UNIVERZITA PARDUBICE
Dopravní Fakulta Jana Pernera

Generátor báze fuzzy pravidel

Bc. Martin Kosina

Diplomová práce

2008

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Katedra informatiky v dopravě
Akademický rok: **2007/2008**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin KOSINA**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Generátor báze fuzzy pravidel**

Zásady pro vypracování:

Cílem diplomové práce je vytvoření počítačové aplikace – generátoru báze fuzzy pravidel. Aplikace by měla umožňovat volbu počtu a tvaru funkcí příslušnosti hodnot proměnných k fuzzy množině, následné vytváření pravidel typu *if-then* a uložení získaných pravidel do souboru vhodného formátu.
Práce bude obsahovat kapitoly:

1. Úvod, motivace řešení úlohy.
2. Teoretické a metodické základy řešení.
3. Analýza úlohy, potřebná vstupní data, návrh výstupů.
4. Algoritmizace úlohy.
5. Programová implementace.
6. Aplikace na konkrétním příkladu.
7. Závěry, zhodnocení zkušeností a doporučení.

Výstupem práce bude samostatně běžící aplikace napsaná v programovém prostředí Delphi. Program musí být odevzdán včetně zdrojových kódů.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **JURA, P.** *Základy fuzzy logiky pro řízení a modelování*. **Brno : VUTIUM, 2003. ISBN 80-214-2261-0.**

2. **Matlab**
   Fuzzy Toolbox Users Guide. The MathWorks, Inc. Dostupné na: <http://www.mathworks.com/access/helpdesk/help/pdf_doc/fuzzy/fuzzy.pdf>.

Vedoucí diplomové práce: **Mgr. Věra Záhorová, Ph.D.**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **4. prosince 2007**

Termín odevzdání diplomové práce: **4. června 2008**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 30. listopadu 2007

Souhrn

Práce je zaměřena na problematiku vyhledávání náhradní nástupištní koleje pro osobní vlak přijíždějící do železniční stanice. Důvodem volby jiné nástupištní koleje může být zpoždění osobního vlaku a z toho plynoucí obsazenost přidělené koleje jiným vlakem. Práce si klade za cíl vytvoření báze pravidel pro takovéto situace. Díky těmto pravidlům by mohlo být v budoucnu postupováno při výběru vhodné koleje v obdobných situacích. Pravidla se generují za pomocí metod fuzzy. Celá problematika je řešena – tam kde to lze – pomocí automatizace a výpočetní techniky. Vstup tvoří provozní data z železniční stanice Praha hlavní nádraží. Za výstup jsou považována pravidla typu *if–then* exportovaná do souboru vhodného formátu.

Klíčová slova

náhradní kolej; zpožděný vlak; fuzzy; mamdani;

Title

The generator of fuzzy rules based

Abstract

Work is focused on building a searching an alternative platform rail for passenger trains arriving at the railway station. The reason for the choice of other platform track may be a delay passenger train and the consequent occupation attributed to the track by another train. Labor aims to build a framework of rules for such situations. Thanks to these rules could be followed in selecting the appropriate track in similar situations. The rules are generated using methods of fuzzy. The whole issue is dealt with – where it can – through automation and computer technology. Entry form the historical data from the railway station Prague Main Railway Station. For output rules are considered the type of *if-then* exported to a file format.

Keywords

alternative track; delay train; fuzzy; mamdani

Poděkování

V celé práci bylo stavěno na základech práce Věry Záhorové a i ostatní postupy s ní byly často konzultovány. Za tyto konzultace a čas jí patří velké poděkování. Dále patří poděkování také společnosti AŽD Praha, která poskytla program GTN a provozní data.

Obsah

Seznam obrázků

Seznam tabulek

1 Cíl práce

Cílem diplomové práce je v uživatelsky příjemném prostředí vytvořit počítačovou aplikaci, jež na bázi historických dat ze skutečného provozu železniční stanice vygeneruje bázi fuzzy pravidel. Pro hodnocení kolejí ve stanici je použito několik kritérii. Uvažujeme zde i okolí systému. Při hodnocení kolejí bereme v úvahu:

- aktuální obsazení kolejí ve stanici
- plánované obsazení kolejí v následujícím časovém období
- rozmístění přípojných vlaků

Při hodnocení, je možné zvolit některou z následujících strategii:

- hodnocení kolejí s prioritou dodržení původního plánu obsazení kolejí vycházející z grafikonu vlakové dopravy
- hodnocení kolejí s prioritou minimalizace přestupní doby mezi zpožděnými a přípojnými vlaky
- hodnocení kolejí s minimalizací zpoždění dalších vlaků, které může vzniknout jako přímý důsledek změny obsazení kolejí

Získané hodnoty, po průchodu nastavenými funkcemi příslušnosti, jsou vyhodnocovány a tvoří základ báze pravidel pro další zpracování v systémech fuzzy. Tento postup nazýváme fuzzyfikací proměnných.

Řešená úloha generátoru pravidel je pouze částí komplexního problému výběru vhodné nástupištní koleje pro osobní vlak.

2 Sběr dat

Základem práce je soubor skutečných dat přijíždějících vlaků do železniční stanice Praha hl.n., od kterých je vyžadována maximální přesnost. Z tohoto důvodu byl již z prvopočátku zavrhnut sběr dat z doposud používaného „papírového“ železničního deníku který – díky lidskému faktoru – obsahuje informace již zkreslené. Neboť který z výpravčích či jiných zaměstnanců železnice by nechtěl, aby právě z jeho nádraží odjížděli / přijížděli vlaky téměř přesně. Díky této skutečnosti byl sběr dat uskutečněn z programu GTN vyvinutý firmou AŽD Praha s.r.o..

2.1 Základní předpoklady

Důležitým upozorněním je, že v této práci abstrahujeme od Pn dopravy (nákladní) a zabýváme se pouze dopravou osobní, jež nám poskytuje potřebná data o přípojích a vzdálenosti nástupišť, jež jsou dále vyhodnocována s ohledem na pohodlí přepravovaného klienta. Dále také není brán ohled na směr jízdy vlaku. Tohoto poznatku se v praxi využívá k přesnému určení přípojných vlaků. Tento aspekt je zevšeobecněn a za přípojný vlak je považován takový vlak, který odjíždí 8minut po ohlášení námi zkoumaného vlaku.

2.2 GVD a ODK

Grafikon dopravy v obecném pojetí chápeme jako grafické znázornění pohybu dopravních spojů (vozidel, vlaků) po dané trase. Spoje jsou zobrazeny jako lomené čáry nebo úsečky na podkladu kartézské soustavy souřadnic. Vodorovná osa obvykle naznačuje plynutí času v rámci provozního dne. Svislá osa značí místa na trase (stanice, zastávky, křižovatky, kontrolní

body, …). Jízda jedním směrem je zobrazena šikmou čarou směřující doprava nahoru, jízda zpět šikmou čarou směřující doprava dolů. Stání vozidla nebo vlaku v mezilehlém nebo koncovém místě je znázorněno vodorovnou částí čáry, rychlejší jízda větším sklonem čáry.

V železniční dopravě se používá pojem grafikon vlakové dopravy (GVD), kde je grafikon doplněn popisky, jež lze souhrnně nazvat „služebními pomůckami“, související s organizací práce na železnici.

Zdroj [5]

Plán obsazení dopravních kolejí (Plán ODK) železniční stanice se kterým bylo uvažováno v této práci je specifickou instancí popisovaného grafikonu vlakové dopravy. Jedná se o předpis, dle kterého jsme schopni sestavit situaci v železniční stanici, pro kterou je vypracován, v kterémkoliv okamžiku a na kterékoliv koleji. Je doplněn řadou pomůcek, které jsou graficky či textově zobrazeny. Celý plán je k dispozici v příloze A této práce.

2.3 GTN

Tento program i v režimu off-line dokáže interpretovat vybraný úsek v železniční stanici téměř totožně jako papírová forma dopravního deníku. Firma AŽD poskytla informace z celého měsíce srpna 2006. Zkoumané období pro daný projekt bylo stanoveno v časovém rámci 1.8.2006 – 10.8.2006. Základní měrnou jednotkou programu pro historická data je jeden den a pro události obsažené v tomto dni na sledovaném železničním úseku je jedna vteřina.

Program Graficko–technologická nadstavba zabezpečovacího zařízení postihuje několik typů událostí a díky funkci elektronické dopravní dokumentace (ELDODO) se v něm zpracovávají a uchovávají informace o uskutečněné vlakové dopravě. Data jsou sbírána přímo ze zabezpečovacích zařízení a jsou prokazatelně přesná a nezpochybnitelná. Tato funkce je stěžejní pro náš sběr dat.

Protokol obsluhy

Protokol obsluhy jež je součástí GTN a v chronologickém sledu uchovává události o dopravních procesech. Dále je interpretuje tabelární formou. Protokol obsluhy eviduje několik základních událostí:

- povinně dokumentovatelné úkony
- jízdu vlaku a stavění vlakových cest
- předání dopravny na místní × dálkový provoz
- odevzdávku dopravní služby

Klíčová vlastnost, jež byla při práci s Protokolem obsluhy nejvíce využita, je filtrování dat.

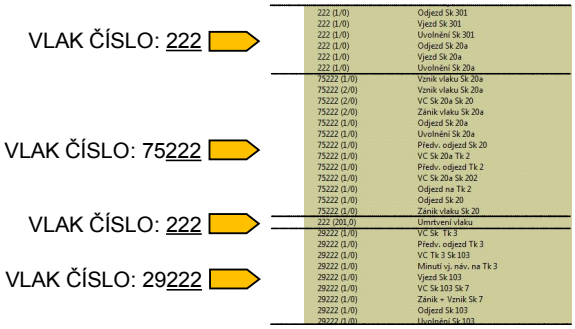
Protokol obsluhy – nedostatky při práci a její automatizace

Při filtrování dat bylo prioritou zvolit takový vlak, jež přijíždí na jinou nástupišťní kolej (Sk – staniční kolej) než jaká je mu přiřazena. V daném časovém období považujeme za předpis aktuální grafikon vlakové dopravy, respektive plán obsazení kolejí. Tímto grafikonem bohužel GTN nedisponovala, i když problematiku GVD částečně řeší.

Druhou nepříjemnou vlastností filtrování v modulu Protokol obsluhy je tzv. fulltextové vyhledávání bez možnosti další parametrizace. Například zvolit příslušný sloupec atributů jako číslo vlaku, odjezd, … . Ve výsledku se vyfiltrují všechny řádky obsahující hledaný řetězec

v kterémkoliv atributu daného řádku. Jako ilustrativní příklad může posloužit vlak číslo 230, který má přijíždět v 18:10 do sledovaného úseku pražského hlavního nádraží. V tu chvíli se ve splněné dopravě celého dne může také objevit vlak 2230 nebo 2301. V podobných případech se hledání pomocí filtrů v tisíci řádkovém výpisu může jevit jako zhola nemožný úkol.

Automatizované zpracování vstupních dat by dle prvního záměru mohlo probíhat ve vlastním programu, jež by byl součástí práce. Odpadly by oba problémy, které jsou nastíněny výše. Bohužel, i tato možnost byla při bližším zkoumání a práci s GTN zamítnuta. Především ze třech důvodů. GTN neumožňuje export Protokolu obsluhy do externího souboru; za druhé jsou vlastní historická data uložena ve zvláštním binárním souboru, který je prakticky bez použití jednotlivých datových záznamů (typu RECORD) nepoužitelná.



Obr. 1 – Použití filtrů v GTN

A třetí nepříjemnou vlastností, jež prakticky vyplývá z použitého modelu sběru dat postaveném na zabezpečovacím zařízení, je nepravidelné párování uložených informací. Uvedenou problematiku jak filtrování, tak párovosti naznačují Obr. 1 a Obr. 2.

Obr. 2 – Nepárovost informací v GTN

Události v obrázku Obr. 2 naznačené prázdnou (bílou) šipkou chybí v záznamech z protokolu obsluhy. Pokud by byly obsaženy, byla by zachována párovost a šla by snadno provést automatická analýza těchto dat.

2.4 Reálná možnost získávání dat

Jedinou reálnou možností, jak získat potřebná data pro řešení dané problematiky bylo porovnání plánu obsazení kolejí železniční stanice Praha hl.n. s filtrovanými daty Protokolu obsluhy z GTN ručně.

Nejprve bylo nutné z plánu obsazení dopravních kolejí po 2. změně GVD pro období od

28.5.2006 žst. Praha hl.n. konvertovat všechny osobní vlaky do tabelární formy s potřebnými atributy.

Struktura atributů z reálného obsazení dopravních kolejí pro jeden vlak:

- Číslo vlaku
- Příjezd
- Odjez
- Kolej

Celkem bylo za jeden den (od 0:00 do 24:00) tímto způsobem vyselektováno 178 vlaků. Což je suma všech osobních vlaků které přijíždějí a zastaví v ŽST. Praha hl.n.

Následně byly jednotlivé dny z GTN porovnávány s tímto seznam, přičemž hlavním kritériem pro výběr daného spoje byl jeho příjezd na jinou kolej, než která mu byla přiřazena plánem obsazení. Výsledky této analýzy jsou k dispozici v přílohách B a C této práce.

Struktura atributů z komparační analýzy dat GTN a GVD pro jeden vlak:

- Datum
- Číslo vlaku
- Předzvěst (kdy se vlak začne evidovat v modulu Protokolu obsluhy)
- Příjezd
- Odjezd
- Kolej

Tímto způsobem bylo získáno celkem 439 spojů v období od 1.8.2006 – 10.8.2006. Tyto vlaky byly zabezpečovací technikou zaznamenány na jiné koleji, než jaká jim byla přiřazena příslušným plánem obsazení.

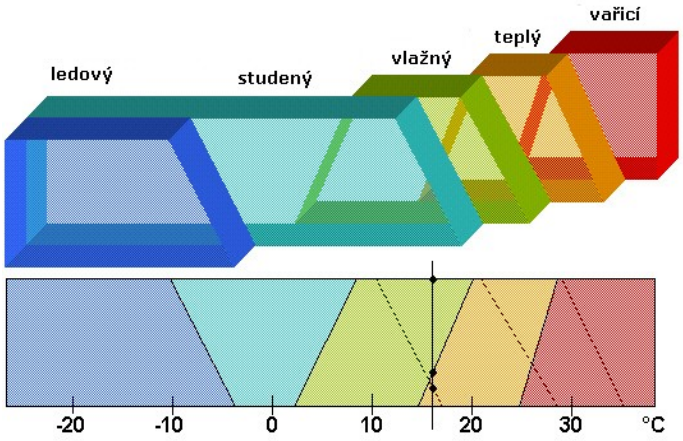
V úvodním textu práce se vyskytla informace o souboru ze skutečného obsazení kolejí, který obsahuje celý měsíc dat. Dozajista by bylo zajímavé zkoumat chování výsledných pravidel o této velikosti. Bereme-li ale v potaz kapitolu o nedostatcích při práci s Protokolem obsluhy, trvala by analýza a rozklíčování celého souboru minimálně třikrát déle. Získávání vstupního souboru dat pro jeden den trvalo přibližně 120minut. Bylo rozhodnuto, že k další práci si postačíme se vzorkem vstupních dat o velikosti deseti dní.

3 Fuzzy model

3.1 Základní pojmy

Vybraná idea fuzzy množin aplikovaných na řešení daného problému je ve své podstatě velmi jednoduchá a přirozená. Není-li možné stanovit přesné hranice jevů (ostrost množin), je každé posuzované hodnotě přiřazena míra vyjadřující její místo a roli v dané třídě. Odtud lze přeložit pojem fuzzy jako mlhavý či nejistý. Tato míra nejistoty je nazývána *stupněm příslušnosti (membership function)*. Třída, v níž je každý prvek charakterizován stupněm příslušnosti v ní, označujeme jako *fuzzy množinu (fuzzy sets)*.

Fuzzy logika je tedy logika mlhavá, nejistá či neurčitá. S rozvojem nových disciplín ve druhé polovině 20. století mezi které se hrdě řadí například teorie systémů, robotika a umělá inteligence se začala čím dál více uplatňovat výpočetní technika. Rozvoj těchto oblastí nutil matematiky, aby se začali zabývat problémem, jak popsat nepřesné pojmy. V roce 1965 profesor kalifornské univerzity L. A. Zadech publikoval zajímavý článek, v němž se zabýval modifikovanou formou teorie množin. Použil matematický nástroj, jež umožňoval popis vágních a nepřesných pojmů. Použil takzvané fuzzy množiny.



Obr. 3 – Zjednodušený model prolínání fuzzy množin

Zdroj: Internetový časopis SAID, ročník 2007/2008

3.2 Funkce příslušnosti

Teorie množin vychází z představy množin, které jsou podmnožinami určité libovolné, ale pevně dané univerzální množiny (univerza) U. O každém prvku z univerza lze jednoznačně říci, zda do té či oné množiny patří. Klasická množina A může být definována výčtem všech prvků, které do ní patří, definicí vlastností, které určují příslušnost prvku k množině, nebo pomocí charakteristické funkce. **Charakteristická funkce** přiřazuje každému prvku x z univerza U hodnotu 1, pokud tento prvek náleží množině A a hodnotu 0, pokud nepatří do množiny A.

