ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta dopravní

Bakalářská práce

Využití genetických algoritmů ve shlukové analýze

Radim Komenda

2023

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K614..... **Ústav aplikované informatiky v dopravě**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Radim Komenda

Studijní program (obor/specializace) studenta:

bakalářský – ITS – Inteligentní dopravní systémy

Název tématu (česky): **Využití genetických algoritmů ve shlukové analýze**

Název tématu (anglicky): Using genetic algorithms in cluster analysis

Zásady pro vypracování

Při zpracování bakalářské práce se řiďte následujícími pokyny:

- Prostudujte a popište principy genetických algoritmů
- Prostudujte využití genetických algoritmů v oblasti shlukové analýzy
- Prostudujte a popište data poskytnutá ze simulátoru
- Navrhněte využití shlukové analýzy pro zpracování poskytnutých dat

ČVUT
FD

Rozsah grafických prací: dle pokynů vedoucího práce

Rozsah průvodní zprávy: minimálně 35 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury:
Mařík a kol.: Umělá inteligence 3, Academia, 2001, ISBN 80-200-0472-6

Mařík a kol.: Umělá inteligence 4, Academia, 2003, ISBN 80-200-1044-0

Zelinka a kol.: Evoluční výpočetní techniky - Principy a aplikace, BEN, 2008, ISBN: 80-7300-218-3

Vedoucí bakalářské práce: **doc. Ing. Vít Fábera, Ph.D.**

Datum zadání bakalářské práce: **7. října 2022**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání bakalářské práce: **7. srpna 2023**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

Fábera V.

doc. Ing. Vít Fábera, Ph.D.
vedoucí
Ústavu aplikované informatiky v dopravě

prof. Ing. Ondřej Přibyl, Ph.D.
děkan fakulty

Potvrzuji převzetí zadání bakalářské práce.

Komenda

Radim Komenda
jméno a podpis studenta

V Praze dne 7. 10. 2022

Poděkování

Rád bych poděkoval vedoucímu práce doc. Ing. Vítu Fáberovi, Ph.D. za jeho cenné rady, odbornou pomoc a trpělivost během mého vypracování této bakalářské práce. Dále bych rád poděkoval své rodině a přátelům za jejich vytrvalou podporu, konstruktivní kritiku a víru v můj úspěch. Bez jejich podpory a porozumění by tato práce nebyla možná.

Prohlášení

a) „Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).“ (pokud nebyla tato závěrečná práce zadána jako utajená dle čl. 15 odst. 11 aktuální Směrnice děkana pro realizaci bakalářských a magisterských studijních programů)

b) „Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.“

V Praze dne 02. 08. 2023

Kennerda

.....

Podpis

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

Využití genetických algoritmů ve shlukové analýze

Bakalářská práce
Srpen 2023
Radim Komenda

ABSTRAKT

Bakalářská práce „*Využití genetických algoritmů ve shlukové analýze*“ se zabývá rešerší v oblasti genetických algoritmů, shlukové analýzy a jejich synergií v oblasti zpracování dat. Dále také navrhuje způsob, jak využít genetických algoritmů a shlukové analýzy při zpracování reálných dat poskytnutých ze simulátoru a EEG senzoru.

ABSTRACT

The bachelor's thesis titled "Utilization of Genetic Algorithms in Cluster Analysis" focuses on research in the fields of genetic algorithms, cluster analysis, and their synergy in data processing. Furthermore, it proposes a way to use genetic algorithms and cluster analysis in the processing of real data provided from a simulator and an EEG sensor.

KLÍČOVÁ SLOVA

Evoluční techniky, genetické algoritmy, shluková analýza

KEYWORDS

Evolutionary techniques, genetic algorithms, cluster analysis

1 Seznam použitých zkratek

GA	Genetický algoritmus
EVT	Evoluční techniky
SGA	Standardní genetický algoritmus
BSF	Best-so-far
SA	Shluková analýza
EEG	Elektroencefalografie
EKG	Elektrokardiografie
EOG	Elektrookulografie
EMG	Elektromyografie

2 Úvod

V současné době, kdy se stále více využívají tzv. *Big Data* neboli obrovské množství dat, na jejichž analýzu již nestačí tradiční databázové nástroje softwarových aplikací, je třeba vytvořit metody a nástroje k jejich analýze a interpretaci. Jedním z přístupů, který se ukazuje jako účinný k řešení takovýchto problémů, jsou genetické algoritmy. Genetické algoritmy se inspirují evolučními principy přírody a zabývají se řešením optimalizačních problémů.

Tato bakalářská práce se zabývá jejich základními principy a dále se zaměřuje na jejich využití v oblasti shlukové analýzy. Shluková analýza je metoda, která data sdružuje do shluků na základě jejich „podobnosti". Práce popisuje základní principy shlukové analýzy, různé metody jejího využití a obsahuje také příklady konkrétního využití. V případě, že jsou data složitějšího charakteru, nebo je dat velké množství, mohou nám napomoci právě genetické algoritmy.

Další částí práce je popis dat ze simulátoru a ze senzoru EEG. Součástí je také návrh jak principy genetických algoritmů a shlukové analýzy využít ke zpracování právě těchto dat.

3 Evoluční techniky

3.1 Úvod do evolučních technik

Genetické algoritmy představují jednu z klasických aplikací evolučních technik a patří mezi první zkoumané oblasti v rámci tohoto oboru. Pro lepší pochopení této problematiky je vhodné se detailněji seznámit s tzv. evolučními technikami (dále zkráceně EVT).

3.2 Historie EVT

Začátek studování EVT sahá až do první poloviny 60. let, kdy studenti v Berlíně při konstrukci převodovek přišli s nápadem pokusit se náhodně kombinovat již vytvořené kombinace, a nalézt tak tu nejlepší z nich. Dá se říci, že se jedná o tzv. *přírodní křížení téhož druhu*. [1]

Ve druhé polovině 60. let se L. J. Fogel [2] a jeho spolupracovníci již snažili o plnohodnotné evoluční programování, konkrétněji evoluci chování konečných automatů a na základě jejich vyvinutých schopností o predikování prostředí, ve kterém se vyvíjely. [3]

3.3 Charakteristika

Nejvýznamnějším využitím EVT je hledání v definovaném prostoru. EVT jsou stochastické algoritmy, které pro lepší vyhledávání využívají genetickou dědičnost a Darwinovský zápas o přežití. V podstatě se jedná o optimalizaci – vyhledávané změny jsou porovnávány s hodnotícím kritériem a musí být vzhledem k němu výhodné. V přírodě je hodnotícím kritériem schopnost reprodukce omezena životně důležitými zdroji. Zároveň platí, že ne všichni zástupci svého druhu jsou stejně kvalitní, ale přepokládá se, že lze jejich kvalitu vždy určit. [3]

EVT na rozdíl od klasických optimalizačních technik nepracují s jedním kandidátem – řešením problému, ale s jejich množinami. Nazýváme je tzv. *populace* jedinců $x_{i,j}$ a mají tento tvar [3]

$$G(t) = \{x_{i,1}, x_{xi,2}, \dots, x_{i,N}\} \tag{3.1}$$

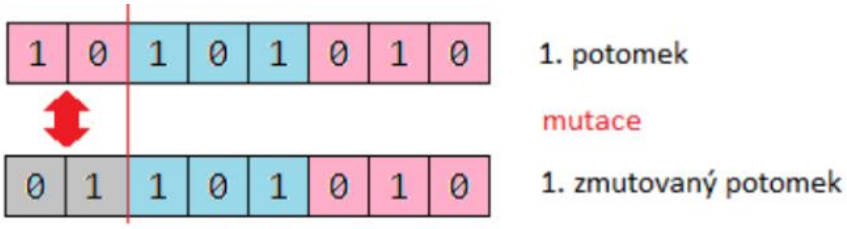
kde t je vývojový čas a N rozsah populace. Všichni zástupci populace jsou ohodnoceni svojí mírou kvality. [3]

3.4 Základní algoritmus EVT

```
begin
  t := 0;
  inicializace G(t);
  ohodnocení G(t);
  while (not zastavovací_pravidlo) do
    begin
      t:= t + 1;
      selekce G(t) z G(t-1);
      změna G(t);
      ohodnocení G(t);
    end
end
```

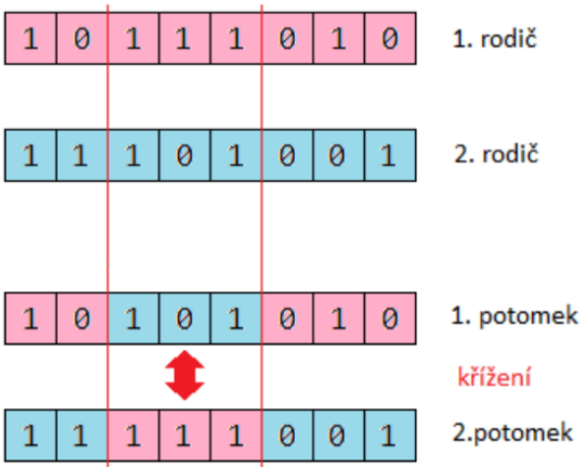
Na počátku algoritmu je vývojový čas *t* nastaven na 0 a v kroku *inicializace* je vytvořena počáteční populace, například pomocí množiny náhodných jedinců. Následuje operace *vyhodnocení*, ve které se u všech jedinců vypočítá jejich kvalita, ale možné je i vypočtení různých vlastností populace. Dalším krokem je tzv. selekce, jedná se o přirozený výběr jedince generace G(t-1) do nové generace G(t) dle jejich ohodnocení. Jedinci bývají vybíráni pravděpodobnostním mechanismem tak, že vyšší pravděpodobnost výběru mají kvalitnější jedinci. Inovativním prvkem je proces *změny pomocí* tzv. *rekombinačních operátorů*. Příklady rekombinačních operátorů jsou mutace a křížení. [4]

„Mutace m vytvoří nového jedince „malou“ změnou původního jedince.“ [5]

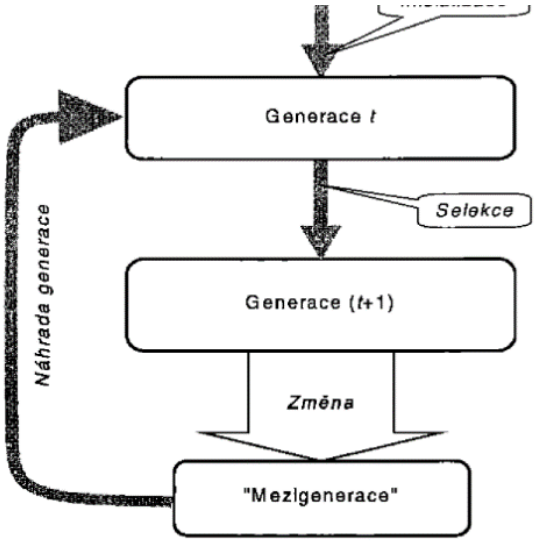


Obrázek 1: Operace mutace potomka. Převzato z [46]

„Křížení x generuje jednoho či více jedinců ze dvou či více původních jedinců – rodičů. Zpravidla se využívá tzv. binární křížení, tedy využití dvou rodičů.“ [5]



Obrázek 2: Operace křížení. Převzato z [46]



Obrázek 1. 1 Vývojový diagram základního algoritmu EVT. Převzato z [47]

„Na rozdíl od algoritmického zápisu se ve vývojovém diagramu vyskytují i pojmy mezigenerace a náhrada generace. Mezigenerace je množina tvořená jak jedinci, kteří byli předmětem změnových operací, tak i jedinci vytvořenými změnami. Náhrada generace pak představuje postup (vývojovou strategii), jak z této smíšené množiny „rodičů“ a „potomků“ vybrat množinu, kterou budeme považovat v dalším vývoji za novou populaci.“ [5]

Zpravidla se využívají dva typy procesu náhrady generace:

- *„Generační, kdy nastane úplná náhrada jedné populace populací následující (…).“* [5]
- *„Postupné, při které změny podstoupí jen část populace a umožňuje koexistenci rodičů a potomků (…)“* [5]

4 Genetické algoritmy

4.1 Úvod do genetických algoritmů

Na počátku stál problém, kterým se zabýval John Holland, americký teoretický biolog. Tímto problémem byla odpověď na otázku: „Proč se navzájem liší jedinci patřící k témuž biologickému druhu?“. Kniha, kterou J. Holland sepsal, je základem pro genetické algoritmy. Jeho dílo dále rozpracoval D. Goldberg. Z jejich práce také vycházel pan Zbigniew Michalevicz, který vytvořil modifikace algoritmů jimi vytvořenými. [1]

„Jak již bylo zmíněno v kapitole o EVT (3.1) genetické algoritmy (dále jen GA) jsou klasickým příkladem varianty EVT (…).“ [6]

Jedná se v podstatě o optimalizaci nějaké cílové funkce, kterou patřičně dobře neznáme. Toto znamená, že naše množina všech přípustných řešení je velice rozsáhlá. Základní myšlenkou GA je využití principů genetiky živých organismů, tedy pohled na přípustná řešení, jako by to byly živé organismy. Postupná simulace vývoje těchto „živých organismů“ má tendenci produkovat čím dál lepší výsledky. [7]

4.2 Charakteristika

Na počátku každého GA stojí zvolení počáteční populace přípustných řešení. Nejčastější je zcela náhodný výběr. Dalším krokem je určení formy, ve které budeme populace kódovat – zapisovat. Obecně je využíváno kódování ve formě řetězců a polí, nabízí se tedy používat binární řetězce – hodnoty 0 a 1. Zde se ukazuje analogie s genetikou, kdy řetězce reprezentují *chromozomy* konečné délky L (s_1, s_2, …, s_L), pozice v řetězci odpovídají *genům* a konkrétní s_i hodnoty pak reprezentují tzv. *alely.* [7]

