanik} během procesu reprodukce s jednobodovým křížením a základním typem mutace je menší než $1 - P_{prez}$. Toto zjednodušení vede k tomu, že můžeme pracovat pouze s kratšími schématy s malým řádem, neboť dlouhá schémata a schémata s vysokým řádem přežívají jen s malou pravděpodobností. Budeme tedy zkoumat jen schémata s délkou $\delta < P_{zanik}(k - 1) + 1$, $\delta \in \mathbb{Z}, \delta \geq 0$. Jako δ_s označíme největší přirozené číslo splňující předchozí nerovnost, tedy největší uvažovanou délku schémat. Pod pojmem schémata délky δ budeme rozumět všechna schémata splňující předchozí nerovnost.

Nejdříve spočítáme, kolik různých schémat délky δ, tedy délky δ_s a kratších, je obsaženo v daném chromozómu. Mějme chromozóm $\boldsymbol{\alpha}$ délky k a „okénko“ délky δ_s začínající na první pozici tohoto chromozómu. Nejdříve spočítáme, kolik schémat délky δ se nachází na tomto „okénku“. Na každé pozici z tohoto „okénka“ může být buď skutečná hodnota chromozómu $\boldsymbol{\alpha}$, nebo žolík. Na pozice mimo toto „okénko“ doplníme žolíky, abychom nepřekročili maximální uvažovanou délku schémat δ_s. V rámci tohoto „okénka“ tedy existuje celkem 2^{δ_s} schémat délky δ. Posuneme „okénko“ o jednu pozici doprava a ponecháme hodnotu posledního znaku v tomto „okénku“ rovnu skutečné hodnotě chromozómu $\boldsymbol{\alpha}$. Znakům mimo toto „okénko“ opět přiřadíme hodnotu žolíku. Na ostatních pozicích mohou být opět buď skutečné hodnoty $\boldsymbol{\alpha}$, nebo žolíci. V rámci tohoto „okénka“ tedy existuje celkem $2^{\delta_s - 1}$ schémat délky δ. Takto můžeme „okénko“ posunout celkem $(k - \delta_s)$–krát. V rámci chromozómu $\boldsymbol{\alpha}$ tedy existuje celkem

$$2^{\delta_s} + (k\ - \delta_s)2^{\delta_s - 1} = (k\ - \delta_s + 2)2^{\delta_s - 1} \tag{2.1}$$

různých schémat délky δ.

Kdybychom chtěli nadhodnotit odhad počtu různých schémat v celé populaci chromozómů, stačí vzorec (2.1) vynásobit počtem chromozómů v populaci n. Ve velké populaci se určitě budou opakovat krátká schémata nebo schémata malého řádu, proto je tento odhad skutečně nadhodnocený. Položme $n = 2^{\delta_s/2}$ a předpokládejme, že počet schémat řádu $\delta_s/2$

a vyššího je menší, nebo roven jedné. Schémata vyššího řádu jsou v populaci zastoupena ve výrazně menším počtu, proto není tento předpoklad až tak omezující. Všimněme si, že počet schémat v populaci z hlediska jejich řádu má binomické rozdělení. Počet schémat řádu menšího než $\delta_s/2$ je rovný počtu schémat řádu vyššího než $\delta_s/2$. Spočítáme-li pouze schémata řádu $\delta_s/2$ a vyššího, dostáváme dolní odhad počtu schémat v populaci ve tvaru

$$n_s \geq n(k\ -\delta_s+2)2^{\delta_s-2},$$

kde n_s značí právě počet schémat v populaci. Dosadíme-li $n = 2^{\delta_s/2}$ podle předpokladu na velikost populace, ihned dostáváme

$$n_s \geq (k\ -\delta_s+2)2^{3\delta_s/2-2} = \frac{k\ -\delta_s+2}{4}n^3.$$

Vidíme, že $n_s = cn^3 = \mathcal{O}(n^3)$, kde c je konstanta. Tím jsme dokončili náznak důkazu tvrzení nesoucí název „implicit parallelism“.

Toto tvrzení vlastně říká, že GA vyhodnotí během každé generace krom n chromozómů ještě asi n^3 schémat, což se děje někde v pozadí algoritmu. Nemusí se tedy uchovávat žádné další informace atd.

Pravděpodobnost přežití schémat a jejich počet v další generaci

Nyní se podíváme na pravděpodobnost přežití schématu v populaci $\mathcal{P}$ a na odhad počtu příkladů schématu v populaci. Zde budeme postupovat stejně jako v Kvasnička et al. (2000).

Začneme jednodušší částí. Jestliže dojde k aplikování mutace na schéma $\boldsymbol{\sigma}$, pak pravděpodobnost přežití tohoto schématu je rovna

$$(1 - P_{mut})^{o(\boldsymbol{\sigma})}. \tag{2.2}$$

Je to pravděpodobnost, že znaky, které neodpovídají volným symbolům, zůstanou nezměněné. Počet takových znaků je roven řádu schématu $o(\boldsymbol{\sigma})$ a pravděpodobnost, že znak nebude mutací změněn, je $(1 - P_{mut})$. Spojením těchto dvou jednoduchých poznatků dostáváme rovnou vzorec (2.2). Pravděpodobnost mutace P_{mut} bývá navíc malé kladné reálné číslo. Zanedbáme-li její druhé a vyšší mocniny, zjednodušší se nám vzorec (2.2) na

$$1 - o(\boldsymbol{\sigma})P_{mut}. \tag{2.3}$$

Jestliže dané schéma vstoupí do procesu křížení, pak je výpočet pravděpodobnosti přežití tohoto schématu o něco složitější. Pokud dané schéma zkřížíme s jiným schématem, pak pro pravděpodobnost zániku tohoto schématu platí vzorec

$$P_{cross}\frac{\delta(\boldsymbol{\sigma})}{k-1},$$

což je pravděpodobnost, že dojde ke křížení a bod křížení bude ležet mezi nějakými dvěma symboly schématu $\boldsymbol{\sigma}$ různými od $*$. Zkřížíme-li schéma s chromozómem z dané populace, jenž je příkladem tohoto schématu, pak toto schéma přežívá automaticky. Počet takových chromozómů označíme $N(\boldsymbol{\sigma}) = |I(\boldsymbol{\sigma}) \bigcap \mathcal{P}|$. Podíl chromozómů v populaci, které nejsou příkladem schématu $\boldsymbol{\sigma}$, je $\left(1 - \frac{N(\boldsymbol{\sigma})}{|\mathcal{P}|}\right)$ a jenom ty mohou aplikováním křížení zničit schéma $\boldsymbol{\sigma}$. Pravděpodobnost, že schéma $\boldsymbol{\sigma}$ přežije křížení, je tedy

$$1 - P_{cross}\frac{\delta(\boldsymbol{\sigma})}{k-1}\left(1 - \frac{N(\boldsymbol{\sigma})}{|\mathcal{P}|}\right). \tag{2.4}$$

Vynecháme-li ve vzorci (2.4) člen $(1 - \frac{N(\boldsymbol{\sigma})}{|\mathcal{P}|})$, získáme pro schéma $\boldsymbol{\sigma}$ vzorec nezávislý na ostatních schématech

$$1 - P_{cross}\frac{\delta(\boldsymbol{\sigma})}{k-1}.$$

Spojením vzorců (2.2) a (2.4) získáme vzorec pro pravděpodobnost přežití schématu operace reprodukce

$$\left[1 - P_{cross}\frac{\delta(\boldsymbol{\sigma})}{k-1}\left(1 - \frac{N(\boldsymbol{\sigma})}{|\mathcal{P}|}\right)\right] \times (1 - P_{mut})^{o(\boldsymbol{\sigma})}. \tag{2.5}$$

Použijeme-li místo vzorce (2.2) vzorec (2.3) a zanedbáme-li ve vzorci (2.5) křížové součiny, dostáváme zjednodušený vzorec pravděpodobnosti přežití schématu

$$1 - P_{cross}\frac{\delta(\boldsymbol{\sigma})}{k-1} - P_{mut}o(\boldsymbol{\sigma}). \tag{2.6}$$

Následující řádky budou patřit počtu příkladů schématu $\boldsymbol{\sigma}$ v populaci $\mathcal{P}$. Přesněji řečeno, jak se mění počet příkladů daného schématu při přechodu od populace $\mathcal{P}_t$ k populaci $\mathcal{P}_{t+1}$. Nejdříve si však zavedeme dva nové pojmy.

Střední hodnotu fitness schématu $\boldsymbol{\sigma}$ v populaci $\mathcal{P}$ definujeme jako

$$F(\boldsymbol{\sigma}) = \frac{1}{N(\boldsymbol{\sigma})}\sum_{\boldsymbol{\alpha} \in I(\boldsymbol{\sigma}) \bigcap \mathcal{P}} F(\boldsymbol{\alpha}).$$

Střední hodnotu fitness chromozómů populace $\mathcal{P}$ definujeme následujícím způsobem

$$\bar{F} = \frac{1}{|\mathcal{P}|}\sum_{\boldsymbol{\alpha} \in \mathcal{P}} F(\boldsymbol{\alpha}).$$

Jestliže se při přechodu od generace t ke generaci $t+1$ omezíme pouze na výběr rodičů, tj. zanedbáme křížení i mutaci chromozómů, pak je počet příkladů schématu $N_{t+1}(\boldsymbol{\sigma})$ v populaci $\mathcal{P}_{t+1}$ určen pomocí počtu příkladů $N_t(\boldsymbol{\sigma})$ v populaci $\mathcal{P}_t$ následujícím způsobem

$$N_{t+1}(\boldsymbol{\sigma}) = N_t(\boldsymbol{\sigma})\frac{F_t(\boldsymbol{\sigma})}{\bar{F}_t}. \tag{2.7}$$

Nechť pro schéma $\boldsymbol{\sigma}$ platí pro všechny generace vztah

$$F_t(\boldsymbol{\sigma}) = \bar{F}_t \times (1 + c), \qquad \forall t \in \mathbb{Z},\ t \geq 0. \tag{2.8}$$

kde konstanta c je malé kladné, nebo velké záporné číslo ($|c| < 1$). Dosadíme-li vztah (2.8) do vzorce (2.7), dostáváme

$$N_{t+1}(\boldsymbol{\sigma}) = N_t(\boldsymbol{\sigma}) \times (1 + c). \tag{2.9}$$

Opakovaným použitím tohoto vztahu nakonec získáme

$$N_t(\boldsymbol{\sigma}) = N_0(\boldsymbol{\sigma}) \times (1 + c)^t. \tag{2.10}$$

V tomto zjednodušeném případě tedy počet příkladů daného schématu exponenciálně roste, nebo klesá v závislosti na tom, zda je konstanta c kladná, či záporná.

Naše úvahy o počtu příkladů daného schématu v populaci probíhaly za předpokladu, že vynecháme procesy křížení a mutace. Přidáme-li je do procesu reprodukce, dostaneme pro počet příkladů schématu v populaci o něco složitější vzorec, který vznikne spojením vzorců (2.5) a (2.7)

$$N_{t+1}(\boldsymbol{\sigma}) \approx N_t(\boldsymbol{\sigma}) \frac{F_t(\boldsymbol{\sigma})}{\bar{F}_t} \left[1 - P_{cross} \frac{\delta(\boldsymbol{\sigma})}{k-1} \left(1 - \frac{N(\boldsymbol{\sigma})}{|\mathcal{P}|}\right)\right] \times (1 - P_{mut})^{o(\boldsymbol{\sigma})}. \tag{2.11}$$

Použijeme-li místo vzorce (2.5) pro pravděpodobnost přežití schématu jeho jednodušší verzi (2.6), získáme

$$N_{t+1}(\boldsymbol{\sigma}) \approx N_t(\boldsymbol{\sigma}) \frac{F_t(\boldsymbol{\sigma})}{\bar{F}_t} \left[1 - P_{cross} \frac{\delta(\boldsymbol{\sigma})}{k-1} - P_{mut} o(\boldsymbol{\sigma})\right]. \tag{2.12}$$

Vzorec (2.12) je na rozdíl od vzorce (2.11) nezávislý na ostatních schématech. Získali jsme tedy odhad počtu příkladů daného schématu, jenž je nezávislý na ostatních schématech.

Na závěr této kapitoly si uvedeme dvě věty z teorie GA a povíme si ke každé pár slov.

Věta 2.1. *Nechť pro populaci $\mathcal{P}_t$ platí, že $N_t(\boldsymbol{\sigma})$ je počet chromozómů, které jsou příkladem schématu $\boldsymbol{\sigma}$. Nechť $F_t(\boldsymbol{\sigma})$ je střední hodnota fitness chromozómů, které jsou příkladem schématu $\boldsymbol{\sigma}$, a $\bar{F}_t$ je střední hodnota fitness všech chromozómů populace $\mathcal{P}_t$. Potom očekávaný počet chromozómů, které jsou příkladem schématu $\boldsymbol{\sigma}$ v následující generaci, je zdola omezený vztahem*

$$N_{t+1}(\boldsymbol{\sigma}) \geq N_t(\boldsymbol{\sigma}) \frac{F_t(\boldsymbol{\sigma})}{\bar{F}_t} \left(1 - P_{zanik}\right),$$

kde

$$P_{zanik} = P_{cross} \frac{\delta(\boldsymbol{\sigma})}{k-1} + P_{mut} o(\boldsymbol{\sigma}).$$

Pokud platí $F_t(\boldsymbol{\sigma}) > \bar{F}_t$, respektive $F_t(\boldsymbol{\sigma}) < \bar{F}_t$ a P_{zanik} je malé, pak z předešlé věty plyne, že očekávaný počet chromozómů, které jsou příkladem daného schématu, roste, respektive klesá s každou další generací. Pravděpodobnost zániku daného schématu v důsledku

operací křížení a mutace P_{zanik} je malá pro tzv. stavební kameny (building blocks), což jsou krátká schémata, tedy schémata s malým $\delta(\boldsymbol{\sigma})$. Tato věta nám vlastně říká, jak GA pracuje se zmíněnými základními kameny. Ty, které mají velkou střední hodnotu fitness, se v populaci udrží a jejich příklady se budou neustále množit. Naopak základní kameny s malou střední hodnotou fitness se nejspíš postupně z populace vytratí.

Konvergence GA

Další věta nám poví, za jakých podmínek GA nalezne globální minimum funkce f s jednotkovou pravděpodobností.

Věta 2.2. *Nechť je v GA posloupnost populací $\mathcal{P}_0, \mathcal{P}_1, \ldots$ monotónní, tj. pro každé $t \in \mathbb{Z}$, $t \geq 0$ platí*

$$\min_{\boldsymbol{\alpha} \in \mathcal{P}_{t+1}} f(\boldsymbol{\alpha}) \leq \min_{\boldsymbol{\alpha} \in \mathcal{P}_t} f(\boldsymbol{\alpha}),$$

potom GA asymptoticky pro $t \to +\infty$ nalezne globální optimum, tj.

$$\boldsymbol{\alpha}_{opt} = \lim_{t \to +\infty} \left(\arg\min_{\boldsymbol{\alpha} \in \mathcal{P}_t} f(\boldsymbol{\alpha}) \right).$$

Důkaz: Při dokazování této věty využijeme znalostí z teorie Markovských řetězců, kterou nalezneme například v Prášková and Lachout (2005).

Používáme jako obvykle GA s jednobodovým křížením a základním typem mutace. Nadále pracujeme s populací $\mathcal{P}$ o n chromozómech.

Na posloupnost populací $\{\mathcal{P}_t\}$ můžeme také nahlížet jako na n–rozměrný Markovský řetězec. Snadno zjistíme, že $\{\mathcal{P}_t\}$ je nerozložitelný, neboť mutace nám zajišťuje, že se z každého stavu dokážeme dostat do libovolného jiného s nenulovou pravděpodobností.

Zkonstruujeme nový Markovský řetězec $\{Y_t\}$, pro který platí

$$Y_t = \begin{cases} y_0, & \boldsymbol{\alpha}_{opt} \in \mathcal{P}_t, \\ \mathcal{P}_t, & \text{jinak.} \end{cases}$$

Podmínka na monotónnost populací se projeví tím, že stav y_0 řetězce $\{Y_t\}$ je absorbční. Zároveň je to jediný absorbční stav řetězce $\{Y_t\}$. Všechny ostatní stavy jsou tedy přechodné. Platí tedy

$$\lim_{t \to +\infty} \mathcal{P}[Y_t = y_0] = 1,$$

což je ekvivalentní s

$$\boldsymbol{\alpha}_{opt} = \lim_{t \to +\infty} \left(\arg\min_{\boldsymbol{\alpha} \in \mathcal{P}_t} f(\boldsymbol{\alpha}) \right).$$

□

Větu 2.2 lze nalézt například v Kvasnička et al. (2000). V důkazu jsme postupovali podobně jako v Hartl (1990).

Monotónnost populací GA dosáhneme např. pomocí strategie šíření `plus(n+m)`, při které jedinci s nejmenší hodnotou účelové funkce zůstávají v populaci, pokud nejsou nahrazeni lepšími. Monotónnosti rovněž dosáhneme, uchováváme-li v populaci nejsilnějšího jedince. Popišme si, jakým způsobem to lze udělat.

Označme nejlepšího jedince z populace $\mathcal{P}_t$ jako $\widehat{\boldsymbol{\alpha}}_t$. Jestliže v populaci $\mathcal{P}_{t+1}$ po reprodukci není přítomen $\widehat{\boldsymbol{\alpha}}_t$ a žádný jedinec z $\mathcal{P}_{t+1}$ není lepší, nebo alespoň stejně dobrý jako $\widehat{\boldsymbol{\alpha}}_t$, vyjmeme jednoho jedince z populace $\mathcal{P}_{t+1}$ a nahradíme jej pomocí $\widehat{\boldsymbol{\alpha}}_t$. Výběr jedince, který bude nahrazen, můžeme provádět různými způsoby. Můžeme například odstranit nejslabšího nebo vybrat některého náhodně.