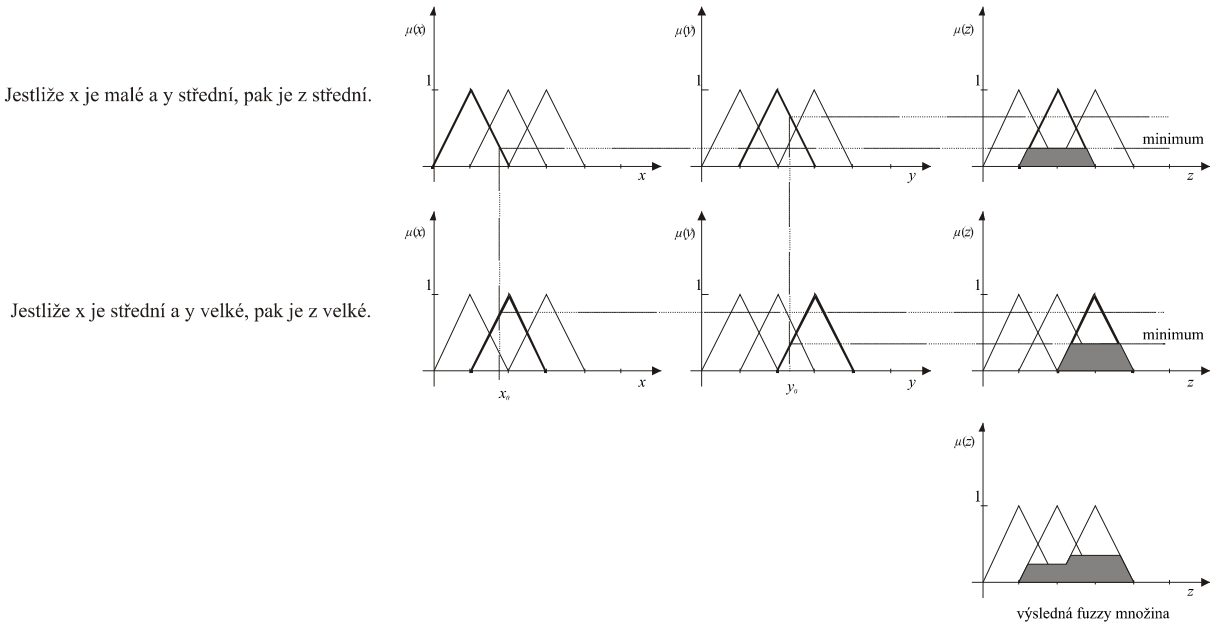
Pro snadné pochopení fuzzy množin, lze použít následující příklad. Dejme tomu, že prvky z množiny A jsou osobní vlaky jedoucí v pondělí z Prahy do Pardubic. Prvky množiny B chápejme jako vlaky jedoucí v úterý taktéž z Prahy do Pardubic. Otázka zní, do které množiny zařadíme osobní vlak, který vyjíždí v podělí ve 23:00 z Prahy a přijede v 01:00 do Pardubic?

Patrně každého napadne odpověď, že vlak patří z poloviny do skupiny A a z poloviny do skupiny B. Právě takovýto způsob myšlení je základem konstrukce fuzzy množin.

V klasické teorii množin může charakteristická funkce nabývat pouze hodnot 0 a 1, teorie fuzzy množin předpokládá, že charakteristická funkce může nabývat všech hodnot z intervalu $\langle 0;1\rangle$. Tato zobecněná charakteristická funkce množiny A se nazývá **funkce příslušnosti** a zpravidla se označuje symbolem $\boldsymbol{\mu_A}$. Hodnota $\mu_A(x)$ vyjadřuje, do jaké míry je prvek x z univerza U prvkem množiny A. Klasické množiny se pak dají vnímat jako zvláštní případ fuzzy množin. V teorii fuzzy množin se pro klasické množiny vžil termín „ostré množiny“.

Fuzzy množina A je tedy jednoznačně určena, pokud ke každému x z univerza přiřadíme hodnotu funkce příslušnosti $\mu_A(x)$, tzn.

$$A=\{(x;\mu_A(x));x\in U\} \tag{1}$$



Obr. 4 – Výsledky získané pomocí implikace Mamdani ze dvou pravidel

1. Jestliže ***x*** je malé a ***y*** střední, pak je ***z*** střední
2. Jestliže ***x*** je střední a ***y*** velké, pak je ***z*** velké

Zdroj: Mgr. Věra Záhorová, PhD., Konstrukce pravidel pro určení nástupištní koleje přijíždějícího vlaku na základě provozních dat

Fuzzy pravidla a fuzzy dedukce (*if–then*)

Fuzzy pravidlo,

$$if\ \ (x\ \ is\ \ A)\ \ \text{AND}\ \ (y\ \ is\ \ B)\ \ then\ \ (z\ \ is\ \ C), \tag{2}$$

představuje pravidlo, kde *x, y, z* jsou prvkem fuzzy množin. Výrazy, jež jsou uzavřeny v kulatých závorkách jsou označovány jako *fuzzy výroky*, jejichž pravdivostní hodnota leží v intervalu $\langle 0;1\rangle$. Fuzzy výrok se skládá ze dvou částí, a to z jazykové proměnné a hodnoty jazykové proměnné. Výroky mohou být spojeny logickými výrazy *and* či *or* a tím dohromady

tvoří *složený fuzzy výrok*. Výsledné pravidlo představující výsledek *fuzzy implikace* se nazývá *konsekvent*. Konsekvent je agregovaná hodnota z výsledku všech pravidel. Pomocí inferenčního mechanizmu získáváme ze vstupních fuzzy množin báze pravidel výstupní fuzzy množiny.

Pravidlo *if–then* (jestliže–pak) představuje fuzzy implikaci, která vyjadřuje související vztah mezi výroky. Nejčastěji používaným typem implikace ve fuzzy regulacích je implikace typu *Mamdani*. Implikaci typu Mamdani je použita i u řešení našeho problému.

Fuzzy inferenční systém (FIS) typu Mamdani lze definovat vztahem:

$$R_m^{(i)}: \qquad if\ (x_1 = A_1^{(i)}) \text{ and } (x_2 = A_2^{(i)}) \ \text{ and } \ldots \text{ and } \ (x_m = A_m^{(i)}) \text{ then } (y = C^{(i)}), \tag{3}$$

kde *i = 1, 2, ..., n* je číslo pravidla a *n* je počet pravidel. Takovýmto fuzzy modelem, se souborem *n* pravidel, lze aproximovat téměř libovolný nelineární systém. Základním charakteristickým znakem fuzzy modelu Mamdani je, že konsekvent každého pravidla tvoří fuzzy výrok.

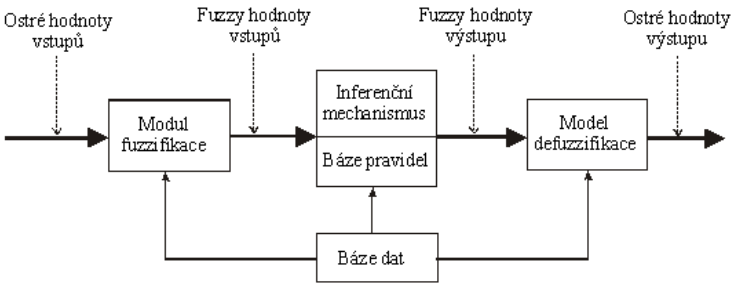
3.3 Jazykové proměnné

Cílem fuzzifikace je přiřadit ke vstupní jazykové proměnné (například ve významu teploty, taku, polohy, hladiny nebo času) pravdivostní hodnoty jedné nebo několika elementárních fuzzy proměnných. Jako fuzzy proměnné chápeme jazykové termy, zkráceně vstupní termy nebo jen termy odpovídající číselné hodnotě vstupní proměnné. Jazykové termy můžeme označovat jako jazykové či literární proměnné. Každý term je definován svou pravdivostí funkcí, která k hodnotám vstupní jazykové proměnné přiřazuje hodnoty pravdivosti tohoto termu (stupeň pravdivosti). V souladu s obvyklou praxí připouští norma pravdivostní funkci pouze ve formě spojité, lomené, po úsecích lineární funkce. Znázorněno graficky jde o dva nebo více spojitě navazujících přímkových úseků (typicky ve tvaru rampy, trojúhelníku, lichoběžníku, ale normy připouští i složitější lomené průběhy).

Zdroj: Fuzzy logika a norma IEC 61131

3.4 Fuzzy systém typu Mamdani

Fuzzy systém je systém, jehož proměnné (všechny nebo jen některé) nabývají hodnot nebo stavů, které nelze definovat reálnými čísly nebo ostrými množinami. Mnohdy se jedná o případy, kdy jsou jednotlivé stavy proměnných definovány slovně. Algoritmus fuzzy systému je schématicky znázorněn na Obr. 4.



Obr. 5 – Struktura fuzzy systému Mamdani, Zdroj [2]

Stěžejní součástí fuzzy systému je báze pravidel typu *if-then* (jestliže-pak), která popisují výstup (konsekvent) systému při určitých hodnotách vstupů (antecendentů). Pomocí inferenčního

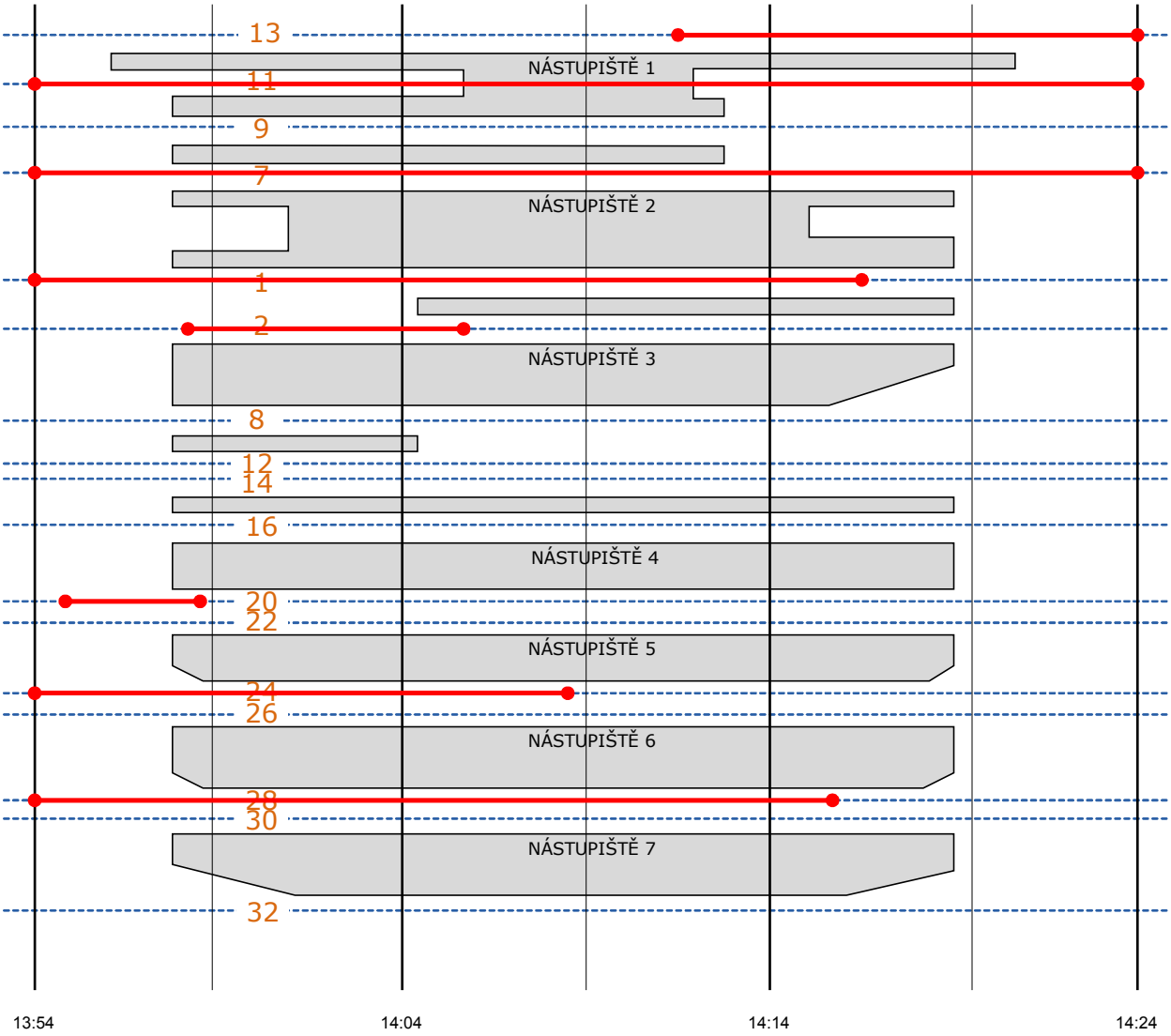
mechanismu získáváme ze vstupních fuzzy množin a báze pravidel výstupní fuzzy množiny. Jedním z často používaných inferenčních mechanismů je implikace Mamdani. Obr. 3 znázorňuje výsledek získaný pomocí implikace Mamdani ze dvou pravidel.

4 Základy teorie ohodnocování a zpracování získaných dat

4.1 Časový snímek obsazení kolejí

Ze získaných dat z obsazení dopravních kolejí žst. Praha hl.n. jsme díky vytvořenému programu schopni zrekonstruovat stav na nástupištích v libovolném časovém okamžiku. Spolu s distanční maticí (Tab. 1) dále můžeme doplnit tento pohled i o fyzické vzdálenosti obsazených kolejí. Pak je možné vytvořit věrný časový snímek obsazení kolejí.

Obr. 6 schématicky naznačuje časové okénko Praha, hlavní nádraží ve zjednodušeném nákresu.



Obr. 6 – Schéma obsazení kolejí

Časové okénko rekonstruujeme pro každý vlak, jež přijíždí na jinou kolej, než která mu byla přidělena plánem obsazení. Tedy pro každý vlak z námi vytvořeného vstupního souboru z reálného obsazení kolejí.

Časové okénko (time slide) začíná v okamžiku, kdy železniční stanice získá prvotní informaci o přijíždějícím vlaku. Tomuto údaji můžeme říkat ohlášení či předzvěst. Od okamžiku ohlášení vlaku ve stanici Praha hl.n. začínáme rekonstruovat stav na nástupištích. Délka tohoto okénka byla stanovena na 30 minut od ohlášení vlaku. Jedná se o hodnotu empirickou, jelikož po dobu sběru dat o vlacích vyplynul poznatek, že doba mezi ohlášením vlaku a jeho odjezdem ze stanice ve většině případů nepřekročí oněch 30 minut.

4.2 Distanční matice nástupišť

Pro další práci se získanými daty je třeba nějakým způsobem ohodnotit vzdálenosti jednotlivých nástupišť, které naznačují obtížnost (nepohodlí), jež musí cestující vynaložit, aby se dostal z jednoho nástupiště na jiné.

Díky získanému blokovému schéma železniční stanice Praha, hlavní nádraží, bylo snadné vytvořit tuto matici. Nebyl použit žádný algoritmus, který by přidělil jednotlivým nástupištím váhu, dle jejich parametrů (délka, počet laviček či zda-li se na nástupišti nachází stánek s občerstvením). Jediným kritériem byl počet podchodů (přechodů) kolejiště.

Tab. 1 – Distanční matice

M =

	1	2	7	8	9	11	13	16	20	22	24	26	28	30	32
1	0	1	0	1	1	1	1	2	2	3	3	4	4	5	5
2	1	0	1	0	2	2	2	1	1	2	2	3	3	4	4
7	0	1	0	1	1	1	1	2	2	3	3	4	4	5	5
8	1	0	1	0	2	2	2	1	1	2	2	3	3	4	4
9	1	2	1	2	0	0	0	3	3	4	4	5	5	6	6
11	1	2	1	2	0	0	0	3	3	4	4	5	5	6	6
13	1	2	1	2	0	0	0	3	3	4	4	5	5	6	6
16	2	1	2	1	3	3	3	0	0	1	1	2	2	3	3
20	2	1	2	1	3	3	3	0	0	1	1	2	2	3	3
22	3	2	3	2	4	4	4	1	1	0	0	1	1	2	2
24	3	2	3	2	4	4	4	1	1	0	0	1	1	2	2
26	4	3	3	2	4	4	4	1	1	0	0	1	1	2	2
28	4	3	4	3	5	5	5	2	2	1	1	0	0	1	1
30	5	4	5	4	6	6	6	3	3	2	2	1	1	0	0
32	5	4	5	4	6	6	6	3	3	2	2	1	1	0	0

Je-li nástupiště dosažitelné z jiného, aniž by cestující musel překonávat podchod či přecházet koleje, byla příslušná distance stanovena na 0. Přičemž za stavu, který byl v roce 2006 na hlavním nádraží byla maximální možná vzdálenost dvou nástupišť 6.