Dalším krokem GA je výběr jedinců, kteří budou utvářet novou generaci. Každý jedinec aktuální populace je ohodnocen na základě cílové (kriteriální, fitness) funkce a je tak tedy určena *„kvalita“* jedince. Pravděpodobnost výběru jedince do další generace je pak zpravidla přímo úměrná jeho kvalitě a je určena vztahem [7]

$$Pr_i = k_i f\left(x_{t,i}\right), \qquad i = 1, \dots, N.$$

(4.1)

„Vztah uvádí pravděpodobnost i-tého jedince v t-té generaci. Koeficient k se určuje takovým způsobem, aby součet pravděpodobností přes celou populaci byl jedna.“ [8] Pokud přepíšeme vztah (1.2) tak, aby byla pravděpodobnost závislá explicitně na číslu populace, bude vypadat takto [8]

$$Pr_i = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}, \qquad i = 1, \dots, N.$$

(4.2)

Abychom mohli vytvářet nové struktury, je potřeba do tohoto výběru jedinců ještě zahrnout *genetické operátory.* Zpravidla využíváme dva, *křížení* a *mutaci*. Tyto operátory jsou popsány v předešlé kapitole o EVT (3.4). [7]

Tyto procesy opakujeme tak dlouho, dokud nedojde ke splnění požadavků – splnění *ukončovací podmínky (zastavovacího pravidla)*. Tato podmínka nemusí být formulována jako nalezení globálního extrému, jelikož jeho hodnota je často neznámá. Můžeme si třeba říci, že je pro nás nějaká hodnota již přijatelná a není třeba za každou cenu hledat lepší výsledek nebo byl vyhodnocen limitní počet generací.[7]

4.3 Standardní genetický algoritmus

```
begin
  t := 0;
  inicializace G(0);
  ohodnocení G(0);
  while (not zastavovací_pravidlo) do
    begin
      t := t + 1;
      selekce G(t) z G(t-1);
      křížení G(t);
      mutace G(t)
      ohodnocení G(t);
    end
end
```

4.4 Výhody a nevýhody GA

Jednou z výhod GA je, že nemusíme přesně znát cílovou funkci a její chování. Velmi dobře pracují s problémy, kde jsou množiny přípustných řešení, stejně jako jejich využití, velice rozsáhlé a dají se aplikovat na řadu problémů.

Hlavní nevýhodou však je, že při hledání nemusíme nalézt globální optimum. [7]

4.5 Modifikace standardních genetických algoritmů

4.5.1 Selekce v konečných populacích

K vyřešení problému SGA, kdy může dojít při rekombinaci k eliminaci dosud nejlepšího výsledku, se nabízí vytvořit paměťovou pozici BSF neboli *best-so-far,* do které průběžně ukládáme dosud nejlepší řešení a porovnáváme ho s řešením

nalezeným pomocí GA. Pokud nalezneme řešení lepší, než je uloženo v BSF, nahradíme jej. [9]

Další modifikace je vytvořena pro lepší práci s malými populacemi, protože celkový proces výběru jedinců je určen spíše pro velké populace. Proto vznikl tzv. *zbytkový stochastický výběr*, kdy pro všechna *i* zavedeme selekci takto [9]

$$m'(i, i+1) = \lfloor Pr_i N \rfloor$$

(4.3)

„(...) tedy deterministicky určíme počet kopií jako celou část statisticky očekávaného počtu kopií. Zbývající jedince pak doplníme tak, že provedeme selekci pouze nad nevyužitými zlomkovými částmi pravděpodobností, tedy pravděpodobností.“ [9]

$$Pr'_i = \frac{Pr_i m(i,t) - m'(i,t+1)}{N - \sum m'(i,t+1)}$$

(4.4)

Můžeme pak počítat buď s navrácením zbytkovou selekcí vybraného jedince do dalšího výběru, nebo bez navrácení takového jedince do dalšího výběru. [9]

4.5.2 Úpravy selekčního tlaku

Řeší problém svázanosti selekce s hodnotou účelové funkce, jelikož nemusíme příliš dobře znát účelovou funkci f(x). Pokud například víme pouze to, že funkce není nezáporná, přičteme nějakou kladnou konstantu, která je „dostatečně velká“. Může pak nastat takový problém, že budou všichni kandidáti stejně pravděpodobní a bude se tedy jednat čistě o náhodné hledání. Proto zavádíme *transformační funkci h*, která konvertuje funkční hodnoty f(x) na selekční pravděpodobnosti. Pr_i tak budeme určovat místo vztahu (4.2) takto [10]

$$\Pr(i) = \frac{h(f(x_i))}{\sum_{j=1}^{N} h(f(x_j))}, \qquad i = 1, \dots, N.$$

(4.5)

Funkce h musí mít takové vlastnosti, aby průměrní jedinci zůstali průměrnými, nadprůměrní byli upřednostňováni a podprůměrní potlačováni – příkladem takovéto funkce h je lineární transformační funkce ve tvaru [10]

$$h(s) = sg(as + b)$$

(4.6)

kde platí, že sg(y) = y pro y> 0 a sg(y) = 0 pro všechny ostatní případy. Hodnoty *a* a *b* je potřeba určit tak, aby pro každou populaci platilo [10]

$$\overline{h(f(x))} = \overline{f(x)} \text{ a současně } h\big(f(x_{max})\big) = k\overline{(fx)}$$

(4.7)

kde k je měřítkový koeficient. *„Slovy můžeme říci, že požadujeme, aby funkce h neměnila střední hodnotu populace, a část s měřítkovým koeficientem říká,*

kolikrát je nejlepší prvek populace lepší než průměr. Empiricky byly zjištěny hodnoty koeficientu v rozsahu 1,3 až 1,8." [10]

4.5.3 Turnajová selekce

Turnajová selekce místo náhodného výběru na základě kvalit jedinců vybírá jedince dle výsledku jejich souboje. Z původní populace se náhodně vytvoří skupinky jedinců, kteří mezi sebou soupeří o to, který prvek bude vybrán do generace následující. Jedinec nebo jedinci s nejvyšším ohodnocením, tedy vítězové, postoupí do další generace. Proces se opakuje, dokud nemá následující generace dostatečný počet jedinců. Tato selekce se kombinuje s ukládáním doposud nejlepšího výsledku a poskytuje nejlepší výsledky. Výhodou turnajové selekce je také její jednoduchá implementace, a to bez ohledu na to, jestli řešený problém minimalizujeme či maximalizujeme. [11]

4.5.4 Náhradové strategie

„SGA funguje na předpokladu, že jsou rodiče jedince při křížení nahrazeni potomky a jedinci podstupující mutaci zmutovanými jedinci." [12] Pokud však vezmeme v potaz, že je možné mutovat jak potomky, tak i původní jedince, dostaneme tzv. *rozšířenou množinu jedinců*. Cílem náhradových strategií je pak vybrat z takovéto množiny požadovaný počet jedinců N. Existuje řada pohledů. Jedním z nich je třeba tzv. *elitismus,* kdy dochází k výběru mezi všemi jedinci jednoho či více nejlepších jedinců, další jedinci se do požadovaného počtu N náhodně vyberou, přičemž jsou upřednostňování potomci a mutanti. Další strategií je tzv. *plně náhodný výběr*, kdy je požadovaný počet jedinců vybraný z rozšířené množiny plně náhodně bez ohledu na jejich kvalitu. Modifikací této strategie je tzv. *prázdná strategie*, kdy se spojí proces náhrady a selekce v následujícím generačním cyklu. [12] „*Dalším příkladem náhradové strategie je také postupný vývoj, kdy místo celé populace podstupuje rekombinaci jen část. Náhradová strategie pak určí, koho mají potomci a mutanti nahradit.*" [12]

4.6 Příklad využití GA

4.6.1 Popis problému

Příklad, který uvedl v článku [13] Miroslav Slivoně, se zabývá problematikou nalezení tzv. hubů neboli terminálů nalézajících se mezi uzly. Úloha je zjednodušena tak, že kapacita hubů není omezena a přiřazení uzlů k hubům je jednoznačné.

4.6.2 Úvod úlohy

Nejprve je potřebné definovat si několik pojmů. Jedním z nich je *distribuční systém,* kterým je zamýšlen proces přepravy zboží od primárních zdrojů k zákazníkům. Přeprava je realizována takovým stylem, že je zboží od několika málo výrobců přepraveno nejprve přes hub, v tomto případě například přes skladiště, následně k prodejci zboží a odtud k zákazníkům. Pokud je možné počet výrobců a zákazníků srovnávat, mluvíme o speciálním případu zvaném *many to many,* tedy od mnohých k mnohým. [13]

Tento styl distribuce je většině z nás dobře známý, jedná se o dopravu poštovních služeb nebo přepravních společností jakýmkoli typem dopravy. [13]

Huby jsou zřizovány z toho důvodu, že je pro tyto styly přepravy typické, že je směrem od primárních zdrojů odesíláno relativně malé množství zásilek, proto se vyplatí svážet zásilky do hubů, jež se nachází v blízkosti zdrojů a až tehdy, kdy se nám zásilky nahromadí, je dovážet do dalších hubů, zřízených v blízkosti zákazníků. Teprve potom jsou rozváženy do konkrétních míst. Tomuto způsobu využívání hubů se také říká *hub-and-spoke.* Výhodou takto realizované přepravy je vyšší ekonomičnost dopravy, protože realizujeme méně jízd a během jízdy převážíme více zásilek najednou, a také nám dovoluje přepravovat zásilky ve vyšších frekvencích. [13]

4.6.3 Předpoklady úlohy

Přeprava mezi uzly je vždy realizována jedním nebo dvěma huby. V úloze tedy není připouštěna doprava přímo mezi uzly. Je však připouštěna přeprava pomocí jednoho uzlu, a to v případě, kdy se jak uzel zdroje, tak uzel cíle oba nacházejí v blízkosti jednoho hubu. [13]

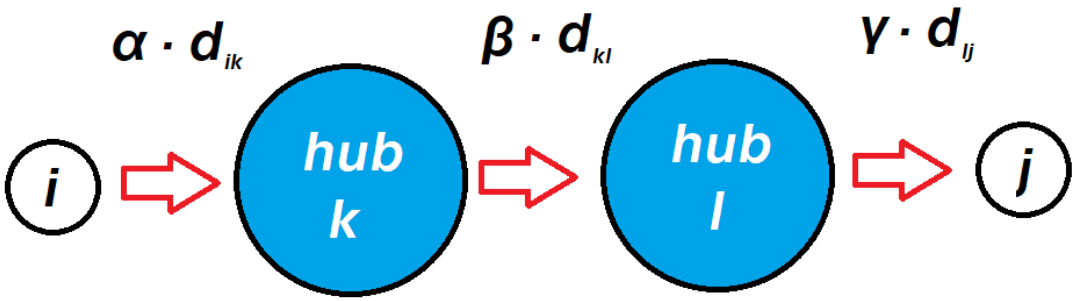
Prostá alokace. Jedná se o již zmíněné jednoznačné přiřazení uzlu některému z hubů.

Neomezená kapacita hubů. Každý hub může přijmout neomezený počet zásilek.

4.6.4 Model dopravní sítě

Dopravní síť je modelována grafem G, množinu uzlů označujeme V a množinu hran H. Každé hraně zároveň náleží ohodnocení d_{ij}, odpovídající vzdálenosti mezi uzly *i* a *j*. Velikost přepravního proudu je označena b_{ij}. [13]

Proces přepravy se skládá ze tří složek: z přepravy zásilky z uzlu *i* do prvního hubu, zde označeného písmenem *k,* následuje přeprava z hubu *k* do hubu *l*, je-li nutné přepravu realizovat pomocí dvou hubů, a následně přeprava z hubu *l* do uzlu *j*. [13]



Obrázek 3: Schéma výpočtu přepravních nákladů

Parametry α, β a γ jsou vytvořeny za účelem rozlišení nákladů v jednotlivých částech přepravy. Parametry α a γ bývají většinou rovny jedné, protože nerozlišujeme přepravní náklady pro svoz, tedy dopravu mezi uzlem *i* a hubem *k*, a rozvoz, tedy přeprava mezi hubem *l* a uzlem *j*. [13]

Hodnota parametru β reflektuje výši úspory nákladů tím, že jsme pro přepravu zvolili tento způsob. Celkové náklady jsou vypočteny pomocí následujícího vzorce [13]

$$c_{ij} = \alpha \cdot d_{ik} + \beta \cdot d_{kl} + \gamma \cdot d_{lj}$$

(4.8)

Rozhodujícím aspektem v této úloze je míra vykonané práce při přepravě mezi uzly *i* a *j*, označme ji například *w*. Vypočítá se součinem velikosti přepravního proudu a přepravních nákladů [13]

$$w_{ij} = b_{ij} \cdot c_{ij}$$

(4.9)

Cílem této úlohy je zmíněnou vykonanou práci minimalizovat. Známe-li fixní náklady označené jako f_i za sledované období v uzlu *i*, lze tuto úlohu zpracovat pomocí GA, přičemž počet hubů bude fungovat jako proměnná v účelové funkci. [13]

4.6.5 Matematický zápis

Je třeba definovat proměnnou, která nám bude indikovat, zdali je uzel *i* přiřazen k hubu *k,* nebo ne. Tuto proměnnou označíme jako h_{ik}, kdy hodnota 1 bude značit, že je uzel přiřazen, a hodnota 0, že nikoliv. Pro uzel *k,* který se stane hubem je hodnota proměnné rovněž rovna 1 a je označena jako h_{kk}. [13]

Následně existuje možnost řešit úlohu dvěma způsoby, s *pevně daným počtem hubů*, nebo s *proměnným počtem hubů*.