Kapitola 3

Aplikace na praktický problém

V celé této kapitole se budeme odkazovat na práci Dvořák et al. (2007), na které jsem spolupracoval s mými kolegy z ročníku – Markem Dvořákem, Tomášem Maradou a Radkou Pickovou. Práci (projekt) jsme si zvolili v rámci povinného předmětu Seminář modelování v ekonomii. Tato práce je velmi rozsáhlá, a tak z ní vybereme pouze ty části, které jsou důležité pro uvedení problému a výpočet pomocí GA.

3.1 Popis problému

Ze všeho nejdříve si představme problém, který následně vyřešíme pomocí teorie řízených Markovských řetězců a GA. Na jaře minulého roku byla Matematicko–fyzikální fakulta Univerzity Karlovy v Praze oslovena vedením Poštovní spořitelny (dále jen PS), aby pro ně zpracovala problém optimalizace počtu jejích pracovníků a úrovně jejich dovedností.

Problém se týkal pracovníků z oddělení zpracování žádostí o spotřebitelské úvěry (dále jen SÚ) a pracovníků z oddělení zpracování žádostí o kreditní karty (dále jen KK). Politika PS je nastavena tak, že pracovníci musí zvládnout vyřídit alespoň 60 % žádostí o daný produkt do tří pracovních dnů a dalších 30 % žádostí od čtyř do osmi pracovních dnů, maximálně 10 % žádostí o daný produkt může být zpracováváno déle než osm pracovních dní. O produkty PS mohou zákazníci žádat na pobočkách České pošty a PS, což označujeme za tzv. Front Office (dále jen FO). Zpracování těchto žádostí probíhá v tzv. Back Office (dále jen BO). Doba, po kterou žádosti putují z FO do BO, se do časových limitů na zpracování žádostí daných politikou PS nepočítá. Na obě oddělení přichází žádosti o dané produkty PS. V určitých ročních obdobích přichází na dané oddělení mnohem více žádostí než v jiných, sledujeme tzv. špičku, v jiných obdobích zase přichází na dané oddělení žádostí mnohem méně, sledujeme tzv. sedlo. To je dáno sezónností a reklamními kampaněmi PS. PS nemůže nabírat a propouštět své zaměstnance podle toho, jaké je zrovna roční období nebo jestli zrovna probíhá reklamní kampaň. V době špiček tedy dané oddělení nestíhá žádosti zpracovávat, naopak v době sedel není na daném oddělení příliš mnoho práce. Obě oddělení SÚ a KK postupují při zpracovávání svého produktu velmi podobně, a tak se vedení PS rozhodlo vyškolit několik specializovaných pracovníků, kteří umějí zpracovávat oba produkty. Tyto pracovníky nazveme univerzály.

Naším úkolem bylo analyzovat data, která nám PS poskytla. Data se týkala právě zpracovávání daných dvou produktů, podrobnější popis přijde později. Dále jsme měli za úkol odhadnout rozdělení počtu příchozích žádostí, odhadnout rozdělení setrvání žádostí v systému a na základě poskytnutých dat vypočítat optimální počet zaměstnanců a úroveň jejich vyškolení.

3.2 Analýza dat

Nebudeme se zde dlouze rozepisovat o tom, jaká data jsme dostali a co jsme z nich zjistili. K samotnému výpočtu optimálního počtu zaměstnanců a úrovně jejich vyškolení toho totiž nebudeme mnoho potřebovat.

Musíme zde ovšem zmínit odhad rozdělení počtu příchozích žádostí, odhad rozdělení poměru schválených, zamítnutých, na doposlání čekajících a stornovaných žádostí za den a to, že ve výpočtu uvažujeme konstantní dobu zpracování žádostí o jednotlivé produkty.

Začněme konstantní dobou zpracování žádostí. Vedení PS nám na jedné ze schůzek řeklo, že délky jednotlivých úkonů procesu posuzování žádosti jsou normovány. Při posuzování žádosti o daný produkt PS má pracovník na každou činnost s tím spojenou určitý počet „normominut“. Proces posuzování žádostí i s „normominutami“ na jednotlivé úkony je možno vidět na obrázku 3.1.

Z diagramu na obrázku 3.1 jsme vypočítali, jakou dobu stráví pracovník oddělení SÚ nebo KK posuzováním jedné žádosti. Z tohoto diagramu je také vidět, že doba, jíž pracovník stráví posuzováním žádosti, která bude ve výsledku schválena, je odlišná od doby, kterou stráví při zpracovávání žádosti zamítnuté. To se týká SÚ i KK. Délky zpracování žádostí:

- SÚ
    1. schválená žádost ... 100 min
    2. zamítnutá žádost ... 35 min
- KK
    1. schválená žádost ... 50 min
    2. zamítnutá žádost ... 32 min

Právě kvůli rozdílným dobám zpracování schválených a zamítnutých žádostí jsme byli nuceni odhadnout rozdělení poměru schválených, zamítnutých, na doposlání čekajících a stornovaných žádostí za den. K modelování tohoto poměru jsme použili Dirichletovo rozdělení, jehož parametry jsme odhadli z dat. Jelikož jsme neznali dobu zpracování žádostí, které čekají na doposlání, rozhodli jsme se je přičíst ke schváleným. Modelovali jsme tedy poměr schválených, zamítnutých a stornovaných žádostí za den, a to zvlášť pro SÚ a KK. Parametry tvaru Dirichletova rozdělení pro zmíněný poměr byly odhadnuty metodou maximální věrohodnosti takto:

- parametry Dirichletova rozdělení pro SÚ ... (38,13; 24,16; 2,14)
- parametry Dirichletova rozdělení pro KK ... (7,70; 5,89; 1,38)

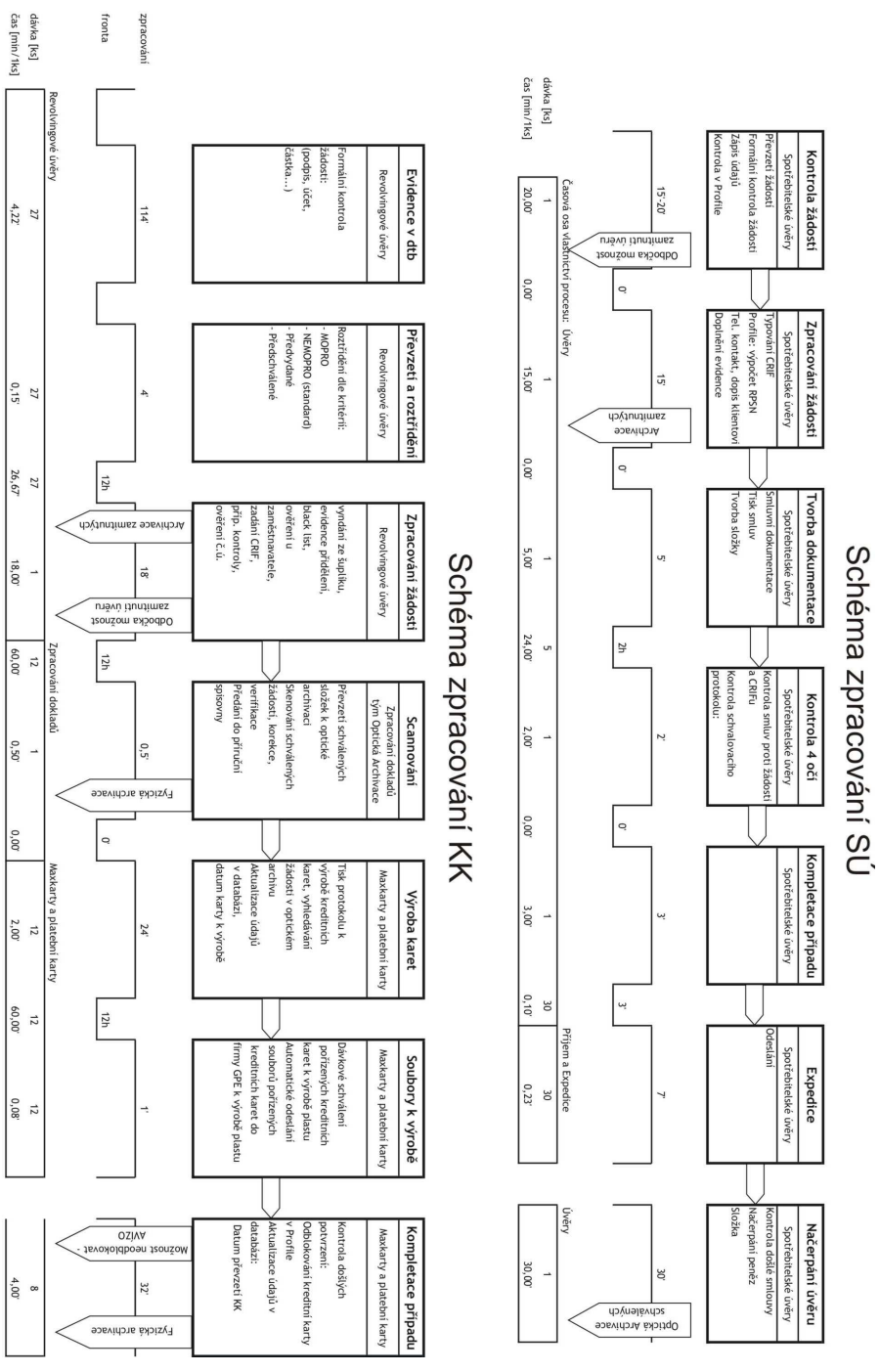
Nakonec jsme byli nuceni stornované žádosti rovněž přičíst ke schváleným. Z diagramu na obrázku 3.1 se totiž nedala vyčíst doba, kterou pracovník daného oddělení stráví zpracováváním žádosti, jež bude nakonec žadatelem stornována.

Když známe poměr schválených a zamítnutých žádostí a délku jejich zpracování („normominuty"), jsme schopni vypočítat, kolik žádostí dokáže jeden pracovník za den vyřídit. To je klíčové při zjišťování, kolik žádostí celkem odejde každý den ze systému při daném počtu zaměstnanců.

Při zkoumání rozdělení počtu příchozích žádostí na jednotlivá oddělení jsme museli vzít na vědomí, že by mělo být nezáporné a diskrétní. Mělo by být omezené, neboť určitě existuje nějaká horní mez počtu příchozích žádostí. Lidí je omezený počet a určitě nebude člověk žádat o více než jeden SÚ nebo KK za den. Uvažovali jsme však i neomezená rozdělení, neboť se s nimi většinou v modelu lépe pracuje. Uvažovali jsme nad různými rozděleními a nakonec jsme zvolili negativně binomické, které se při řešení problémů hromadné obsluhy hojně využívá. Parametry tohoto rozdělení jsme odhadli z poskytnutých dat pro obě oddělení zvlášť metodou momentů a metodou maximální věrohodnosti. Jelikož se počty příchozích žádostí na jednotlivá oddělení v průběhu roku značně lišily, rozhodli jsme se odhadnout parametry pro každý měsíc v roce také zvlášť.

Pro oddělení KK jsme neměli data pokrývající celý rok, nýbrž jen období od února do října včetně. Navíc se nám pomocí metody maximální věrohodnosti nepodařilo zkonstruovat použitelné odhady pro červen a srpen. Rozhodli jsme se proto pro použití odhadů metodou momentů. U měsíce srpna jsme měli problém i při použití metody momentů, neboť výběrový průměr vycházel větší než výběrový rozptyl, což při modelování pomocí negativně binomického rozdělení nesmí nastat. Tuto nesnáz jsme odstranili umělým zvýšením rozptylu. Pro zbylé tři měsíce, pro něž jsme data vůbec neměli, jsme se rozhodli použít nejvyšší hodnotu výběrového průměru i rozptylu a to kvůli tomu, abychom v našem výpočtu počty zaměstnanců spíše nadhodnocovali. Navíc v listopadu a prosinci přichází žádostí zpravidla spíše nadbytek.

V tabulce 3.1, respektive 3.2 můžeme vidět výběrové průměry a výběrové rozptyly počtu příchozích žádostí za daný měsíc na oddělení SÚ, respektive KK. Z nich pak pomocí metody momentů nebo metody maximální věrohodnosti vypočítáme parametry negativně binomického rozdělení. Výběrové průměry nalezneme ve sloupci $\overline{x}_i$, výběrové rozptyly ve sloupci s_i^2, parametry negativně binomického rozdělení nalezneme v dalších sloupcích. Horní index $^{(MM)}$ značí odhad vyrobený pomocí metody momentů, zatímco horní index $^{(ML)}$ značí odhad vyrobený pomocí metody maximální věrohodnosti.



Obrázek 3.1: Znázornění procesu zpracování produktů.

měsíc	$\overline{x}_i$	s_i^2	$\widehat{p}^{(MM)}$	$\widehat{r}^{(MM)}$	$\widehat{p}^{(ML)}$	$\widehat{r}^{(ML)}$
10/06	168,591	2186,634	0,077	14,084	0,083	15,283
11/06	253,476	3190,562	0,079	21,875	0,085	23,597
12/06	109,000	3493,222	0,031	3,511	0,034	3,849
1/07	85,318	379,656	0,225	24,731	0,202	21,566
2/07	103,100	174,516	0,591	148,841	0,623	170,103
3/07	120,909	223,706	0,540	142,213	0,560	153,682
4/07	117,600	492,253	0,239	36,914	0,257	40,605
5/07	216,381	3859,948	0,056	12,850	0,053	12,136
6/07	197,905	892,190	0,222	56,412	0,266	71,782
7/07	136,450	515,418	0,265	49,130	0,282	53,658
8/07	115,304	167,040	0,690	256,984	0,737	322,435
9/07	144,895	695,433	0,208	38,134	0,213	39,249
10/07	204,429	2817,757	0,073	15,991	0,063	13,798

Tabulka 3.1: Odhady parametrů MM a ML v negativně binomickém rozdělení pro SÚ.

měsíc	$\overline{x}_i$	s_i^2	$\widehat{p}^{(MM)}$	$\widehat{r}^{(MM)}$	$\widehat{p}^{(ML)}$	$\widehat{r}^{(ML)}$
2/07	20,650	67,818	0,304	9,040	0,329	10,102
3/07	27,455	50,260	0,546	33,052	0,582	38,296
4/07	31,950	145,418	0,220	8,996	0,225	9,262
5/07	24,667	46,033	0,536	28,476	0,586	34,863
6/07	34,429	40,957	0,841	181,560	0,000	0,000
7/07	12,700	17,589	0,722	32,987	0,781	45,352
8/07	11,652	11,419	1,020	−582,214	NA	∞
9/07	14,000	23,111	0,606	21,512	0,647	25,622
10/07	19,150	117,713	0,163	3,721	0,185	4,343

Tabulka 3.2: Odhady parametrů MM a ML v negativně binomickém rozdělení pro KK.

3.3 Model

Po seznámení s teorií řízených Markovských řetězců (dále jen MŘ) jsme se rozhodli, že tuto teorii aplikujeme na náš problém. V této sekci se nejprve seznámíme s teorií MŘ s oceněním stavu a řízených MŘ a následně si ukážeme, jak jsme tuto teorii aplikovali na projekt zadaný PS. Už jsme se zmínili, že teorii Markovských řetězců lze najít například v Prášková and Lachout (2005). Teorii řízených Markovský řetězců jsme nastudovali z Koole (2001).

3.3.1 Markovské řetězce s oceněním stavu

Nechť $\{X_t, t \in \mathbb{N}\}$ je diskrétní, časově homogenní MŘ s množinou stavů $\mathcal{X}$, jenž splňuje následující podmínky:

1. Řetězec $\{X_t, t \in \mathbb{N}\}$ je nerozložitelný.
2. Množina stavů $\mathcal{X}$ je konečná.
3. Řetězec $\{X_t, t \in \mathbb{N}\}$ není periodický.

Pro takovýto řetězec existuje stacionární rozdělení, označme jej π_*.

Předpokládejme, že dokážeme ocenit každý stav $\mathbf{x} \in \mathcal{X}$, ve kterém se řetězec nachází, pomocí tzv. *oceňovací funkce*

$$r : \mathcal{X} \to \mathbb{R},\ \mathbf{x} \in \mathcal{X} \mapsto r(\mathbf{x}).$$

Nachází-li se řetězec $\{X_t\}$ v čase t ve stavu $\mathbf{x}$, tj. $X_t = \mathbf{x}$, potom přinese zisk $r(\mathbf{x})$. Při studiu MŘ s oceněním stavů nás obvykle zajímá očekávaný zisk g pro $t \to +\infty$, neboli

$$g = \lim_{t\to+\infty} \mathrm{E} r(X_t) = \sum_{\mathbf{x}\in\mathcal{X}} \pi_*(\mathbf{x}) r(\mathbf{x}).$$

Podle zákona velkých čísel také platí

$$g = \lim_{T\to+\infty} \frac{1}{T} \sum_{t=0}^{T-1} r(X_t).$$

Zavedeme si ještě celkový očekávaný zisk za období $\{0, 1, \ldots, T-1\}$ za podmínky, že řetězec začíná ve stavu $\mathbf{x}$, jako

$$V_T(\mathbf{x}) = \sum_{t=0}^{T-1} \sum_{\mathbf{y}\in\mathcal{X}} p_t(\mathbf{x}, \mathbf{y}) r(\mathbf{y}),$$

kde $p_t(\mathbf{x}, \mathbf{y})$ je pravděpodobnost přechodu ze stavu $\mathbf{x}$ do stavu $\mathbf{y}$ za čas t. Všimněme si, že

$$\begin{aligned}\sum_{\mathbf{x}\in\mathcal{X}} \pi_*(\mathbf{x})V_T(\mathbf{x}) &= \sum_{\mathbf{x}\in\mathcal{X}} \pi_*(\mathbf{x}) \sum_{t=0}^{T-1} \sum_{\mathbf{y}\in\mathcal{X}} p_t(\mathbf{x}, \mathbf{y})r(\mathbf{y}) \\ &= \sum_{t=0}^{T-1} \sum_{\mathbf{y}\in\mathcal{X}} \sum_{\mathbf{x}\in\mathcal{X}} \pi_*(\mathbf{x})p_t(\mathbf{x}, \mathbf{y})r(\mathbf{y}) \\ &= \sum_{t=0}^{T-1} \sum_{\mathbf{y}\in\mathcal{X}} \pi_*(\mathbf{y})r(\mathbf{y}) = gT.\end{aligned}$$

Dále si zaveďme rozdíl zisku, jenž přinese řetězec začínající ve stavu $\mathbf{x}$, a zisku, který přinese řetězec začínající v ustáleném stavu, jako

$$V(\mathbf{x}) = \lim_{T\to+\infty} \left(V_T(\mathbf{x}) - gT\right).$$

$V_{T+1}(\mathbf{x})$ lze vyjádřit jako

$$V_{T+1}(\mathbf{x}) = V_T(\mathbf{x}) + \sum_{\mathbf{y}\in\mathcal{X}} \pi_T(\mathbf{y})r(\mathbf{y}),$$

kde $\pi_T(\mathbf{y})$ je pravděpodobnost, že se řetězec $\{X_t\}$ nachází v čase T ve stavu $\mathbf{y}$. Jelikož máme díky předpokladům zaručenou existenci stacionárního rozdělení, konverguje $\pi_T(\mathbf{y})$ pro $T \to +\infty$ k $\pi_*(\mathbf{y})$. Platí tedy

$$V_{T+1}(\mathbf{x}) = V_T(\mathbf{x}) + g + o(1), \tag{3.1}$$

kde pro člen $o(1)$ platí $\lim_{T\to+\infty} o(1) = 0$. V tomto případě nám značení symbolu o koliduje se značením řádu schématu, jež se však vyskytovalo pouze v kapitole 2. Doufáme, že to čtenáře příliš nezmátne.