Jednotlivé hodnoty postupným odčítáním z náčrtku kolejiště byly zaneseny do matice, přičemž průsečík řádku a sloupce (matici lze samozřejmě číst i v opačném pořadí) s příslušným označením nástupiště určuje vzdálenost potřebnou k překonání dané vzdálenosti. V případě železniční stanice Praha hl.n. i v případech většiny ostatních stanic nenalezneme číslování kolejí v přesném iteračním kroku jako například 1, 2, 3, 4, … *n.* Po pozornějším prostudování distanční

matice lze zjisti, že některé koleje jsou vynechány. Geografické uspořádání kolejí a jejich označení není striktně iterační. Principy značení a uspořádání kolejí v železničních stanicích jsou stanovovány na základech legislativy a historických zvyklostí. V práci je počítáno pouze s takovými kolejemi, kterých se týká osobní přeprava a lze je nalézt v datech z GTN.

S tímto úskalím musí distanční matice samozřejmě počítat a její model musí postihovat všechny tyto odchylky.

Z časového okénka lze relativně snadno získat hodnoty do první části následující tabulky.

Tab. 2 – Obsazení kolejí a ohodnocení vhodnosti přiřazení koleje přijíždějícímu vlaku

Vlak číslo 676, předzvěst 13:54, kolej číslo: 22, plánovaná kolej 26, příjezd 13:54										
Vlak číslo	Nová kolej	Blízko plánované koleje	Volná za jak dlouho	Volná jak dlouho	Minut do pravidelného odjezdu přípoje	Blízko plánované koleje	Volná před ohlášením	Volná dostatečně dlouho	Přípoj	Celkem
No	Kolej	X_B	X_V	X_D	X_P	K_B	K_V	K_D	K_P	Suma
25904	13	5	ihned	16:00		0,29	1,00	0,53	0,00	1,82
9931	11	5	není volná	není volná		0,29	0,00	0,00	0,00	x
	9	5	ihned	dostatečně		0,29	1,00	1,00	0,00	2,29
29658	7	4	není volná	6:00		0,43	0,00	0,20	0,00	x
704	1	4	21:00	9:00	21:00	0,43	0,30	0,30	0,00	1,03
679	2	3	ihned	5:00		0,57	1,00	0,17	0,00	1,74
	8	3	ihned	dostatečně		0,57	1,00	1,00	0,00	2,57
	12	3	ihned	dostatečně		0,57	1,00	1,00	0,00	2,57
	14	3	ihned	dostatečně		0,57	1,00	1,00	0,00	2,57
	16	2	ihned	dostatečně		0,71	1,00	1,00	0,00	2,71
960	20	2	ihned	3:00		0,71	1,00	0,10	0,00	1,81
	22	1	ihned	dostatečně		0,86	1,00	1,00	0,00	2,86
251	24	1	16:00	14:00	16:00	0,86	0,47	0,47	0,00	1,79
	26	1	ihned	dostatečně		0,86	1,00	1,00	0,00	2,86
701	28	0	21:00	9:00	21:00	1,00	0,30	0,30	0,00	1,60
	30	1	ihned	dostatečně		0,86	1,00	1,00	0,00	2,86
	32	1	ihned	dostatečně		0,86	1,00	1,00	0,00	2,86

V tabulce se objeví všechny nástupišťní koleje z dané stanice. Staniční koleje obsazené vlakem vneseme do tabulky do správného řádku. Řádky jsou předpisem staničních kolejí. Sloupec „Vlak číslo“ je zde pouze ilustrativně pro kontrolu. Ve výsledných pravidlech ani dále pro nás nemá žádnou informační hodnotu.

Hodnota X_B je výsledkem průniku té koleje, na kterou měl vlak dle předpisu přijet s příslušnou hodnotou posuzované staniční koleje v distanční matici kolejiště. Určuje tedy počet nástupišť, která oddělují uvažovanou kolej od koleje přiřazené vlaku plánem obsazení kolejí.

X_V je ohodnocení doby, za kterou bude námi posuzovaná kolej v kolejišti volná od časového okamžiku předzvěst. V tomto sloupečku se vedle časového ohodnocení v minutách můžou objevit ještě lingvistické proměnné typu „ihned“ a „není volná“. „Ihned“ je ekvivalentně rovna 0, tedy kolej může být ihned obsazena. Popřípadě se zde může objevit „není volná“, což značí, že v době ohlášení vlaku nemůže vlak při svém příjezdu do stanice využít tuto kolej.

X_D symbolizuje hodnotu, říkající, jak dlouho bude posuzovaná kolej volná. Stejně jako u X_V se zde může objevit jak časová jednotka v minutách, tak se zde může objevit „dostatečně“, popřípadě „není volná“

Ohodnocení sloupečku X_P informuje o čase, za který přípojný vlak opustí nádraží. Nutno zdůraznit, že počátečním okamžikem je opět předzvěst námi vybraného vlaku. U hodnocení X_P se nemusí objevit žádná hodnota ani u jedné z kolejí, jelikož dle normy ČD D4 článek 96. – 108. můžeme zjednodušeně považovat za přípoj ten vlak, který odjíždí do 8. minut od příjezdu hlavního vlaku.

Pokud tedy hodnota X_P bude v rozmezí 0 – 8, budeme v následujících krocích s tímto vlakem pracovat jako s vlakem přípojným.

Data pro prvních šest sloupečků nám snadno poskytlo časové okénko. S ostatními údaji v následujících sloupcích provádíme operace na základě údajů z první části. A to tak, aby výsledné dílčí hodnoty byly v intervalu <0; 1>. Tyto hodnoty přestavují ohodnocení vhodnosti přiřazení jednotlivých kolejí zkoumaného vlaku. Hodnotu 0 přiřazujeme nejméně vhodné koleji a hodnotu 1 nejvhodnější.

Při výpočtu kriterií K_X byly využity následující vztahy:

Kritérium pro blízko plánované koleje K_B:

$$K_B = 1 - \frac{X_B}{max(x_b)} \quad , \tag{3}$$

přičemž $max(x_b)$ je maximální počet nástupišť mezi kolejemi. Pro uvažovanou stanici je maximální vzdálenost nástupišť 6. Použitý vzorec tedy má tvar:

$$K_B = 1 - \frac{X_B}{7} \quad . \tag{4}$$

Kritérium pro volná před ohlášením K_V:

$$K_V = \begin{cases} 0, pro\ X_V = \text{není volná} \\ 1 - \frac{X_V}{30} \in \langle 0;30 \rangle \\ 1, pro\ X_V = \text{ihned} \end{cases} \tag{5}$$

Konstanta 30 je stanovena s ohledem na délku trvání časového okénka.

Kritérium pro volná dostatečně dlouho K_D:

$$K_D = \begin{cases} 0, pro\ X_D = \text{není volná} \\ \frac{X_D}{30} \in \langle 0;30 \rangle \\ 1, pro\ X_D = \text{dostatečně} \end{cases} \tag{6}$$

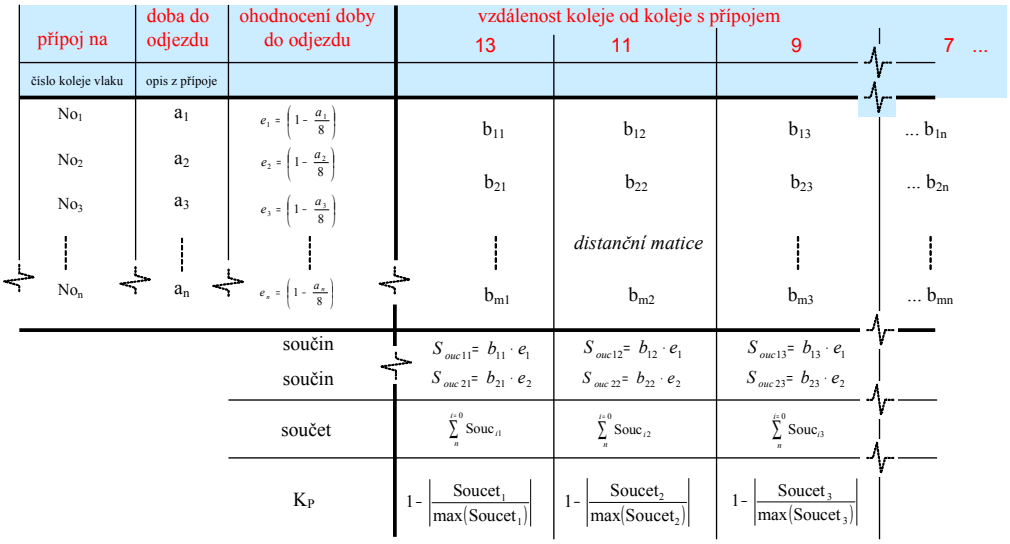
Kritérium pro přípoj K_P

Způsob pro výpočet hodnoty K_P je poněkud složitější. Jeho princip je naznačen na obrázku Obr. 7. Výsledná doba do odjezdu přípojů je transformována pomocí vztahu (7).

$$\text{ohodnocení doby do odjezdu} = 1 - \frac{doba\ do\ odjezdu}{8} \tag{7}$$

přičemž konstanta 8 je již zmiňovaný nejzazší možný čas odjezdu přípoje. Uvažujme jiný případ vlaku, než který je naznačen v tabulce Tab. 2. V této tabulce bohužel žádná hodnota přípoje X_P neodpovídá rozmezí <0; 8>. Uvažujme tedy vlak 9517 téhož dne. Tento vlak má dokonce 4 vhodné přípoje.

Princip ohodnocení doby do odjezdu je jasný ze vzorce (7). Všechny ostatní sloupce dané koleje jsou ohodnocením vzdáleností zvolené koleje vůči jednotlivým kolejím na nástupištích.



Obr. 7 – Postu při výpočtu tabulky Tab. 4

Součinem označujeme hodnotu, která je výsledkem násobení vzdálenosti koleje od koleje s přípojem (*b*) s ohodnocením doby do odjezdu (*e*). Součet je suma součinů pro stejné koleje s danými přípoji.

Kritérium K_P je normovaná hodnota v rozmezí <0; 1>, postihující váhu součtu kolejí. Cílem je dosáhnout takové hodnoty, kdy 0 znamená nejméně vhodná a 1 nejvíce vhodná. Převrácenou hodnotu získáváme z jednoho prostého důvodu. Kolej s nejvyšším součtem je pro nás nejméně vhodná, proto je snaha přiblížit její váhu co nejvíce 0.

Nakonec se hodnoty všech kritérií K_X v tabulce Tab. 2 sečtou. Koleje, které jsou v době ohlášení vlaku obsazeny, „volná za jak dlouho“ či „volná jak dlouho“ nabudou hodnot „není volná“, dále s nimi do celkového ohodnocení nepočítáme a nahradíme hodnotu symbolem 'x'.

Kolej, která získá nejvyšší ohodnocení je považována za nejvhodnější. Opět může nastat případ, kdy několik kolejí bude ohodnoceno stejně a to maximální hodnotou. V tomto případě jsou všechny tyto koleje stejně vhodné a ve výsledku budou figurovat všechny tyto hodnoty.

Vybrané koleje můžeme porovnat s kolejí, na kterou byl poslán vlak dispečerem a hodnotit, zda-li se dispečer rozhodl stejně.

Tab. 4 – Výpočet kritéria K_p

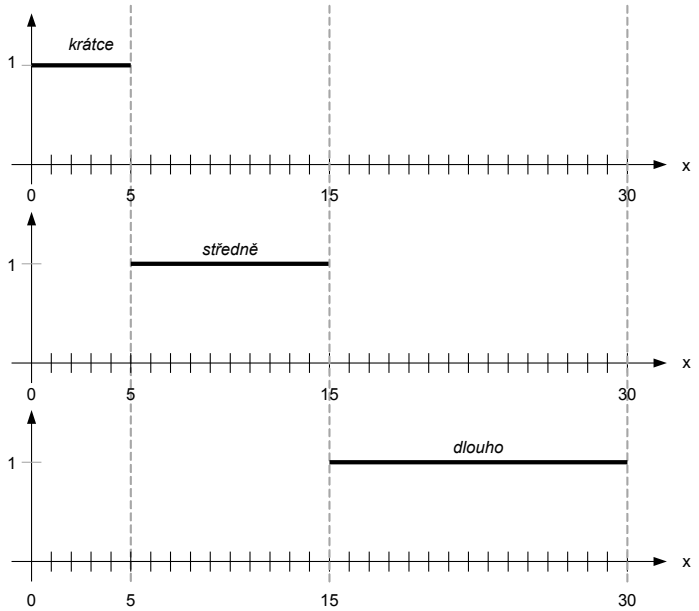
přípoj na	doba do odjezdu	ohodnocení doby do odjezdu	13	11	9	7	1	2	8	12	14	16	20	22	24	26	28	30	32
1	8:00	0,000	1	1	1	0	0	1	1	1	2	2	2	3	3	4	4	5	5
2	1:00	0,875	2	2	2	1	1	0	0	1	1	1	1	2	2	3	3	4	4
24	3:00	0,625	4	4	4	3	3	2	2	2	2	1	1	0	0	1	1	2	2
28	8:00	0,000	5	5	5	4	4	3	3	3	3	2	2	1	1	0	0	1	1
		součin	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		součin	1,75	1,75	1,75	0,88	0,88	1,50	0,00	0,88	0,88	0,88	0,88	1,75	1,75	2,63	2,63	3,50	3,50
		součin	2,50	2,50	2,50	1,88	1,88	1,25	1,25	1,25	1,25	0,63	0,63	0,00	0,00	0,63	0,63	1,25	1,25
		součin	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		součet	4,25	4,25	4,25	2,75	2,75	1,25	1,25	2,13	2,13	1,50	1,50	1,75	1,75	3,25	3,25	4,75	4,75
		K_P	0,11	0,11	0,11	0,42	0,42	0,74	0,74	0,55	0,55	0,68	0,68	0,63	0,63	0,32	0,32	1	1

V případě tabulky Tab. 4 jsou první čtyři řádky obsazeny údaji z přípojů.

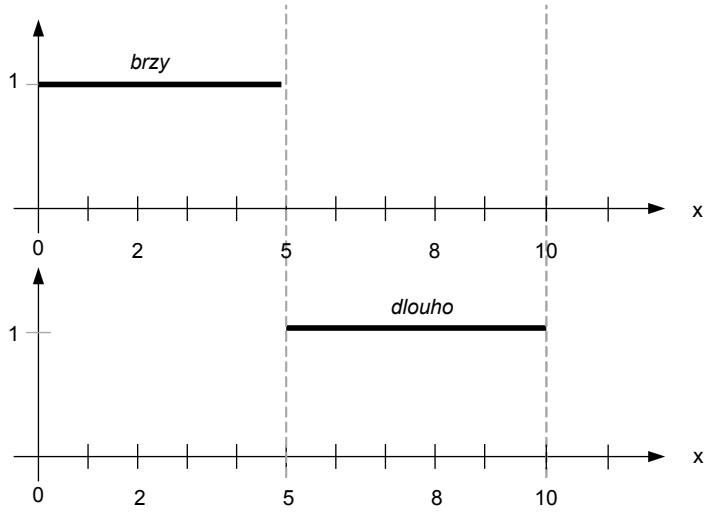
V krajní situaci může nastat stav, kdy nelze nalézt žádný vhodný přípoj. Stejně tomu bylo i u vlaku 676. V takovém případě nebude tabulka obsahovat řádky s přípoji a ani řádky se součiny. Řádek součet bude vyplněn nulami. Řádek K_P, který se promítne do tabulky, je vhodné vyplnit také nulami. V tabulce Tab. 2 by nebyl ovlivněn výsledek, ani pokud by místo nul byly dosazeny jedničky. Ovšem problém nastává u tabulky Tab. 6.

4.3 Vytváření pravidel z dat

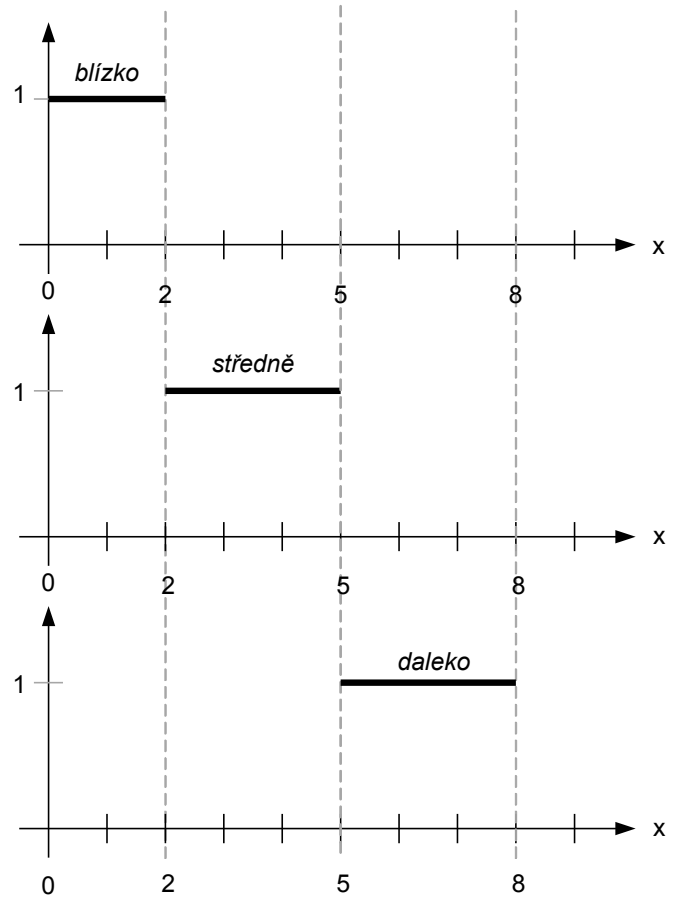
Jak již bylo řečeno, při tvorbě pravidel je zapotřebí stanovit tvar funkcí příslušnosti hodnot vstupů a výstupů k fuzzy množinám tak, aby byl pokryt celý rozsah možných hodnot. Funkce příslušnosti použité v našem případě jsou na obrázcích 8, 9, 10 a 11. Mezní hodnoty funkcí byly opět stanoveny intuitivně. Výsledné hodnoty po průchodu funkcemi příslušnosti jsou zpracovány do tabelární formy v tabulce Tab. 6. Pro správnou interpretaci funkcí bylo zapotřebí každou funkci rozdělit na dílčí subfunkce.