Pokud budeme řešit úlohu s pevně daným počtem hubů označeným jako *p*, bude naším úkolem vlastně splnit tento matematický zápis, který je zároveň účelovou funkcí

$$\min\left(\sum_i \sum_j b_{ij}\left(\sum_k \alpha \cdot d_{ik} \cdot h_{ik} + \sum_k \sum_l \beta \cdot d_{kl} \cdot h_{ik} \cdot h_{jl} + \sum_l \gamma \cdot d_{jl} \cdot h_{jl}\right) + \sum_k h_{kk} \cdot f_k\right)$$

(4.10)

Platí podmínky:

$\sum_k h_{kk} = p$

(4.11)

$\sum_k h_{ik} = 1 \qquad pro\ i \in V$

(4.12)

$h_{ik} \leq h_{kk} \qquad pro\ i,k\ \in V$

(4.13)

$h_{ik} \in \{0,1\} \qquad pro\ i,k\ \in V$

(4.14)

Účelová funkce popsaná ve vztahu (4.10) popisuje celkové náklady, tedy součet přepravní práce a fixních nákladů, což je složka, která není ve výpočtu nezbytně nutná, ale slouží ke zvýhodnění uzlů, kde je již vybudována infrastruktura atd. Podmínka (4.11) garantuje prostou alokaci. Podmínka (4.12) zajišťuje, že přeprava nemůže být realizována napřímo mezi dvěma uzly. [13]

Úloha s proměnným počtem hubů je řešena podobně, ale nebude zavedena podmínka (4.11).

4.6.6 Popis GA

Algoritmus vychází z algoritmu GAHUB2, vytvořeném J. Kraticou a kolektivem. Některé prvky využívají algoritmu z postupu popsaného zde [14]. Autoři přebrali některé prvky postupu ze článku zde [15].

4.6.7 Zakódování

„Zakódování úlohy lze popsat následovně: délka chromozomu (datového řetězce), tedy počet genů, je rovna počtu uzlů v grafu G. Chromozom je rozdělen do dvou segmentů, z nichž první říká, zda je uzel hubem, nebo ne, a odpovídá tedy proměnné h_{kk}*. Druhý indikuje, ke kterému hubu je uzel přiřazen. Je potřeba si uvědomit, že ačkoliv je hub nejblíže položený k uzlu, ještě to neznamená, že je přiřazení uzlu k takovému hubu optimální. Záleží totiž ještě na rozložení přepravních proudů. Uzel má proto tedy vytvořen seznam hubů, které jsou seřazeny dle vzdálenosti od uzlu vzestupně. Druhý segment má pak hodnoty určené stylem: 0 znamená, že je uzel přiřazen nejbližšímu hubu, 1 znamená, že je uzel přiřazen druhému nejbližšímu hubu atd. Tento způsob zachovává přípustnosti řešení i po provedení operací křížení a mutace.“* [13]

4.6.8 Generování počáteční populace

Pro úlohu s pevným počtem hubů platí, že první segment nabývá hodnoty 1 s pravděpodobností p/n, kde p je počet hubů a n je počet uzlů v grafu G. Může nastat situace, kdy je třeba opravit hodnoty prvního segmentu genů, protože se může lišit skutečný počet hubů od požadovaného počtu hubů p. [13]

Druhý segment je ve tvaru binárního řetězce s výchozí hodnotou 0. Poslední bit druhého segmentu genu nabývá hodnot 1 s pravděpodobností $1/n$, přičemž každý bit blíže začátku binárního čísla má pravděpodobnost poloviční oproti předešlému. Tato strategie zaručuje, že většina uzlů bude přiřazena k nejbližšímu hubu, menší počet uzlů ke druhému nejbližšímu atd. [13]

Algoritmus je možné upravit tak, aby zohledňoval celkovou velikost přepravního toku v uzlech. Toho docílíme tak, že si seřadíme uzly sestupně dle velikosti celkového přepravního toku v uzlech. První ¾ populace jsou vytvořeny tak, že se huby vybírají pouze z prvních 2/3 tohoto seznamu. U zbylých uzlů se pak huby vybírají z celé množiny uzlů.

Pro úlohu s proměnným počtem hubů platí, že počet hubů p je pro ¾ populace generován v rozmezí $1-n/4$. Pro zbylé jedince je to pak v rozmezí $n/4-n/2$, kde n je celkový počet uzlů. [13] Toto je postup převzatý z [15]Dále je postup generace stejný.

4.6.9 Selekce

Při vytváření nové generace je využíváno *elitismu*, popsaného zde (4.5.4). Dále je selekce v tomto příkladě řešena *turnajovým způsobem*, který je popsaný již dříve (4.5.3), a vítězem se stává jedinec s minimální fitness hodnotou (hledáme minimum účelové funkce). Následně jsou vytvořeny dvojice, které budou podrobeny křížení a mutaci, přičemž rozdíl velikosti dané populace a počtu jedinců, které elitismus „poslal" do další generace přednostně, určuje počet takových dvojic. [13]

4.6.10 Křížení

U úlohy s pevným počtem hubů probíhá ve směru zprava doleva a hledá takovou pozici *i*, kde má první jedinec v prvním segmentu genu hodnotu 1 a druhý 0. Po nalezení této pozice dojde k výměně celých genů na této pozici. Souběžně probíhá proces opačný, tedy ve směru zleva doprava se hledá taková pozice *j*, u které platí, že první jedinec má v prvním segmentu hodnotu 0 a druhý jedna. Po nalezení takové pozice opět dojde k výměně celých genů. Tyto procesy se opakují tak dlouho, dokud platí $j \geq i$. Tento způsob zaručí, že oba vzniklí jedincí budou obsahovat právě *p* hubů. Výchozí pravděpodobnost p_c je nastavena na 0,85. [13]

„Pro úlohu s proměnným počtem uzlů probíhá křížení tak, že se náhodně vygeneruje r v rozmezí 1 až n-1, kde n je počet uzlů. Za pozicí r se chromozomy překříží". [13]

4.6.11 Mutace

Oba segmenty genu s nějakou pravděpodobností genů zmutují. Pro první je tato pravděpodobnost je p_m^1/n, pro druhý pak p_m^2/n, kde *n* je počet uzlů. Tyto pravděpodobnosti platí pro poslední bit, pro každý další bit směrem k začátku je pravděpodobnost poloviční oproti předešlému. [13]

4.6.12 Modifikace algoritmu

Během evolučního cyklu může dojít k tomu, že geny většiny jedinců budou na některé pozici stejné. Při nalezení těchto genů se zvýší jejich pravděpodobnost mutace. [13]

5 Shluková analýza

5.1 Úvod do shlukové analýzy

Shluková analýza (dále jen SA) se jako taková zabývá metodami a algoritmy takovým způsobem, že data, která jsou si více podobná sdružuje do skupin – shluků. Také se zabývá uspořádáním takto sdružených dat. SA nám dovoluje najít strukturu mezi objekty. Dále ji však nevysvětluje, čistě nás informuje, že nějaká struktura existuje.

5.2 Použití

SA se dá využít jak v informatice, tak například v biologii, v průzkumu atd. Existují dva typy shlukování – *konvenční* a *konceptuální*. [16]

„Konvenční shlukování se zabývá měřením podobnosti dvou znaků. Podobnost dvou znaků je funkcí jejich vlastností.“ [17] Konceptuální shlukování využívá jednak podobností vlastností dvou znaků a rovněž také popisný jazyk a okolí. Popisný jazyk popisuje třídy objektů. Okolí jsou pak „sousedi“ objektu – jedná se tedy o množinu sousedících objektů. [17]

Dalším dělením SA je dělení pomocí typu shluků - tzv. *pevné shlukování*, kdy objekt může být zařazen pouze do jednoho shluku. Naopak tzv. *překrývající shlukování* připouští i to, že je objekt zařazen do více shluků najednou. Tyto shluky se pak tedy překrývají. [17]

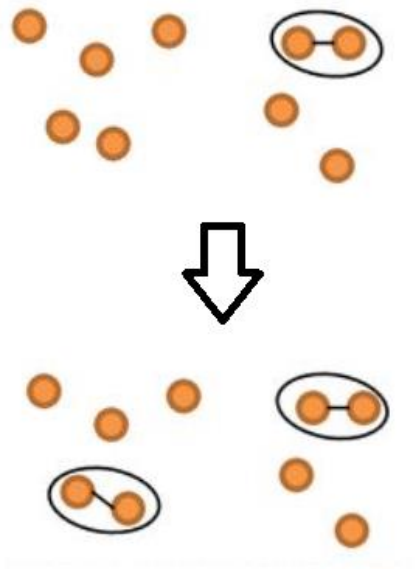
Algoritmy SA můžeme rozdělit na algoritmy *hierarchické* a *nehierarchické.*

5.3 Hierarchické metody

Hierarchické algoritmy vytváří z dříve nalezených shluků shluky nové. Hierarchické algoritmy se dále dělí na *aglomerativní* a *divizní*. [16]

5.3.1 Aglomerativní metody

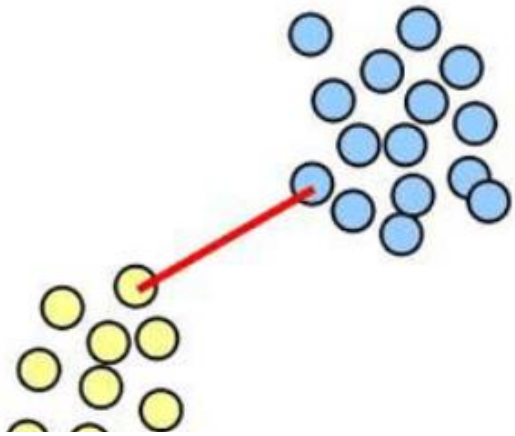
Algomerativní přístup pracuje s každým prvkem jako se shlukem a slučováním tento shluk „narůstá". Na začátku tedy máme n jednoprvkových shluků. Podle daného kritéria slučujeme nejpodobnější shluky. Nevýhodou je, že chyby, které mohou vzniknout už na začátku shlukování, se odhalí až později a jednotlivé kroky nelze změnit. [18]



Obrázek 4: Aglomerativní hierarchické shlukování. Převzato z [48]

5.3.1.1 Metoda nejbližšího souseda

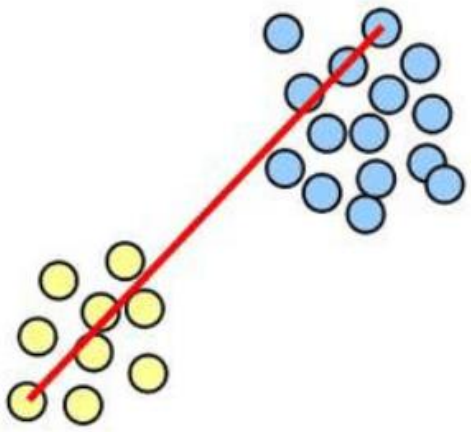
Kritériem je nejmenší vzdálenost. Shluk je vytvořen z takových prvků či shluků, které jsou od sebe vzdálený co nejméně. Vybírá se minimální vzdálenost mezi prvky z každých dvou shluků. Pokud existují prvky se stejnou vzdáleností, může dojít ke *zřetězení,* a to může vést ke špatnému výsledku SA. [18]



Obrázek 5: Metoda nejbližšího souseda. Převzato z [49]

5.3.1.2 *Metoda nejvzdálenějšího souseda*

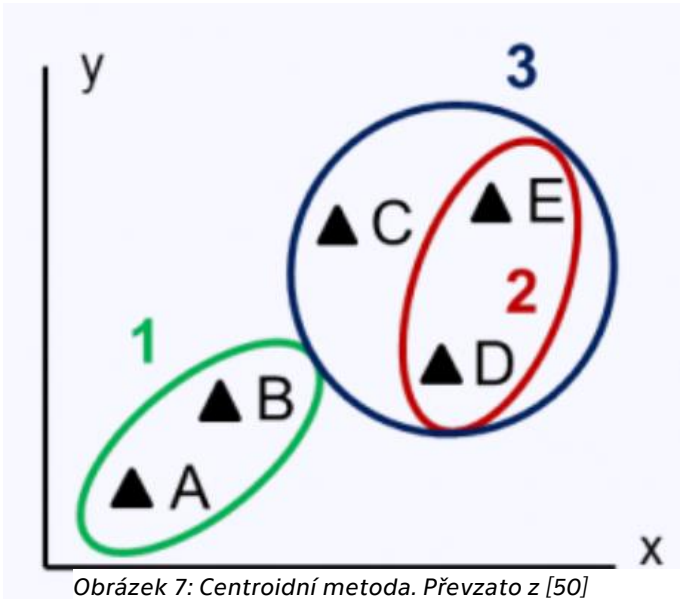
Opakem metody nejbližšího souseda je metoda nejvzdálenějšího souseda, kdy spojujeme do shluků prvky nebo shluky které jsou od sebe nejvzdálenější. Při řešení vzdálenosti dvou shluků vezmeme maximální vzdálenost každých jejich dvou prvků. Tato metoda zabraňuje vzniku *zřetězení*. [18]



Obrázek 6: Metoda nejvzdálenějšího souseda. Převzato z [49]

5.3.1.3 *Centroidní metoda*

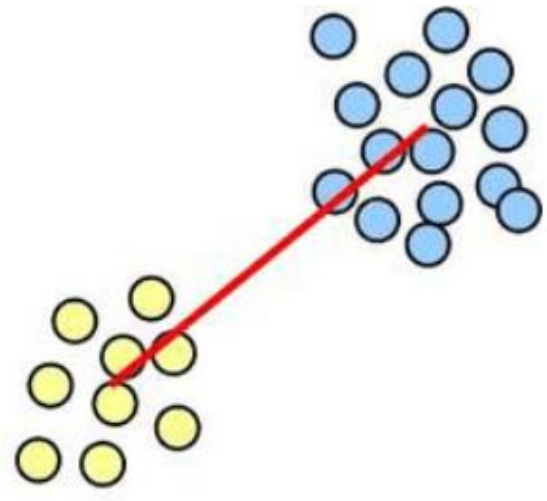
Při shlukování se používá vzdálenost těžišť prvků nebo shluků. [18] *„Dojde ke sloučení shluků či prvků takových, které mají nejmenší vzdálenost mezi těžišti.“* [18]