$V_{T+1}(\mathbf{x})$ lze však počítat i jiným způsobem

$$V_{T+1}(\mathbf{x}) = r(\mathbf{x}) + \sum_{\mathbf{y}\in\mathcal{X}} p(\mathbf{x}, \mathbf{y})V_T(\mathbf{y}). \tag{3.2}$$

Spojením vzorců (3.1) a (3.2) ihned dostáváme

$$V_T(\mathbf{x}) + g + o(1) = r(\mathbf{x}) + \sum_{\mathbf{y}\in\mathcal{X}} p(\mathbf{x}, \mathbf{y})V_T(\mathbf{y}).$$

Odečteme-li gT od obou stran rovnice a necháme-li $T \to +\infty$, dostáváme tzv. *Poissonovu rovnici*

$$V(\mathbf{x}) + g = r(\mathbf{x}) + \sum_{\mathbf{y}\in\mathcal{X}} p(\mathbf{x}, \mathbf{y})V(\mathbf{y}). \tag{3.3}$$

Máme tedy celkem $|\mathcal{X}|$ Poissonových rovnic (3.3) (pro každý stav $\mathbf{x} \in \mathcal{X}$ jednu) o $|\mathcal{X}|+1$ neznámých. Přidáme-li ještě počáteční podmínku $V(\mathbf{0}) = 0$, získáme jednoznačné řešení soustavy.

Na začátku této části jsme mimo jiné předpokládali, že pracujeme s řetězcem, jenž není periodický. Tento předpoklad lze v případě diskrétních MŘ s konečnou množinou stavů vynechat, odvozování výše uvedených vzorců by pak ovšem bylo složitější.

3.3.2 Řízené Markovské řetězce

Mnohdy můžeme ovlivnit to, jakého stavu MŘ nabude v dalším časovém okamžiku. Jestliže máme tuto možnost, mluvíme pak o tzv. *řízení Markovských řetězců*. Předpokládejme, že máme k dispozici množinu akcí $\mathcal{A}$, s jejichž pomocí můžeme řetězec nějak ovládat. Jestliže se řetězec $\{X_t\}$ nachází ve stavu $\mathbf{x}$, pak na základě pevně zvoleného řízení R zvolíme z množiny $\mathcal{A}$ akci a, tedy

$$R : \mathcal{X} \to \mathcal{A},\ \mathbf{x} \mapsto R(\mathbf{x}) = a.$$

Možnost ovlivnit, jakého stavu řetězec nabude v dalším časovém okamžiku, se projeví na možné závislosti pravděpodobnosti přechodu ze stavu $\mathbf{x}$ do stavu $\mathbf{y}$ na akci a, tedy $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}, a, \mathbf{y})$. Obecně může i ocenění stavu záviset na provedené akci, to jsme ovšem při řešení projektu nepoužili, a tak se tomu věnovat nebudeme. Pracujeme-li s řízením, přejde rovnice (3.3) do tvaru

$$V^R(\mathbf{x}) + g^R = r(\mathbf{x}) + \sum_{\mathbf{y} \in \mathcal{X}} p(\mathbf{x}, a, \mathbf{y}) V^R(\mathbf{y}),$$

kde horní index ^{R} značí závislost na řízení R. Označme pomocí $V_t^R(\mathbf{x})$ celkový očekávaný zisk, jenž řetězec $\{X_t\}$ začínající ve stavu $\mathbf{x}$ přinese do času t při řízení R. Při řešení problémů z oblasti řízení MŘ je obvykle naším cílem najít optimální řízení, tj. $\arg\max_R V_T^R(\mathbf{x})/T$. Jelikož je množství různých řízení daného MŘ rovno $|\mathcal{X}| \cdot |\mathcal{A}|$, je hledání optimálního řízení značně složitou záležitostí náročnou na čas. Proto jej hledáme pomocí následujícího algoritmu:

1. Vezmi nějaké řízení R.
2. Spočti g^R a $V^R(\mathbf{x})$ pro všechny stavy $\mathbf{x}$.
3. Najdi lepší řízení R'. Pokud neexistuje, skonči.
4. $R = R'$, jdi na krok 2.

Třetí krok tohoto algoritmu je nejtěžší. Lepší řízení hledáme následujícím způsobem:

$$R'(\mathbf{x}) = \arg\max_{a \in \mathcal{A}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \mathcal{X}} p(\mathbf{x}, a, \mathbf{y}) V^R(\mathbf{y}) \right\}.$$

Pokud je pro každý stav $\mathbf{x}$ dosaženo maxima pomocí R, našli jsme optimální řízení. Pro optimální řízení R^* platí *Belmanova rovnice optimality*

$$V^{R^*}(\mathbf{x}) + g^{R^*} = \max_{a \in \mathcal{A}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \mathcal{X}} p(\mathbf{x}, a, \mathbf{y}) V^{R^*}(\mathbf{y}) \right\}.$$

Více o tomto tématu lze nalézt například v knize White (1993).

Prokletí dimenzionality

Teorie řízených Markovských řetězců je sice krásná a teoreticky se s její pomocí dají řešit rozličné problémy, avšak z hlediska praxe jsou v ní mnohá úskalí. Jeden z nejvážnějších problémů této teorie je často se vyskytující „prokletí dimenzionality“. S tímto problémem se obvykle setkáváme, pokud je řetězec vícerozměrný nebo stavový prostor příliš veliký. Jde o to, že na uchování dat potřebných k výpočtu optimálního řízení bychom potřebovali pevný disk s obrovskou kapacitou. Tak veliký zkrátka neexistuje. Nehledě na to, kolik času by procesor počítače potřeboval k tak obrovskému výpočtu.

„Prokletí dimenzionality“ se vyskytlo i v našem problému, a tak explicitní výpočet nebyl možný.

Východisko

Vyskytne-li se při řešení praktického problému „prokletí dimenzionality“, doporučuje se nejdříve použít tzv. agregování. To spočívá v tom, že co nejvíce věcí sloučíme dohromady. Například sloučíme zákazníky do větších skupin po $20, 50, \ldots$, které budou do systému přicházet s větší intenzitou. Můžeme také sloučit obslužné stanice, které budou následně pracovat rychleji atd. Tento přístup nám v našem problému ovšem nepomohl.

Dalším možným řešením je použít aproximativní dynamické programování. Jelikož jsme ho v našem projektu použili pouze okrajově, nebudu se zde o něm šířeji rozepisovat. Pomocí něj jsme chtěli zjistit, jak moc se naše pevně zvolené řízení liší od optimálního.

V neposlední řadě se při řešení problémů z oblasti řízení MŘ používají simulace. Podle odhadnutých nebo zadaných rozdělení se generují vstupy do systému a doby obsluhy a sleduje se vývoj systému. S vyšším počtem opakování simulace získáváme lepší výsledek, ale zároveň nám roste čas potřebný k výpočtu, proto je volba počtu opakování velmi důležitá. Sofistikované způsoby simulace složitých systémů jsou popsány například v Chang et al. (2007).

K tomuto řešení jsme se přiklonili v našem projektu i my; neměli jsme vlastně ani na vybranou, neboť alternativa neexistovala.

3.4 Aplikace teorie

V této části si ukážeme, jakým způsobem jsme představenou teorii aplikovali na náš problém. Nejdříve si uvedeme, jakou reprezentaci systému jsme zvolili, a popíšeme si algoritmus, jenž jsme použili pro výpočet optimálního počtu a rozložení zaměstnanců. Následně aplikujeme teorii GA.

3.4.1 Reprezentace systému

MŘ popisující stavy systému byl celkem 19–ti rozměrný. Prvních 9 stavů popisovalo počet žádostí o SÚ. Přesněji, i–tá složka obsahovala informaci o tom, kolik žádostí o SÚ je v systému již i pracovních dní, $i = 1, \ldots, 8$. Devátá složka řetězce uváděla, kolik žádostí o SÚ je v systému déle než 8 pracovních dní. Analogicky, dalších 9 složek MŘ bylo určeno pro KK. Pro $i = 10, \ldots, 17$ obsahovala i–tá složka informaci o tom, kolik žádostí o KK je v systému již $(i-9)$ pracovních dní. Osmnáctá složka pak vyjadřovala, kolik žádost o KK je v systému déle než 8 pracovních dní. Jelikož řetězec nebyl časově homogenní, byl devatenáctou složkou čas.

Ještě jsme uvažovali nad modelem, ve kterém by byl systém hromadné obsluhy PS popsán pomocí sedmirozměrného řetězce. První tři rozměry by náležely SÚ. V první složce bychom drželi informaci o tom, kolik žádostí je v systému první, druhý nebo třetí pracovní den. Druhá složka by udávala, kolik žádostí je v systému od 4 do 8 pracovních dnů. Třetí složka by udávala počet žádostí, které jsou v systému déle než 8 pracovních dnů. V dalších třech rozměrech bychom uchovávali stejné informace pro KK. Sedmou složkou by byl opět čas.

Avšak pokud nevíme, kolik žádostí o SÚ je v systému už třetí pracovní den, potom nevíme, kolik žádostí přesunout z první složky do druhé. V tomto modelu není jasné, kolik žádostí se má na konci dne přesunout z první složky do druhé, ze druhé do třetí, ze čtvrté do páté a z páté do šesté. Tuto myšlenku tedy nešlo realizovat.

Pro obsluhování žádostí jsme zvolili systém „FIFO“, což je zkratka pro *first in – first out*. Český ekvivalent pro „FIFO“ je fronta. Nejdříve je tedy obsloužen ten zákazník, který přišel jako první. V našem případě to znamená, že nejdříve jsou vyřizovány žádosti nacházející se v deváté a osmnácté složce, dále se vyřizují žádosti v osmé a sedmnácté složce atd. Poslední jsou vyřizovány žádosti z první a desáté složky MŘ.

3.4.2 Oceňovací funkce

Oceňovací funkce, jíž jsme si představili v části 3.3.1, v našem případě stavy pokutovala, nepřinášela tedy užitek. Návod jak ocenit jednotlivé stavy jsme od PS nedostali, a tak jsme byli nuceni nastavit oceňovací funkci sami. Z logiky věci by neměly být pokutovány ty složky MŘ, ve kterých držíme počet žádostí, které jsou v systému 3 a méně pracovních dnů. Složky odpovídající žádostem zůstávajícím v systému od 4 do 8 pracovních dnů by

měly být pokutovány stejně. Devátá a osmnáctá složka by měly být pokutovány nejvíce, neboť žádostí zůstávajících v systému déle než 8 pracovních dní může být maximálně 10 %. Poslední, tj. devatenáctá složka by neměla být pokutována, neboť je pouze pomocná.

Snažili jsme se nastavit oceňovací funkci tak, aby optimální řešení vycházelo pro takové nastavení počtu a rozložení pracovníků, při kterém jen těsně zvládají plnit politiku PS. To odpovídá tomu, že chceme minimalizovat mzdové náklady při plnění limitů stanovených PS.

Zvolili jsme lineární oceňovací funkci, tedy

$$r(\mathbf{x}) = \mathbf{v}^T\mathbf{x},\ \mathbf{x} \in \mathcal{X},$$

kde

$$v_i = \begin{cases} 0, & i = 1,2,3 \\ 2000, & i = 4,5,\ldots,8 \\ 10000, & i = 9 \\ 0, & i = 10,11,12 \\ 10000, & i = 13,14,\ldots,17 \\ 20000, & i = 18 \\ 0, & i = 0 \end{cases}$$

3.4.3 Množina akcí a řízení MŘ

Jestliže pracujeme s nenulovým počtem univerzálních pracovníků, musíme každý den určit, kolik z nich bude pracovat na oddělení SÚ a kolik na oddělení KK. Množinou akcí byla tedy všechna možná rozdělení univerzálních pracovníků na oddělení SÚ a KK.

Řízení, jež jsme na základě logických úvah zvolili, rozdělovalo univerzální pracovníky na začátku dne na základě žádostí zůstávajících v systému z předchozích dní a očekávaného počtu nově příchozích žádostí.

Na konci dne nevyřízené žádosti o SÚ i KK zestárnou o jeden den, tj. posunou se v řetězci o jednu složku doprava s výjimkou desáté a osmnácté složky. Víme, kolik žádostí nebylo zpracováno předchozí den, tj. pro $i = 2,\ldots,9,11,\ldots 18$ známe i-tou složku řetězce, ale ještě nevíme, kolik bude nových žádostí, tj. neznáme první a desátou složku řetězce. Do první složky dosadíme očekávanou hodnotu (střední hodnotu) počtu příchozích žádostí o SÚ pro daný den a do desáté složky dosadíme očekávanou hodnotu (střední hodnotu) počtu příchozích žádostí o KK pro daný den. Provedeme výpočet, kolik žádostí zvládnou obsloužit samotní úvěráři a kreditkáři. Pro stav, který zbude po tomto zpracování, oceníme zvlášť úvěrovou část a karetní část. V poměru, v jakém jsou náklady na SÚ a náklady na KK, rozdělíme univerzální pracovníky na práci na SÚ a na KK. Pokud je ocenění po zpracování nulové pro úvěrovou i karetní část (to znamená, že úvěráři a kartáři zpracují vše sami bez pomoci univerzálů), spočte se stav, který by nastal, kdyby se daný den nic neobsloužilo, a ten se ocení. Poté se všichni zaměstnanci rozdělí tak, aby počet lidí pracujících na úvěrech a počet lidí pracujících na kreditkách byl stejný jako poměr nákladů za úvěrovou a kreditní část. Takto rozhodujeme každý pracovní den v roce.

Jak již bylo řečeno, univerzální pracovníky rozdělujeme na začátku dne. Jejich obsazení se na oddělení SÚ a KK v průběhu dne nemění.

3.4.4 Simulace

Zde si ve zkratce popíšeme, jak jsme simulovali příchody žádostí o SÚ a KK do PS.

V našem projektu jsme čtyřikrát simulovali „průběh roku“ s 10 000 opakováními a jednou pouze s 5 000 opakováními. Při hledání optima pomocí GA budeme zásadně provádět simulaci s 10 000 opakováními.

Jeden „průběh roku“ obsahuje vygenerované počty příchozích žádostí na obě oddělení pro každý den a také vygenerované poměry schválených, zamítnutých a stornovaných žádostí pro každý den. Počty příchozích žádostí o SÚ a KK generujeme z negativně binomického rozdělení s parametry odhadnutými z dat pomocí metody momentů. Poměry schválených, zamítnutých a stornovaných žádostí generujeme z Dirichletova rozdělení s parametry odhadnutými z dat pomocí metody maximální věrohodnosti.

K dispozici máme „normominuty“ pracovníků, a tak jsme schopni určit, kolik žádostí se za daný den vyřídí (odejdou ze systému).

Víme, kolik žádostí je v systému kolik dní na konci každého dne. To vyčteme ze stavu, ve kterém se nalézá MŘ popisující systém. Pomocí funkce r oceníme stav, ve kterém se řetězec nalézá v daný den. Takto oceníme každý den v roce, všechna tato ocenění sečteme a následně k tomu přičteme ještě mzdové náklady, čímž získáme celkové roční náklady PS při daném počtu a rozložení zaměstnanců. Mzdovými náklady rozumíme roční náklady na všechny zaměstnance oddělení SÚ a KK (včetně univerzálních pracovníků).

V předchozím odstavci jsme narazili na mzdové náklady, ale ještě jsme si neuvedli, jak jsme je určili. Máme tři druhy zaměstnanců – úvěráře, kreditkáře a univerzály. Předpokládáme, že jsou v rámci jednoho druhu pracovníků platy stejné. Ve skutečnosti tomu tak sice nejspíš nebude, ale tento předpoklad je k výpočtu nutný. Uvažujeme ovšem, že v jednotlivých kategoriích se platy různí. Jelikož jsme si byli vědomi toho, že tyto údaje jsou tajné, požádali jsme vedení PS o prozrazení výše platu v jednotlivých kategoriích alespoň k nějakému referenčnímu platu. PS nám však tato data neposkytla, a tak jsme zvolili výše platů zaměstnanců v našem modelu sami.