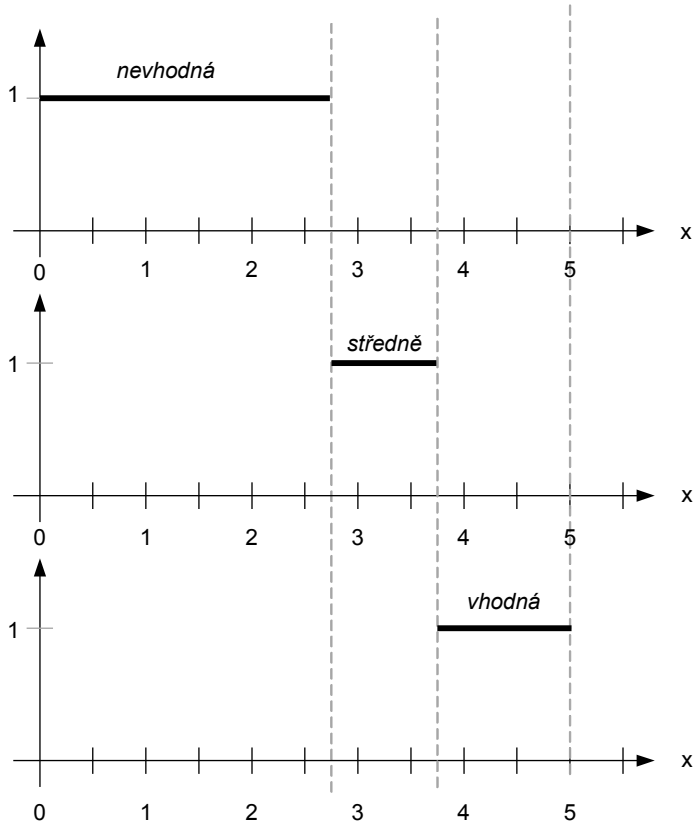
Obr. 8 – Funkce použitá pro tvorbu *if-then* pravidel proměnné „kolej volná před ohlášením vlaku ze sousední stanice“ a tvorbě pravidel „kolej volná po ohlášení vlaku ze sousední stanice“



Obr. 9 – Funkce použitá pro tvorbu *if-then* pravidel z proměnných „kolej blízko obvyklé“ a „přípoj blízko koleje“



Obr. 10 – Funkce použitá pro tvorbu *if-then* pravidel z proměnné „přípoj odjezd“



Obr. 11 – Funkce použitá pro tvorbu *if-then* pravidel z proměnné „celkové ohodnocení koleje zpožděnému vlaku“

Z tabulky Tab. 6 lze snadno zpětně získat algoritmy (funkce) použité při její konstrukci. Levá strana obsahuje námi již zjištěné hodnoty doplněné o dva přípoje s atributy „blízko koleje“ a „odjezd za“. Jedná se o jakýsi průnik s tabulkami Tab. 2 a Tab. 4.

4.4 Histogram přípojů

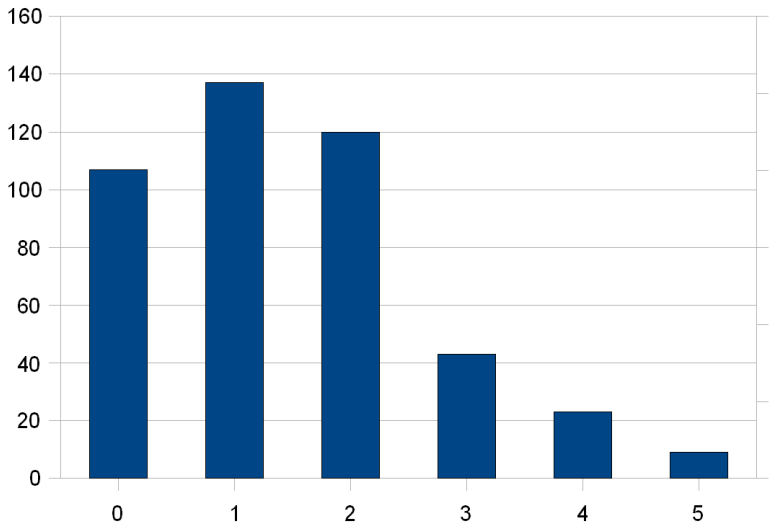
Z tabulky Tab. 6 vychází soubor všech možných jazykových proměnných z reálného zkoumání kolejiště se kterými budeme nadále pracovat. Ovšem otázkou zůstává, proč je počet relevantních přípojů stanoven právě na dva?

Zkoumáním získaných dat vyplynulo, že přijíždějící vlak může disponovat od 0 do 5 přípojů s různou četností v různých dnech. Abychom postihli co možná největší soubor pravidel a ohodnotili každou možnost, bylo by optimální zkoumat chování pravidel *if–then* ne u dvou, ale u pěti možných přípojů. Výsledná tabulka by oproti nynější podobě tabulky Tab. 6 byla na levé straně o 6 sloupců a na pravé straně také o 6 sloupců větší.

Z analýzy problému a dostupnosti všech potřebných dat vyplynulo, že takto rozsáhlý soubor jazykových proměnných postihne o necelých 10% více variant oproti dvou přípojům. Proto můžeme abstrahovat a považovat dva přípojné vlaky za optimální počet přípojů.

Tab. 5 – Četnosti osobních vlaků přijíždějící na jinou kolej, než která je jim přidělena z GVD nádraží.

	četnost					
den	0	1	2	3	4	5
10.08.2006	6	10	11	4	2	0
09.08.2006	8	11	10	1	4	1
08.08.2006	16	17	11	1	2	1
07.08.2006	10	15	12	3	4	1
06.08.2006	14	10	10	7	2	0
05.08.2006	8	11	13	3	1	1
04.08.2006	12	10	17	4	1	1
03.08.2006	12	20	13	6	0	2
02.08.2006	11	18	10	7	3	1
01.08.2006	10	15	13	7	4	1
∑	**107**	**137**	**120**	**43**	**23**	**9**



Obr. 12 – Histogram četností počtu přípojných vlaků

Díky nejistotě v počtu přípojů k námi zkoumanému osobnímu vlaku, mohou nastat v první části tabulky Tab. 6 čtyři případy práce s přípoji.

a) Existuje více přípojů než dva

V takovém případě, kdy množina přípojů je větší než dva, vybereme dva nejvhodnější přípoje. Tedy přípoje jejichž celkové ohodnocení dosáhlo jedné z nejvyšších hodnot. Vybereme tedy dva nejlépe ohodnocené přípoje.

b) Existují právě dva přípoje

V případě, že existují právě dva přípoje jsou oba vybrány a procházejí fuzzyfikací.

c) Existuje pouze jeden přípoj

Nastane-li situace, že množina vhodných přípojů obsahuje pouze jeden prvek (přípoj) je tento samozřejmě vybrán a druhý přípoj je ohodnocen jako kdyby byl na stejném nástupišti a odjížděl ihned. Tedy blízko plánované koleje ohodnotíme 0 a čas do odjezdu také 0 minutami. Takto hodnotíme přípoj z důvodu, aby výrazným způsobem neovlivnil celkové hodnocení.

d) Neexistuje ani jeden přípoj

Tato situace lze řešit obdobně jako situace s jedním přípojem, s tím rozdílem, že strategii ohodnocování aplikujeme i na první přípojný vlak.

Tab. 6 – Generování pravidel z dat

	data								pravidla							
nová kolej	blízko	volná před	volná dlouho	přípoj 1		přípoj 2		celkem	blízko	volná před	volná dlouho	přípoj 1		přípoj 2		vhodnost
				blízko	odjezd za	blízko	odjezd za					blízko	za	blízko	za	
13	2	ihned	dostatečně	1	3	2	2	3,00	středně	dlouho	dlouho	blízko	brzy	středně	brzy	středně
11	2	ihned	3	1	3	2	2	2,10	středně	dlouho	krátce	blízko	brzy	středně	brzy	nevhodná
9	2	ihned	dostatečně	1	3	2	2	3,07	středně	dlouho	dlouho	blízko	brzy	středně	brzy	středně
7	1	ihned	29	0	3	1	2	3,43	blízko	dlouho	dlouho	blízko	brzy	blízko	brzy	středně
1	1	8	22	0	3	1	2	2,98	blízko	středně	dlouho	blízko	brzy	blízko	brzy	středně
2	0	ihned	dostatečně	1	3	0	2	3,78	blízko	dlouho	dlouho	blízko	brzy	blízko	brzy	vhodná
8	0	3	27	1	3	0	2	3,61	blízko	krátce	dlouho	blízko	brzy	blízko	brzy	vhodná
12	1	není	dostatečně	1	3	0	2	x	blízko	není	není	blízko	brzy	blízko	brzy	nevhodná
14	1	7	23	2	3	1	2	3,10	blízko	středně	dlouho	středně	brzy	blízko	brzy	středně
16	1	ihned	dostatečně	2	3	1	2	3,55	blízko	dlouho	dlouho	středně	brzy	blízko	brzy	vhodná
20	1	ihned	dostatečně	2	3	1	2	3,55	blízko	dlouho	dlouho	středně	brzy	blízko	brzy	vhodná
22	2	ihned	dostatečně	3	3	2	2	3,20	středně	dlouho	dlouho	středně	brzy	středně	brzy	středně
24	2	ihned	dostatečně	3	3	2	2	3,20	středně	dlouho	dlouho	středně	brzy	středně	brzy	středně
26	3	11	24	4	3	3	2	2,25	středně	středně	dlouho	středně	brzy	středně	brzy	nehodná
28	3	ihned	dostatečně	4	3	3	2	2,82	středně	dlouho	dlouho	středně	brzy	středně	brzy	středně
30	4	ihned	dostatečně	5	3	4	2	2,43	středně	dlouho	dlouho	daleko	brzy	středně	brzy	nevhodná
32	4	ihned	dostatečně	5	3	4	2	2,43	středně	dlouho	dlouho	daleko	brzy	středně	brzy	nevhodná

Dosazené hodnoty do levé části tabulky Tab. 6 necháme modifikovat danými funkcemi příslušnosti a výsledné jazykové proměnné doplníme do její pravé části.

Získaný soubor pravidel může být značně rozsáhlý a je třeba jej redukovat. V prvé řadě se vyřadí opakující se pravidla. Ovšem četnost výskytu této kombinace musí zůstat dostupná pro další potřeby programu. Nastane-li situace, kdy jsou vygenerovaná pravidla se stejnými antecendenty, ale různými konsekventy, je za správné považováno pravidlo s vyšší četností výskytu. Další redukce počtu pravidel spočívá ve vyřazení pravidel málo obecných. Při tomto postupu samozřejmě dochází k částečné ztrátě informací.

4.5 Třídící algoritmus

Posledním a zároveň nejdůležitějším krokem celé algoritmizace problému výběru vhodné nástupištní koleje pro osobní vlak je třídění a shlukování báze fuzzy pravidel. Nejprve je třeba si uvědomit význam pravidel a jejich návaznosti. Tento význam vychází ze třech proměnných,

které rozlišujeme v souboru báze pravidel. Těmi jsou informace o prvním a druhém přípoji a proměnné vhodnost. Proměnnou vhodnost považujeme za výsledek předchozích údajů.

Proměnné o přípoji jedna mohou nabývat hodnot „blízko“, „daleko“ a „středně“ a hodnot o odjezdu „brzy“ a „dlouho“. To samé lze tvrdit o druhém přípojném vlaku. Vzniká tedy 36možností, jež mohou dohromady nabývat oba přípoje $((3 \cdot 2) \cdot (3 \cdot 2))$. Základní pracovní skupinou tedy bude třicetišestice. Po třicetišesticích se bude dále celý soubor pravidel zpracovávat.

Třicetišestice bude nadále dělena na tři čtveřice a tři osmice. Základní skupinu rozdělenou na tyto podskupiny lze dle významu ještě dělit a provádět nad nimi operace.

Prvním krokem zpracování souboru báze pravidel je setřídit soubor. Pravidlo pro třídění je vcelku jednoduché a lze si ho snadno představit jako operace s tabulkou v jakémkoliv tabulkovém procesoru. Všechna výsledná pravidla (představit si je lze jako řádky s osmi proměnnými) srovnáme dle prvního sloupce a to proměnné blízko. Ta nabývá hodnot „blízko“, „středně“ a „daleko“. Ve výsledném souboru dat bude ve skupině několik řádků s proměnou „blízko“, dále několik řádků s proměnnou „daleko“ a na konci souboru několik řádků s proměnou „středně“. De facto lze říci, že jsme základní soubor rozdělili do třech skupin. V dalším kroku vezmeme postupně každou z těchto skupin a srovnáme ji dle druhé proměnné „volná před“. Proměnná „volná před“ nabývá hodnot: „dlouho“, „krátce“, „není“ a „středně“. Po dvou uvedených krocích – za podmínek, že jsou zohledněna i ta pravidla, která v původním souboru vůbec nenastala a jejichž četnost je tedy 0 – je báze dat roztříděna do 12ti skupin. Naznačený postup je aplikován na další proměnné a to: „volná dlouho“ a oba přípoje. Dle výsledné proměnné „vhodnost“ se samozřejmě soubor netřídí.

V takto utříděné bázi dat jsou od každého záznamu tři varianty, jež se liší pouze ve výsledné proměnné „vhodnost“. Četnosti u jednotlivých záznamů v trojici mohou být různé. Četnosti vyjadřují kolikrát daný případ kombinací proměnných v kolejišti nastal. Řešení tohoto kroku v případě našeho algoritmu je naznačen v kapitole 5.3 výběr nejčetnějšího pravidla.

Výstupem kroku nazvaného výběr maxima z trojice musí být vždy jeden záznam. A to nejčetnější. Mohou nastat situace, kdy daný případ v kolejišti nenastal (četnost je tedy v každém řádku trojice 0), nebo všechny tři varianty mají stejnou četnost. U trojic, kde nenastávají takové extrémní situace, je výběr zcela jednoduchý. Vybereme ten řádek z trojice, jehož četnost je maximální a se zbylými dvěma řádky dále nepracuji. Vraťme se nyní k situacím, kdy každý řádek ve trojici má stejnou četnost. Výběr řádku je pak zcela libovolný. V případě aplikování tohoto algoritmu byl vybrán vždy řádek první, četnost nastavena na 0 a se zbylými dvěma se dále nepočítá. Tento postup platí jak pro nulovou četnost. V případě stejných četností nejsme schopni vybrat nejsilnější pravidlo a postupujeme stejně jako v případě nulových četností.

Ani takto setříděný a rozdělený soubor pravidel není prozatím logicky správný, aby byla postihnuta kombinace obou přípojů. Je třeba aplikovat další krok na třicetišestici, jež dokáže zaručit to, že logicky spolu související pravidla budou u sebe, aby mohl být aplikován krok dělení na čtveřice a osmice.

Bylo zapotřebí vytvořit jakousi násadu, která po aplikaci na základní matici zaručí správné párování spolu sovisejících pravidel. Násadu lze chápat jako třicetišestici která má jediný sloupeček a v něm je číselnou hodnotou vyjádřena pozice řádku. Pokud aplikujeme srovnání třicetišestice dle této proměnné, dostaneme setříděný soubor pravidel dle logických návazností. Princip tohoto postupu musel být aplikován ručně. Vybrané řádky z matice byly ohodnoceny čísly od 1 do 36 tak, jak spolu tyto řádky významově souvisejí. Tato násada je pak aplikována na každou 36tici a po srovnání dle této násady, aby čísla byly od 1 do 36 spolu tvoří logické skupiny.

Tab. 7 – Logická násada určená ke třídění

pozice_v_původní_matici	pozice_v_logické_matici
1	1
2	2
7	3
8	4
29	5
30	6
35	7
36	8
15	9
16	10
21	11
22	12
5	13
25	14
12	15
32	16
26	17
31	18
6	19
11	20
3	21
13	22
10	23
20	24
4	25
9	26
14	27
19	28
17	29
27	30
24	31
34	32
18	33
23	34
28	35
33	36

Po takto aplikovaných algoritmech je soubor roztříděný na již zmíněné 36tice a ty dále na čtveřice a osmice.

V prvním kroku, po setřídění dle „logické násady“, se zaměříme na první tři čtveřice. Prví čtveřice ponese v logické násadě označení: 1, 2, 3 a 4, druhá 5, 6, 7 a 8 a z třetí čtveřice 9, 10, 11 a 12. Dílčím cílem třídícího algoritmu je zabezpečit, aby vždy druhý a třetí řádek z čtveřice po jeho aplikaci disponoval stejnou proměnnou „vhodnost“. Jedná se o řádky 2, 3; 6, 7 a 10, 11 v tabulce Tab. 7 naznačeny šedým podkladem.