Obrázek 7: Centroidní metoda. Převzato z [50]

5.3.1.4 *Metoda průměrné vazby*

V této metodě se jako kritérium podobnosti využívá průměr vzdáleností mezi každými dvěma prvky, náležících do dvojice shluků. Ty, co mají minimální průměr vzdálenosti, jsou vyhodnoceny jako nejpodobnější a dojde k jejich sloučení. [18]



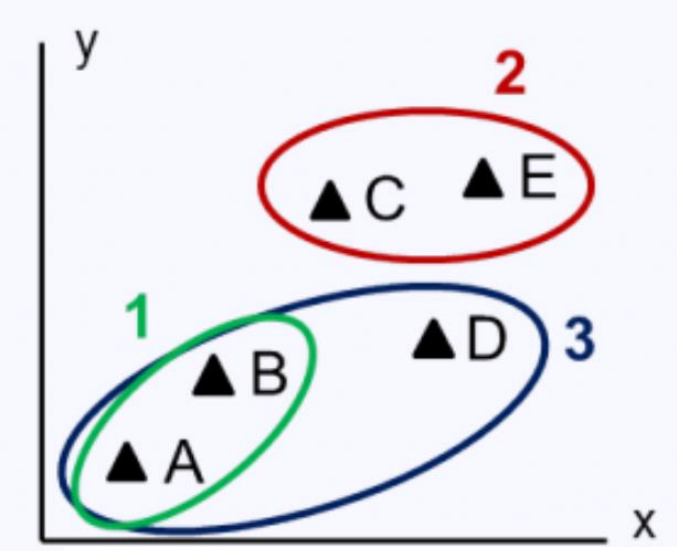
Obrázek 8: Metoda průměrné vazby. Převzato z [49]

5.3.1.5 *Mediánová metoda*

Podobnost se určuje pomocí vzdálenosti mediánů dvou shluků. Sloučí se ty shluky, jejichž vzdálenost je minimální. [18]

5.3.1.6 *Wardova metoda*

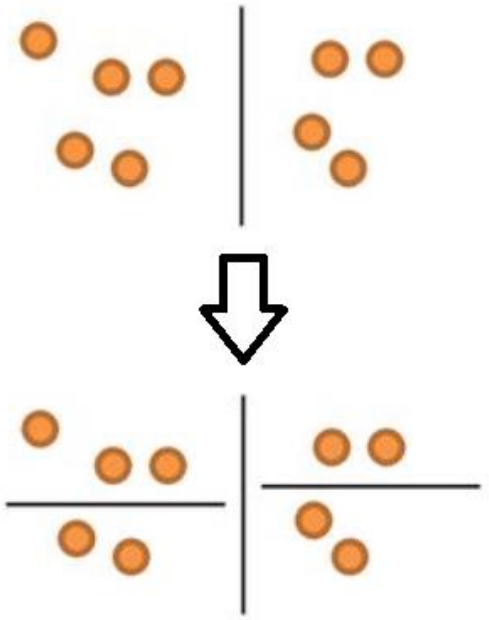
Kritériem podobnosti je součet druhých mocnin odchylek každého prvku od těžiště shluku, jehož je součástí.



Obrázek 9: Wardova metoda. Převzato z [50]

5.3.2 Divizní hierarchické shlukování

Divizní přístup naopak bere všechny prvky jako jeden velký shluk a ten pak dělí. Tento přístup je výpočetně velice náročný a je omezen na malý počet prvků.



Obrázek 10: Divizní hierarchické shlukování. Převzato z [48]

5.3.2.1.1 MacNaughton–Smithova metoda

Hlavním cílem této metody je snížení časové náročnosti divizních algoritmů i za cenu toho, že výsledné řešení nemusí být optimální. Přesto je však stále pomalejší než aglomerativní algoritmy. [18] „*Pomocí středních vzdáleností se vybere prvek uvnitř shluku, který vytvoří shluk nový a na základě rozdílů středních vzdáleností prvků původního a prvků nového shluku se prvek přiřadí do nového shluku, zůstane v původním.*“ [18]

5.4 Nehierarchické metody shlukování

Dle předem daného podobnostního kritéria rozkládají danou množinu do podmnožin. První rozklad se dále nedělí, ale optimalizuje se tak, aby byla vzájemná vzdálenost a odlišnost optimální a došlo k rovnoměrnému rozložení prvků. Většina metod začíná s předem daným rozkladem a s tím pak pracuje. Výsledkem těchto metod je však většinou pouze lokální optimum rozkladu. Definování shluků probíhá v jednom kroku. [19]

Nehierarchické metody dále dělíme na *optimalizační metody*, které přeřazují prvky mezi shluky, a *analýzy modů*, které berou shluky jako místa s větší koncentrací prvků. [19]

Na začátku analýzy je třeba určit optimální počet shluků, ten je buď pevně daný anebo proměnný. Dalším krokem je určení počátečního rozkladu. Je možné buď náhodně přiřadit prvky shlukům, nebo využít nějaké hierarchické metody. Další

možností je vytvoření tzv. *vzorových množin*, kdy se nějakým způsobem vytvoří počáteční rozklad a následuje výběr vzorové množiny prvků dle daného kritéria, vypočítá se vzdálenost všech prvků k vzorovým množinám a přiřadí se k nejbližší z nich. Následuje opětovné vypočítání vzorových množin. Proces se opakuje, dokud nejsou rozklady stejné. [19]

Metod nehierarchických algoritmů je hned několik, nejznámější z nich je však MacQueenova metoda, známá jako metoda K-means.

5.4.1 Metoda K-means

Cílem této metody je nalézt takové skupiny prvků, jejichž vnitroskupinová podobnost je co největší, přičemž princip tvorby shluků je stejný jako ve Wardově metodě (5.3.1.6). Na začátku je opět třeba určit počáteční rozklad do *k* shluků (odtud název K-means) libovolnou metodou. Dalším krokem je určení centroidů shluků. Následuje zhodnocení pozic všech objektů. Následně podle vzdáleností od různých centroidů rozhazujeme prvky mezi shluky. Rozhodující je nejmenší vzdálenost. Dalším krokem je přepočítání shluků. Následuje opakování zhodnocení pozic prvků a jejich rozhazování mezi shluky, dokud nedojdeme k optimálnímu řešení. [20]

Tato metoda je zároveň modifikována několika způsoby. Jedním z nich je, že místo počátečního rozkladu vezmeme *k* prvků a ty určíme jako centroidy. Dále je postup stejný. Další možností je provést přepočet centroidů po každém přesunu prvku, a nejen na konci cyklu. [20]

Metoda K-means je citlivá na odlehlé hodnoty, což je jednou z jejích nevýhod. Další nevýhodou je správné určení *k*. Většinou je potřeba provést tuto metodu pro několik hodnot *k*, abychom se vyvarovali tomu, že nalezneme pouze lokální optimum. [20]

5.4.2 Příklad využití shlukové analýzy

5.4.2.1 Úvod příkladu

Příklad je zpracován ve článku od Pavly Jindrové [21]. Zabývá se disparitou českých krajů. Ačkoliv byl sepsán již v roce 2009, na zobrazení využití shlukové analýzy je vhodný, protože je dobře popsaný a není příliš složitý na pochopení.

Disparitu určuje na základě rozdílů v regionální politice. Regionální politika je koncepční a výkonná činnost státu prostřednictvím regionálních orgánů. Nejzávažnější rozdíly patří *administrativní faktory*, které se zabývají celkovou strukturou území kraje a místem v hierarchii státu. Dále *sociálně-ekonomické faktory* čili faktory zabývající se sociální politikou a ekonomickým rozvojem kraje. V neposlední řadě pak *demografické faktory*, které se zabývají statistikami týkajících se obyvatel – např. počtem obyvatel, hustotou zalidnění a věkovým zastoupením. Obecně se dá očekávat, že se tyto faktory budou chovat nejvíce dynamicky v oblasti velkých měst a jejich okolí. [21]

Ke zpracování nejen tohoto příkladu je potřeba dostatek dat, získaných ze statistických údajů jednotlivých krajů. Zároveň je třeba uvědomit si, že kraje svojí politikou zohledňují své priority a nevyvíjí se všechny stejným způsobem. I když máme dostatek dat, nelze vždy zohlednit všechny parametry vývoje krajů. [21]

„Pro statistické a analytické údaje a pro potřeby EU byly vymezeny statistické územní jednotky NUTS 1, 8 jednotek NUTS 2 a 14 jednotek NUTS 3, jednotky NUTS 3 odpovídají krajům.“ [21]

5.4.2.2 Parametry příkladu

Tato práce porovnává kraje v rámci několika vybraných parametrů a hodnotí je dle stavu v roce 2006. Jelikož se dá předpokládat, že město Praha bude mít významný vliv na výsledky, je tento příklad nejdříve zpracován pro všech 14 krajů, včetně kraje Hlavní město Praha, a následně bez něj. [21]

Kraje jsou hodnoceny dle 17 kritérií, těmi jsou

- počet okresů, počet obcí, počet obcí se statusem města
- rozloha kraje v km^2, vzdálenost metropole kraje od hlavního města
- počet obyvatel kraje, hustota obyvatel na km^2, podíl městského obyvatelstva
- HDP regionu, tvorba HDP v kraji, podíl kraje na HDP v ČR, HDP na obyvatele, HDP na zaměstnance, daňová výtěžnost, průměrná hrubá měsíční mzda, čistý disponibilní důchod domácnosti na obyvatele, vývoz regionu

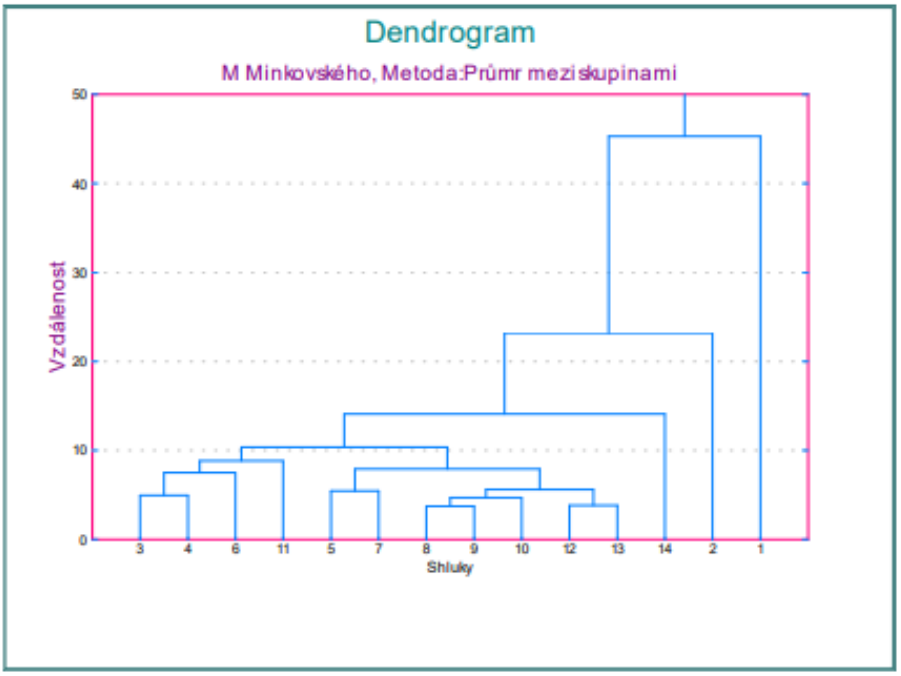
5.4.2.3 Vyhodnocení

Příklad využívá SA pomocí programu Unistat 5.6. Toto zpracování umožní zobrazit podobnost krajů v České republice. V příkladu je použita metoda průměrné vzdálenosti popsaná zde (5.3.1.4). [21] Pro měření vzdálenosti byla využita *Hammingova vzdálenost*, která je metrikou pro měření datových řetězců a dokáže říci, v kolika místech se řetězce liší. [22] Příklad obsahuje i dendrogramy, které napomáhají k lepšímu pochopení výsledků.

Data, pomocí kterých je příklad zrealizován, jsou získána z Českého statistického úřadu.