Jelikož pracovníci oddělení KK rozhodují zpravidla o malých částkách, připsali jsme jim 1–násobek referenčního platu. Pracovníkům oddělení SÚ jsme dali 1,2–násobek referenčního platu. A konečně univerzální pracovníci, jakožto nejvíce vyškolení, od nás dostali 1,45–násobek referenčního platu. Z databáze Českého statistického úřadu jsme zjistili, že průměrná výše platů v sektoru bankovnictví je 40 000 Kč, což se nám zdálo rozumné. Referenční plat byl tedy stanoven na 40 000 Kč. K tomu jsme samozřejmě ještě museli přičíst odvody zaměstnavatele. Platy v jednotlivých kategoriích tak, jak jsme je určili, jsou uvedeny v tabulce 3.3.

Pro dané rozložení zaměstnanců spočítáme pro každý průběh celkové roční náklady. Ty sečteme a vydělíme počtem simulací, tedy číslem 10 000. Tím získáme průměrné celkové roční náklady, které se snažíme minimalizovat.

kategorie zaměstnance	násobek ref. platu	hrubá měsíční mzda	měsíční mzdové náklady PS	roční mzdové náklady PS
úvěrář	1,2	48 000	64 800	777 600
kreditkář	1	40 000	54 000	648 000
univerzál	1,45	58 000	78 300	939 600

Tabulka 3.3: Mzdové náklady na zaměstnance.

V simulaci rovněž počítáme, jak zvládají zaměstnanci plnit limity stanovené vedením PS. Mějme poměr vyjádřený ve tvaru $(s_1/s_2/s_3)$, kde $s_1, s_2, s_3 \geq 0$ a $s_1+s_2+s_3 = 1$. Poměr počítáme pro každý měsíc, získáme tedy celkem 120 000 takovýchto poměrů.

Při porovnávání poměrů jsme se rozhodli brát za nejdůležitější třetí složku, následně druhou a pak teprve první. Platí tedy

$$(0{,}4/0{,}3/0{,}3) > (0{,}5/0{,}3/0{,}2) > (0{,}4/0{,}5/0{,}1) > (0{,}5/0{,}4/0{,}1) > \\ > (0{,}65/0{,}25/0{,}1) > (0{,}65/0{,}3/0{,}05) > (1/0/0),$$

kde $\mathbf{x} < \mathbf{y}$ znamená, že $\mathbf{x}$ je lepší než $\mathbf{y}$. Takto seřadíme všechny dosažené poměry a zapamatujeme si ten nejhorší. Jelikož jsme však zvyklí pracovat na 5% hladině spolehlivosti, rozhodli jsme se z takto získaných poměrů vyřadit 5 % nejhorších a následně si zapamatovat nejhorší z těchto „očištěných“ poměrů.

Tyto dva poměry se nám vypíší u nejlepších dosažených řešení, což nám pomůže při diskuzi optimálního počtu a rozložení zaměstnanců.

3.4.5 Hledání optima pomocí specializovaného algoritmu

Nejdříve si zavedeme značení, které budeme používat v celé této části:

v ... celkový počet zaměstnanců,

a_v ... počet úvěrářů z celkového počtu v zaměstnanců,

b_v ... počet kreditkářů z celkového počtu v zaměstnanců,

c_v ... počet univerzálů z celkového počtu v zaměstnanců.

Vzhledem k tomu, že jiní pracovníci než úvěráři, kreditkáři nebo univerzálové nás při našem zkoumání nezajímají, musí vždy platit $v = a_v + b_v + c_v$.

Algoritmus probíhá následovně:

- urči startovní hodnoty $v_0, a_{v_0}, b_{v_0}, c_{v_0}$
- $v = v_0$
- urči optimální rozložení a průměrné náklady pro v zaměstnanců
- opakuj

– přidej jednoho univerzála ($v = v + 1, a_v = a_{v_0}, b_v = b_{v_0}, c_v = c_{v-1} + 1$)
– urči optimální rozložení a průměrné roční náklady pro v zaměstnanců
– pokud je splněna ukončovací podmínka, tak skonči

Popišme si jednotlivé části algoritmu.

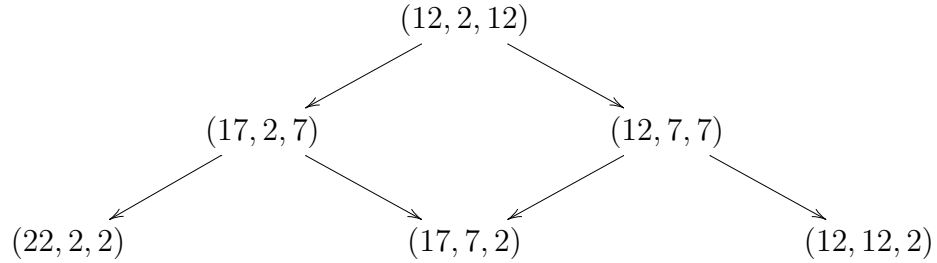
Určení startovacích hodnot

Startovací celkový počet zaměstnanců jsme určili následujícím způsobem. Pro SÚ a KK zvlášť jsme vzali celkový očekávaný počet žádostí za celý rok a vydělili ho počtem žádostí, které zvládne za rok vyřídit jeden zaměstnanec daného oddělení. Výsledná dvě čísla jsme sečetli, a tím dostali minimální počet zaměstnanců, který PS potřebuje. Označme tento minimální počet jako v_0.

Startovací rozložení pracovníků jsme určili na základě velmi jednoduché úvahy. Našli jsme pracovní den, ve kterém jsme v rámci celého roku očekávali nejmenší počet příchozích žádostí o SÚ. Tento minimální počet žádostí jsme vydělili počtem žádostí, které dokáže za jeden den vyřídit pracovník oddělení SÚ, a tím jsme získali minimální počet zaměstnanců oddělení SÚ a_{v_0}. Stejným způsobem jsme postupovali i při výpočtu minimálního počtu zaměstnanců oddělení KK b_{v_0}. Startovací počet univerzálních pracovníků jsme zvolili jako $c_{v_0} = v_0 - a_{v_0} - b_{v_0}$.

Určení optimálního rozložení zaměstnanců pro pevný počet zaměstnanců v

Mějme vektor rozložení pracovníků ve tvaru (a_v, b_v, c_v), kde samozřejmě platí $c_v = v - a_v - b_v$. Kvůli časové úspoře probíhá určení optimálního rozložení pracovníků pro pevné v ve dvou krocích. Na začátku máme pravděpodobně více univerzálů, než by bylo zapotřebí, proto přesouváme skupinu univerzálů buď mezi úvěráře, nebo mezi kreditkáře. V našem projektu jsme volili skupiny o pěti, nebo třech lidech. To provádíme tak dlouho, dokud můžeme univerzály po skupinách přesouvat. Na následujícím diagramu je vidět, jak dochází k přesunům.



(3.4)

Pro všechna tato získaná rozložení pracovníků (včetně toho, ze kterého jsme startovali – v diagramu úplně nahoře) spočítáme průměrné celkové roční náklady. Dále pokračujeme pouze s rozložením s nejnižšími náklady. Dejme tomu, že nám nejmenší náklady vyšly pro rozdělení $(17, 2, 7)$.

V druhém kroku přesouváme pracovníky šesti různými směry:

– přesunutí jednoho úvěráře mezi univerzály,

– přesunutí jednoho kreditkáře mezi univerzály,

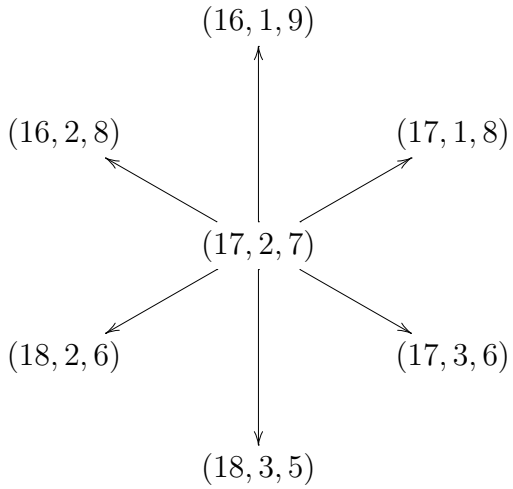
– přesunutí jednoho úvěráře a jednoho kreditkáře mezi univerzály,

– přesunutí jednoho univerzála mezi úvěráře,

– přesunutí jednoho univerzála mezi kreditkáře,

– přesunutí jednoho univerzála mezi úvěráře a jednoho univerzála mezi kreditkáře.

Grafické znázornění těchto přesunů je možné vidět na následujícím diagramu.



(3.5)

Pro všechna tato rozložení zaměstnanců spočítáme opět roční náklady. Ze směrů přesunu pracovníků vybereme samozřejmě ten s nejmenšími ročními náklady. Pokud pro toto nové rozložení zaměstnanců vychází roční náklady menší než pro rozložení zaměstnanců, ze kterého jsme startovali v druhém kroku, posuneme se tímto směrem, tj. přesuneme pracovníky odpovídajícím způsobem. Takto postupujeme, dokud se roční náklady zmenšují.

Ukončovací podmínka

Vlastnosti funkce, kterou se snažíme minimalizovat, jsou nám neznámé. Jejím vstupem je dané rozložení zaměstnanců a výstupem jsou roční náklady při tomto rozložení. Při zvyšování počtu zaměstnanců směrem od celkového počtu 0 se celkové náklady nejdříve snižují, neboť se zmenšují pokuty za nedodržování limitů stanovených PS, a to až do té doby, než dosáhneme minima. Zde nastává problém, protože nevíme, jak se funkce v oblasti okolo globálního minima chová. Mohou zde být lokální minima, která nejsou zároveň globálními. Přidáváme-li však stále další zákazníky, funkce začne opět růst. To je způsobeno tím, že pracovníků je již dostatečný počet na to, aby zvládali vyřizovat všechny žádosti v rámci prvních tří pracovních dnů, a tak nejsme vůbec penalizováni za neplnění stanovených limitů. Vzrůstají však mzdové náklady a díky tomu celkové roční náklady rostou.

Jelikož nevíme, jak se funkce chová okolo globálního minima, nastavili jsme ukončovací podmínku tak, aby zastavila algoritmus, pokud roční náklady dvakrát po sobě neklesnou. Tedy přidáme-li dvakrát po sobě jednoho pracovníka a náklady se ani v jednom případě nezmenší, algoritmus skončí.

Výsledky dosažené pomocí specializovaného algoritmu

Zde si uvedeme, jakých výsledků jsme dosáhli pomocí právě představeného algoritmu. Výsledky jsou shrnuty v tabulkách 3.4, 3.5, 3.6, 3.7 a 3.8. Jednotlivé simulace se liší počtem opakování a tím, kolik lidí čítaly skupiny, po kterých jsme přesouvali univerzály mezi úvěráře nebo kreditkáře v prvním kroku určování optimálního rozložení zaměstnanců při jejich pevném počtu. Výpočet, ve kterém měla simulace 10 000 opakování a ve kterém jsme univerzály přesouvali ve skupinkách po pěti, trval déle než 7 hodin.

V tabulkách je v prvním sloupci počet zaměstnanců. Ve druhém sloupci je rozložení zaměstnanců po prvním kroku přesouvání, tj. po přesouvání po skupinách, pro daný celkový počet zaměstnanců. Ve čtvrtém sloupci je konečné optimální rozložení zaměstnanců pro daný počet zaměstnanců. „Max. poměry“ udávají pro konečné rozložení zaměstnanců nejhorší plnění limitů stanovených vedením PS. „Max. poměry s hlad.“ udávají plnění těchto limitů na hladině spolehlivosti 5 %.

v	Rozložení 1. krok	Náklady 1. krok	Rozložení 2. krok	Náklady 2. krok	Max. poměry		Max. poměry s hlad.	
					SÚ	KK	SÚ	KK
27	(23, 2, 2)	1 629 870 421	(21, 1, 5)	1 574 428 149	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
28	(23, 2, 3)	905 307 482	(22, 2, 4)	876 341 006	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,05/0,95)
29	(23, 2, 4)	545 695 488	(23, 2, 4)	545 695 488	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,28/0,72)
30	(26, 2, 2)	366 685 727	(26, 3, 1)	327 292 695	(0/0/1)	(0/0,05/0,95)	(0/0/1)	(0/0,90/0,10)
31	(26, 2, 3)	230 196 859	(27, 3, 1)	198 294 690	(0/0/1)	(0/0,01/0,99)	(0/0,19/0,81)	(0/0,90/0,10)
32	(26, 2, 4)	140 493 011	(26, 2, 4)	140 493 011	(0/0/1)	(0/0,05/0,95)	(0/0,45/0,55)	(0/0,90/0,10)
33	(29, 2, 2)	89 832 818	(29, 3, 1)	78 503 103	(0/0/1)	(0/0,46/0,54)	(0,10/0,77/0,13)	(0/0,90/0,10)
34	(29, 2, 3)	61 863 585	(30, 3, 1)	56 018 714	(0/0/1)	(0/0,48/0,52)	(0,33/0,57/0,10)	(0/0,91/0,09)
35	(29, 2, 4)	44 777 491	(30, 3, 2)	43 824 052	(0/0/1)	(0/1/0)	(0,24/0,66/0,10)	(0,61/0,39/0)
36	(32, 2, 2)	38 411 574	(31, 3, 2)	36 424 394	(0,03/0,15/0,82)	(0,01/0,99/0)	(0,34/0,56/0,10)	(0,62/0,38/0)
37	(32, 2, 3)	33 422 190	(32, 3, 2)	32 894 682	(0,19/0,24/0,57)	(0,09/0,91/0)	(0,54/0,46/0)	(0,64/0,36/0)
38	(32, 2, 4)	31 744 811	(33, 3, 2)	31 339 656	(0,29/0,33/0,38)	(0,10/0,90/0)	(0,65/0,35/0)	(0,65/0,35/0)
39	(35, 2, 2)	31 762 695	(34, 3, 2)	30 976 209	(0,35/0,64/0,01)	(0,06/0,94/0)	(0,79/0,21/0)	(0,65/0,35/0)
40	(35, 2, 3)	31 567 399	(34, 5, 1)	31 043 732	(0,35/0,64/0,01)	(0,16/0,84/0)	(0,95/0,05/0)	(0,95/0,05/0)
41	(35, 5, 1)	31 555 193	(35, 5, 1)	31 555 193	(0,23/0,77/0)	(0,23/0,77/0)	(1/0/0)	(0,95/0,05/0)

Tabulka 3.4: Průběh simulace – Počet opakování = 10 000, přesouvání po skupinách o třech univerzálech.

v	Rozložení 1. krok	Náklady 1. krok	Rozložení 2. krok	Náklady 2. krok	Max. poměry		Max. poměry s hlad.	
					SÚ	KK	SÚ	KK
27	(19, 2, 6)	1 630 842 961	(19, 1, 7)	1 574 063 699	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
28	(24, 2, 2)	901 235 677	(24, 2, 2)	901 235 677	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
29	(24, 2, 3)	563 313 260	(23, 2, 4)	544 356 539	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,28/0,72)
30	(24, 2, 4)	351 870 807	(24, 2, 4)	351 870 807	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,64/0,36)
31	(24, 2, 5)	250 219 785	(25, 2, 4)	222 514 452	(0/0/1)	(0/0/1)	(0/0,18/0,82)	(0/0,90/0,10)
32	(24, 2, 6)	159 397 480	(24, 2, 6)	159 397 480	(0/0/1)	(0/0,24/0,76)	(0,09/0,26/0,65)	(0/0,90/0,10)
33	(29, 2, 2)	89 430 162	(29, 3, 1)	77 933 334	(0/0/1)	(0/0,29/0,71)	(0,10/0,76/0,14)	(0/0,90/0,10)
34	(29, 2, 3)	61 382 067	(30, 3, 1)	55 905 978	(0/0/1)	(0/0,37/0,63)	(0,40/0,50/0,10)	(0/0,91/0,09)
35	(29, 2, 4)	44 831 177	(29, 3, 3)	43 786 521	(0/0,02/0,98)	(0/1/0)	(0,34/0,56/0,10)	(0,86/0,14/0)
36	(29, 2, 5)	40 863 549	(31, 3, 2)	36 409 974	(0/0,31/0,69)	(0,01/0,99/0)	(0,35/0,55/0,10)	(0,62/0,38/0)
37	(29, 2, 6)	35 687 630	(32, 3, 2)	32 867 847	(0,07/0,40/0,53)	(0,10/0,90/0)	(0,51/0,43/0,06)	(0,64/0,36/0)
38	(34, 2, 2)	32 347 684	(33, 3, 2)	31 366 732	(0,24/0,63/0,13)	(0,11/0,89/0)	(0,66/0,34/0)	(0,65/0,35/0)
39	(34, 2, 3)	31 245 887	(34, 3, 2)	30 984 313	(0,02/0,98/0)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
40	(34, 2, 4)	31 726 000	(36, 4, 0)	31 280 455	(0,32/0,52/0,16)	(0,80/0,20/0)	(0,80/0,20/0)	(1/0/0)
41	(34, 2, 5)	32 566 146	(35, 5, 1)	31 562 447	(0,33/0,67/0)	(0,20/0,80/0)	(1/0/0)	(0,95/0,05/0)

Tabulka 3.5: Průběh simulace – Počet opakování = 10 000, přesouvání po skupinách o pěti univerzálech.