Proměnná „vhodnost“ může být u těchto řádků stejná, aniž by na ní byly provedeny jakékoliv úpravy. V těchto situacích si není třeba čtveřice všímat. Pokud tomu tak není, použijeme několik metod v různých situacích, aby tato symetričnost druhých a třetích řádků z čtveřic byla zajištěna.

Pokud je tedy druhý a třetí řádek ve čtveřici různý v proměnné „vhodnost“ a zároveň má první a čtvrtý řádek stejnou výslednou proměnnou „vhodnost“, zvolíme postup takový, aby druhá a třetí proměnná „vhodnost“ byla v souladu s první a čtvrtou. Ve výkonné části algoritmu nesmíme opomenout nastavit četnost z měněných řádků (druhého či třetího) na nulovou. Nulová hodnota četností u řádku, který již prošel zpracováním nám říká, že bude významově nepodstatný, avšak podstatný pro úplnost čtveřice.

Opačně může nastat situace, kdy je první a čtvrtý řádek různý ve své výsledné osmé proměnné „vhodnost“. V takovémto případě je postup následující. Výběrem maxima z četností je určen druhý či třetí řádek. Ten, který má vyšší hodnotu četnost. V souladu s vybraným četnějším řádkem je změněna „vhodnost“ u řádku s nižší četností a četnost takového řádku nastavena na 0.

V extrémní situaci, kdy nelze použít ani jeden z nabízených algoritmů, tedy 1. a 4. řádek mají různou výstupní proměnnou a druhý a třetí řádek mají stejné četnosti, si lze vybrat. Zda-li zvolíme druhý řádek a četnost třetího nastavím na 0, či zda-li zvolíme třetí řádek a četnost druhého bude nastavena na nulu.

Tento algoritmus je aplikován vždy na první tři čtveřice z třicetišestice v celém souboru dat.

Nyní jsou na řadě zbylé tři osmice. Osmice je z logických důvodů dále členěna na dvě dvojice a jednu čtveřici. V následující kroku uvažujme první dvojici. Stejný postup je aplikován i na druhou dvojici.

Filozofie problému je stále stejná. Ve dvojici zajistit, aby výsledné hodnoty osmé proměnné „vhodnost“ byly v souladu. Opět nastává řada situací a také řada řešení. Když první (resp. druhá) dvojice z osmice nemají stejnou proměnnou „vhodnost“ zaměříme se již intuitivně na četnosti. V situaci, kdy jsou četnosti stejné, lze provést výběr náhodně. Vybereme prví „vhodnost“ z dvojice a dosadíme ji do druhého řádku a četnost onoho druhého řádku nastavíme na 0. A nebo lze výběr provést opačně.

Tab. 8 – Předpis pro převod stejné 8. mé proměnné ve skupině

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost	
0	středně	středně	středně	blízko	brzy	blízko	brzy	**vhodná**	
1	středně	středně	středně	blízko	brzy	blízko	dlouho	**nevhodná**	Zaručit stejnou 8.mou proměnnou
1	středně	středně	středně	blízko	dlouho	blízko	brzy	**středně**	
0	středně	středně	středně	blízko	dlouho	blízko	dlouho	**vhodná**	
0	středně	středně	středně	středně	brzy	středně	brzy	**vhodná**	
1	středně	středně	středně	středně	brzy	středně	dlouho	**středně**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	středně	dlouho	středně	brzy	**vhodná**	
0	středně	středně	středně	středně	dlouho	středně	dlouho	**vhodná**	
0	středně	středně	středně	daleko	brzy	daleko	brzy	**vhodná**	
0	středně	středně	středně	daleko	brzy	daleko	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	daleko	dlouho	daleko	brzy	**vhodná**	
0	středně	středně	středně	daleko	dlouho	daleko	dlouho	**vhodná**	
0	středně	středně	středně	blízko	brzy	středně	brzy	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	středně	brzy	blízko	brzy	**vhodná**	
0	středně	středně	středně	blízko	dlouho	středně	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	středně	dlouho	blízko	dlouho	**vhodná**	
0	středně	středně	středně	středně	brzy	blízko	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
1	středně	středně	středně	středně	dlouho	blízko	brzy	**středně**	
0	středně	středně	středně	blízko	brzy	středně	dlouho	**vhodná**	
1	středně	středně	středně	blízko	dlouho	středně	brzy	**nevhodná**	
0	středně	středně	středně	blízko	brzy	daleko	brzy	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	daleko	brzy	blízko	brzy	**vhodná**	
0	středně	středně	středně	blízko	dlouho	daleko	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	daleko	dlouho	blízko	dlouho	**vhodná**	
0	středně	středně	středně	blízko	brzy	daleko	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	blízko	dlouho	daleko	brzy	**vhodná**	
0	středně	středně	středně	daleko	brzy	blízko	dlouho	**vhodná**	
0	středně	středně	středně	daleko	dlouho	blízko	brzy	**vhodná**	
0	středně	středně	středně	daleko	brzy	středně	brzy	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	středně	brzy	daleko	brzy	**vhodná**	
0	středně	středně	středně	daleko	dlouho	středně	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	středně	dlouho	daleko	dlouho	**vhodná**	
0	středně	středně	středně	daleko	brzy	středně	dlouho	**vhodná**	Zaručit stejnou 8.mou proměnnou
0	středně	středně	středně	daleko	dlouho	středně	brzy	**vhodná**	
0	středně	středně	středně	středně	brzy	daleko	dlouho	**vhodná**	
0	středně	středně	středně	středně	dlouho	daleko	brzy	**vhodná**	

V situacích, kdy jsou četnosti různé, selektivním výběrem bude zvolena četnější varianta – též lze nazvat jako varianta, která v kolejišti nastane častěji. Varianta menší četnosti bude uvedena do souladu s četnější variantou. Opět nesmíme opomenout na změnu ohodnocení četnosti méně četné varianty na 0. Tento postup je aplikován na první dvě dvojice z každé osmice v 36tici z celého souboru dat.

V dalším výběru se budeme zajímat o zbylou čtveřici z osmice. Tento krok musí striktně nastat až po zpracování prvních dvou dvojic, jelikož je zapotřebí, aby předešlé hodnoty byly již nově ohodnoceny. Na tyto nové hodnoty se budeme odvolávat a pracovat s nimi.

Zajímat nás bude pouze ta čtveřice, u které jsou různé hodnoty výstupní proměnné „vhodnost“. Ta, která má všechny výstupní proměnné stejné bude zpracována v následujících krocích. U čtveřic, které nejsou v souladu a je třeba s nimi pracovat mohou nastat následující případy.

Nejprve je vhodné se podívat, zda-li první a druhá dvojice z osmice má stejnou výstupní proměnnou vhodnost. Pokud tomu tak je, prioritou se stává uvedení výstupní proměnné z čtveřice do souladu s dvojicemi. V situacích, když některá výstupní proměnná z čtveřice již v souladu je, přeskočíme ji. V opačném případě její 8. proměnnou uvedeme v soulad s dvojicemi a její četnost nastavíme na 0. A to ze stejného důvodu, který již byl uveden v předchozím textu.

Pokud nastane situace, kdy první dvě dvojice nejsou ve svých výstupních proměnných stejné, zaměříme se sumu četností těchto dvojic. Je-li četnější první dvojice, uvedeme v soulad čtveřici

s výstupní proměnnou „vhodnost“ první dvojice a naopak. Při porovnání výstupních proměnných čtveřic kopírujeme postupu popsaný v předcházejícím odstavci.

A konečně jako poslední stav může nastat situace, kdy výstupní proměnné dvojic nejsou stejné a suma jejich četností je stejná. V takovýchto extrémních případech je postup zcela na řešiteli, zda-li si vybere výstupní proměnnou z prví či druhé dvojice. Doporučením by mohla být poučka, aby rozhodnutí výběru ovlivnilo co nejméně řádků. Ovšem je to pouze doporučení vycházející ze zkoumání daného problému. Výkonný postup je opět stejný jako u předchozích případů.

Po průchodu všemi naznačenými kroky se dostáváme k základům shlukovacího algoritmu. Pokud toto všechno byly přípravné kroky, jež je nutno projít, aby shlukování bylo co nejefektivnější, další postup je cílem našeho snažení.

Pro hladký průběh shlukování je třeba opět zůstat u základního členění třicetišestice a to na tři čtveřice a tři osmice. Neboť na nich bude aplikováno shlukování. Je to nutnou podmínkou pro zachování logicky souvisejících pravidel. Další členění (na jednice, dvojice a čtveřice) již nebereme v potaz.

Tab. 9 – Princip hodnocení dvojic a čtveřic, aby výsledné proměnné byly v souladu

četnost	…	vhodnost		četnost	…	vhodnost	
0	…	vhodná		0	…	vhodná	
1	…	nevhodná		*1*	…	*nevhodná*	1. a 4. řádek mají stejné proměnné, vybraná hodnota musí být s nimi v souladu
1	…	středně		*0*	…	*nevhodná*	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
1	…	středně		*0*	…	*vhodná*	1. a 4. řádek mají stejné proměnné, vybraná hodnota musí být s nimi v souladu
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
2	…	vhodná		2	…	vhodná	
1	…	vhodná		*0*	…	*nevhodná*	1. a 4. řádek se liší v 8.mé proměnné, vybraná hodnota z 2. či 3. řádku, která má větší četnost
5	…	nevhodná		5	…	nevhodná	
2	…	středně		2	…	středně	
0	…	vhodná		0	…	vhodná	
0	…	vhodná	►►►	0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
1	…	středně		*0*	…	*vhodná*	Řádky 1,2 a 3,4 obsahují stejnou 8.mou proměnnou, vybraná hodnota je s nimi v souladu
0	…	vhodná		0	…	vhodná	
1	…	nevhodná		*0*	…	*vhodná*	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	V souladu, proto s nimi v prvních částech algoritmu nepracujeme
0	…	vhodná		0	…	vhodná	
0	…	vhodná		0	…	vhodná	
5	…	středně		5	…	středně	
6	…	středně		6	…	středně	
6	…	vhodná		6	…	vhodná	
5	…	vhodná		5	…	vhodná	
1	…	vhodná		*0*	…	*středně*	
2	…	středně		2	…	středně	Řádky 1,2 a 3,4 obsahují různé 8.mé proměnné, četnosti jsou stejné, výběr je náhodný
3	…	nevhodná		*0*	…	*středně*	
4	…	vhodná		*0*	…	*středně*	

Za podmínek, kdy ve skupině – nyní zkoumáme první tři čtveřice – nastane situace, kdy výstupní proměnná „vhodnost“ je u všech čtyřech řádků totožná, je vhodné na této čtveřici provést shluknutí. Nově tedy vzejde z čtveřice pouze jeden řádek a to první. Neznamená to ale, že ostatní řádky zahodíme, jako v kroku výběru nejčetnější z trojice. Nyní je důležité zachovávat strukturu původní třicetišestice. Shlukneme tedy čtveřici do jednoho pravidla, ale řádky po ostatních zůstanou. Na jejich místech budou jakési „díry“. U shlukování je třeba také zachovat četnosti všech řádků, tomu lze vyhovět tak, že řádek, jež zbude, bude nově ohodnocena četnost a to sumou četností všech čtyř řádků. To ale není jediná úprava. Díky tomuto shlukování je jasné, že proměnná u přípoje jedna a u přípoje dva - odjezd (ta, jejíž logika rovná celý soubor dat) je nepodstatná a můžeme jí označit za nedůležitou. Dohodou bylo, jako symbol nedůležitosti,

stanoveno znaménko 'x', jež je přiřazeno namísto lingvistické proměnné za proměnou p1 odjezd a p2 odjezd. Tento postup aplikujeme na všechny třicetišestice v souboru.

V dalším postupu se zaměříme na shlukování třech osmic z původní třiceti šestice. Filozofie shlukování je o něco odlišná, než u čtveřic. Osmice je v podstatě složena ze dvou čtveřic a proto i výsledek shlukování musí být dvojnásobný. Nevzejde nám tedy jedno pravidlo, ale pravidla dvě. Co se týče označování odjezdu přípojných vlaků zůstaneme u dohodnuté konvence. To samé platí i o součtu četností, i když trošku upravenou variantu pro případ dvou výsledných pravidel.

Tab. 10 – Shlukování

četnost	…	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	…	vhodnost
8	…	blízko	brzy	blízko	brzy	…	vhodná
1	…	blízko	brzy	blízko	dlouho	…	*nevhodná*
0	…	blízko	dlouho	blízko	brzy	…	*nevhodná*
8	…	blízko	dlouho	blízko	dlouho	…	vhodná
3	…	středně	brzy	středně	brzy	…	vhodná
0	…	středně	brzy	středně	dlouho	…	*vhodná*
3	…	středně	dlouho	středně	brzy	…	vhodná
3	…	středně	dlouho	středně	dlouho	…	vhodná
2	…	daleko	brzy	daleko	brzy	…	vhodná
0	…	daleko	brzy	daleko	dlouho	…	*nevhodná*
5	…	daleko	dlouho	daleko	brzy	…	nevhodná
2	…	daleko	dlouho	daleko	dlouho	…	středně
5	…	blízko	brzy	středně	brzy	…	vhodná
6	…	středně	brzy	blízko	brzy	…	vhodná
4	…	blízko	dlouho	středně	dlouho	…	vhodná
7	…	středně	dlouho	blízko	dlouho	…	vhodná
8	…	středně	brzy	blízko	dlouho	…	vhodná
0	…	středně	dlouho	blízko	brzy	…	*vhodná*
9	…	blízko	brzy	středně	dlouho	…	vhodná
0	…	blízko	dlouho	středně	brzy	…	*vhodná*
1	…	blízko	brzy	daleko	brzy	…	vhodná
2	…	daleko	brzy	blízko	brzy	…	vhodná
3	…	blízko	dlouho	daleko	dlouho	…	vhodná
4	…	daleko	dlouho	blízko	dlouho	…	vhodná
5	…	blízko	brzy	daleko	dlouho	…	vhodná
6	…	blízko	dlouho	daleko	brzy	…	vhodná
7	…	daleko	brzy	blízko	dlouho	…	vhodná
8	…	daleko	dlouho	blízko	brzy	…	vhodná
5	…	daleko	brzy	středně	brzy	…	středně
6	…	středně	brzy	daleko	brzy	…	středně
6	…	daleko	dlouho	středně	dlouho	…	vhodná
5	…	středně	dlouho	daleko	dlouho	…	vhodná
0	…	daleko	brzy	středně	dlouho	…	*středně*
2	…	daleko	dlouho	středně	brzy	…	středně
0	…	středně	brzy	daleko	dlouho	…	*středně*
0	…	středně	dlouho	daleko	brzy	…	*středně*

▸▸▸

četnost	…	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	…	vhodnost
8	…	blízko	brzy	blízko	brzy	…	vhodná
1	…	blízko	brzy	blízko	dlouho	…	*nevhodná*
0	…	blízko	dlouho	blízko	brzy	…	*nevhodná*
8	…	blízko	dlouho	blízko	dlouho	…	vhodná
9	…	středně	x	středně	x	…	vhodná
						…	
						…	
2	…	daleko	brzy	daleko	brzy	…	vhodná
0	…	daleko	brzy	daleko	dlouho	…	*nevhodná*
5	…	daleko	dlouho	daleko	brzy	…	nevhodná
2	…	daleko	dlouho	daleko	dlouho	…	středně
5+4+8+0	…	blízko	x	středně	x	…	vhodná
6+7+9+0	…	středně	x	blízko	x	…	vhodná
						…	
						…	
						…	
						…	
1+3+5+6	…	blízko	brzy	daleko	brzy	…	vhodná
2+4+7+8	…	daleko	brzy	blízko	brzy	…	vhodná
	…					…	
	…					…	
	…					…	
	…					…	
	…					…	
5	…	daleko	brzy	středně	brzy	…	středně
6	…	středně	brzy	daleko	brzy	…	středně
6	…	daleko	dlouho	středně	dlouho	…	vhodná
5	…	středně	dlouho	daleko	dlouho	…	vhodná
0	…	daleko	brzy	středně	dlouho	…	*středně*
2	…	daleko	dlouho	středně	brzy	…	středně
0	…	středně	brzy	daleko	dlouho	…	*středně*
0	…	středně	dlouho	daleko	brzy	…	*středně*

Základní předpoklad pro shlukování je stejný. Pokud se celá osmice neliší ve výsledné proměnné „vhodnost“ je připravena ke shlukování. V podstatě rozdělíme osmici na dvě spolu logicky související čtveřice. A to 1., 3., 5. a 6. řádek a 2., 4., 7. a 8. řádek. Zjednodušeně lze říci, že ponecháme první a druhý řádek, které jsou jakýmisi představiteli oněch dvou čtveřic. V naznačených řádcích (1, 3, 5, 6) provedeme součet četností a výslednou četnost uvedeme jako četnost do prvního řádku. Obdobně provedeme operaci i u druhé skupiny, kde sumu četností vkládáme do druhého řádku. Všechny ostatní řádky v osmici, krom prvního a druhého, považujeme za „díry“. Provedeme nahrazení nepodstatných parametrů z prvního a druhého přípoje symbolem 'x'. Tímto postupem jsme shlukli první osmici. V třicetišestici tento postup aplikujeme ještě dvakrát a stejně tak v dalších třicetišesticích celého souboru báze pravidel.