	Počet okresů	Počet obcí	Počet obcí se statutem města	Rozloha regionu v km^2	Vzdálenost regionální metropole od hl. města ČR	Počet obyvatel regionu	Hustota obyvatel na 1 km^2	Podíl městského obyvatelstva	Hrubá přidaná hodnota (v mil.Kč)	Tvorba hrubého domácího produktu v regionu (mil.Kč)	Podíl regionu na HDP ČR v % (ČR = 100)	HDP na 1 obyvatele regionu v Kč	HDP na 1 zaměstnance v Kč	Daňová výtěžnost (Kč/obyvatel)	Průměrná hrubá měsíční mzda v Kč	Čistý disponibilní důchod domácností na 1 obyvatele v Kč	Vývoz regionu v běžných cenách (mil. Kč)
Hlavní město Praha	1	1	1	496	0	1188126	2395	100,0	706464	784492	24,3	662815	893280	32452	26460	204845	128276
Středočeský kraj	12	1146	77	11015	0	1175254	107	54,7	298969	331990	10,3	284594	620017	14203	22866	159516	417807
Jihočeský kraj	7	623	52	10057	135	630006	63	65,1	160656	178400	5,5	283701	588879	15257	18962	145990	93448
Plzeňský kraj	7	501	50	7561	94	554537	73	67,4	146634	162829	5,0	294501	574886	17647	20101	150920	175237
Karlovarský kraj	3	132	30	3315	133	304602	92	81,0	65849	73122	2,3	240082	491172	15197	17942	133574	51192
Ústecký kraj	7	354	53	5335	82	823265	154	79,9	188249	209041	6,5	253939	590186	14235	20290	133598	144828
Liberecký kraj	4	215	39	3163	102	430774	136	78,7	103170	114565	3,5	266553	563632	14511	19432	139236	93794
Královéhradecký kraj	5	448	44	4758	112	549643	116	67,8	135267	150207	4,6	273541	565867	14109	17871	146033	89661
Pardubický kraj	4	451	33	4519	101	507751	112	61,3	117336	130295	4,0	257090	541675	13794	18073	141859	154022
Kraj Vysočina	5	704	33	6796	123	511645	75	58,4	122129	135618	4,2	265339	579942	15075	19185	142876	92050
Jihomoravský kraj	7	673	48	7196	200	1132563	157	62,7	291371	323553	10,0	286079	596082	15917	18217	144537	151666
Olomoucký kraj	5	398	30	5267	245	639894	121	57,8	134573	149436	4,6	233705	519024	14227	17563	136240	83051
Zlínský kraj	4	304	30	3964	287	589839	149	60,9	135172	150102	4,6	254466	554153	12836	18432	141908	103270
Moravskoslezský kraj	6	299	39	5427	348	1249290	230	76,4	304315	337926	10,5	270360	632397	14825	21323	136584	225492

Tabulka 1: Tabulka vyhodnocovaných dat pro jednotlivé kraje. Převzato z [21]



Obrázek 11 Dendrogram vyhodnocení všech 14 krajů. Převzato z [21]

Označení krajů na dendrogramu

1 – Hlavní město Praha

2 – Středočeský kraj

3 – Jihočeský kraj

4 – Plzeňský kraj

5 – Karlovarský kraj

6 – Ústecký kraj

7 – Liberecký kraj

8 – Královéhradecký kraj

9 – Pardubický kraj

10 – Kraj Vysočina

11 – Jihomoravský kraj

12 – Olomoucký kraj

13 – Zlínský kraj

14 – Moravskoslezský kraj

„Na dendrogramu je zřejmé, že jsou data rozdělena do čtyř základních skupin.“ [21]

První skupina je tvořena pouze Hlavním městem Praha, jelikož se významně liší od ostatních krajů. Nejnižších čísel dosáhl tento kraj v parametrech: počet okresů, počet obcí, počet obcí se statusem města – počty jsou rovné jedné. Nulová je pak vzdálenost od hlavního města ČR. Ostatní parametry jsou buď nejvyšší nebo velmi vysoké v porovnání s ostatními kraji. [21]

Druhá skupina je tvořena Středočeským krajem. Hodnoty parametrů vyplývají hlavně z toho, že se jedná o kraj položený nejblíže Praze. Ostatní kraje převyšuje hlavně v parametrech počet obcí, počet obcí se statusem města, v rozloze kraje, a ve vývozu regionu. [21]

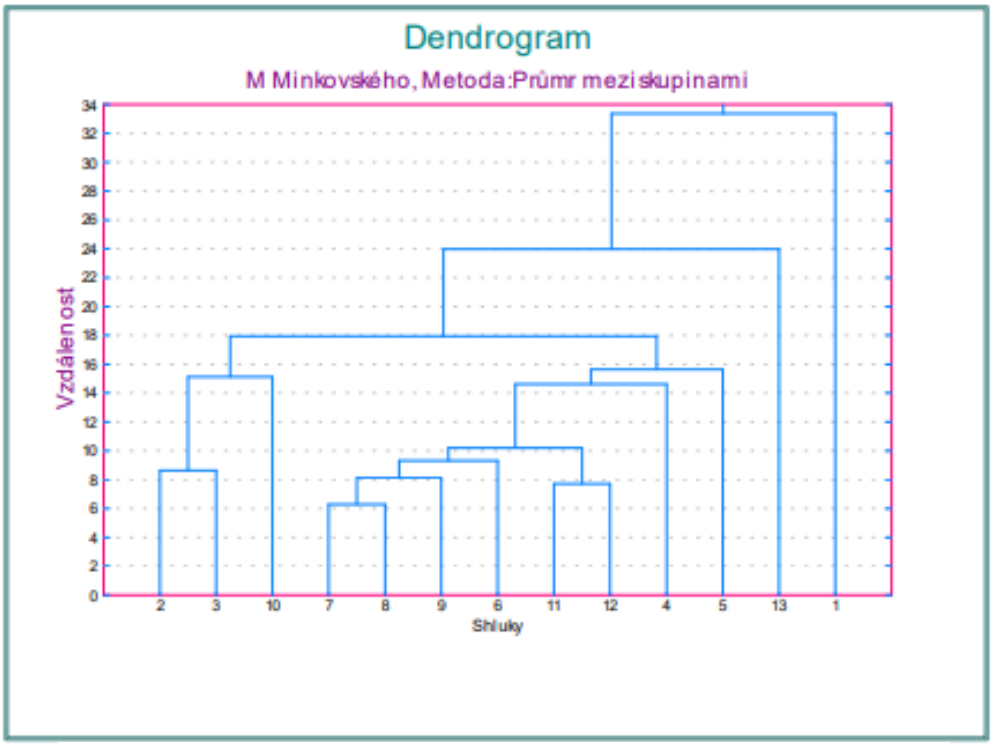
Třetí skupina je tvořena Moravskoslezským krajem – nejlidnatějším krajem ČR. Parametry kraje se nacházejí ve středu v porovnání s ostatními kraji kromě počtu obyvatel a vzdálenosti metropole od hlavního města. [21]

Ostatní kraje tvoří *čtvrtou skupinu*, která je dále rozdělená do dvou větví, první tvoří Jihočeský, Plzeňský, Ústecký a Jihomoravský. Druhou větev tvoří kraje Karlovarský, Liberecký, Královéhradecký, Pardubický, kraj Vysočina a kraje Olomoucký a Zlínský. [21]

Kraje první větve se významně liší v počtu obyvatel. Jihomoravský kraj má dvojnásobný počet obyvatel oproti Ústeckému kraji. V některých parametrech se kraje velice přibližují, těmi jsou počty obcí se statusem města a hodnota čistého disponibilního důchodu domácnosti na 1 obyvatele. Největší podobnosti dosahují kraje Jihočeský a Plzeňský. [21]

Největší podobnosti ze všech 14 sledovaných krajů dosahují Královéhradecký, Pardubický, Olomoucký a Zlínský kraj. [21]

Jak již bylo zmíněno, kraj Hlavní město Praha značně ovlivňuje výsledky SA a je tedy vhodné provést ji ještě jednou pro ostatních 13 krajů.



Obrázek 12: Dendrogram vyhodnocení krajů mimo Prahu. Převzato z [21]

Označení krajů na dendrogramu

1 – Středočeský kraj

2 – Jihočeský kraj

3 – Plzeňský kraj

4 – Karlovarský kraj

5 – Ústecký kraj

6 – Liberecký kraj

7 – Královéhradecký kraj

8 – Pardubický kraj

9 – Kraj Vysočina

10 – Jihomoravský kraj

11 – Olomoucký kraj

12 – Zlínský kraj

13 – Moravskoslezský kraj

Tentokrát se výsledky rozdělují do tří skupin. *První skupinu* tvoří Středočeský kraj, což znázorňuje, že má Středočeský kraj velký význam v regionální politice. [21]

Druhá skupina je tvořena Moravskoslezským krajem z důvodu, že je velice podobný v ekonomických faktorech Středočeskému kraji. [21]

Třetí skupina je tvořena ostatními kraji a dělí se do dvou větví. *První větev* tvoří Jihočeský, Plzeňský a Jihomoravský, ostatní kraje tvoří *druhou větev.* Rozdělení druhé větvě není stejné jako při zahrnutí Hlavního města Prahy, což je znázorněno v obrázcích (Obrázek 11) a (Obrázek 12). Největší podobnosti mezi všemi 13 kraji dosahují kraje Královéhradecký a Pardubický kraj. [21]

6 Shluková analýza pomocí genetických algoritmů

6.1 Obecné využití

V případě SA je GA jedním z přístupů, který může být použit k nalezení optimálního rozdělení datových bodů do shluků. Jak již bylo zmíněno, GA jsou inspirací z přírody a využívají principy evoluce a genetiky k hledání nejlepšího řešení. SA se snaží rozdělit data do shluků, které mají vysokou podobnost mezi datovými body uvnitř shluků a nízkou podobnost mezi shluky. Vhodným využitím GA tak může být nalezení takového rozdělení datových bodů, které minimalizuje rozdíl mezi shluky a maximalizuje podobnost uvnitř shluků.

V GA je populace jedinců, kde každý jedinec představuje jedno možné rozdělení datových bodů do shluků. Jedinci jsou reprezentováni genotypem, který může být například binární řetězec. Každý bit pak odpovídá přiřazení jednoho datového bodu do jednoho shluku.

Hodnotící funkce (fitness funkce) je klíčovým prvkem GA v problematice SA. Hodnotící funkce slouží k ohodnocení každého jedince v populaci a na základě kvality jsou datové body přiřazeny do shluků. Hodnotící funkce by mohla zohledňovat míru podobnosti datových bodů uvnitř shluků (např. vzdálenost mezi body) a rozdíl mezi shluky (např. vzdálenost mezi centroidy shluků).

Dále by se na jedince aplikovaly operace křížení a mutace již zmíněné v kapitole (3.4).

Nová generace jedinců je vytvořena aplikací genetických operací a je vybrána ta nejlepší část populace pomocí selekčního mechanismu. Tento proces se opakuje po určený počet generací nebo dokud není splněno určité kritérium ukončení, například konvergence k optimálnímu řešení.

GA mají schopnost prozkoumat velký prostor řešení a hledat globální optima. Tím mohou pomoci odhalit skryté vzorce a struktury v datech, které by jinak mohly zůstat nepovšimnuty. Při správném nastavení parametrů GA, jako je velikost populace, pravděpodobnost křížení a mutace, mohou být nalezena optimální shlukování, která lépe odhalují strukturu dat a vzájemnou podobnost mezi jednotlivými datovými body.

Lze tedy říci, že GA poskytují vhodný rámec pro řešení problémů SA a mohou být efektivním nástrojem pro objevování skrytých vzorců a struktur v datech.

6.2 Existující knihovny

Existuje několik knihoven, které dokáží pracovat s GA, a pomocí nich řešit SA. Ať už to dokáží samy o sobě, nebo v kombinaci s jinými knihovnami. V této práci je popsáno pět knihoven. Všechny kromě Scikit-learn, která sama o sobě neumí pracovat s GA, a je třeba ji doplnit o možnosti jiné knihovny, jsou založeny na práci s GA a nabízejí různé funkce. Mezi popisované knihovny patří:

- DEAP
- Scikit-Learn
- JGAP
- ECJ
- Opt4j

6.2.1 DEAP (Distributed Evolutionary Algorithms in Python)

DEAP je výkonná knihovna pro implementaci evolučních algoritmů v Pythonu, včetně GA. [23]

Při tvorbě GA je prvním krokem rozhodnutí, jak data kódovat. Nejčastějším typem je binární řetězec, kde by každý bit mohl představovat například přiřazení jedince do jednoho shluku. Kupříkladu pokud chceme pět jedinců rozdělit do tří shluků, můžeme postupovat následovně:

- Jedinec 1 – 100 (shluk 1)
- Jedinec 2 – 010 (shluk 2)
- Jedinec 3 – 001 (shluk 3)
- Jedinec 4 – 010 (shluk 2)
- Jedinec 5 – 100 (shluk 1)

Binární řetězec by pak vypadal následovně: 100010001010100

Dalším důležitým krokem je definice hodnotící funkce. Hodnotící funkce by mohla zohledňovat podobnost datových bodů uvnitř shluku a rozdíl mezi shluky. Můžeme využít několik metod, které již byly popsány v kapitole (5.3) a v kapitole (5.4). Příkladem je třeba vzdálenost mezi datovými body nebo vzdálenost mezi centroidy shluků.

DEAP jako taková nabízí již předdefinované operátory křížení a mutace, ale zároveň poskytuje možnost definovat si operátory vlastní.

Následuje definice GA. Toto zahrnuje nastavení parametrů jako jsou velikost populace (v případě analýzy dat zkoumaných v kapitole (7) by se jednalo o počet datových bodů), počet generací, které chceme vyprodukovat, pravděpodobnost křížení a mutace a v neposlední řadě volbu selekčního mechanismu.

Jakmile máme splněny předchozí body, je možné GA spustit. Na konci algoritmu je vybrán jedinec, který představuje optimální rozřazení datových bodů do shluků.

6.2.2 Scikit-learn

Jedná se o knihovnu vytvořenou v jazyce Python zaměřenou na strojové učení. Obsahuje množství algoritmů pro analýzu dat včetně algoritmů pro shlukování. Nezaměřuje se však primárně na GA a lze ji tak k tomuto účelu využít v kombinaci s jinou knihovnou. Například s již zmíněnou knihovnou DEAP. [24]

Proces tvorby algoritmu je stejný až na to, že k samotné implementaci algoritmu je třeba využít také jiné knihovny. Vznikne tedy algoritmus, který využívá robustních funkcí knihovny scikit-learn ve spojení s GA z knihovny jiné. Vlastnosti takto vytvořených algoritmů tedy závisí i na zvolené knihovně pro implementaci genetického algoritmu.