v	Rozložení 1. krok	Náklady 1. krok	Rozložení 2. krok	Náklady 2. krok	Max. poměry		Max. poměry s hlad.	
					SÚ	KK	SÚ	KK
27	(19, 2, 6)	1 636 246 240	(19, 1, 7)	1 577 032 782	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
28	(24, 2, 2)	909 507 917	(22, 2, 4)	872 564 178	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,05/0,95)
29	(24, 2, 3)	565 794 172	(23, 2, 4)	543 157 076	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,28/0,72)
30	(24, 2, 4)	349 744 053	(24, 2, 4)	349 744 053	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,65/0,35)
31	(24, 2, 5)	249 734 871	(25, 2, 4)	223 753 398	(0/0/1)	(0/0/1)	(0/0,17/0,83)	(0/0,90/0,10)
32	(24, 2, 6)	160 502 304	(24, 2, 6)	160 502 304	(0/0/1)	(0/0,20/0,80)	(0,03/0,31/0,66)	(0/0,90/0,10)
33	(29, 2, 2)	89 534 281	(29, 3, 1)	78 382 191	(0/0/1)	(0/0,51/0,49)	(0,14/0,73/0,13)	(0/0,90/0,10)
34	(29, 2, 3)	61 353 623	(30, 3, 1)	55 822 153	(0/0/1)	(0/0,52/0,48)	(0,07/0,83/0,10)	(0/0,92/0,08)
35	(29, 2, 4)	44 890 285	(30, 3, 2)	43 826 366	(0/0,05/0,95)	(0/1/0)	(0,27/0,63/0,10)	(0,60/0,40/0)
36	(29, 2, 5)	41 102 859	(30, 3, 3)	36 576 259	(0/0,17/0,83)	(0,03/0,97/0)	(0,42/0,48/0,10)	(0,89/0,11/0)
37	(29, 2, 6)	35 683 566	(29, 2, 6)	35 683 566	(0,18/0,15/0,67)	(0/1/0)	(0,45/0,52/0,03)	(0,76/0,24/0)
38	(34, 2, 2)	32 384 762	(33, 3, 2)	31 284 668	(0,21/0,28/0,51)	(0,08/0,92/0)	(0,66/0,34/0)	(0,65/0,35/0)
39	(34, 2, 3)	31 305 041	(34, 3, 2)	31 006 417	(0,23/0,69/0,08)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
40	(34, 2, 4)	31 713 431	(36, 4, 0)	31 265 995	(0,39/0,59/0,02)	(0,84/0,16/0)	(0,79/0,21/0)	(1/0/0)
41	(34, 2, 5)	32 571 060	(35, 5, 1)	31 560 526	(0,29/0,71/0)	(0,24/0,76/0)	(1/0/0)	(0,95/0,05/0)

Tabulka 3.6: Průběh simulace – Počet opakování = 10 000, přesouvání po skupinách o pěti univerzálech.

v	Rozložení 1. krok	Náklady 1. krok	Rozložení 2. krok	Náklady 2. krok	Max. poměry		Max. poměry s hlad.	
					SÚ	KK	SÚ	KK
27	(19, 2, 6)	1 640 259 578	(19, 1, 7)	1 576 178 208	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
28	(24, 2, 2)	903 638 589	(22, 2, 4)	876 610 285	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,05/0,95)
29	(24, 2, 3)	567 684 998	(23, 2, 4)	542 385 292	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,28/0,72)
30	(24, 2, 4)	352 614 373	(24, 2, 4)	352 614 373	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,65/0,35)
31	(24, 2, 5)	250 688 717	(25, 2, 4)	224 295 702	(0/0/1)	(0/0/1)	(0/0,18/0,82)	(0/0,90/0,10)
32	(24, 2, 6)	159 282 327	(24, 2, 6)	159 282 327	(0/0/1)	(0/0,35/0,65)	(0/0,34/0,66)	(0/0,90/0,10)
33	(29, 2, 2)	89 227 365	(29, 3, 1)	77 676 597	(0/0/1)	(0/0,30/0,70)	(0/0,87/0,13)	(0/0,91/0,09)
34	(29, 2, 3)	61 480 570	(30, 3, 1)	56 279 653	(0/0/1)	(0/0,51/0,49)	(0,40/0,50/0,10)	(0/0,92/0,08)
35	(29, 2, 4)	44 952 663	(29, 3, 3)	43 693 384	(0/0,11/0,89)	(0/1/0)	(0,19/0,71/0,10)	(0,87/0,13/0)
36	(29, 2, 5)	40 803 744	(31, 3, 2)	36 469 290	(0/0,23/0,77)	(0,01/0,99/0)	(0,38/0,52/0,10)	(0,63/0,37/0)
37	(29, 2, 6)	35 672 097	(31, 3, 3)	32 995 716	(0,06/0,39/0,55)	(0,10/0,90/0)	(0,54/0,46/0)	(0,91/0,09/0)
38	(34, 2, 2)	32 378 432	(33, 3, 2)	31 304 013	(0,17/0,57/0,26)	(0,08/0,92/0)	(0,66/0,34/0)	(0,64/0,36/0)
39	(34, 2, 3)	31 299 923	(34, 3, 2)	30 994 659	(0,33/0,54/0,13)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
40	(34, 2, 4)	31 710 477	(34, 5, 1)	30 995 682	(0,20/0,79/0,01)	(0,22/0,78/0)	(0,94/0,06/0)	(0,95/0,05/0)
41	(34, 2, 5)	32 583 332	(35, 5, 1)	31 557 477	(0,20/0,80/0)	(0,29/0,71/0)	(1/0/0)	(0,95/0,05/0)

Tabulka 3.7: Průběh simulace – Počet opakování = 10 000, přesouvání po skupinách o pěti univerzálech.

v	Rozložení 1. krok	Náklady 1. krok	Rozložení 2. krok	Náklady 2. krok	Max. poměry		Max. poměry s hlad.	
					SÚ	KK	SÚ	KK
27	(23, 2, 2)	1 628 685 365	(21, 2, 4)	1 572 033 022	(0/0/1)	(0/0/1)	(0/0/1)	(0/0/1)
28	(23, 2, 3)	896 320 895	(22, 2, 4)	875 609 576	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,05/0,95)
29	(23, 2, 4)	541 085 274	(23, 2, 4)	541 085 274	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,29/0,71)
30	(26, 2, 2)	362 159 926	(26, 3, 1)	325 672 652	(0/0/1)	(0/0/1)	(0/0/1)	(0/0,90/0,10)
31	(26, 2, 3)	230 426 390	(27, 3, 1)	199 012 760	(0/0/1)	(0/0,30/0,70)	(0/0,20/0,80)	(0/0,91/0,09)
32	(26, 2, 4)	140 265 381	(28, 3, 1)	120 479 850	(0/0/1)	(0/0,21/0,79)	(0,20/0,30/0,50)	(0/0,91/0,09)
33	(29, 2, 2)	89 446 288	(29, 3, 1)	79 081 576	(0/0/1)	(0/0,40/0,60)	(0,20/0,67/0,13)	(0/0,91/0,09)
34	(29, 2, 3)	61 340 007	(30, 3, 1)	56 259 310	(0/0/1)	(0/0,57/0,43)	(0,20/0,70/0,10)	(0/0,88/0,08)
35	(29, 2, 4)	44 756 518	(29, 3, 3)	43 747 388	(0/0,11/0,89)	(0,01/0,99/0)	(0,28/0,62/0,10)	(0,86/0,14/0)
36	(32, 2, 2)	38 443 028	(31, 3, 2)	36 496 902	(0,10/0,21/0,69)	(0,04/0,96/0)	(0,33/0,57/0,10)	(0,62/0,38/0)
37	(32, 2, 3)	33 307 236	(32, 3, 2)	32 829 769	(0,14/0,31/0,55)	(0,05/0,95/0)	(0,54/0,46/0)	(0,64/0,36/0)
38	(32, 2, 4)	31 683 083	(33, 3, 2)	31 318 654	(0,31/0,60/0,09)	(0,15/0,85/0)	(0,66/0,34/0)	(0,65/0,35/0)
39	(35, 2, 2)	31 813 638	(34, 3, 2)	31 022 252	(0,41/0,58/0,01)	(0,10/0,90/0)	(0,80/0,20/0)	(0,65/0,35/0)
40	(35, 2, 3)	31 556 584	(36, 4, 0)	31 252 816	(0,01/0,99/0)	(0,75/0,25/0)	(0,79/0,21/0)	(1/0/0)
41	(35, 5, 1)	31 555 229	(35, 5, 1)	31 555 229	(0,21/0,79/0)	(0,27/0,73/0)	(1/0/0)	(0,95/0,05/0)

Tabulka 3.8: Průběh simulace – Počet opakování = 5 000, přesouvání po skupinách o třech univerzálech.

3.5 Adaptace problému na teorii genetických algoritmů

Ze všeho nejdříve si popíšeme náš optimalizační problém matematicky. Zavedeme si funkci

$$f : \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 \to \mathbb{R},$$

kde pod $\mathbb{N}_0$ rozumíme přirozená čísla včetně nuly. Funkce f přiřadí danému složení zaměstnanců PS na odděleních SÚ a KK průměrné celkové roční náklady, tj. výsledek simulace. Máme tedy za úkol nalézt

$$\mathbf{x}_{opt} = \underset{\mathbf{x}\in\mathbb{N}_0\times\mathbb{N}_0\times\mathbb{N}_0}{\arg\min} f(\mathbf{x}). \tag{3.6}$$

O průběhu funkce f nevíme zhola nic. Umíme ji pouze vyčíslit. K hledání minima takovéto funkce na množině $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$ můžeme použít nejaký speciální algoritmus, což jsme udělali při řešení tohoto problému v rámci semináře modelování v ekonomii, nebo můžeme zkusit některý z evolučních algoritmů, které se díky své robustnosti dají použít na širokou škálu optimalizačních problémů.

Jak již bylo řečeno, přidáváme-li zaměstnance poté, co jsme dosáhli optimálního počtu, začnou celkové roční náklady růst. Množinu $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$ můžeme tedy s klidným svědomím „rozumně“ omezit.

V době zpracovávání projektu pracovalo na oddělení SÚ asi 37 zaměstnanců a na oddělení KK asi 23 zaměstnanců. Z analýzy dat jsme zjistili, že při takovém složení zaměstnanců by měli v PS zvládat vyřizovat téměř vždy všechny žádosti v rámci prvních tří pracovních dnů. Počet univerzálů byl velmi nejasný, a proto jsme je při analýze dat do našeho zkoumání nezařadili.

Zavedeme si novou množinu

$$D_0 = \{0, 1, \ldots, 63\} \times \{0, 1, \ldots, 31\} \times \{0, 1, \ldots, 31\}, \tag{3.7}$$

která vznikne omezením $\mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0$. Její první rozměr odpovídá počtu pracovníků na oddělení SÚ, další rozměry odpovídají počtu pracovníků z oddělení KK a počtu univerzálů. Zjednodušili jsme tedy problém nalezení minima (3.6) na

$$\mathbf{x}_{opt} = \underset{\mathbf{x}\in D_0}{\arg\min} f(\mathbf{x}), \tag{3.8}$$

kde množina D_0 je definovaná v (3.7).

Množinu D_0 snadno zakódujeme pomocí binárního řetězce. Každý rozměr zakódujeme zvlášť a následně všechny tři kódy spojíme. Množinu $\{0, 1, \ldots, 63\}$ zakódujeme šestimístným binárním řetězcem. Oba další rozměry stačí kódovat pomocí pětimístného binárního řetězce stejně jako v příkladu z části 1.2.4. Dostáváme tak šestnáctimístný kód, který udává složení zaměstnanců, například řetězec (011110|10001|10100) odpovídá vektoru $(30, 17, 20)^T$. Formálně definujeme bijekci

$$\Phi : D_0 \to \{0, 1\}^{16}, \mathbf{x} \mapsto \boldsymbol{\alpha},$$

která převádí vektor celých čísel z množiny D_0 na binární kód.

Tím jsme převedli minimalizaci funkce f na množině D_0, viz (3.8), na problém hledání minima funkce $f \circ \Phi^{-1}$ na množině $\{0,1\}^{16}$

$$\begin{aligned} \mathbf{x}_{opt} &= \Phi^{-1}(\boldsymbol{\alpha}_{opt}), \\ \boldsymbol{\alpha}_{opt} &= \arg\min_{\boldsymbol{\alpha}\in\{0,1\}^{16}} f(\Phi^{-1}(\boldsymbol{\alpha})). \end{aligned}$$

Takovýto druh optimalizačního problému jsme si popsali v kapitole 1. Na tomto příkladu si vyzkoušíme, jak GA funguje. Vyzkoušíme různé typy GA představených v kapitole 1 a všechny je nakonci této kapitoly zhodnotíme.

3.6 Hledání optima pomocí genetických algoritmů

V této části je možné najít výsledky dosažené pomocí GA, které budou vždy shrnuty v přehledných tabulkách. Připomeňme, že při výpočtu optimálního počtu a rozložení zaměstnanců pomocí GA jsme „průběh roku“ simulovali vždy 10 000–krát.

3.6.1 Strategie `plus(15+100)`

Jak napovídá název této části, první GA použitý pro hledání minima průměrných celkových ročních nákladů se šířil strategií `plus(15+100)`. Dále byl zvolen základní typ mutace (1.7) a jednobodové křížení pro každý podřetězec zvlášť. Fitness byla počítána dle vzorce (1.3) s úpravou (1.4). Parametr ε, jemuž se rovná minimální hodnota fitness F_{min}, byl nastaven na 0,01. Pravděpodobnost křížení chromozómů P_{cross} byla nastavena na 0,9 a pravděpodobnost mutace jednoho genu P_{mut} na 0,05. Pravděpodobnost reprodukce byla nastavena na 1.

Algoritmus s touto strategií šíření poměrně rychle konverguje k nějakému suboptimálnímu řešení, avšak globální optimum najde jen zřídka. V našem případě to ale příliš nevadí, použijeme GA s touto strategií šíření a různé jeho další varianty, abychom zjistili, kde se zhruba globální optimum nalézá. Poté ještě více omezíme množinu přípustných řešení, kterou prohledáme důkladněji pomocí algoritmu se strategií šíření `čárka`.

Popišme si zmíněné křížení podrobněji. Aplikací operace křížení O_{cross} na chromozómy $\boldsymbol{\alpha}, \boldsymbol{\beta}$ vzniknou jako obvykle dva chromozómy $\boldsymbol{\alpha}', \boldsymbol{\beta}'$. Jak ale přesně křížení probíhá? Generujeme náhodně tři body křížení $p \in \{1,2,\ldots,6\}$, $q \in \{7,8,\ldots,11\}$ a $r \in \{12,13,\ldots,16\}$.

$$\alpha_i' = \begin{cases} \alpha_i, & i = 1,2,\ldots,p,7,8,\ldots,q,12,13,\ldots,r \\ \beta_i, & i = p+1,p+2,\ldots,6,q+1,q+2,\ldots,11,r+1,r+2,\ldots,16 \end{cases}$$
$$\beta_i' = \begin{cases} \beta_i, & i = 1,2,\ldots,p,7,8,\ldots,q,12,13,\ldots,r \\ \alpha_i, & i = p+1,p+2,\ldots,6,q+1,q+2,\ldots,11,r+1,r+2,\ldots,16 \end{cases}$$

Pokud se při křížení stane, že některý z bodů křížení p, q nebo r nabude nejvyšší hodnoty množiny, ze které je generován, vymění si chromozómy celý příslušný podchromozóm.

Výpočet se zastavil, pokud byly všechny chromozómy v populaci stejné nebo když GA dosáhl třísté generace, avšak k druhému případu nikdy nedošlo. Nejvyšší počet dosažených generací byl 13.

Při výpočtu se ukládá celkem 15 nejlepších nalezených řešení, což nám pomůže při komentování dosažených výsledků. Výsledky získané pomocí tohoto GA nalezneme v tabulkách 3.9, 3.10 a 3.11, ze kterých lze vyčíst, jak dlouho výpočet trval, kolik generací GA prošel nebo počet navštívených chromozómů.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 5, 1)	31 029 713	(0/1/0)	(0,19/0,81/0)	(0,94/0,06/0)	(0,95/0,05/0)
(34, 4, 1)	31 055 555	(0,34/0,63/0,03)	(0,25/0,75/0)	(0,94/0,06/0)	(0,95/0,05/0)
(36, 4, 0)	31 268 808	(0,34/0,64/0,02)	(0,28/0,72/0)	(0,94/0,06/0)	(0,95/0,05/0)
(33, 6, 1)	31 562 283	(0,33/0,60/0,07)	(0,28/0,72/0)	(0,94/0,06/0)	(0,95/0,05/0)
(34, 5, 2)	31 837 019	(0,33/0,64/0,03)	(0,25/0,75/0)	(0,94/0,06/0)	(0,95/0,05/0)
(36, 4, 1)	31 861 804	(0,33/0,64/0,03)	(0,21/0,79/0)	(0,94/0,06/0)	(0,95/0,05/0)
(36, 4, 2)	32 494 853	(0,36/0,64/0)	(0,26/0,74/0)	(0,94/0,06/0)	(0,95/0,05/0)
(36, 5, 2)	33 128 393	(0,33/0,63/0,04)	(0,25/0,75/0)	(0,95/0,05/0)	(0,95/0,05/0)
(32, 4, 1)	33 614 619	(0,34/0,64/0,02)	(0,27/0,73/0)	(0,94/0,06/0)	(0,95/0,05/0)
(38, 2, 3)	33 669 404	(0,34/0,64/0,02)	(0,28/0,72/0)	(0,94/0,06/0)	(0,95/0,05/0)
(38, 4, 2)	34 033 296	(0/1/0)	(0,19/0,81/0)	(0,94/0,06/0)	(0,95/0,05/0)
(38, 5, 2)	34 668 354	(0,33/0,64/0,03)	(0,23/0,77/0)	(0,94/0,06/0)	(0,95/0,05/0)
(38, 4, 3)	34 959 807	(0,35/0,64/0,01)	(0,28/0,72/0)	(0,94/0,06/0)	(0,95/0,05/0)
(32, 16, 1)	36 201 479	(0,34/0,64/0,02)	(0,28/0,72/0)	(0,94/0,06/0)	(0,95/0,05/0)
(27, 0, 16)	36 282 332	(0,34/0,64/0,02)	(0,26/0,74/0)	(0,94/0,06/0)	(0,95/0,05/0)

Čas zahájení výpočtu	17 : 15 : 00	Počet generací algoritmu	8
Čas ukončení výpočtu	22 : 10 : 41	Počet navštívených chromozómů	487