Na první pohled by se mohlo zdát, že je soubor po průchodu všemi zmíněnými operacemi již setříděn a můžeme jej považovat za výsledek celého snažení. Ale opak je pravdou. Část souboru je opravdu setříděna a shluknuta správně, ale pokud bereme v potaz to, co prováděl první krok v extrémní situaci při výběru řádku s maximální hodnotou četnost, musí být jasné, že u řádků s nulovou četností je vybraná hodnota „vhodnost“ pouze náhodná. Zjednodušeně řečeno, když byl prováděn výběr nejčetnějšího řádku z trojice a všechny řádky v trojici měli stejnou četnost, byl vybrán libovolný řádek – samozřejmě i „vhodnost“ – a jeho četnost ohodnocena nulou. A tuto náhodnost musíme nyní eliminovat ve výsledných hodnotách. V hodnotách jak u čtveřic, tak u osmic.

Projdeme tedy znovu celý soubor po skupině třicetišestic a rozdělíme na shlukovací skupiny 3× čtveřice a 3× osmice. Hledáme takovou čtveřici či osmici, ve které doposud nebylo

provedeno shlukování. Pokud existuje, je třeba zjistit, zda-li v takovéto skupině nastává situace, kde řádky s nulovou četností brání shluknutí skupiny. Algoritmus takového kroku je následující.

Pokud ve čtveřici či osmici mají řádky s nenulovou četností stejnou výslednou proměnnou „vhodnost“ a ostatní řádky mají četnost „0“, považujeme řádky s nulovou četností za řádky z náhodného výběru a změníme jejich výslednou proměnnou „vhodnost“ na takovou, která je v souladu s výslednými proměnnými z řádků s nenulovou četností. Dále postupujeme různě u čtveřic a osmic a aplikujeme shlukovací algoritmus.

Výpis báze pravidel ze souboru, na kterém byly aplikovány všechny předcházející kroky se opět řídí pravidelnou strukturou třicetišestic, který je dále dělitelný na tři čtveřice a tři osmice. Vypisují se všechna pravidla kromě „děr“, ale i s nimi je nutno počítat do 36tice. Výpis lze uspořádat dle výstupní proměnné „vhodnost“ popřípadě dle pozice řádku v třicetišestici. Takto shluknutý a vypsaný soubor báze pravidel můžeme považovat za výsledek shlukovacího algoritmu.

Dále lze s výpisem ještě pracovat a zpříjemnit jeho čtení či zvýšit informační hodnotu. Zajímavým ukazatel může být i tzv. „zohledněno pravidel“ jež nám v procentech ukazuje podíl četnosti pravidel, které byly na počátku třídění v třicetišestici a četnosti pravidel zbylých po průchodu shlukovacím algoritmem. Jsme tak snadno schopni určit procento ztracené informace.

4.6 Matlab

Při tvorbě programu generátoru fuzzy pravidel vyvstal zajímavý požadavek. Použít mezivýsledky z levé části tabulky Tab. 6 a převést je do souboru vhodného formátu, který bude srozumitelný pro program Matlab.

MATLAB je výkonný softwarový nástroj pro vědeckotechnické výpočty a návrh pokročilých algoritmů. Obsahuje širokou škálu matematických i vizualizačních funkcí. Jedním z jeho hlavních rysů je maticové zpracování, tj. matematické funkce se nepočítají v cyklu po jednotlivých hodnotách, ale jako celé matice hodnot najednou. To činí tento nástroj mnohem rychlejším oproti jiným matematickým softwarům. Dále jsou v Matlabu zahrnuty nástroje pro numerické výpočty a řešení obyčejných i diferenciálních rovnic. Matlab je program od společnosti The MathWorks.

Bylo nutné, aby výstupní soubor šlo snadno parametrizovat. Tuto parametrizaci lze chápat jako nastavení, které údaje z tabulky Tab. 6 chceme či nechceme zahrnout do konečné podoby souboru.

Matlab, konkrétně funkce FuzzyToolbox, nejsou nastaveny tak, aby pro ně byly srozumitelné námi nastavené lingvistické proměnné použité v levé části tabulky Tab. 6. Bylo třeba přidat podmínky, které takovouto proměnnou převedou na její ekvivalentní číselné ohodnocení. Dále byl výpis rozšířen o další parametry, jelikož k dalšímu zpracování bylo třeba více či méně informací, než které jsou obsaženy v již zmíněné tabulce. Nově přibyl sloupeček ohodnocující u aktuálního vlaku kolej, na který byla souprava – dle provozních dat – vyslána dispečerem vlakové dopravy. Formálně bylo určeno, že taková nástupištní kolej na níž přijela souprava bude ohodnocena 1 a ostatní 0.

Slovní proměnné u sloupečku „volná před“ byly pozměněny takto:

- ihned.....0
- není volná.....31
- jinak.....číslo

0, jelikož je kolej okamžitě volná a tedy doba čekání na takový stav je 0 minut. 31, jelikož doba, než nastane takováto situace je delší, než zkoumané časové okno (to je 30 minut) a proto tedy 31.

Slovní proměnné u sloupečku „volná dlouho“ byly pozměněny takto:

- dostatečně.....31
- není volná.....0
- jinak.....číslo

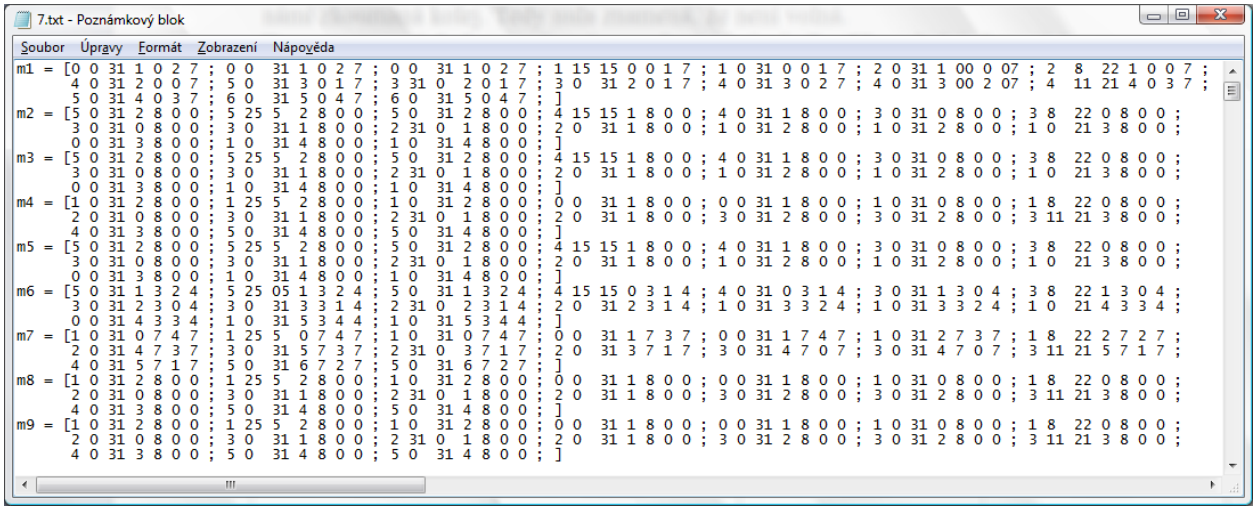
31 je podobně jako u předchozího případu delší, než námi zkoumané časové okénko a naznačuje, že kolej je volná déle, než námi zkoumaný úsek. 0 naznačuje čas, po který je volná námi zkoumaná kolej. Tedy nula znamená, že není volná.

Do výsledného souboru nezahrnujeme z levé části tabulky T3 poslední sloupeček a to proměnnou „celkem“.

Požadovaný formát souboru

Jelikož Matlab pracuje především s hodnotami uspořádaných do matic, musel být výsledný soubor také maticově uspořádám. Výsledný soubor byl tedy textového formátu a přizpůsoben syntaxi maticového zápisu, kterému Matlab rozumí. Syntaxe je naznačena ve vzorci (7) a výsledný soubor s daty je ilustrativně zobrazen na Obr. 13.

$$M_{\text{číslo matice}}(n) = [X_{b13}\ \text{volná před}_{13}\ \text{volná dlouho}_{13}\ p1_{\text{kolej}13}\ p1_{\text{min}13}\ p2_{\text{kolej}13}\ p2_{\text{min}13}\ od_{13};$$
$$X_{b11}\ \text{volná před}_{11}\ \text{volná dlouho}_{11}\ p1_{\text{kolej}11}\ p1_{\text{min}11}\ p2_{\text{kolej}11}\ p2_{\text{min}11}\ od_{11};$$
$$\ldots \quad (7)$$



Obr. 13 – Náhled obsahu souboru s maticemi pro Matlab

Celkem byly vytvořeny čtyři typy souborů, které se lišili pouze ve struktuře exportovaných dat. Všechny případy struktur jsou obsaženy v tabulce Tab. 11.

Tab. 11 – Výstupní charaktery souborů pro další zpracování v Matlabu

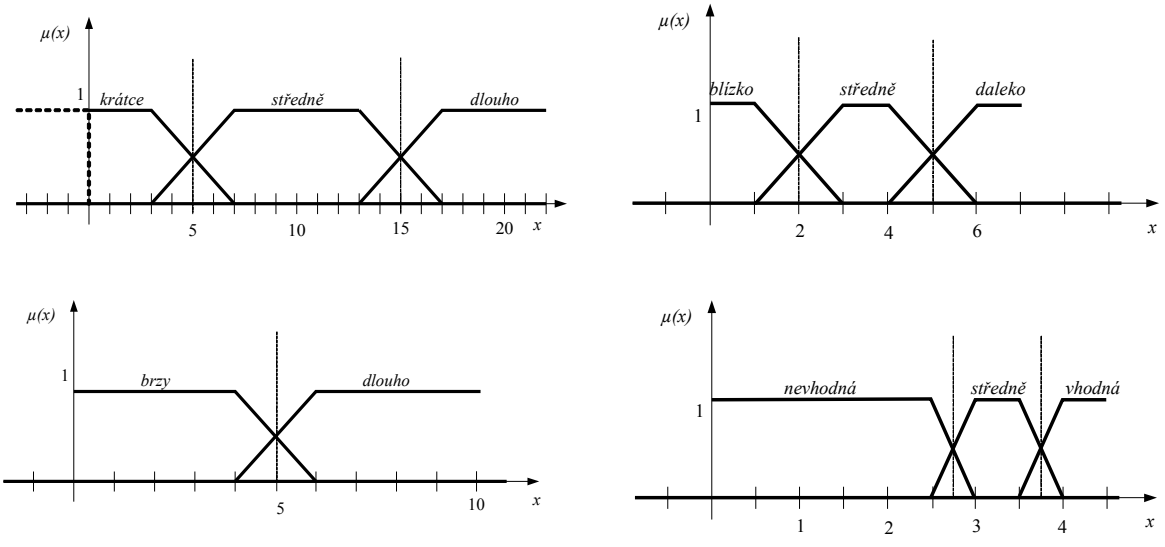
1. soubor	2. soubor	3. soubor	4. soubor
■ blízko ■ volná před ■ volná dlouho ■ p1 blízko ■ p1 min ■ p2 blízko ■ p2 min	■ blízko ■ volná před ■ volná dlouho ■ p1 blízko ■ p1 min ■ p2 blízko ■ p2 min ■ rozhodnutí dispečera	■ blízko ■ volná před ■ volná dlouho ■ p1 blízko ■ p2 blízko	■ blízko ■ volná před ■ volná dlouho ■ p1 blízko ■ p2 blízko ■ rozhodnutí dispečera

Konstrukce fuzzy regulátoru prostřednictvím Matlabu

Úkolem celé práce je generátor báze fuzzy pravidel. Tyto pravidla budou dále zpracovávána za pomocí Matlabu. V tomto prostředí existuje nadstavba jménem FuzzyToolbox, díky níž a výstupním pravidlům z mého programu se dá vytvořit Fuzzy regulátor. Fuzzy Logic Toolbox je určen pro návrh a diagnostiku systémů na bázi fuzzy logiky. FLT disponuje interaktivní grafickou nadstavbou, která dovoluje sledovat a přesně dolaďovat chování fuzzy systému. Editor funkcí příslušnosti a editor pravidel jsou základními součástmi práce se systémem. Díky grafickým metodám (point and click) usnadňuje uživateli přepisování rozsáhlých fuzzy pravidel ručně.

Dále se zpracovávají data za pomocí funkce evalfis. Ta má dva parametry. Proměnnou s načtenou bází fuzzy pravidel v maticovém formátu a vektor vstupních veličin, které budou vyhodnoceny. Jedná se tedy o regulátor. Výsledkem je sloupcový vektor s tolika řádky, kolik měla na vstupu matice báze pravidel.

Při tvorbě fuzzy regulátoru můžeme využít následujících funkcí příslušnosti.



Obr. 14 – Funkce příslušnosti použité při tvorbě fuzzy regulátoru

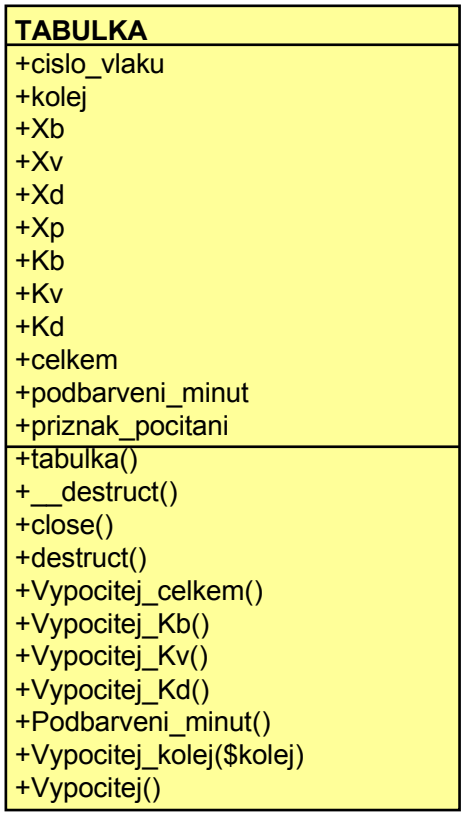
5 Program GBP Fuzzy

5.1 Popis tříd

V popisu tříd se omezíme pouze na třídy z první části zpracování dat. Celkem lze program pro určení náhradní nástupištní koleje rozdělit do třech základních algoritmů plus další algoritmy jež interpretují data a umožňují vstup a výstup. V první části programu je třeba zpracovat vstupní soubor dat. Vstupní data budou uložena do databáze respektive databází. Z databází se informace získávají na základě některých kritérii, respektive z databáze vybíráme právě ta data, která potřebujeme pro aktuální kolej či vlak.

Třída Tabulka

Prvním výsledkem průchodu dat algoritmy by měla být tabulka pro jednotlivé vlaky přijíždějící na jinou, než předepsanou kolej a jejich přípoje v daném časovém okénku. Pro tuto situaci je navržena třída Tabulka. Třída Tabulka je instancí právě jedné nástupištní koleje a celkový obraz nádraží nám tvoří pole těchto tabulek. Celkem je tedy pro jeden zpožděný vlak nutno vytvořit 17 instancí třídy Tabulka. Sluší se podotknout, že při konstrukci třídy neznáme všechny parametry, které je třeba dále dopočítat. K dopočítání potřebných informací do základní třídy Tabulka (především atributu „celkem“) je zapotřebí provést několik sub-výpočtů. Pro tyto dílčí výpočty je navržena třída Kp_kolej.



Obr. 15 – Třída Tabulka

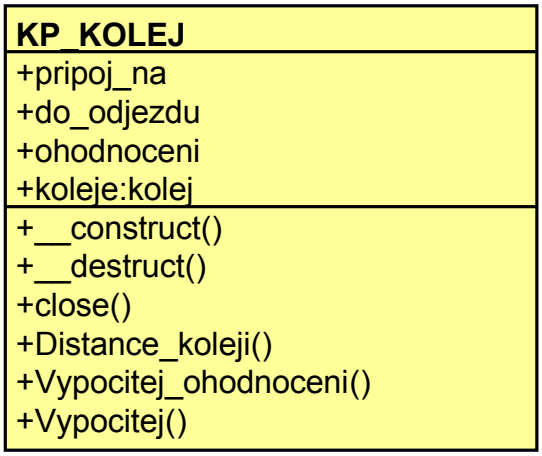
Vzhledem ke zvolenému programovacímu jazyku PHP je třeba vysvětlit některé odlišnosti od „klasických“ programovacích nástrojů, jakými jsou C++, C#, Java či dnes již méně rozšířené Delphi. Co se týče atributů, není snad nutné dodávat žádný komentář. Zcela vycházejí z řešeného

problému a jejich vysvětlení již bylo naznačeno v teoretické části této práce.