Některé nabízené funkce:

- **Algoritmy shlukování:** Příkladem jsou algoritmy K-means, DBScan, hierarchické shlukování a další.
- **Algoritmy snižování dimenzionality:** Knihovna nabízí algoritmy pro snižování dimenzionality, jako například PCA (Principal Component Analysis), t-SNE (t-Distributed Stochastic Neigbor Embedding) nebo LLE (Locally Linear Embedding).
- **Algoritmy klasifikace a regrese:** Scikit-learn nabízí široký výběr algoritmů pro klasifikaci a regresi, např. rozhodovací stromy, podpůrné vektory, lineární regresi a další. Tyto algoritmy umožňují předpověď hodnot nebo klasifikaci nových dat na základě trénovacích dat.
- **Nástroje pro evaluaci:** Příkladem takových funkcí jsou funkce pro výpočet přesnosti, zpětné vazby, F-míry, ROC křivky a další. Těmito funkcemi dokážeme porovnat výkonnosti různých modelů
- **Preprocessing dat:** Normalizace, standardizace, kategorické proměnné a další.
- **Křížová validace:** Ta nám pomáhá odhadnout výkonnost modelu a zjišťovat, zda je možné model generalizovat na nová data.

Porovnávání Scikit-learn s knihovnami zaměřenými na GA nedává příliš smysl, protože se Scikit-learn zaměřuje na analýzu dat a slouží jako pomocník knihovnám založených pro práci s GA.

6.2.3 JGAP (Java Genetic Algorithms Package)

Na rozdíl od DEAP je JGAP knihovnou pro implementaci, jak již název napovídá, v Javě. Je navržena tak, aby byla co nejjednodušší na využití, ale zároveň vysoce upravitelná. [25]

Tvorba algoritmu je obdobná jako u knihovny DEAP.

Pokud tyto dvě knihovny porovnáme, zjistíme, že se jednak liší jazyky, ve kterých jsou implementovány – DEAP je v jazyce Python a JGAP v jazyce Java. Obě knihovny nabízejí poměrně velkou flexibilitu, avšak DEAP je o něco málo flexibilnější než JGAP, a to hlavně tím, že nabízí vetší počet funkcí. Obě však dávají možnost definovat si vlastní genetické operátory. DEAP nabízí více podrobných příkladů konkrétních využití knihovny v oblasti shlukové analýzy než JGAP, ale jinak jsou obě dobře dokumentovány na svých webových stránkách. Hlavní výhodou JGAP je však jeho jednoduchost pro začátečníky. DEAP za cenu více funkcí platí větší složitostí a člověk musí knihovnu poměrně důkladně prostudovat.

6.2.4 ECJ (Evolutionary Computation in Java)

Jedná se také o knihovnu v jazyce Java, která si zakládá na své flexibilitě. Poskytuje možnost většinu tříd definovat dynamicky za běhu pomocí souboru s parametry. [26]

Sestrojení GA je opět podobné jako v předešlých případech, je potřeba určit typ reprezentace jedince, definovat hodnotící funkci, definovat si vlastní, nebo využít

již předem definované genetické operátory, definice algoritmu samotného, určení jeho parametrů, a nakonec jeho spuštění.

Pokud porovnáme JGAP a ECJ, což jsou knihovny, které jsou obě implementovány v jazyce Java, uvidíme podobný případ jako při porovnávání JGAP s DEAP. ECJ je náročnější, flexibilnější v počtu funkcí a s větší možností rozšířit knihovnu. Jelikož je ECJ starší, existuje více příkladů využití a více zdrojů informací ohledně knihovny. Výkonově jsou si knihovny podobné a rychlost algoritmu úzce souvisí s řešeným problémem.

6.2.5 Opt4J

Jedná se o open source knihovnu implementovanou v jazyce Java. Obsahuje několik optimalizačních algoritmů, a to včetně GA. Cílem je proces optimalizace pro uživatele zjednodušit pomocí modulové implementace. Součástí je také grafická vizualizace optimalizace a grafické rozhraní. [27]

Při porovnání s předešlými knihovnami je potřeba opět zdůraznit modularitu této knihovny, na které si zakládá. Jednotlivé funkce GA implementuje pomocí modulů a lze ji jednoduše rozšířit o další moduly, které jsou v dané úloze potřebné. Jedná se o knihovnu jednodušší k použití, zejména díky již zmíněnému grafickému rozhraní.

6.2.6 Tabulka podporovaných funkcí jednotlivých knihoven

Funkce	DEAP	ECJ	JGAP	Opt4J
Genetická reprezentace	Ano	Ano	Ano	Ano
Registrace operátorů	Ano	Ano	Ano	Ano
Hodnotící funkce	Ano	Ano	Ano	Ano
Evoluční cyklus	Ano	Ano	Ano	Ano
Modulární architektura	Ano	Ano	Ne	Ano
Vizualizace vývoje	Ano	Ano	Ne	Ano
Multiobjektivní optimalizace	Ano	Ano	Ne	Ano
Evoluce symbolického vyjádření	Ano	Ano	Ano	Ne
Elitismus	Ano	Ano	Ano	Ne přímo
Paralelní výpočet	Ano	Ano	Ne	Ano
Evoluce neuronových sítí	Ano	Ne	Ne	Ne
Snižování dimenzionality	Ne	Ne	Ne	Ne
Preprocessing dat	Ne	Ne	Ne	Ne
Křížová validace	Ne	Ne	Ne	Ne

Tabulka 2: Tabulka podporovaných funkcí jednotlivých knihoven

Pro porovnávání jednotlivých knihoven a tvorbu tabulky bylo potřebné pročíst jednotlivé dokumentace. Pro DEAP je dokumentace zde [23]. Dokumentace scikit-learnu se nachází zde [28], odkaz na dokumentaci knihovny ECJ 27 je zde [29] a pro knihovnu Opt4J zde [30]

Genetická reprezentace: daná knihovna poskytuje nástroje pro reprezentaci jedinců a řeší způsob zakódování jedinců. Typickými formami jsou řetězce (např. binární), stromové struktury atd. Funkce tedy znamená, že knihovna nabízí nástroje pro práci s různými formami kódování jedinců. Zahrnuje generaci počáteční

populace, křížení, mutaci, ohodnocení a další operace s genetickou operací spojené. [31]

Registrace operátorů: Knihovna umožňuje uživateli registrovat vlastní genetické operátory, tj. operátory selekce, křížení a mutace. Díky této funkci může uživatel definovat vlastní operátory. [32]

Hodnotící funkce: Knihovna umožňuje uživatelům definovat a používat vlastní hodnotící funkci. [32]

Evoluční cyklus: Knihovna umožňuje provedení evolučního cyklu. Znamená to, že se mohou procesy selekce, křížení, mutace vyhodnocení kvality a nahrazení části původní populace novými jedinci, opakovat. Tato funkce je pro GA zásadní a umožňuje postupné zlepšování výsledků. [32]

Modulární architektura: Knihovna je navržena tak, aby byla snadno upravitelná a rozšířitelná. Uživatel tak může knihovnu upravit podle požadavků úlohy, kterou řeší. [33]

Vizualizace vývoje: Knihovna disponuje nástroji pro vizualizaci vývoje GA, tj. grafy zobrazující změny kvalit jedinců v průběhu generací a vizualizace slouží k lepšímu pochopení výsledků a umožňuje GA ladit a optimalizovat.

Multiobjektivní optimalizace: Knihovna dokáže řešit úlohy s více než jednou cílovou funkcí a optimalizuje tedy více než jedno kritérium. Příkladem u SA je minimalizace rozptylu uvnitř shluků a současně maximalizace vzdálenosti mezi shluky. [34]

Evoluce symbolického vyjádření: Knihovna dokáže pracovat se symbolickými výrazy. Těmi jsou např. logické operace, matematické výrazy atd.

Elitismus: Jedná se o strategii, která zaručí, že nejlepší jedinec aktuální populace postoupí do další generace, aniž by prošel křížením či mutací. Tato funkce zabraňuje ztrátě nejlepších řešení. [12]

Paralelní výpočet: Knihovna dokáže využít více výpočetních prostředků k současnému vypočítávání a zpracování dat, proces je pak tedy rychlejší a může probíhat několik procesů najednou. [35]

Evoluce neuronových sítí: Knihovna poskytuje nástroje pro evoluci neuronových sítí. Výsledkem je optimalizace architektury, váhových parametrů a struktury neuronových sítí. [36]

Snižování dimenzionality: Knihovna dokáže redukovat počet dimenzí v mnohorozměrných datech. Vyvaruje se tak redundanci a sníží se výpočetní náročnost při zachování důležitých vlastností dat. [37]

Preprocessing dat: Knihovna poskytuje nástroje k tzv. předzpracování dat. Tj. odstranění chybějících hodnot, normalizace, škálování a kódování kategoriálních dat. Data jsou tak lépe připravena a GA s nimi dokáže efektivněji pracovat. [38]

Křížová validace: Knihovna je schopná provést křížovou validaci. Ta je důležitá pro hodnocení modelů a pro odhad výkonnosti. [39]

6.3 Příklad využití knihovny DEAP

Tento příklad ukáže, jak lze obecně využít knihovnu DEAP k vytvoření kódu pro SA pomocí GA.

```
import random

from deap import algorithms, base, creator, tools
```

Zde importujeme potřebné moduly pro vytvoření kódu. [40]

```
#Data pro shlukovou analýzu

data = [...]
```

Zde se nachází proměnná pro importování dat. [40]

```
#Počet shluků

num_clusters = 3
```

Určíme si počet shluků dle typu úlohy, kterou řešíme. [40]

```
#Počet genů v chromozomu

num_genes = len(data[0])
```

Zde určujeme počet genů. Určíme ho jako délku datového bodu. V tomto případě předpokládáme, že mají všechny datové body stejnou délku a použijeme tedy délku prvního z nich. [41]

```
#Vytvoření fitness a jedince

creator.create("FitnessMin", base.fitness, weights=(-1.0,))

creator.create("Individual", list, fitness=creator.FitnessMin)
```

Jelikož v případě shlukové analýzy můžeme měřit například minimální vzdálenost mezi datovými body či středy shluků, hodí se nám využití třídy FitnessMin, která implementuje fitness funkci pro minimalizaci. [40]

```
#Inicializace jedince

def create_individual():

        return[random.randint(0, num_clusters – 1) for _ in  range(num_genes)]
```

Zde vytváříme náhodně inicializované jedince [42] s geny přiřazenými k jednotlivým shlukům. Hodnoty od 0 do num_clusters – 1 představují možné přiřazení datových bodů ke shlukům. [43]

```
#Vyhodnocení jedince

def evaluate_individual(individual):

        # TODO: Implementuje vyhodnocení jedince, např. vzdálenost mezi
datovými body a
```

```
            středy shluků

        fitness = ...

        return fitness,
```

Následuje vyhodnocení jedince, způsobů, jak určit kvalitu čili fitness jedince je několik, příkladem může být již zmíněná vzdálenost.

```
#Genetické operátory

toolbox = base.Toolbox()

toolbox.register("individual", tools.initIterate, creator.Individual,
create_individual)

toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("evaluate", evaluate_individual)

toolbox.register("mate", tools.cxTwoPoint)

toolbox.register("mutate", tools.mutUniformInt, low=0, up=num_clusters - 1,
indpb=0.1)

toolbox.register("select", tools.selTournament, tournsize=3)
```

Zde jsou inicializovány jednotlivé genetické operátory. Využity jsou operátory předdefinované knihovnou DEAP. Operátor *cxTwoPoint* slouží ke křížení dvou jedinců pomocí dvoubodového křížení. Při tomto křížení jsou vybrány dva body v chromozomu a geny mezi těmito body jsou prohozeny mezi rodiči. Operátor *mutUniformInt* provádí mutaci náhodným výběrem jednoho, či více genů a nahrazením jejich hodnoty za náhodně vybranou v rámci rozsahu. [40]

```
def main():
        #Velikost populace
        Population_size = 100


        #Počet generací
        Num_generations = 50


#Vytvoření počáteční populace
population = toolbox.population(n=population_size)


#Spuštění evolučního cyklu
for generation in range(num_generations):
        #Vyhodnocení populace
```

```
        fitnesses = map(toolbox.evaluate, population)
        for individual, fitness in zip(population, fitnesses):
            individual.fitness.values = fitness

        #Výběr nové populace
        offspring = toolbox.select(population, len(population))

        #Kopie potomků pro další generaci
        offspring = list(map(toolbox.clone, offspring))

        #Aplikace křížení a mutace na potomky
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < 0.5:
                toolbox.mate(child1, child2)
                del child1.fitness.values
                del child2.fitness.values

        for mutant in offspring:
            if random.random() < 0.2:
                toolbox.mutate(mutant)
                del mutant.fitness.values

        #Nahrazení populace potomky
        population[:] = offspring

#Výběr nejlepšího jedince
best_individual = tools.selBest(population, k=1)[0]
best_fitness = best_individual.fitness.values[0]

print("Nejlepší jedinec:", best_individual)
print("Fitness:", best_fitness)
```

```
if __name__ == "__main__":
    main()
```

Poslední částí kódu je samotný evoluční cyklus. Nejdříve dojde k vyhodnocení populace, vybere se nová populace na základě hodnot fitness a aplikují se genetické operátory. [44] Potomci poté nahradí původní generaci a proces se opakuje po definovaný počet generací. Na konci se vypíše nejlepší jedinec a vypíše se jeho fitness hodnota. [45]. Evoluční cyklus se opakuje, dokud není splněna podmínka.