Tabulka 3.9: Výsledky GA se strategií šírení `plus(15+100)`

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(33, 5, 1)	30 967 807	(0,28/0,48/0,24)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 3, 2)	30 992 100	(0,26/0,44/0,30)	(0,20/0,80/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 2, 3)	31 261 430	(0,28/0,45/0,27)	(0,16/0,84/0)	(0,79/0,21/0)	(0,96/0,04/0)
(35, 4, 1)	31 287 488	(0,10/0,64/0,26)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 3, 3)	31 475 899	(0,28/0,46/0,26)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 6, 1)	31 562 083	(0,08/0,66/0,26)	(0,21/0,79/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 7, 1)	31 589 706	(0,14/0,63/0,23)	(0,16/0,84/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 4, 1)	31 601 886	(0,08/0,68/0,24)	(0,22/0,78/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 7, 1)	31 680 232	(0,26/0,46/0,28)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)
(36, 4, 1)	31 868 006	(0,09/0,67/0,24)	(0,19/0,81/0)	(0,79/0,21/0)	(0,96/0,04/0)
(36, 1, 3)	32 002 977	(0,15/0,70/0,15)	(0,23/0,77/0)	(0,79/0,21/0)	(0,96/0,04/0)
(36, 2, 2)	32 022 101	(0,10/0,65/0,25)	(0,12/0,88/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 8, 1)	32 241 495	(0,11/0,63/0,26)	(0,16/0,84/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 7, 2)	32 366 782	(0,28/0,43/0,29)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(37, 4, 1)	32 588 863	(0,26/0,44/0,30)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)

Čas zahájení výpočtu	11 : 05 : 58	Počet generací algoritmu	9
Čas ukončení výpočtu	16 : 23 : 35	Počet navštívených chromozómů	534

Tabulka 3.10: Výsledky GA se strategií šíření `plus(15+100)`

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(31, 3, 5)	32 177 798	(0,30/0,56/0,14)	(0,54/0,46/0)	(0,67/0,33/0)	(1/0/0)
(36, 0, 4)	32 209 743	(0,23/0,62/0,15)	(0,56/0,44/0)	(0,67/0,33/0)	(1/0/0)
(31, 2, 6)	32 244 234	(0,29/0,50/0,21)	(0,51/0,49/0)	(0,67/0,33/0)	(1/0/0)
(31, 4, 4)	32 270 841	(0,30/0,51/0,19)	(0,54/0,46/0)	(0,67/0,33/0)	(1/0/0)
(31, 6, 4)	32 638 740	(0,30/0,53/0,17)	(0,50/0,50/0)	(0,67/0,33/0)	(1/0/0)
(31, 4, 5)	32 820 985	(0,27/0,59/0,14)	(0,53/0,47/0)	(0,67/0,33/0)	(1/0/0)
(32, 6, 4)	32 826 879	(0,31/0,63/0,06)	(0,53/0,47/0)	(0,67/0,33/0)	(1/0/0)
(31, 5, 4)	32 915 830	(0,28/0,63/0,09)	(0,54/0,46/0)	(0,67/0,33/0)	(1/0/0)
(30, 4, 6)	32 968 753	(0,29/0,48/0,23)	(0,56/0,44/0)	(0,67/0,33/0)	(1/0/0)
(36, 2, 4)	33 063 845	(0,29/0,59/0,12)	(0,55/0,45/0)	(0,67/0,33/0)	(1/0/0)
(31, 3, 4)	33 066 574	(0,28/0,60/0,12)	(0,53/0,47/0)	(0,67/0,33/0)	(1/0/0)
(30, 6, 4)	33 241 593	(0,31/0,52/0,17)	(0,51/0,49/0)	(0,67/0,33/0)	(1/0/0)
(38, 2, 2)	33 324 071	(0,30/0,60/0,10)	(0,54/0,46/0)	(0,67/0,33/0)	(1/0/0)
(31, 2, 4)	33 377 097	(0,30/0,55/0,15)	(0,46/0,54/0)	(0,67/0,33/0)	(1/0/0)
(30, 10, 2)	33 405 216	(0,29/0,47/0,24)	(0,54/0,46/0)	(0,67/0,33/0)	(1/0/0)

Čas zahájení výpočtu	1 : 04 : 51	Počet generací algoritmu	10
Čas ukončení výpočtu	7 : 23 : 09	Počet navštívených chromozómů	633

Tabulka 3.11: Výsledky GA se strategií šíření `plus(15+100)`

3.6.2 Odstranění Hammingovy bariéry

V této části se zaměříme na odstranění Hammingovy bariéry. Nejdříve k tomu použijeme Grayovo kódování a pak inverzní operátor. Jinak ponecháme algoritmus nezměněný.

Grayovo kódování

Zde využijeme pouze převod z Grayova kódování na standartní binární kód. Na začátku generujeme populaci chromozómů náhodně, není ji tedy nutné převádět na Grayův kód, a tak můžeme říci, že už je v Grayově kódu vygenerovaná. V průběhu výpočtu se převádí binární kód chromozómu na vektor rozložení zaměstnanců. Ten je následně vstupem simulace, jež vrací průměrné celkové roční náklady. Přidáme před převod binárního kódu na vektor rozložení zaměstnanců ještě převod Grayova kódování na standartní binární kód. Tento převod navíc nebude téměř vadit, neboť samotný výpočet průměrných celkových ročních nákladů při daném rozložení zaměstnanců trvá procesoru mnohem déle.

Výsledky dosažené pomocí GA, ve kterém bylo použito Grayovo kódování, je možno nalézt v tabulkách 3.12 a 3.13.

Inverzní operátor

Druhým způsobem jak odstranit Hammingovu bariéru je inverzní operátor, jenž byl upraven podobně jako křížení. Inverze probíhá odděleně pro každý „podchromozóm“ zvlášť. I tuto operaci si popíšeme detailněji. Pravděpodobnost inverze P_{inv} byla nastavena na 0,01 pro každý „podchromozóm“. Začneme s prvním „podchromozómem“ nacházejícím se na indexech $1, 2, \ldots, 6$. Nejdříve generujeme náhodně číslo z intervalu $[0, 1)$ a pokud je menší než 0,01 generujeme náhodně bod inverze a_1 z množiny $\{2, 3, \ldots, 6\}$. Až do tohoto bodu ponecháme chromozóm nezměněný, od tohoto bodu včetně až do pozice 6 včetně nahradíme všechny geny jejich komplimenty, tj.

$$\alpha_i^{'} = \begin{cases} \alpha_i, & i = 1, 2, \ldots, a_1 - 1 \\ 1 - \alpha_i, & i = a_1, a_1 + 1 \ldots, 6 \end{cases} \tag{3.9}$$

Podobně postupujeme i pro druhý „podchromozóm“, který zabírá pozice $7, 8, \ldots, 11$. Opět nejdříve generujeme náhodné číslo z $[0, 1)$; je-li menší než P_{inv}, dojde k inverzi. Generujeme další bod křížení a_2 z množiny $\{8, 9, \ldots, 11\}$. Geny s indexy $7, 8, \ldots, a_2 - 1$ ponecháme beze změny, zatímco na pozicích $a_2, a_2 + 1, \ldots, 11$ nahradíme geny jejich komplimenty, tj.

$$\alpha_i^{'} = \begin{cases} \alpha_i, & i = 7, 8, \ldots, a_2 - 1 \\ 1 - \alpha_i, & i = a_2, a_2 + 1, \ldots, 11 \end{cases} \tag{3.10}$$

Postup pro třetí „pochromozóm“ vynecháme, neboť je zcela analogický.

Výsledky GA s inverzním operátorem můžeme zhlédnout v tabulkách 3.14 a 3.15.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 4, 1)	31 077 457	(0,21/0,47/0,32)	(0,03/0,97/0)	(0,79/0,21/0)	(0,57/0,43/0)
(35, 4, 1)	31 295 808	(0,24/0,57/0,19)	(0,01/0,99/0)	(0,79/0,21/0)	(0,57/0,43/0)
(37, 4, 0)	31 596 595	(0,13/0,44/0,43)	(0,01/0,99/0)	(0,79/0,21/0)	(0,57/0,43/0)
(35, 4, 0)	31 613 630	(0,21/0,59/0,20)	(0,04/0,96/0)	(0,79/0,21/0)	(0,57/0,43/0)
(32, 4, 2)	31 648 219	(0,21/0,54/0,25)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(36, 4, 1)	31 878 506	(0,20/0,48/0,32)	(0,03/0,97/0)	(0,79/0,21/0)	(0,57/0,43/0)
(32, 6, 3)	32 471 902	(0,23/0,59/0,18)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(37, 4, 1)	32 594 246	(0,21/0,52/0,27)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(31, 6, 4)	32 625 218	(0,22/0,59/0,19)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(31, 5, 5)	32 736 344	(0,24/0,53/0,23)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(31, 4, 5)	32 806 310	(0,22/0,57/0,21)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(32, 6, 4)	32 828 192	(0,24/0,56/0,20)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(30, 5, 5)	33 084 284	(0,22/0,54/0,24)	(0,02/0,98/0)	(0,79/0,21/0)	(0,57/0,43/0)
(31, 7, 5)	33 636 534	(0,22/0,54/0,24)	(0,03/0,97/0)	(0,79/0,21/0)	(0,57/0,43/0)
(39, 4, 1)	34 129 898	(0,21/0,58/0,21)	(0,05/0,95/0)	(0,79/0,21/0)	(0,57/0,43/0)

Čas zahájení výpočtu	21 : 29 : 46	Počet generací algoritmu	9
Čas ukončení výpočtu	2 : 35 : 42	Počet navštívených chromozómů	498

Tabulka 3.12: Výsledky GA se strategií šírení `plus(15+100)` a Grayovým kódováním.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(31, 2, 6)	32 236 608	(0,45/0,54/0,01)	(0,20/0,80/0)	(0,68/0,32/0)	(1/0/0)
(31, 2, 7)	32 857 450	(0,33/0,60/0,07)	(0,34/0,66/0)	(0,68/0,32/0)	(1/0/0)
(31, 5, 4)	32 907 105	(0,20/0,75/0,05)	(0,22/0,78/0)	(0,68/0,32/0)	(1/0/0)
(31, 2, 5)	33 048 899	(0,46/0,52/0,02)	(0,35/0,65/0)	(0,68/0,32/0)	(1/0/0)
(31, 2, 8)	33 237 156	(0,22/0,77/0,01)	(0,24/0,76/0)	(0,68/0,32/0)	(1/0/0)
(31, 7, 4)	33 284 589	(0,39/0,61/0)	(0,33/0,67/0)	(0,68/0,32/0)	(1/0/0)
(31, 2, 4)	33 395 464	(0,43/0,57/0)	(0,27/0,73/0)	(0,68/0,32/0)	(1/0/0)
(33, 2, 8)	34 500 458	(0,35/0,62/0,03)	(0,32/0,68/0)	(0,68/0,32/0)	(1/0/0)
(27, 2, 12)	34 644 872	(0,34/0,62/0,04)	(0,39/0,61/0)	(0,68/0,32/0)	(1/0/0)
(31, 2, 10)	34 869 891	(0,20/0,75/0,05)	(0,36/0,64/0)	(0,68/0,32/0)	(1/0/0)
(27, 2, 10)	35 054 650	(0,29/0,58/0,13)	(0,37/0,63/0)	(0,68/0,32/0)	(1/0/0)
(27, 2, 11)	35 102 690	(0,21/0,76/0,03)	(0,42/0,58/0)	(0,68/0,32/0)	(1/0/0)
(27, 2, 13)	35 248 654	(0,23/0,76/0,01)	(0,38/0,62/0)	(0,68/0,32/0)	(1/0/0)
(27, 5, 11)	35 337 897	(0,20/0,71/0,09)	(0,38/0,62/0)	(0,68/0,32/0)	(1/0/0)
(25, 2, 14)	35 565 378	(0,21/0,77/0,02)	(0,32/0,68/0)	(0,68/0,32/0)	(1/0/0)

Čas zahájení výpočtu	18 : 38 : 00	Počet generací algoritmu	9
Čas ukončení výpočtu	23 : 48 : 22	Počet navštívených chromozómů	504

Tabulka 3.13: Výsledky GA se strategií šírení `plus(15+100)` a Grayovým kódováním.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 3, 2)	31 001 386	(0,30/0,39/0,31)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 4, 2)	31 224 706	(0,32/0,46/0,22)	(0,12/0,88/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 2, 3)	31 275 432	(0,32/0,43/0,25)	(0,09/0,91/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 3, 3)	31 440 089	(0,32/0,42/0,26)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 6, 1)	31 636 953	(0,33/0,53/0,14)	(0,14/0,86/0)	(0,79/0,21/0)	(0,66/0,34/0)
(32, 6, 2)	31 751 723	(0,33/0,50/0,17)	(0,14/0,86/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 5, 2)	31 852 663	(0,32/0,39/0,29)	(0,09/0,91/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 0, 5)	32 000 718	(0,32/0,42/0,26)	(0,10/0,90/0)	(0,79/0,21/0)	(0,66/0,34/0)
(36, 2, 2)	32 033 485	(0,32/0,44/0,24)	(0,12/0,88/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 6, 2)	32 316 435	(0,34/0,52/0,14)	(0,16/0,84/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 2, 2)	32 347 000	(0,32/0,41/0,27)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
(32, 8, 2)	32 425 357	(0,32/0,48/0,20)	(0,15/0,85/0)	(0,79/0,21/0)	(0,66/0,34/0)
(32, 6, 1)	32 485 513	(0,31/0,41/0,28)	(0,12/0,88/0)	(0,79/0,21/0)	(0,66/0,34/0)
(31, 8, 2)	32 554 896	(0,35/0,50/0,15)	(0,10/0,90/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 5, 3)	32 585 262	(0,35/0,55/0,10)	(0,16/0,84/0)	(0,79/0,21/0)	(0,66/0,34/0)

Čas zahájení výpočtu	8 : 38 : 35	Počet generací algoritmu	9
Čas ukončení výpočtu	21 : 08 : 27	Počet navštívených chromozómů	569

Tabulka 3.14: Výsledky GA se strategií šírení `plus(15+100)` a inverzním operátorem.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 3, 2)	31 008 829	(0,35/0,63/0,02)	(0,12/0,88/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 2, 3)	31 290 037	(0,38/0,52/0,10)	(0,12/0,88/0)	(0,78/0,22/0)	(0,65/0,35/0)
(35, 3, 2)	31 293 636	(0,38/0,61/0,01)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 3, 2)	31 401 087	(0,39/0,53/0,08)	(0,16/0,84/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 2, 4)	31 403 589	(0,39/0,54/0,07)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,357/0)
(33, 1, 5)	31 726 765	(0,38/0,62/0)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 2, 3)	31 764 970	(0,39/0,53/0,08)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
(35, 2, 2)	31 829 992	(0,37/0,63/0)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 0, 5)	32 027 676	(0,39/0,58/0,03)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
(35, 0, 4)	32 286 029	(0,38/0,53/0,09)	(0,14/0,86/0)	(0,78/0,22/0)	(0,65/0,35/0)
(33, 1, 6)	32 382 239	(0,36/0,63/0,01)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 2, 5)	32 576 654	(0,39/0,55/0,06)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 2, 6)	32 732 349	(0,39/0,53/0,08)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 1, 6)	32 851 962	(0/1/0)	(0,12/0,88/0)	(0,79/0,21/0)	(0,65/0,35/0)
(32, 2, 3)	33 604 260	(0/1/0)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)

Čas zahájení výpočtu	16 : 32 : 35	Počet generací algoritmu	8
Čas ukončení výpočtu	21 : 34 : 01	Počet navštívených chromozómů	494

Tabulka 3.15: Výsledky GA se strategií šírení `plus(15+100)` a inverzním operátorem.

3.6.3 Strategie `čárka(15,100)`

Další typ GA, jenž vyzkoušíme, bude stejný jako první zmíněný, ovšem s jinou strategií šíření. Do další generace bude postupovat 15 nejsilnějších chromozómů z populace $\mathcal{Q}$ čítající 100 chromozómů. Kvůli této strategii vypadlo z GA elitářství, které zaručovalo neustálé vylepšování jedinců. Při této strategii šíření si tedy můžeme při přechodu od jedné generace ke druhé i pohoršit. Za to algoritmus navštíví více různých chromozómů. Navíc ke zhoršení moc často nedochází, neboť nejsilnější jedinci mají nejvyšší pravděpodobnost výběru do další generace. Jelikož ovšem nemůžeme zaručit, že v poslední generaci bude přítomný nejlepší nalezený chromozóm, ukládáme pro jistotu 15 nejlepších jedinců, což nám opět pomůže i při diskuzi nejlepšího rozložení zaměstnanců.