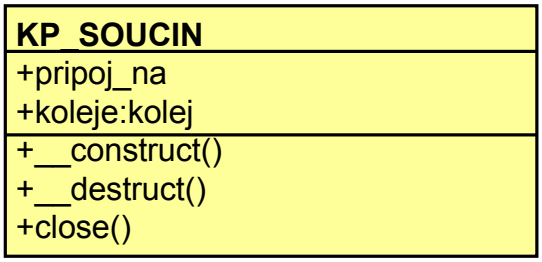
Zajímavější situace nastává při tvorbě konstruktorů a destruktorů tříd. Zde nastává několik drobných odlišností od již zmiňovaných programovacích jazyků. V případě námi použité programovací techniky je třeba destruktor zavolat ručně, jelikož si v paměti neponecháváme všechny vlaky, ale pouze aktuální s nímž pracujeme. Kdybychom tak nečinili, dalo by se vypozorovat, že v určitých specifických případech by mohla nastat nekonzistence dat a docházelo by ke ztrátě či zkreslení informace jež dané třídy nesou. Samozřejmě na konci práce by se destruktor zavolal sám, ale to by již docházelo ke zmiňovaným stavům.

Třída Kp_kolej, Kp_soucin a Kp_soucet

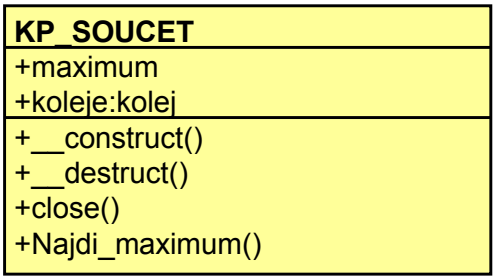
Kp_kolej je třída, jež nám umožní realizovat pomocné výpočty. Velkou výhodou použití objektu oproti statickému poli či podobné struktuře je fakt, že do zpracování údajů o přijíždějícím vlaku nejsme schopni určit počet přípojných vlaků. Ve statickém poli bychom zbytečně alokovali systémové prostředky pro 5 možných přípojů. Výsledky z třídy Kp_kolej jsou zapracovány do třídy Kp_součin. Třída Kp_soucin je naznačena na obrázku Obr. 17. Výsledky ze součinu jsou zpracovány ve třídě Kp_soucet a z této třídy se dále vrací do základní třídy Tabulka. U třídy Kp_soucet vzniká pouze jedna instance oproti třídě Kp_soucin, kde je počet instancí úměrný počtu vhodných přípojů daného vlaku.



Obr. 16 – Třída Kp_kolej



Obr. 17 – Třída Kp_soucin

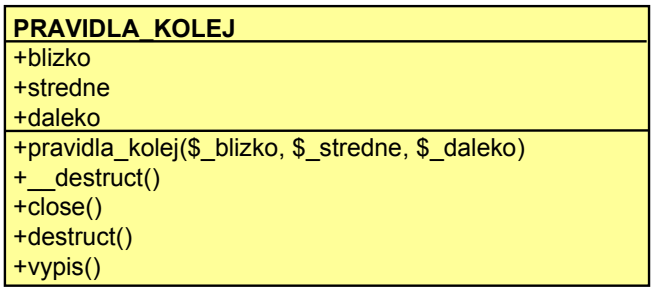


Obr. 18 – Třída Kp_soucet

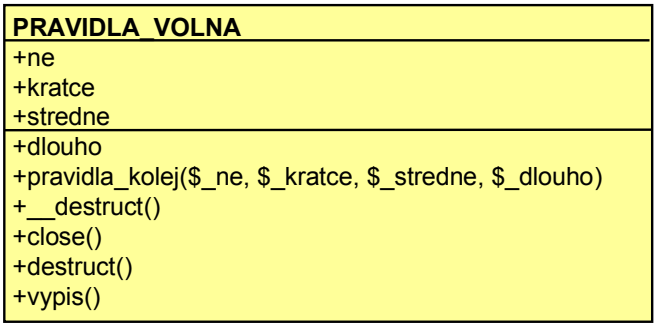
Výsledky všech tříd a jimi tvořených tabulek jsou ukládány do databáze k jejich pozdějšímu zpracování a popřípadě i další analýze.

Třída Pravidla_kolej, Pravidla_odjezd, Pravidla_volna

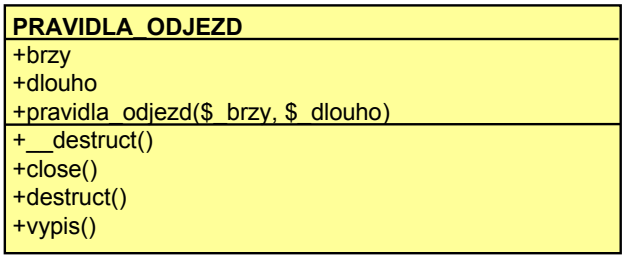
Když jsou již ohodnoceny všechny potřebné údaje o vlacích a kolejišti, je třeba získané výsledky zpracovat. Zpracováním je míněn průchod funkcemi příslušnosti a následná tvorba pravidel typu *if–then*. Funkce příslušnosti jsou zcela variabilní. V řešeném problému je využíváno čtyř funkcí příslušnosti. Informace pro třídy Pravidla_kolej, Pravidla_odjezd a Pravidla_volná jsou načítány z externího ini souboru. Při průchodu funkcemi příslušnosti jsou jednotlivé hodnoty „zasílány“ do příslušných tříd a vrací se správné lingvistické proměnné dle nastavených parametrů.



Obr. 19 – Třída Pravidla_kolej



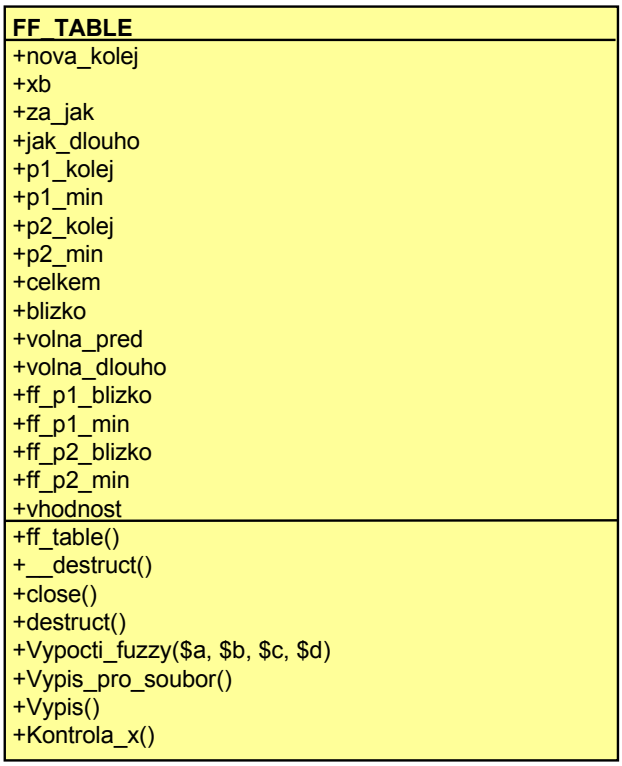
Obr. 20 – Třída Pravidla_volna



Obr. 21 – Třída Pravidla_odjezd

Třída FF_table

Všechny výsledky jsou po průchodu funkcemi příslušnosti ukládány do instance třídy FF_Table (FuzzyFikace). Zde obdobně jako u třídy Tabulka tvoří jedna instance třídy FF_table jeden řádek a symbolizuje jednu nástupištní kolej. Tabulka v sobě také uchovává informace o hodnotách, které symbolizují jednotlivé přípoje ještě před průchodem funkcemi pro tvorbu *if – then* pravidel.



Obr. 22 – Třída FF_table

Všechny třídy jsou svou orientací, nebo lépe řečeno filozofii, zaměřeny spíše na logiku řešené problematiky, nikoliv na vyšší efektivnost algoritmů a tříd. Tento záměr vychází z podstaty řešení problému, který má určit za pomocí fuzzy logiky vhodnou nástupištní kolej. Od prvopočátků řešení problému nebylo jasné, jaké metodiky bude zapotřebí využít k řešení. Stejně tomu bylo i u dílčích výpočtů. Nejednou nastala situace, kdy po validaci výsledků bylo od algoritmu zcela upuštěno, popřípadě byl do základu přepracován.

Jedinou bernou mincí byla data, která měla vstupovat, popřípadě vystupovat z již zmíněných algoritmů. U dat byla vždy jasná struktura, počet i přibližné hodnoty, ke kterým se bylo zapotřebí dobrat.

Z těchto důvodů bylo třeba konstrukce většiny tříd směřovat spíše do věcné části řešení problému.

5.2 Popis zajímavých funkcí

Celá aplikace GBP (Generátor Báte fuzzy Pravidel) je psána v programovacím jazyce PHP. Aplikace využívá spojení s databází MySQL, kde jsou shromažďována všechna data. Původně měla být celá aplikace postavena na programovacím jazyce Delphi společnosti Borland. Od tohoto záměru bylo upuštěno. Jazyk Delphi neumožňuje tak pohodlně a snadno přistupovat k databázím a zaostává i velkou režii výpočetních prostředků počítače, která je potřeba ke spojení s databází. Oproti tomu, je jazyk PHP přímo designován a optimalizován pro práci s databázemi. Z důvodů kompatibility byla použita verze PHP 5 a MySQL 4,1. Optimálnější by byla práce s databází verze 5, která umožňuje spoustu zajímavých možností, jak zrychlit práci, ale i v dnešní době využívá většina dostupných serverů databázi ve verzi 4. Kompatibilita tedy vyhrála nad rychlostí a optimalitou.

Použité technologie programování

Z existence předchozí kapitoly popisu tříd jasně vyplývá, že programování není čistě strukturované, ale je použito i objektové programování. U jazyků, jakým je PHP, je častěji aplikováno pouze strukturované programování občas doplněné použitím funkcí. Tento aspekt lze snadno také přisoudit skutečnosti, že objekty se v PHP objevili až ve verzi 4 a teprve ve verzi 5 s nimi lze zacházet téměř stejně, jako v jazycích typu C a J.

V dalších podkapitolách bude pohled zaměřen na vysvětlení zajímavých částí kódu.

Výběr vlaků v časovém okénku

V první části programu je třeba zjistit, jaký je stav na nádraží, když je nám ohlášen přijíždějící vlak na jinou kolej, než která je mu předepsána. Potřebujeme získat údaje o tom, které vlaky jsou na nádraží, které vlaky teprve přijedou a popřípadě kdy který vlak odjede. Časové okénko začíná v době, kdy je ohlášen námi zkoumaný vlak a končí 30 minut po této události. V tuto chvíli program spolupracuje se dvěma databázemi. A to databází zpožděných vlaků a databází vlaků z grafikonu vlakové dopravy. První databáze se jmenuje vlak_skutecnost a druhá vlak_gvd. Díky jednoduchým dotazům do databáze vybereme postupně všechny vlaky přijíždějící na ostatní koleje a každý z nich nám určí vlaky, které jsou v dané chvíli z našeho pohledu důležité.

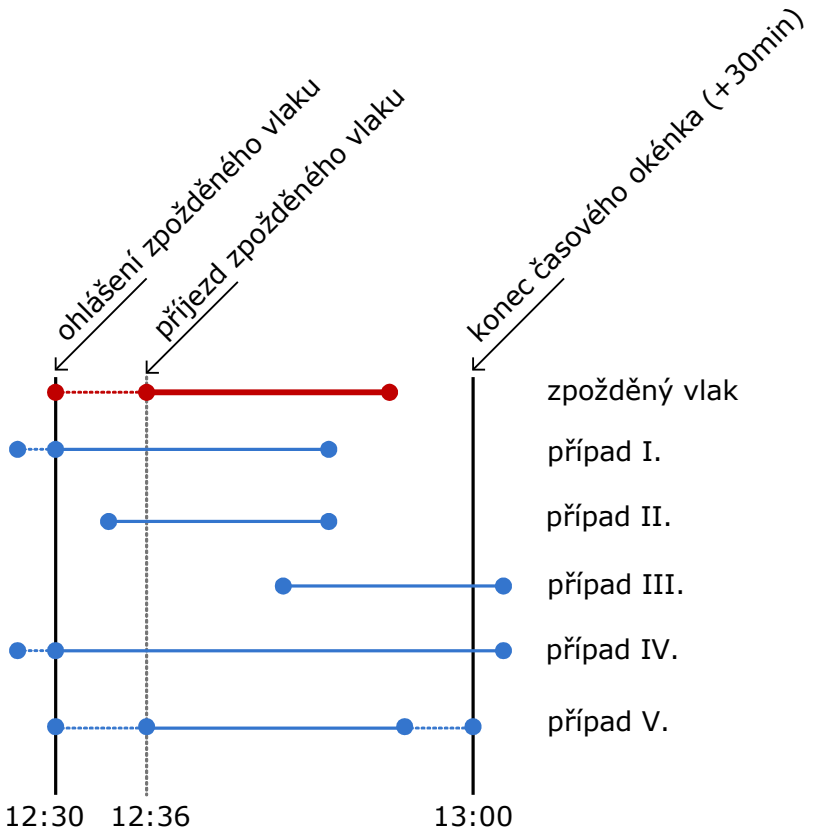
Por výběr takovýchto vlaků je třeba ošetřit všechny stavy, které mohou s daným vlakem nastat. Výsledky ukládáme do tabulky Tab. 2 do příslušných instancí třídy Tabulka, symbolizující jednotlivé nástupištní koleje.

U vlaku je nutno zjistit tyto stavy (ošetřit situace):

- přípojný vlak přijede dříve nebo současně s ohlášením zpožděného vlaku a zároveň neodjede dříve, než skončí časové okénko
- když přípojný vlak přijíždí déle než je ohlášen zpožděný vlak a přípojný vlak od jíždí dříve, než skončí časové okénko
- když přípojný vlak přijede déle, než je ohlášen zpožděný vlak a přípojný vlak od jede déle, než je námi zkoumané časové okénko
- když přípojný vlak přijede dříve nebo zároveň s ohlášením zpožděného vlaku odjede déle, než je námi stanovené časové okénko

- když příjezd zpožděného vlaku je totožný nebo větší s příjezdem přípoje a odjezd přípoje je menší nebo roven délce časového okénka

Tyto stavy by měly postihnout celé dění v kolejišti. Po jejich rozboru je program schopen vyplnit první část tabulky Tab. 2 a prakticky dopočítat s pomocí tabulky Tab. 4 druhou část tabulky Tab. 2.



Obr. 23 – Možnosti umístění vlaků v časovém okénku

Jako ukázku lze aplikovat složitější stav kombinací případů I. a III. Předpokládejme, že oba případné přípoje míří na stejnou kolej 13. Přípoj A na tuto kolej přijíždí ve 12:15 a odjíždí ve 12:40. Přípoj B přijíždí na tu samou kolej ve 12:42 a odjíždí ve 13:15. Algoritmus je schopen postihnout i takovouto situaci a první řádek tabulky Tab. 2 vyplní následovně.

Tab. 12 – Aplikace algoritmu časového okénka na vzorek dat

Vlak číslo	Nová kolej	Blízko plánované koleje	Volná za jak dlouho [min]	Volná jak dlouho [min]	Minut do pravidelného odjezdu přípoje	Blízko plánované koleje	Volná před ohlášením	Volná dostatečně dlouho	Přípoj	Celkem
No	Kolej	X_B	X_V	X_D	X_P	K_B	K_V	K_D	K_P	Suma
	13		10	2						

Počítání kombinací pravidel

Jako další zajímavou funkci programu lze označit algoritmus, který vygeneruje všechny možné druhy kombinací pravidel a u každé kombinace nahlíží do databáze, zda-li se kombinace vyskytla v reálné situaci na kolejišti a pokud ano, sečte tyto případy a výslednou četnost vloží do

databáze.

Tento algoritmus je velmi náročný na strojový čas. Je nutno brát v úvahu, že počet kombinací, které mohou vzniknout v kolejišti je: $3 \cdot 4 \cdot 4 \cdot 3 \cdot 2 \cdot 3 \cdot 2 \cdot 3 = 5184$. Jak je to možné? U každé proměnné v pravé části tabulky Tab. 6 mohou nastat stavy:

blízko	= [blízko, daleko, středně]	(3 stavy)
volná před	= [dlouho, středně, krátce, není]	(4 stavy)
volná dlouho	= [dlouho, středně, krátce, není]	(4 stavy)
přípoj 1 kolej	= [blízko, středně, daleko]	(3 stavy)
přípoj 1 odjezd	= [brzy, dlouho]	(2 stavy)
přípoj 2 kolej	= [blízko, středně, daleko]	(3 stavy)
přípoj 2 odjezd	= [brzy, dlouho]	(2 stavy)
vhodnost	= [vhodná, středně, nevhodná]	(3 stavy)

Vhodnou kombinací cyklů FOR pro každou proměnnou lze celkem snadno dosáhnou všech možných kombinací. Bohužel za cenu delší doby výpočtu. Uprostřed cyklů je položen dotaz do databáze, který sečte počet řádků odpovídajících daným proměnným. Pokud neexistuje v databázi z reálného zkoumání žádná kombinace odpovídající vzoru, vloží do databáze aktuální kombinaci s četností 0. V opačném případě je četnost stanovena počtem výskytu kombinace.