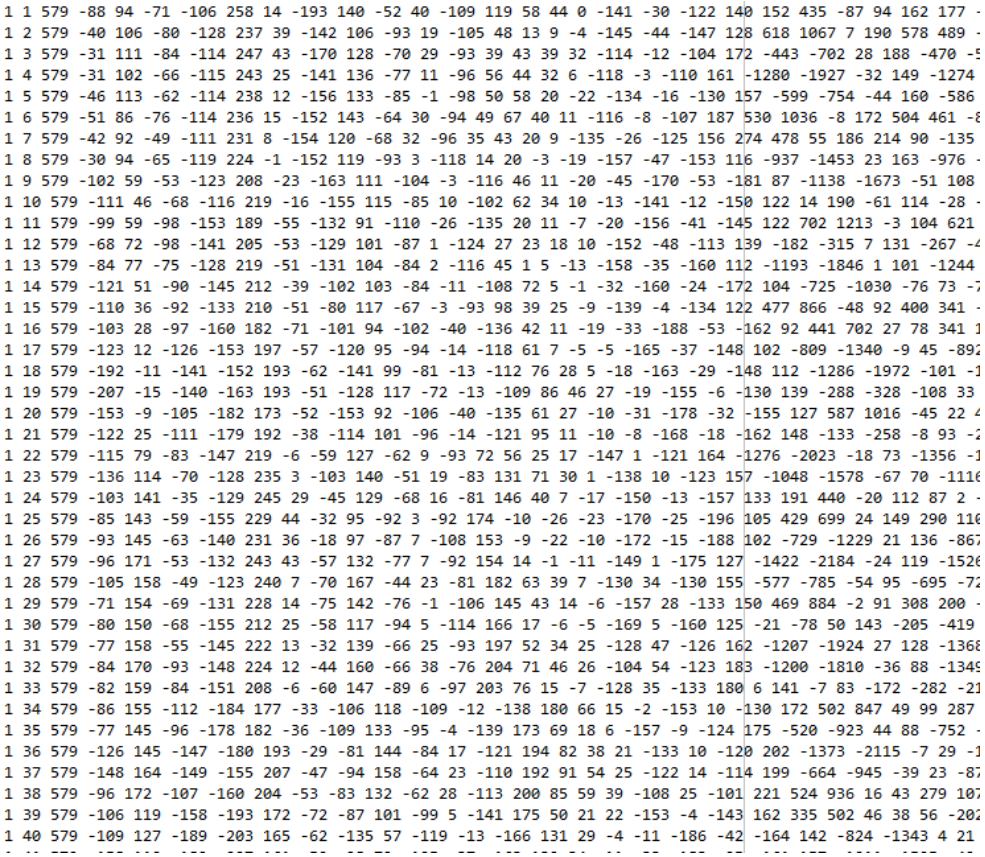
7 Popis dat poskytnutých ze simulátoru

7.1 Rozdělení dat

Poskytnutá data se dělí na data ze simulátoru a EEG (elektroencefalografie) data. Dále se dělí na data bez spánkové deprivace a se spánkovou deprivací. Soubory s daty jsou v příloze. K rozlišení původu dat napomáhá jejich pojmenování, přesněji řečeno poslední písmenko názvu, které je buď *s (simulátor)* nebo *e (encefalograf).*

7.2 Úprava výchozích dat

Původní data byla poskytnuta v textovém dokumentu. Za účelem přehlednosti byla převedena do excelového sešitu.



Obrázek 13: Původní formát dat

Číslo červené	Číslo řádku	Reakční čas	Čas od zapnutí záznamu	X-ová souřadnice středu auta	Y-ová souřadnice středu auta (výška)	Z-ová souřadnice	Rychlost	Synchr.	Města	Čas. značky	flag	natočení volantu	plyn	brzda	spojka (nepouž.)	pneumatika 1 mimo silnici	penumatika 2	pneumatika 3	pneumatika 4	semafor
1	1	579	91387	2498.031006	0.860077	-1182.34021	24.104773	0	0	6	0	-5	290	0	1000	0	0	0	0	2
1	2	579	91402	2498.037109	0.860283	-1181.978882	24.102182	0	0	6	0	-5	288	0	1000	0	0	0	0	2
1	3	579	91417	2498.043213	0.860488	-1181.617554	24.099564	0	0	6	0	-5	288	0	1000	0	0	0	0	2
1	4	579	91435	2498.050537	0.860734	-1181.18396	24.096416	0	0	6	0	-5	288	0	1000	0	0	0	0	2
1	5	579	91450	2498.056641	0.860938	-1180.822632	24.093821	0	0	6	0	-7	290	0	1000	0	0	0	0	2
1	6	579	91465	2498.0625	0.861142	-1180.461304	24.091223	0	0	6	0	-7	288	0	1000	0	0	0	0	2
1	7	579	91480	2498.067383	0.861345	-1180.099976	24.0886	0	0	6	0	-7	288	0	1000	0	0	0	0	2
1	8	579	91495	2498.072266	0.861548	-1179.738647	24.085968	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	9	579	91513	2498.078125	0.86179	-1179.305054	24.082811	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	10	579	91528	2498.083008	0.861991	-1178.943726	24.080185	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	11	579	91543	2498.087891	0.862191	-1178.582397	24.07756	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	12	579	91558	2498.092041	0.862391	-1178.221069	24.074848	0	0	6	0	-8	283	0	1000	0	0	0	0	2
1	13	579	91573	2498.095703	0.86259	-1177.859741	24.07222	0	0	6	0	-8	293	0	1000	0	0	0	0	2
1	14	579	91591	2498.100098	0.862828	-1177.42627	24.06916	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	15	579	91606	2498.10376	0.863026	-1177.065552	24.066551	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	16	579	91621	2498.107422	0.863222	-1176.704834	24.064028	0	0	6	0	-8	293	0	1000	0	0	0	0	2
1	17	579	91636	2498.111084	0.863418	-1176.344116	24.061544	0	0	6	0	-8	290	0	1000	0	0	0	0	2
1	18	579	91651	2498.113525	0.863613	-1175.983398	24.058985	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	19	579	91669	2498.116455	0.863846	-1175.550537	24.055862	0	0	6	0	-8	288	0	1000	0	0	0	0	2
1	20	579	91684	2498.118896	0.86404	-1175.189819	24.053225	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	21	579	91699	2498.121338	0.864232	-1174.829102	24.050587	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	22	579	91714	2498.123291	0.864423	-1174.468384	24.047956	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	23	579	91729	2498.124512	0.864614	-1174.107666	24.045332	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	24	579	91747	2498.125977	0.864842	-1173.674805	24.042187	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	25	579	91762	2498.127197	0.865031	-1173.314087	24.039568	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	26	579	91777	2498.128174	0.86522	-1172.953369	24.036953	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	27	579	91792	2498.128174	0.865407	-1172.592651	24.034302	0	0	6	0	-11	286	0	1000	0	0	0	0	2
1	28	579	91810	2498.128174	0.865631	-1172.15979	24.031078	0	0	6	0	-11	286	0	1000	0	0	0	0	2
1	29	579	91825	2498.128174	0.865817	-1171.799072	24.028423	0	0	6	0	-11	288	0	1000	0	0	0	0	2
1	30	579	91840	2498.128174	0.866002	-1171.438843	24.02582	0	0	6	0	-9	288	0	1000	0	0	0	0	2
1	31	579	91855	2498.127441	0.866186	-1171.078735	24.023201	0	0	6	0	-8	286	0	1000	0	0	0	0	2
1	32	579	91870	2498.126221	0.86637	-1170.718628	24.020576	0	0	6	0	-8	288	0	1000	0	0	0	0	2

Obrázek 14: Formát dat po úpravě

7.2.1 Popis dat ze simulátoru

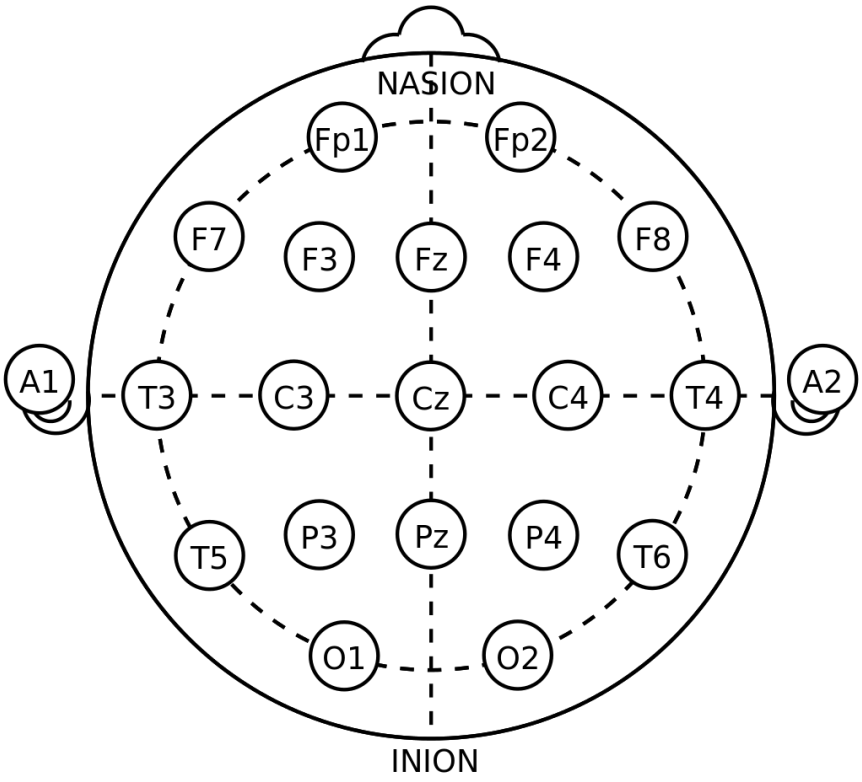
Data jsou rozdělená do několika sloupců, zde jsou významy sloupců pro data ze simulátoru:

1. Pořadové číslo červené (stejné pro *x* řádků)
2. Číslo řádku (vždy *1* až *x*, pro každou červenou – *10 s* před červenou)
3. Reakční čas (stejné pro *x* řádků)
4. Čas od zapnutí záznamu [ms]
5. X-ová souřadnice „středu“ auta [m]
6. Y-ová souřadnice „středu“ auta [m]
7. Z-ová souřadnice „středu“ auta [m]
8. Rychlost [m/s]
9. Synchronizace
10. Města
11. Časové značky
12. Flag (nepoužívané)
13. Natočení volantu (-1000 až 1000)
14. Sešlápnutí plynového pedálu (0 až 1000)
15. Sešlápnutí brzdového pedálu (0 až 1000)
16. Spojka (nepoužívané)
17. Pneumatika (1) mimo silnici (jiná hodnota než nula)
18. Pneumatika (2) mimo silnici (jiná hodnota než nula)
19. Pneumatika (3) mimo silnici (jiná hodnota než nula)
20. Pneumatika (4) mimo silnici (jiná hodnota než nula)

21. Semafor (0…červená, 1…oranžová, 2…zelená)

7.2.2 Popis dat z EEG

1. Pořadové číslo červené (stejné pro *x* řádků)
2. Číslo řádku (vždy *1* až *x*, pro každou červenou – *10 s* před červenou)
3. Reakční čas (stejné pro *x* řádků)
4. Fp1 (levá přední pólová elektroda, F = frontální lalok)
5. Fp2 (pravá přední pólová elektroda, F = frontální lalok)
6. F7 (levá přední elektroda, F = frontální lalok)
7. F3 (levá přední elektroda, F = frontální lalok)
8. Fz (střední přední elektroda, F = frontální lalok)
9. F4 (pravá přední elektroda, F = frontální lalok)
10. F8 (pravá přední elektroda, F = frontální lalok)
11. T3 (levá elektroda, T = temporální lalok)
12. C3 (levá elektroda umístěná nad oblastí mozku)
13. Cz (elektroda umístěna v tzv. *centrálním nulovém bodě hlavy*)
14. C4 (pravá elektroda umístěná nad oblastí mozku)
15. T4 (pravá elektroda, T = temporální lalok)
16. T5 (levá elektroda, T = temporální lalok)
17. P3 (levá elektroda, P = parietální lalok)
18. Pz (střední elektroda, P = parietální lalok)
19. P4 (pravá elektroda, P = parietální lalok)
20. T6 (pravá elektroda, T = temporální lalok
21. O1 (levá elektroda, O = okcipitální lalok)
22. O2 (pravá elektroda, O = okcipitální lalok)
23. A1 (elektroda na levém ušním laloku)
24. A2 (elektroda na pravém ušním laloku)
25. EOG1 (elektroda měřící oční pohyby)
26. EOG2 (elektroda měřící oční pohyby)
27. EOG3 (elektroda měřící oční pohyby)
28. EOG4 (elektroda měřící oční pohyby)
29. EMG1 (elektroda měřící elektrickou aktivitu svalů)
30. EMG2 (elektroda měřící elektrickou aktivitu svalů)
31. EMG3 (elektroda měřící elektrickou aktivitu svalů)
32. EMG4 (elektroda měřící elektrickou aktivitu svalů)
33. EKG1 (elektroda měřicí elektrickou aktivitu srdce)
34. EKG2 (elektroda měřicí elektrickou aktivitu srdce)
35. Foto (nepoužívané)
36. Index (po čtyřech)



Obrázek 15: Schéma rozložení elektrod podle sytému 10-20. Převzato z [51]

8 Návrh využití shlukové analýzy pro zpracování dat

8.1 Návrh genetického algoritmu

Důležité je zvolit si, jak budou jednotlivci (chromozomy) reprezentováni. Jeden příklad je takový, že pokud chceme rozdělit například 100 záznamů do 5 shluků, chromozom můžeme chápat jako vektor o délce 100, kde je každý prvek číslo 1-5 a reprezentuje tedy přiřazení k jednotlivým shlukům. Následně vygenerujeme počáteční populaci chromozomů a náhodně jim přidělíme shluky. Fitness funkci navrhneme tak, aby byla vnitroshluková vzdálenost co nejmenší a mezishluková vzdálenost co největší. Následujícím krokem je využití genetických operátorů křížení a mutace, dále je možno využít například turnajové selekce. Nahradíme původní generaci za novou a celý postup opakujeme, dokud nedosáhneme uspokojivého řešení.