I tento typ GA modifikujeme pomocí inverzního operátoru a Grayova kódování. V tabulkách 3.16, 3.17, 3.18, 3.19, 3.20 a 3.21 najdeme výsledky různých GA se strategií `čárka(15,100)`.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(31, 0, 8)	32 796 234	(0,26/0,71/0,03)	(0,08/0,92/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 2, 8)	33 014 507	(0,17/0,49/0,34)	(0,02/0,98/0)	(0,68/0,32/0)	(0,94/0,06/0)
(31, 1, 8)	33 106 849	(0,24/0,58/0,18)	(0,06/0,94/0)	(0,68/0,32/0)	(0,94/0,06/0)
(31, 2, 8)	33 237 552	(0,19/0,70/0,11)	(0,09/0,91/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 2, 8)	33 469 614	(0,19/0,55/0,26)	(0,09/0,91/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 4, 8)	33 727 996	(0,19/0,63/0,18)	(0,11/0,89/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 0, 10)	33 967 516	(0,21/0,60/0,19)	(0,13/0,87/0)	(0,68/0,32/0)	(0,94/0,06/0)
(31, 4, 8)	34 338 478	(0,19/0,73/0,08)	(0/1/0)	(0,69/0,31/0)	(0,94/0,06/0)
(29, 5, 8)	34 383 167	(0,19/0,71/0,10)	(0,10/0,90/0)	(0,69/0,31/0)	(0,94/0,06/0)
(28, 4, 8)	34 399 556	(0,19/0,69/0,12)	(0,02/0,98/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 0, 12)	34 406 709	(0,20/0,72/0,08)	(0,16/0,84/0)	(0,68/0,32/0)	(0,94/0,06/0)
(28, 1, 12)	34 741 041	(0,17/0,50/0,33)	(0,04/0,96/0)	(0,68/0,32/0)	(0,94/0,06/0)
(28, 8, 8)	35 175 283	(0,19/0,62/0,19)	(0,11/0,89/0)	(0,68/0,32/0)	(0,94/0,06/0)
(27, 1, 12)	35 282 049	(0,19/0,71/0,10)	(0,17/0,83/0)	(0,68/0,32/0)	(0,94/0,06/0)
(28, 6, 10)	35 445 950	(0,24/0,63/0,13)	(0,10/0,90/0)	(0,68/0,32/0)	(0,94/0,06/0)

Čas zahájení výpočtu	22 : 07 : 24	Počet generací algoritmu	8
Čas ukončení výpočtu	3 : 16 : 21	Počet navštívených chromozómů	510

Tabulka 3.16: Výsledky GA se strategií šíření `čárka(15,100)`.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 5, 1)	31 002 890	(0,28/0,68/0,04)	(0,26/0,74/0)	(0,95/0,05/0)	(0,95/0,05/0)
(34, 4, 1)	31 022 671	(0,48/0,50/0,02)	(0,25/0,75/0)	(0,95/0,05/0)	(0,95/0,05/0)
(33, 2, 4)	31 350 722	(0,29/0,69/0,02)	(0,24/0,76/0)	(0,95/0,05/0)	(0,96/0,04/0)
(33, 6, 1)	31 522 780	(0,47/0,51/0,02)	(0,25/0,75/0)	(0,95/0,05/0)	(0,95/0,05/0)
(33, 1, 5)	31 677 371	(0,29/0,68/0,03)	(0,14/0,86/0)	(0,95/0,05/0)	(0,95/0,05/0)
(34, 2, 4)	31 688 442	(0,31/0,68/0,01)	(0,24/0,76/0)	(0,95/0,05/0)	(0,95/0,05/0)
(32, 2, 4)	31 700 139	(0,28/0,70/0,02)	(0,18/0,82/0)	(0,95/0,05/0)	(0,95/0,05/0)
(32, 6, 2)	31 703 558	(0,24/0,76/0)	(0,19/0,81/0)	(0,95/0,05/0)	(0,95/0,05/0)
(34, 0, 5)	31 939 475	(0,30/0,68/0,02)	(0,21/0,79/0)	(0,95/0,05/0)	(0,95/0,05/0)
(32, 2, 5)	32 047 958	(0,31/0,69/0)	(0,21/0,79/0)	(0,95/0,05/0)	(0,95/0,05/0)
(33, 2, 5)	32 071 232	(0,31/0,68/0,01)	(0,26/0,74/0)	(0,94/0,06/0)	(0,95/0,05/0)
(35, 0, 5)	32 158 606	(0,26/0,72/0,02)	(0,22/0,78/0)	(0,95/0,05/0)	(0,95/0,05/0)
(33, 4, 4)	32 222 478	(0,30/0,67/0,03)	(0,26/0,74/0)	(0,95/0,05/0)	(0,95/0,05/0)
(34, 2, 5)	32 565 705	(0,26/0,69/0,05)	(0,16/0,84/0)	(0,95/0,05/0)	(0,95/0,05/0)
(37, 2, 2)	32 596 901	(0,28/0,69/0,03)	(0,19/0,81/0)	(0,95/0,05/0)	(0,95/0,05/0)

Čas zahájení výpočtu	10 : 34 : 45	Počet generací algoritmu	12
Čas ukončení výpočtu	21 : 08 : 27	Počet navštívených chromozómů	526

Tabulka 3.17: Výsledky GA se strategií šíření `čárka(15,100)`.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(33, 4, 2)	31 020 387	(0,31/0,62/0,07)	(0,02/0,98/0)	(0,78/0,22/0)	(0,58/0,42/0)
(34, 4, 1)	31 064 155	(0,30/0,63/0,07)	(0,02/0,98/0)	(0,78/0,22/0)	(0,58/0,42/0)
(34, 4, 2)	31 210 011	(0,32/0,60/0,08)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(36, 4, 0)	31 275 681	(0,29/0,68/0,03)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(34, 6, 1)	31 621 778	(0,33/0,60/0,07)	(0,03/0,97/0)	(0,79/0,21/0)	(0,58/0,42/0)
(35, 4, 2)	31 785 325	(0,32/0,54/0,14)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(35, 2, 2)	31 789 439	(0,29/0,69/0,02)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(33, 5, 3)	32 004 503	(0,33/0,64/0,03)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(34, 4, 3)	32 078 489	(0,33/0,65/0,02)	(0,03/0,97/0)	(0,79/0,21/0)	(0,58/0,42/0)
(36, 2, 3)	32 177 180	(0,31/0,60/0,09)	(0,01/0,99/0)	(0,79/0,21/0)	(0,58/0,42/0)
(36, 6, 0)	32 567 958	(0,33/0,61/0,06)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(37, 4, 1)	32 584 643	(0,31/0,57/0,12)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
(35, 4, 3)	32 688 677	(0,26/0,71/0,03)	(0,03/0,97/0)	(0,79/0,21/0)	(0,58/0,42/0)
(36, 6, 1)	32 845 126	(0,35/0,63/0,02)	(0,01/0,99/0)	(0,78/0,22/0)	(0,57/0,43/0)
(34, 4, 4)	32 848 580	(0,31/0,64/0,05)	(0,02/0,98/0)	(0,79/0,21/0)	(0,58/0,42/0)
Čas zahájení výpočtu		13 : 58 : 05	Počet generací algoritmu		14
Čas ukončení výpočtu		20 : 48 : 48	Počet navštívených chromozómů		687

Tabulka 3.18: Výsledky GA se strategií šírení `čárka(15,100)` a Grayovým kódováním.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(34, 3, 2)	30 976 068	(0,28/0,69/0,03)	(0,16/0,84/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 1, 5)	32 000 254	(0,33/0,64/0,03)	(0,12/0,88/0)	(0,80/0,20/0)	(0,66/0,34/0)
(36, 2, 2)	32 017 880	(0,33/0,63/0,04)	(0,15/0,85/0)	(0,80/0,20/0)	(0,66/0,34/0)
(35, 3, 3)	32 037 803	(0,27/0,68/0,05)	(0,17/0,83/0)	(0,79/0,21/0)	(0,66/0,34/0)
(35, 5, 2)	32 412 629	(0,32/0,65/0,03)	(0,16/0,84/0)	(0,79/0,21/0)	(0,66/0,34/0)
(37, 2, 2)	32 587 431	(0,33/0,65/0,02)	(0,15/0,85/0)	(0,80/0,20/0)	(0,66/0,34/0)
(30, 2, 8)	32 985 677	(0,36/0,63/0,01)	(0,15/0,85/0)	(0,80/0,20/0)	(0,66/0,34/0)
(34, 3, 5)	33 125 448	(0,35/0,63/0,02)	(0,13/0,87/0)	(0,80/0,20/0)	(0,65/0,35/0)
(37, 5, 2)	33 893 163	(0,31/0,68/0,01)	(0,14/0,86/0)	(0,80/0,20/0)	(0,66/0,34/0)
(29, 10, 4)	34 160 773	(0,36/0,63/0,01)	(0,10/0,90/0)	(0,80/0,20/0)	(0,66/0,34/0)
(35, 5, 4)	34 226 782	(0,32/0,65/0,03)	(0,12/0,88/0)	(0,80/0,20/0)	(0,66/0,34/0)
(29, 8, 4)	34 296 308	(0,40/0,59/0,01)	(0,16/0,84/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 2, 7)	34 336 676	(0,24/0,75/0,01)	(0,11/0,89/0)	(0,80/0,20/0)	(0,66/0,34/0)
(29, 11, 4)	34 806 534	(0,36/0,63/0,01)	(0,14/0,86/0)	(0,79/0,21/0)	(0,66/0,34/0)
(29, 11, 5)	34 850 906	(0,35/0,62/0,03)	(0,13/0,87/0)	(0,79/0,21/0)	(0,66/0,34/0)

Čas zahájení výpočtu	20 : 52 : 03	Počet generací algoritmu	7
Čas ukončení výpočtu	2 : 28 : 30	Počet navštívených chromozómů	550

Tabulka 3.19: Výsledky GA se strategií šíření `čárka(15,100)` a Grayovým kódováním.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(33, 5, 1)	30 975 923	(0,28/0,36/0,36)	(0,10/0,90/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 3, 2)	31 004 061	(0,25/0,43/0,32)	(0,26/0,74/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 4, 2)	31 026 256	(0,32/0,37/0,31)	(0,26/0,74/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 4, 1)	31 069 277	(0,27/0,33/0,40)	(0,21/0,79/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 3, 3)	31 156 176	(0,32/0,38/0,30)	(0,15/0,85/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 4, 2)	31 235 307	(0,27/0,31/0,42)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 2, 3)	31 275 301	(0,22/0,40/0,38)	(0,16/0,84/0)	(0,79/0,21/0)	(0,96/0,04/0)
(35, 3, 2)	31 309 782	(0,25/0,32/0,43)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 3, 2)	31 339 115	(0,27/0,32/0,41)	(0,21/0,79/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 3, 3)	31 446 155	(0,17/0,32/0,51)	(0,20/0,80/0)	(0,79/0,21/0)	(0,96/0,04/0)
(35, 5, 1)	31 569 551	(0,24/0,43/0,33)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 4, 1)	31 597 373	(0,29/0,34/0,37)	(0,08/0,92/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 7, 1)	31 609 912	(0,26/0,34/0,40)	(0,26/0,74/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 1, 5)	31 724 218	(0,25/0,32/0,43)	(0,13/0,87/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 5, 3)	31 807 503	(0,27/0,35/0,38)	(0,21/0,79/0)	(0,79/0,21/0)	(0,96/0,04/0)

Čas zahájení výpočtu	11 : 42 : 50	Počet generací algoritmu	9
Čas ukončení výpočtu	16 : 53 : 08	Počet navštívených chromozómů	522

Tabulka 3.20: Výsledky GA se strategií šíření `čárka(15,100)` a inverzním operátorem.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(31, 0, 8)	32 755 593	(0,31/0,64/0,05)	(0,14/0,86/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 2, 8)	32 985 590	(0,23/0,57/0,20)	(0,20/0,80/0)	(0,68/0,32/0)	(0,94/0,06/0)
(31, 1, 8)	33 078 287	(0,39/0,53/0,08)	(0,13/0,87/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 1, 9)	33 232 548	(0,20/0,51/0,29)	(0,17/0,83/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 2, 8)	33 444 144	(0,29/0,61/0,10)	(0,20/0,80/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 0, 10)	33 504 237	(0,21/0,58/0,21)	(0,21/0,79/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 3, 8)	33 589 699	(0,21/0,55/0,24)	(0,14/0,86/0)	(0,68/0,32/0)	(0,94/0,06/0)
(30, 2, 9)	33 702 459	(0,20/0,56/0,24)	(0,13/0,87/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 4, 8)	33 707 418	(0,22/0,56/0,22)	(0,13/0,87/0)	(0,68/0,32/0)	(0,94/0,06/0)
(31, 0, 10)	33 769 642	(0,23/0,57/0,20)	(0,32/0,68/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 0, 10)	33 930 791	(0,25/0,59/0,16)	(0,24/0,76/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 1, 10)	34 065 508	(0,25/0,57/0,18)	(0,25/0,75/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 1, 11)	34 099 485	(0,22/0,60/0,18)	(0,15/0,85/0)	(0,68/0,32/0)	(0,94/0,06/0)
(28, 1, 11)	34 185 533	(0,27/0,59/0,14)	(0,14/0,86/0)	(0,68/0,32/0)	(0,94/0,06/0)
(29, 12, 2)	34 332 575	(0,23/0,59/0,18)	(0,24/0,76/0)	(0,68/0,32/0)	(0,94/0,06/0)

Čas zahájení výpočtu	17 : 20 : 03	Počet generací algoritmu	15
Čas ukončení výpočtu	1 : 29 : 07	Počet navštívených chromozómů	835

Tabulka 3.21: Výsledky GA se strategií šíření `čárka(15,100)` a inverzním operátorem.

Další omezení množiny přípustných řešení

Jelikož už jsme pomocí předešlých typů GA zjistili, že optimální počet zaměstnanců bude asi 37–40 a optimální počet úvěrářů bude asi 32–35, je na čase množinu D_0 trochu omezit. Zmenšíme prostor přípustných řešení, jenž algoritmus prohledává. Zaveďme si proto novou množinu

$$D_1 = \{30, 31, \ldots, 37\} \times \{0, 1, \ldots, 7\} \times \{0, 1, \ldots, 7\},$$

která vznikne omezením množiny D_0. Množinu D_1 stačí kódovat pomocí devítimístného binárního řetězce. Na každý její rozměr stačí třímístný binární kód. Tím jsme počet prvků množiny přípustných řešení redukovali z $|D_0| = 65\,536$ na $|D_1| = 512$.

Nebudeme používat strategii šíření `plus(m+n)`, protože ta často rychle konverguje k nějakému lokálnímu minimu. Použijeme strategii šíření `čárka(15,100)`, s jejíž pomocí navštívíme více chromozómů. Ostatní parametry GA, tj. P_{cross}, P_{mut}, ε, atd., ponecháme stejné.

K odstranění Hammingovy bariéry použijeme tentokrát jen Grayovo kódování, neboť se jeví lépe než inverzní operátor. Navíc jak již bylo řečeno, v našem případě je výpočetní čas strávený převodem Grayova kódování na standartní zcela zanedbatelný oproti času potřebnému k výpočtu průměrných nákladů.

Výsledky dosažené pomocí GA na této zmenšené množině přípustných řešení jsou shrnuty v tabulkách 3.22, 3.23, 3.24, 3.25 a 3.26.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
$(34, 3, 2)$	30 970 945	(0,22/0,67/0,11)	(0,06/0,94/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(34, 5, 1)$	31 021 779	(0,21/0,65/0,14)	(0,17/0,83/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 2, 3)$	31 235 181	(0,22/0,65/0,13)	(0,15/0,85/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(36, 4, 0)$	31 251 794	(0,23/0,67/0,10)	(0,12/0,88/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(35, 3, 2)$	31 289 090	(0,21/0,66/0,13)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(32, 6, 2)$	31 710 996	(0,23/0,62/0,15)	(0,13/0,87/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(34, 5, 2)$	31 834 082	(0,24/0,73/0,03)	(0,15/0,85/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(36, 1, 3)$	31 951 709	(0,22/0,68/0,10)	(0,14/0,86/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(36, 2, 2)$	32 002 349	(0,23/0,70/0,07)	(0,16/0,84/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(31, 3, 5)$	32 124 735	(0,20/0,77/0,03)	(0,15/0,85/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(36, 2, 3)$	32 170 048	(0,23/0,68/0,09)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(35, 5, 0)$	32 215 215	(0,25/0,74/0,01)	(0,15/0,85/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(36, 5, 1)$	32 245 116	(0,21/0,54/0,25)	(0,17/0,83/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 6, 2)$	32 308 739	(0,27/0,72/0,01)	(0,17/0,83/0)	(0,80/0,20/0)	(0,65/0,35/0)
$(34, 3, 4)$	32 309 130	(0,24/0,70/0,06)	(0,14/0,86/0)	(0,80/0,20/0)	(0,65/0,35/0)

Čas zahájení výpočtu	17 : 19 : 06	Počet generací algoritmu	4
Čas ukončení výpočtu	19 : 29 : 45	Počet navštívených chromozómů	201

Tabulka 3.22: Výsledky GA se strategií šíření `čárka(15,100)` na množině D_1.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(33, 5, 1)	30 993 891	(0,31/0,43/0,26)	(0,23/0,77/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 3, 3)	31 147 277	(0,28/0,33/0,39)	(0,19/0,81/0)	(0,79/0,21/0)	(0,96/0,04/0)
(36, 4, 0)	31 283 112	(0,28/0,44/0,28)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 3, 3)	31 479 673	(0,27/0,34/0,39)	(0,25/0,75/0)	(0,79/0,21/0)	(0,96/0,04/0)
(35, 2, 3)	31 575 881	(0,25/0,51/0,24)	(0,25/0,75/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 6, 1)	31 583 476	(0,28/0,28/0,44)	(0,15/0,85/0)	(0,79/0,21/0)	(0,96/0,04/0)
(37, 4, 0)	31 594 367	(0,22/0,48/0,30)	(0,15/0,85/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 4, 2)	31 608 191	(0,29/0,43/0,28)	(0,24/0,76/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 7, 1)	31 612 225	(0,30/0,44/0,26)	(0,20/0,80/0)	(0,79/0,21/0)	(0,96/0,04/0)
(33, 4, 1)	31 640 995	(0,29/0,32/0,39)	(0,26/0,74/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 5, 3)	31 826 981	(0,31/0,29/0,40)	(0,22/0,78/0)	(0,79/0,21/0)	(0,96/0,04/0)
(35, 1, 4)	31 886 881	(0,27/0,37/0,36)	(0,23/0,77/0)	(0,79/0,21/0)	(0,96/0,04/0)
(34, 1, 5)	32 015 517	(0,42/0,35/0,23)	(0,17/0,83/0)	(0,79/0,21/0)	(0,96/0,04/0)
(36, 1, 3)	32 021 390	(0,34/0,37/0,29)	(0,18/0,82/0)	(0,79/0,21/0)	(0,96/0,04/0)
(32, 1, 5)	32 159 888	(0,34/0,52/0,14)	(0,27/0,73/0)	(0,79/0,21/0)	(0,96/0,04/0)

Čas zahájení výpočtu	22 : 13 : 23	Počet generací algoritmu	4
Čas ukončení výpočtu	0 : 11 : 05	Počet navštívených chromozómů	179