8.2 Definice jednotlivých shluků pro data z EEG

Při analýze dat získaných pomocí EEG senzoru je hned několik možností, jak definovat jednotlivé shluky. Můžeme data kupříkladu rozdělit do *časových oken* a každá tato část by reprezentovala jeden shluk. Dalším příkladem je definice shluků na základě *význačných rysů*, těmi jsou frekvenční charakteristiky, jelikož různé stavy bdělosti mají různé frekvenční charakteristiky v EEG datech. Sledovat také můžeme vzory v EOG a EMG datech, kdy aktivita očního pohybu nebo svalová aktivita může indikovat různé stavy. Jako shluky můžeme také považovat *význačné události*, které se objevují před upadnutím do spánku, jako například náhlé skoky v signálu.

8.3 Zpracování dat ze simulátoru

Definujeme si vlastnosti, které se nám budou hodit ke zhodnocení stavu řidiče a pozice vozidla. Ze souřadnic a rychlosti vozidla můžeme využít k identifikaci neobvyklého chování, například náhlé zrychlení nebo zpomalení.

Z natočení volantu, sešlápnutí plynového a brzdového pedálu můžeme zase zjistit, jak aktivně reaguje řidič na události na vozovce. Informace o pozice pneumatiky zase říká, jestli řidič vozovku opustil, nebo ne. To může indikovat únavu řidiče.

Při kombinaci s daty z EEG můžeme propojit fyzické reakce řidiče s jeho chováním za volantem.

8.4 Definice jednotlivých shluků pro data ze simulátoru

Opět můžeme shluky definovat jako *časová okna*, které obsahují průměry a směrodatné odchylky výše zmíněných vlastností. Jinou možností je definice shluků na základně *specifických chování* – normální jízda, zpomalení, zrychlení, náhle a silné zatočení volantem. Je také možné nechat GA definovat automaticky na základě kombinace vlastností.

9 Závěr

Cílem této práce bylo prostudovat principy genetických algoritmů. Popsal jsem, jak genetické algoritmy fungují, důležité termíny, které jsou potřebné k pochopení celé problematiky genetických algoritmů, a ukázal jejich funkčnost na konkrétním příkladě.

Dalším cílem bylo prostudovat principy shlukové analýzy. Představil jsem shlukovou analýzu nejprve obecně, dále jsem ji rozdělil do jednotlivých podskupin a následně popsal to, jak fungují.

V práci také řeším propojení metody genetických algoritmů a shlukové analýzy, kdy představuji konkrétní příklad, popisuji existující knihovny různých programovacích jazyků a jejich silné stránky a následně je mezi sebou porovnávám. Pokusil jsem se také vytvořit kostru příkladu psaného za pomocí knihovny DEAP.

Posledním bodem mé práce bylo popsat data poskytnutá doc. Ing. Vítem Fáberou, Ph.D. a ústavem K616 a navrhnout, jak by se dala synergie genetických algoritmů a shlukové analýzy využít k jejich analýze. Zde se nejvíce projevuje moje neznalost praktické stránky problému a mnou navrhované způsoby nejsou příliš konkrétní.

Jednotlivé body práce jsem popsal a prostudoval jsem manuály jednotlivých knihoven zabývajících se genetickými algoritmy.

Rád bych na tuto práci navázal diplomovou prací na konkrétním využití GA a na praktickém využití principů GA.

10 Použité zdroje

[1] Úvod. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 117. ISBN 80-200-0472-6.

[2] FOGEL, L., A. OWENS a M. WALSH. *Artificial intelligence through simulated evolution*. New York: Wiley, 1966. ISBN 0471265160.

[3] Základní charakteristiky evolučních výpočetních technik. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 118. ISBN 80-200-0472-6.

[4] Základní charakteristiky evolučních výpočetních technik. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 119. ISBN 80-200-0472-6.

[5] Základní charakteristiky evolučních výpočetních technik. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 120. ISBN 80-200-0472-6.

[6] Genetické algoritmy. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 121. ISBN 80-200-0472-6.

[7] LUNER, Petr. Jemný úvod do genetických algoritmů. In: *Computer Graphics Group* [online]. Praha: Computer Graphics Group [cit. 2023-08-06]. Dostupné z: https://cgg.mff.cuni.cz/~pepca/prg022/luner.html

[8] Ohodnocení jedinců a selekce. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 122-123. ISBN 80-200-0472-6.

[9] Selekce v konečných populacích. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. 1. Praha: Academia, 2001, s. 133-134. ISBN 80-200-0472-6.

[10] Úpravy selekčního tlaku. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. 1. Praha: Academia, 2001, s. 135. ISBN 80-200-0472-6.

[11] Turnajová selekce. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. 1. Praha: Academia, 2001, s. 137. ISBN 80-200-0472-6.

[12] Náhradové strategie. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. 1. Praha: Academia, 2001, s. 137-138. ISBN 80-200-0472-6.

[13] SLIVONĚ, Miroslav. ŘEŠENÍ PROBLÉMU LOKACE HUBŮ POMOCÍ GENETICKÉHO ALGORITMU. *Perner's Contacts* [online]. 2008, **3**(4), 96-102 [cit. 2023-08-02]. Dostupné z: https://pernerscontacts.upce.cz/index.php/perner/article/view/1322/1106

[14] KRATICA, J., Z. STANIMIROVIC, D. TOSIC a V. FILIPOVIC. Wo Genetic Algorithms for Solving the Uncapacitated Single Allocation p-Hub Median Problem. *European Journal of Operational Research* [online]. 2007, **182**(1), 15-28 [cit. 2023-08-06]. ISSN 0377-2217. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0377221706006163

[15] TOPCUOGLU, H., F. CORUT, M. ERMIS a G. YILMAZ. Solving the uncapacitated hub location problem using genetic algorithms. *Computers & Operations Research* [online]. 2005,

32(4), 967-984 [cit. 2023-08-06]. Dostupné z: https://www.sciencedirect.com/science/article/abs/pii/S030505480300279X

[16] KUČERA, Jiří. Shluková analýza. In: *Informační systém Masarykovy univerzity* [online]. Brno [cit. 2023-08-02]. Dostupné z: https://is.muni.cz/th/172767/fi_b/5739129/web/web/main.html

[17] SPOHNEROVÁ, Kateřina. *Shluková analýza*. Omolouc, 2010. Diplomová práce. Vedoucí práce Mgr. Jaroslav Marek, Ph. D.

[18] KUČERA, Jiří. Hierarchické metody shlukování. In: *Informační systém Masarykovy univerzity* [online]. Brno [cit. 2023-08-02]. Dostupné z: https://is.muni.cz/th/172767/fi_b/5739129/web/web/hiermet.html

[19] KUČERA, Jiří. Nehierarchické metody shlukování. In: *Informační systém Masarykovy univerzity* [online]. Brno [cit. 2023-07-26]. Dostupné z: https://is.muni.cz/th/172767/fi_b/5739129/web/web/nehiermet.html

[20] Metoda k-průměrů. In: *Matematická biologie: e-learningová učebnice* [online]. Brno: Institut biostatistiky a analýz Lékařské fakulty Masarykovy univerzity [cit. 2023-08-02]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-nehierarchicka-analyza--metoda-k-prumeru

[21] JINDROVÁ, Pavla. Disparity krajů ČR. *Scientific Papers of the University of Pardubice.Series D.Faculty of Economics and Administration* [online]. 2009, (14), 56-63 [cit. 2023-08-02]. ISSN ISSN 1211555X. Dostupné z: https://www.proquest.com/docview/2265542200?accountid=15618&parentSessionId=ud9yEVssotxVoNS0LHW53hDUwZ5BNfYRzlHdX92qsj4%3D&pq-origsite=primo

[22] Co je Hammingova vzdálenost. In: *Icy Science* [online]. [cit. 2023-08-06]. Dostupné z: https://cs.theastrologypage.com/hamming-distance

[23] DEAP documentation. In: *DEAP* [online]. DEAP project, c2009-2023 [cit. 2023-08-02]. Dostupné z: https://deap.readthedocs.io/en/master/

[24] Scikit-learn. In: *Scikit-learn* [online]. Scikit-Learn Consortium at Inria Foundation, 2007 [cit. 2023-08-02]. Dostupné z: https://scikit-learn.org/stable/

[25] JGAP. In: *Sourceforge* [online]. San Diego: SourceForge Headquarters [cit. 2023-08-02]. Dostupné z: https://sourceforge.net/projects/jgap/

[26] ECJ 27. In: *Department of Computer Science* [online]. Fairfax: George Mason University [cit. 2023-08-02]. Dostupné z: https://cs.gmu.edu/~eclab/projects/ecj/

[27] OPT4J. In: *Github* [online] [cit. 2023-08-02]. Dostupné z: https://sdarg.github.io/opt4j/

[28] User guide. In: *Https://scikit-learn.org/stable/* [online]. Scikit-Learn Consortium at Inria Foundation, 2007 [cit. 2023-08-06]. Dostupné z: https://scikit-learn.org/stable/user_guide.html

[29] The ECJ Owner's Manual. In: *ECJ* [online]. Fairfax: George Mason University, 2019 [cit. 2023-08-06]. Dostupné z: https://cs.gmu.edu/~eclab/projects/ecj/manual.pdf

[30] Opt4J - A Modular Framework for Meta-heuristic Optimization. *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation* [online]. New

York: Association for Computing Machinery, 2011, 1723-1730 [cit. 2023-08-02]. ISSN 978-1-4503-0557-0. Dostupné z: https://dl.acm.org/doi/proceedings/10.1145/2001576

[31] Genetic Representation. In: *Academic Accelerator* [online]. Boise, Idaho: Academic Accelerator's headquarters [cit. 2023-08-02]. Dostupné z: https://academic-accelerator.com/encyclopedia/genetic-representation

[32] MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001. ISBN 80-200-0472-6.

[33] What is modular programming?. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021 [cit. 2023-08-02]. Dostupné z: https://www.javatpoint.com/what-is-modular-programming

[34] Multiobjective Optimization. In: *ScienceDirect* [online]. Amsterdam: Elsevier, 2023 [cit. 2023-08-02]. Dostupné z: https://www.sciencedirect.com/topics/engineering/multiobjective-optimization

[35] What is Parallel Computing?. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021 [cit. 2023-08-02]. Dostupné z: https://www.javatpoint.com/what-is-parallel-computing

[36] LAZAROU, Conor. Evolving Neural Networks. In: *Towards Data Science* [online]. Toronto: TDS Team [cit. 2023-08-02]. Dostupné z: https://towardsdatascience.com/evolving-neural-networks-b24517bb3701

[37] Introduction to Dimensionality Reduction Technique: What is Dimensionality Reduction?. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021 [cit. 2023-08-02]. Dostupné z: https://www.javatpoint.com/dimensionality-reduction-technique

[38] Data Preprocessing in Machine learning. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021 [cit. 2023-08-02]. Dostupné z: https://www.javatpoint.com/data-preprocessing-machine-learning

[39] Cross-Validation in Machine Learning. In: *JavaTpoint* [online]. JavaTpoint, c2011-2021 [cit. 2023-08-02]. Dostupné z: https://www.javatpoint.com/cross-validation-in-machine-learning

[40] One Max Problem. In: *Github-DEAP* [online]. DEAP project [cit. 2023-08-02]. Dostupné z: https://deap.readthedocs.io/en/master/examples/ga__onemax.html

[41] Python len() Function. In: *W3Schools* [online]. W3Schools, c1999-2023 [cit. 2023-08-02]. Dostupné z: https://www.w3schools.com/python/ref__func__len.asp

[42] Python Random randint() Method. In: *W3Schools* [online]. W3Schools, c1999-2023 [cit. 2023-08-02]. Dostupné z: https://www.w3schools.com/python/ref__random__randint.asp

[43] Benchmarks: Benchmarks Tools. In: *Github-DEAP* [online]. DEAP project [cit. 2023-08-02]. Dostupné z: https://deap.readthedocs.io/en/master/api/benchmarks.html?highlight=def%20create__individual()%3A#deap.benchmarks.tools.scale.scale

[44] Python map() function. In: *Geeks For Geeks* [online]. Geeksforgeeks [cit. 2023-08-02]. Dostupné z: https://www.geeksforgeeks.org/python-map-function/

[45] Python print() Function. In: *W3Schools* [online]. W3Schools, c1999-2023 [cit. 2023-08-02]. Dostupné z: https://www.w3schools.com/python/ref_func_print.asp

[46] Křížení a mutace. In: *Wikipedia* [online]. Miloš Křivan, 2022 [cit. 2023-08-06]. Dostupné z: https://cs.wikipedia.org/wiki/Genetick%C3%BD_algoritmus#/media/Soubor:K%C5%99%C3%AD%C5%BEen%C3%AD_mutace.png

[47] Vývojový diagram základního algoritmu EVT. In: MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ. *Umělá inteligence (3)*. Praha: Academia, 2001, s. 120. ISBN 80-200-0472-6.

[48] JANOUŠKOVÁ, Eva a Ladislav DUŠEK. Hierarchické aglomerativní metody shlukování. In: *SlidePlayer* [online]. SlidePlayer.cz Inc., 2023 [cit. 2023-08-02]. Dostupné z: https://slideplayer.cz/slide/13739987

[49] Metoda nejbližšího souseda. In: *Nauč se Python* [online]. [cit. 2023-08-02]. Dostupné z: https://naucse.python.cz/2020/pydata-ostrava-jaro/pydata/clustering/

[50] *Centroidní metoda: Matematická biologie: e-learningová učebnice* [online]. In: . Brno: Institut biostatistiky a analýz Lékařské fakulty Masarykovy univerzity [cit. 2023-08-02]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--vicerozmerne-metody-pro-analyzu-dat--shlukova-analyza--shlukova-hierarchicka-analyza--hierarchicke-shlukovani--hierarchicke-aglomerativni-shlukovani

[51] 21 electrodes of International 10-20 system for EEG. In: *Wikipedia* [online]. [cit. 2023-08-06]. Dostupné z: https://commons.wikimedia.org/wiki/File:21_electrodes_of_International_10-20_system_for_EEG.svg

Seznam obrázků

Seznam tabulek

Seznam příloh