Tabulka 3.23: Výsledky GA se strategií šíření `čárka(15,100)` na množině D_1.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
$(34, 3, 2)$	31 006 718	(0,38/0,59/0,03)	(0,07/0,93/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(33, 5, 1)$	31 013 314	(0,31/0,68/0,01)	(0,06/0,94/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 5, 1)$	31 057 796	(0,31/0,68/0,01)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 4, 1)$	31 093 376	(0,32/0,68/0)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(33, 3, 3)$	31 162 959	(0,06/0,94/0)	(0,04/0,96/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 4, 2)$	31 238 217	(0,33/0,67/0)	(0,06/0,94/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(35, 3, 2)$	31 310 925	(0,32/0,67/0,01)	(0,06/0,94/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 2, 3)$	31 314 576	(0,34/0,65/0,01)	(0,07/0,93/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(34, 3, 3)$	31 441 626	(0,38/0,57/0,05)	(0,13/0,87/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(33, 7, 1)$	31 616 899	(0,34/0,54/0,12)	(0,10/0,90/0)	(0,79/0,21/0)	(0,66/0,34/0)
$(34, 2, 4)$	31 722 102	(0,08/0,92/0)	(0,09/0,91/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(33, 1, 5)$	31 740 533	(0,31/0,66/0,03)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(32, 2, 4)$	31 775 793	(0,39/0,59/0,02)	(0,12/0,88/0)	(0,79/0,21/0)	(0,66/0,34/0)
$(34, 5, 2)$	31 868 097	(0,39/0,61/0)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
$(36, 3, 2)$	31 907 476	(0,03/0,97/0)	(0,11/0,89/0)	(0,79/0,21/0)	(0,66/0,34/0)

Čas zahájení výpočtu	18 : 53 : 50	Počet generací algoritmu	3
Čas ukončení výpočtu	20 : 35 : 21	Počet navštívených chromozómů	146

Tabulka 3.24: Výsledky GA se strategií šíření `čárka(15,100)` na množině D_1.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
(33, 5, 1)	30 959 545	(0,19/0,64/0,17)	(0,12/0,88/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 3, 2)	30 986 225	(0,18/0,51/0,31)	(0,16/0,84/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 5, 1)	31 036 641	(0,19/0,73/0,08)	(0,09/0,91/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 4, 1)	31 054 475	(0,19/0,73/0,08)	(0,14/0,86/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 2, 3)	31 251 745	(0,32/0,63/0,05)	(0,17/0,83/0)	(0,79/0,21/0)	(0,65/0,35/0)
(35, 4, 1)	31 285 003	(0,20/0,74/0,06)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
(35, 3, 2)	31 296 536	(0,18/0,53/0,29)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 3, 2)	31 308 921	(0,30/0,57/0,13)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 2, 4)	31 384 578	(0,19/0,60/0,21)	(0,08/0,92/0)	(0,79/0,21/0)	(0,66/0,34/0)
(35, 2, 3)	31 572 315	(0,20/0,67/0,13)	(0,09/0,91/0)	(0,79/0,21/0)	(0,65/0,35/0)
(35, 4, 0)	31 574 163	(0,21/0,79/0)	(0,11/0,89/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 4, 1)	31 586 071	(0,31/0,60/0,09)	(0,13/0,87/0)	(0,79/0,21/0)	(0,66/0,34/0)
(34, 1, 4)	31 633 058	(0,20/0,76/0,04)	(0,10/0,90/0)	(0,79/0,21/0)	(0,65/0,35/0)
(33, 2, 3)	31 667 318	(0,33/0,64/0,03)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)
(34, 2, 4)	31 706 722	(0,19/0,63/0,18)	(0,15/0,85/0)	(0,79/0,21/0)	(0,65/0,35/0)

Čas zahájení výpočtu	17 : 37 : 04	Počet generací algoritmu	6
Čas ukončení výpočtu	19 : 57 : 38	Počet navštívených chromozómů	219

Tabulka 3.25: Výsledky GA se strategií šíření `čárka(15,100)` a Grayovým kódováním na množině D_1.

rozložení	náklady	pomery SÚ	pomery KK	pomery s hlad. SÚ	pomery s hlad. KK
$(33, 5, 1)$	31 039 376	(0,20/0,40/0,40)	(0,26/0,74/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(34, 5, 1)$	31 066 342	(0,17/0,37/0,46)	(0,27/0,73/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(33, 4, 2)$	31 074 350	(0,24/0,41/0,35)	(0,25/0,75/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(34, 4, 1)$	31 118 042	(0,23/0,39/0,38)	(0,22/0,78/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(33, 3, 3)$	31 173 437	(0,18/0,41/0,41)	(0,26/0,74/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(36, 4, 0)$	31 304 930	(0,36/0,20/0,44)	(0,25/0,75/0)	(0,77/0,23/0)	(0,96/0,04/0)
$(33, 2, 4)$	31 423 580	(0,37/0,25/0,38)	(0,25/0,75/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(32, 3, 3)$	31 580 702	(0,19/0,38/0,43)	(0,22/0,78/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(32, 4, 2)$	31 737 061	(0,19/0,42/0,39)	(0,26/0,74/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(33, 1, 5)$	31 743 862	(0,18/0,45/0,37)	(0,24/0,76/0)	(0,77/0,23/0)	(0,96/0,04/0)
$(33, 4, 3)$	31 804 611	(0,18/0,40/0,42)	(0,18/0,82/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(33, 6, 2)$	31 816 400	(0,36/0,20/0,44)	(0,21/0,79/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(34, 7, 1)$	32 057 408	(0,38/0,27/0,35)	(0,21/0,79/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(35, 6, 1)$	32 158 834	(0,14/0,34/0,52)	(0,25/0,75/0)	(0,78/0,22/0)	(0,96/0,04/0)
$(36, 2, 3)$	32 180 916	(0,22/0,39/0,39)	(0,23/0,77/0)	(0,78/0,22/0)	(0,96/0,04/0)

Čas zahájení výpočtu	21 : 13 : 11	Počet generací algoritmu	5
Čas ukončení výpočtu	23 : 19 : 02	Počet navštívených chromozómů	194

Tabulka 3.26: Výsledky GA se strategií šíření `čárka(15,100)` a Grayovým kódováním na množině D_1.

3.7 Vyhodnocení dosažených výsledků

Při řešení tohoto komplikovaného problému v rámci semináře modelování v ekonomii jsme na základě výsledků z tabulek 3.4 – 3.8 za nejlepší řešení označili rozložení $(33, 3, 2)$. Pro rozložení zaměstnanců $(34, 3, 2)$ sice vycházejí menší průměrné celkové roční náklady, ale při rozložení $(33, 3, 2)$ zvládají zaměstnanci na 5% hladině spolehlivosti plnit limity stanovené vedením PS a mzdové náklady jsou při tomto rozložení menší.

Projdeme-li tabulky s výpočty pomocí GA, nalezneme i několik lepších rozložení zaměstnanců než $(33, 3, 2)$. V tabulkách 3.9, 3.11, 3.13 a 3.15 se nacházejí pro celkový počet zaměstnanců 37 rozložení s nižšími mzdovými náklady, při nichž zaměstnanci zvládají na 5% hladině spolehlivosti plnit stanovené limity. Jedná se o rozložení $(32, 4, 1)$, $(31, 2, 4)$ a $(32, 2, 3)$. Bohužel se tyto výsledky nevyskytují i v dalších tabulkách, což snižuje jejich cenu. Rozložení $(32, 2, 3)$ sice specializovaný algoritmus našel, ale bylo vyhodnoceno jako horší než $(32, 3, 2)$. Proto specializovaný algoritmus nespočítal pro toto rozložení plnění limitů, a tak jsme jej nemohli zhodnotit. Rozložení $(32, 4, 1)$ a $(31, 2, 4)$ možná vůbec nenavštívil. I když nemáme tyto výsledky potvrzené dalšími simulacemi, můžeme je připsat k dobru GA.

Jak víme, rozložení s nejnižšími průměrnými celkovými ročními náklady nalezené pomocí specializovaného algoritmu bylo $(34, 3, 2)$. Pomocí GA jsme zjistili, že pro rozložení zaměstnanců $(33, 5, 1)$ vycházejí nižší celkové náklady. Není tomu tak ovšem vždy. V tabulce 3.24 vycházejí celkové náklady pro rozložení $(34, 3, 2)$ lépe, ve všech ostatních tabulkách s výsledky, kde jsou přítomna obě tato rozložení, je vždy lepší rozložení $(33, 5, 1)$. Můžeme tedy prohlásit, že nejlepší řešení je buď $(33, 5, 1)$, nebo $(34, 3, 2)$. Otázka, které z nich je lepší, je složitá. Ovšem vzhledem k tomu, že ve většině případů vychází lépe rozložení $(33, 5, 1)$, označíme jej za lepší.

Specializovaný algoritmus rozložení $(33, 5, 1)$ nejspíš vůbec nenavštívil. Při přesouvání zaměstnanců se specializovaný algoritmus vždy posune do rozložení s nižšími celkovými náklady. Z rozložení $(34, 3, 2)$ do $(33, 5, 1)$ ovšem neexistuje přímá cesta, tj. nemůžeme se posunout z $(34, 3, 2)$ do $(33, 5, 1)$. Museli bychom například přesunout jednoho úvěráře mezi univerzály a následně dvakrát přesunout jednoho univerzála mezi kartáře. To jsou celkem tři přesuny. Nacházíme-li se v rozložení $(34, 3, 2)$ a přesuneme-li jednoho úvěráře mezi univerzály nebo jednoho univerzála mezi kartáře, celkové náklady se nejspíš nezmenší, neboť náklady při rozložení $(33, 3, 3)$ i $(34, 4, 1)$ vycházejí hůře než při $(34, 3, 2)$. To znamená, že nachází-li se specializovaný algoritmus v rozložení $(34, 3, 2)$, nemůže se nikdy dostat do rozložení $(33, 5, 1)$. Pomocí GA jsme tedy pravděpodobně našli globální minimum průměrných celkových ročních nákladů, což můžeme označit za další úspěch GA při řešení tohoto problému. Specializovaný algoritmus uvázl v jednom z lokálních minim, nutno říci že dosti blízkém globálnímu.

Jak již bylo řečeno, používáme-li GA k nalezení optima nějaké funkce, měla by být daná funkce snadno vyčíslitelná. To v našem případě rozhodně splněno není, a proto výpočty trvali dlouho. Z tabulek lze vysledovat, že výpočet pomocí GA prohledávající-

ho množinu D_0 trval obvykle asi 5–6 hodin. Vyskytují se samozřejmě i výjimky, viz tabulky 3.14 a 3.17, v těchto dvou případech byl výpočet pozastaven, takže v těchto dvou případech je výpočetní čas irelevantní. Z tabulky 3.21 můžeme vyčíst, že výpočet trval více než 8 hodin. To je způsobeno tím, že v tomto případě navštívil GA 835 chromozómů, zatímco v ostatních případech GA prošel obvykle 500–600 chromozómů. Pomocí GA se nám při prohledávání množiny D_0 nepodařilo nalézt zdaleka ve všech případech optimum, tedy rozložení $(33, 5, 1)$, případně $(34, 3, 2)$. Kdybychom chtěli dosáhnout větší úspěšnosti, museli bychom zvolit jinou strategii šíření, množina D_0 je totiž příliš veliká. Strategie `plus(15+100)` a `čárka(15,100)` se na množině D_0 přiblížily optimu. S jejich pomocí jsme získali povědomí o tom, kde se nalézá optimum z hlediska průměrných celkových ročních nákladů, a tak jsme mohli následně omezit množinu přípustných řešení D_0.

Z tabulek shrnující výpočty pomocí GA lze vysledovat, že Grayovo kódování ani inverzní operátor výpočet příliš nevylepšily. Víme, že globální minimum nastává pro rozložení $(33, 5, 1)$. Kdyby se v populaci vyskytovaly jen chromozómy kódující rozložení $(x, *, *)$, kde $x \leq 31$ a $*$ je nějaké číslo z množiny $\{0, 1, \ldots, 31\}$, potýkali bychom se v první složce s problémem Hammingovy bariéry. Oceňovací funkce byla však nastavena tak, že nedodržování limitů bylo tvrdě pokutováno. Takováto rozložení jsou tedy většinou velmi tvrdě pokutována oceňovací funkcí za neplnění politiky PS, neboť nezvládají včas vyřizovat žádosti o SÚ. Větší šanci na přežití mají tedy rozložení s 33 a více úvěráři, která žádosti o SÚ zvládají vyřizovat včas. Jakmile se takováto rozložení v populaci objeví, mají velkou šanci, že se v ní udrží. Pokud se k počtu 33 úvěrářů přibližujeme se shora, tj. od hodnot $34, 35, \ldots$, problém s Hammingovou bariérou nenastává. Nepřítomnost problému Hammingovy bariéry si v tomto případě tedy vysvětlujeme tím, že se algoritmus většinou přibližoval k optimálnímu počtu a rozložení zaměstnanců od rozložení s vyšším počtem úvěrářů.

Po zmenšení množiny D_0 na množinu D_1 se výpočetní čas zkrátil z 5–6 hodin na zhruba 2 hodiny. Navíc se úspěšnost GA mnohokrát zvýšila. V každém výpočtu označil GA jako nejlepší řešení buď rozložení $(33, 5, 1)$, nebo $(34, 3, 2)$. Můžeme tedy s klidným svědomím prohlásit, že při prohledávání množiny D_1 dával GA velmi dobré výsledky v rozumném čase.

Na závěr této části dodejme, že výpočtů pomocí GA bylo zhruba třikrát více, vybrali jsme proto jen ty zajímavé, které jsou uvedené v tabulkách výše.

Závěr

Cílem této práce bylo představit čtenáři GA a následně mu ukázat, jak je lze využít v praxi při řešení skutečných optimalizačních problémů.

Nejdříve jsme si představili základní typ GA s jednobodovým křížením a základním typem mutace a pomocí jednoduchého příkladu jsme si ukázali jeho průběh. Následoval popis různých vylepšení a adaptací. V nepříliš dlouhé kapitole věnované teorii byly uvedeny základní teoretické výsledky týkající se GA. Ve třetí kapitole jsme nastolili problém z oblasti optimalizace řízení lidských zdrojů, na kterém jsme demonstrovali sílu GA.

V kapitole prezentující myšlenky GA jsme si představili algoritmus a širokou škálu jeho modifikací. Neuvedli jsme však například koncept tzv. „Messy chromozómů“. GA s „Messy chromozómy“ je dalším východiskem při odstraňování problému Hammingovy bariéry. Dále jsme si neuvedli například paralelní GA. Obě modifikace lze najít například v Kvasnička et al. (2000). Vzhledem k tomu, že jsme je nevyužili při výpočtu, rozhodli jsme se vynechat i jejich popis.

Při řešení tohoto problému jsme využili teorii řízených Markovských řetězců, s jejíž pomocí jsme modelovali systém hromadné obsluhy PS. K následné optimalizaci počtu a úrovně vyškolení pracovníků PS jsme použili jednak specializovaný algoritmus, jednak různé typy GA.

Specializovaný algoritmus vždy označil jako optimální rozložení $(34, 3, 2)$. Otázkou zůstává, jestli to připíšeme k jeho kladům či záporům, neboť algoritmus nejspíš nikdy nenavštívil rozložení $(33, 5, 1)$, které se zdá být lepší. Naproti tomu jsme pomocí GA sice získali toto lepší řešení, ale ne ve všech případech. V tom je právě síla a zároveň slabina stochastických optimalizačních algoritmů – nemají tendenci uváznout v nějakém lokálním optimu, za to ovšem platíme daň v podobě nejistoty nalezení globálního optima. Nutnost opakovat výpočet kvůli nejistotě optimality dosažených výsledků považujeme za hlavní nedostatek GA.

Z hlediska času potřebného k výpočtu se jeví lépe GA, jenž skočil ve většině případů dříve.

Po omezení množiny přípustných řešení jsme dosáhli pomocí GA skvělých výsledků – úspěšnost v nalezení globálního optima se rapidně zvýšila a výpočetní čas klesl zhruba na 2 hodiny! V tomto případě tedy GA jasně předčil specializovaný algoritmus.

Celkově můžeme shrnout, že GA vynikají zvláště svou robustností a rychlostí. I přes jejich robustnost je však musíme používat s rozmyslem. Při řešení problému optimalizace počtu pracovníků PS a úrovně vyškolení považuji za jednu z klíčových věcí právě zmenšení množiny přípustných řešení.

Kvůli špatnému nastavení oceňovací funkce se bohužel rozložení zaměstnanců, ve kterém se nachází globální minimum průměrných celkových ročních nákladů PS na oddělení SÚ a KK, neshoduje s optimálním rozložením, při němž zaměstnanci zvládají plnit požadované limity jen s minimální rezervou. Oceňovací funkci jsme před výpočty pomocí GA nemohli změnit, neboť bychom pak nemohli srovnat výsledky obdržené pomocí specializovaného algoritmu a výsledky dosažené pomocí GA. Z hlediska srovnání těchto dvou typů optimalizačních algoritmů je ovšem špatné nastavení oceňovací funkce irelevantní, takže z tohoto úhlu pohledu jsme dosáhli svého.

Literatura

H. S. Chang, M. C. Fu, J. Hu, a S. I. Marcus. *Simulation–based Algorithms for Markov Decision Processes*. Springer, New York, 2007.

M. Dvořák, J. Krtek, T. Marada, a R. Picková. Optimalizace řízení lidských zdrojů. 2007.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison–Wesley Publishing Company, Inc., 1989.

R. F. Hartl. A global convergence proof for a class of genetic algorithms. Technische Univerzität Wien, 1990.

J. Klaschka. On Ordering of Splits, Gray Code, and Some Missing References. COMPSTAT Symposium, 2004.

G. Koole. Lecture notes Stochastic Optimization. http://www.math.vu.nl/obp/edu/so/notes.pdf, 2001.

V. Kvasnička, J. Pospíchal, a P. Tiňo. *Evolučné algoritmy*. STU Bratislava, 2000.

Z. Prášková a P. Lachout. *Základy náhodných procesů*. Nakladatelství Karolinum, Praha, 2005.

C. R. Reeves a J. E. Rowe. *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Kluwer Academic Publishers, 2002.

D. J. White. *Markov Decision Processes*. John Wiley & Sons, New York, 1993.