Otázkou však zůstává, zda-li by se i po dlouhodobějším zkoumání reálné situace na nádraží objevily všechny možné kombinace alespoň jednou.

Výběr nejčetnějšího pravidla

Když skončí algoritmus z předcházejícího příkladu, je třeba soubor setřídit postupně od první proměnné až k sedmé proměnné. Po tomto kroku přebírá iniciativu algoritmus, který z možné trojkombinace jednoho pravidla vybere to nejčetnější. Pokud takové není, zvolí náhodné, ohodnotí jeho četnost 0 a toto pravidlo uloží do databáze. Ostatní dvě smaže. Zde nastává již zmíněné omezení databáze ve verzi 4. Účinným prostředkem na tuto situaci by byly databázové pohledy (VIEW), popřípadě databázové procedury a funkce, které jsou zatím dostupné pouze v robustnějších databázích typu Oracle. S jejich pomocí by šel celý problém řešit, aniž by PHP kód musel čekat na výsledky z databáze a dle nich se rozhodovat. Ale i tak lze tento krok usutečnit díky několika SQL dotazům. V programu je sice použit jiný dotaz na výběr řádku z trojice s maximálním ohodnocením, ale je zde na místě uvést i tento.

```
SELECT MAX(test.id), test.tmp, test.cetnost FROM test, (SELECT MAX(cetnost)
AS cetnost, tmp FROM test GROUP BY tmp)t WHERE t.cetnost = test.cetnost AND
t.tmp = test.tmp GROUP BY test.tmp;
```

(8)

Na základě výsledku dotazu (8) je program schopen říci databázi, že navrácený řádek z trojice musí zůstat a ostatní dva mohou být smazány.

Otázka ale zní, proč aplikovat takto složitý dotaz, když k vypsání řádku s maximální četností postačí mnohem jednodušší. Předpokládejme, že jsou v databázi uloženy varianty pravidel vždy po třech a pravidla, která spolu souvisí mají stejnou hodnotu pojmenovanou jako „tmp“. O četnostech je již psáno výše. Takže v podstatě postačí dotaz se syntaxí (9).

```
SELECT MAX(cetnost) AS cetnost, * FROM fuzzy_sdruzeni GROUP BY tmp;
```

(9)

Bohužel, tento dotaz je špatný. Po jeho aplikaci na bázi dat nemusí být na první pohled patrné, že výsledek je nesprávný. Ale analýze zdrojových dat a porovnáním s výsledkem dostaneme nejčetnější hodnotu z trojice, ale další parametry označené symbolem * jsou zvoleny náhodně. Pro ilustraci problému je uveden následující příklad.

Tab. 13 – Problém nejčetnějšího pravidla

blízko	dlouho	dlouho	blízko	brzy	blízko	brzy	vhodná	125	1365
blízko	dlouho	dlouho	blízko	brzy	středně	brzy	středně	10	1365
blízko	dlouho	dlouho	blízko	brzy	středně	brzy	nevhodná	8	1365
blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	vhodná	1	1366
blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	středně	0	1366
blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	nevhodná	9	1366

Tab. 14 – Problém nejčetnějšího pravidla – výsledek

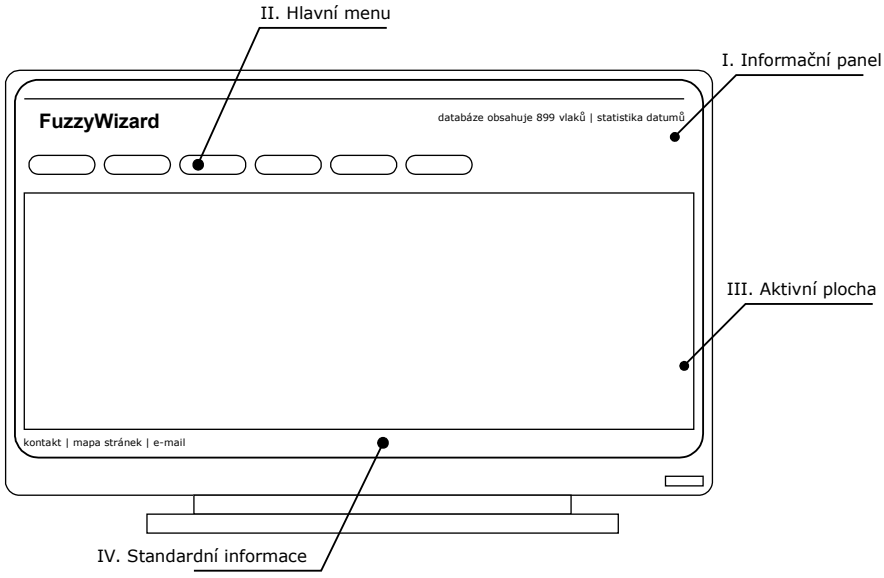
blízko	dlouho	dlouho	blízko	brzy	středně	brzy	vhodná	125	1365
blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	vhodná	9	1366

U první skupiny vrátil dotaz výsledek správný, kdežto u druhé vrátil první řádek a četnost maximální ze skupiny. Kdyby maximum v první skupině bylo na jiném, než na prvním řádku, výsledek by byl také špatný. Díky tomuto místu, byly i další výpočty chybné a dalo mnoho zkoumání, proč jsou výsledky jiné od předpokládaných.

6 Popis a práce s programem GBP

Celý program využívá služeb webového serveru Apache, který byl při vývoji aplikace provozován na lokálním počítači (locahostu). Z důvodů maximální kompatibility je celý kód psán univerzálně i pro nižší verze použitého programovacího jazyka. Díky webové platformě lze program bez větších úprav nasadit na kterémkoliv dostupném webhostingu, a to jak komerčním, tak neplaceném. Jedninou podmínkou je, aby hosting podporoval překlad PHP skriptů a databázi MySQL. V tomto ohledu je zvolená platforma univerzálnější, než předpokládaný jazyk Delphi. Postačí, když aplikace bude nainstalována na serveru dostupném ze sítě Internet a uživatel, aniž by musel mít kopii programu vždy po ruce, může aplikaci obsluhovat z jakéhokoliv počítače, který disponuje připojením k Internetu. Výhodou může být i dostupnost webových prohlížečů v mobilních zařízeních. Není třeba instalovat žádné programy. Výkonná část se odehrává na serveru a na klientu (prohlížeči), v našem případě mobilním zařízení, jsou zobrazovány pouze výsledky algoritmizací.

Po zadání příslušné adresy do webového prohlížeče se objeví aplikace nazvaná FuzzyWizard. Wizard vychází z anglického slova čaroděj a v informační technice a počítačových aplikacích je slovo Wizard považováno v přeneseném slova smyslu také za průvodce. Takže se zobrazí úvodní obrazovka „průvodce fuzzyfikací“. Hlavní okno programu je rozděleno do několika podsekcí. Horní panel tvoří sekci „informační“. Na tento informační panel navazuje panel hlavního menu. Z hlavního menu ovládáme celý program a volíme akce, které budou aktuálně prováděny. Pod hlavním menu je aktivní plocha. V aktivní ploše se zobrazují mezivýsledky a výsledky výpočtů plus řada dalších informací. Díky aktivní ploše jsou zobrazovány a parametrizovány jednotlivé kroky. Úplně ve spodní části obrazovky jsou klasické informace, které jsou od webové aplikace vyžadovány. Je to mapa stránek, e-mail na autora, kontakty,...



Obr. 24 – Základní členění obrazovky programu GBP

6.1 Načtení dat ze souborů a jejich formát

Aby mohl program pracovat, potřebuje ke svým výpočtům zdrojová data. Data, nad nimiž bude nadále provádět operace. Sběr dat je popsán v prvních kapitolách této práce. Program je schopen zpracovat pouze jeden grafikon, ale to ve své podstatě naprosto postačuje. V roce 2006

bychom tedy mohli zkoumat všechna data od druhé změny grafikonu. Pro tato data máme informace uložené v souboru grafikon.csv. Program načítání dat je navržen tak, aby podporoval zmíněné CSV soubory.

CSV je zkratka anglických slov *comma-separated values*, tedy hodnoty oddělené čárkami. V našich souborech jsme čárky zaměnili za středníky. Tento formát podporuje většina tabulkových procesorů ve svých exportech i importech. Byl zvolen z jednoho prostého důvodu. Data o vlacích nelze automatizovaně získávat, a jako prostředek pro sběr dat většina uživatelů použije nějaký tabulkový procesor. Po exportu provozních dat z takového programu jej snadno nahrajeme do naší databáze. Obráceně lze říci, že výstupní soubor dat po průchodu všemi příslušnými algoritmy v programu bude chtít každý uživatel nějakým způsobem analyzovat. Pomocí CSV souboru si výsledky z programu importuje do svého tabulkového procesoru a může je podrobit dalšímu zkoumání. To ale předbíháme.

Pro správnou funkci programu je třeba mít dva druhy získaných dat. Data o vlacích, které obsahuje grafikon vlakové dopravy. O těchto datech lze říci, že jsou předpisem k simulaci nádraží za ideálních podmínek. A druhou násadu dat s vlaky, která je získána ze skutečného kolejiště, tedy dopravního deníku (elektronického). V případě námi zkoumaných deseti dní, jsme připravili pro přehlednost násadu deseti souborů, každý s jedním dnem ze skutečného zkoumání kolejiště.

Formát dat ve vstupním souboru určeném k sestavení předepsaného stavu v kolejišti na nádraží z GVD by měl striktně dodržovat následující charakter a v tomto pořadí.

Tab. 15 – Formát dat o vlaku z plánu obsazení nádraží

číslo vlaku	příjezd	odjezd	kolej
9541	05:33:00	05:36:00	22

A vstupní formát vlaků, přijíždějící na jinou kolej, než která je určena tímto předpisem musí obsahovat právě tyto atributy a v tomto pořadí.

Tab. 16 – Formát dat o vlaku ze skutečného zkoumání

datum	číslo vlaku	ohlášení	příjezd	odjezd	kolej
2006-08-07	9503	07:32:00	07:39:00	07:41:00	2

Mohlo by se zdát až zbytečné dělit časy i na vteřiny. Ve vstupních souborech našich výpočtů bylo od těchto údajů abstrahováno a výpočet probíhal pouze v minutách. Ovšem program je zcela kompatibilní i s počítáním na vteřiny. Druhým důvodem proč mají časy tento formát je ten, že s takovýmto formátem je kompatibilní databáze a nemusí docházet k časově náročným konverzím.

Veškeré přístupy uživatele programu k databázím se uskutečňují z položky hlavního menu „Práce s databází“. Odtud je uživatel schopen databáze doplňovat, mazat v nich uložená data a připravovat je na další kroky výpočtů. Není tomu jinak ani v případě vkládání připravených CSV souborů. V položce „Načtení dat s vlaky z GVD ze souboru“ pomocí dialogového okna nalistovat připravený soubor a odeslat jej na server. Stejně tak můžeme celou databázi s vlaky z GVD situace smazat.

Naprosto totožná je situace s vlaky z reálného zkoumání. K tomuto účelu slouží formulář: Načtení dat s vlaky ze skutečného zkoumání. Problém nenastane ani v tom, když máme rozděleny dny do jednotlivých souborů. Zkrátka lze postupně přidávat den po dni. Každý den se přidá do databáze k předchozímu. S tím je počítáno a vlaky jsou proto ještě označeny atributem

datum, který by jinak nebyl potřeba.

Jak v položce „Práce s databází“ tak na „Informačním panelu“ si lze ověřit, zda-li byly vloženy všechny vlaky a podívat se na jejich vstupní statistiky.

Po vložení potřebných vstupních dat si ještě předtím, než je začneme zpracovávat můžeme vypsat statistiku přípojných vlaků.

6.2 Histogram přípojných vlaků

V celém programu a všech výpočtech předpokládáme existenci dvou přípojných vlaků. Pokud tomu tak není, je snaha o ohodnocení této situace. Díky položce v hlavním menu „Histogramy“ jsme na základě algoritmu výběru vlaku v časovém okénku schopni určit počet přípojů osobního vlaku přijíždějícího na jinou kolej, než která mu byla přidělena grafikonem vlakové dopravy.

Graficky zpracovaný histogram ukazuje počet možností a jejich četností, které mohou nastat v kolejišti za námi zkoumané období. Na ose x je počet možných přípojů a na ose y je vynesena četnost takovýchto situací. Jako nejsilnější možnost, se ze vzorku zkoumaných dat, ukázala návaznost vlaku na jeden přípoj. Přitom maximální počet přípojů byl vyhodnocen na 5.

Další hodnocení je radno ponechat na závěr práce.

6.3 Funkce příslušnosti

Aby byl program univerzální a umožňoval zcela libovolně nastavit hodnoty funkcí příslušnosti, které určují jazykovou proměnnou ve výstupním souboru, lze snadno upravit hranice těchto funkcí. V hlavním menu postačí vybrat volbu „Funkce příslušnosti“.

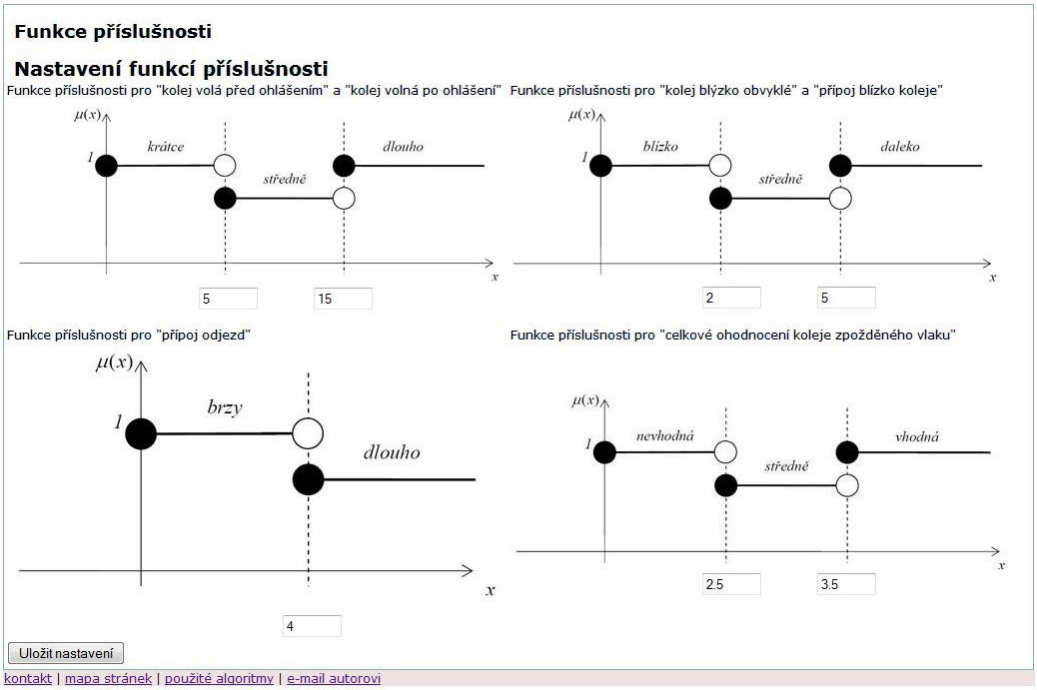
Funkce příslušnosti jsou defaultně nastaveny na hodnoty, se kterými bylo počítáno v celé práci. Hodnoty byly zvoleny na základě výsledků, které poskytly algoritmy. Celá aktivní plocha je snadno čitelná a funkce jsou schématicky naznačeny. Postačí pouze změnit číselné hodnoty hranic a uložit nastavení. Ukládání probíhá pomocí tlačítka ve spodní části aktivní plochy. Je nutno podotknout, že funkce jsou citlivé i na desetiny a oddělovačem desetinných míst je interpunkční znaménko tečka.

Nastavení, které zadá uživatel a uloží, se objeví i při příštím spuštění programu. Nastavené hraniční hodnoty funkcí příslušnosti se ukládají do pomocného ini souboru a při příštím spuštění se z tohoto souboru zase načtou. S těmito hodnotami je počítáno v dalších krocích algoritmu.

Z vědomostech o funkcích příslušnosti, které byly popsány v teoretické části této práce vyplývá, že je třeba znát alespoň 4 hodnoty, ne-li více, k určení hranic. Ale v nastavení programu funkcí příslušnosti jsou u jednotlivých voleb maximálně dvě možná nastavení hranic. Tento problém lze vysvětlit jednoduchým způsobem.

Funkce příslušnosti by měla vždy začínat od 0 do maxima. Maximum je pro každou funkci jiné, ale vždy neměnné. Proto v nastavení můžeme abstrahovat od maximální a minimální hodnoty. Díky tomu lze i eliminovat možnost chyby lidského faktoru, kdyby v nastavení minima a maxima byla jiná hodnota, než která dané funkci náleží.

Ze schématického hlediska je také upuštěno od prolíná funkcí. Černě vyplněnými kruhy jsou označeny hraniční hodnoty, které do intervalu ještě patří a bílými kruhy s černou konturou jsou označeny hraniční hodnoty, které již do dané části funkce nepatří. Analogicky lze odvodit matematický zápis se znaménky ostrého ohraničení a kulatých závorek.



Obr. 25 – Schématický náhled funkcí příslušnosti aplikované v programu

6.4 Databáze vlaků a statistika datum

Obě funkcionality lze označit pouze za informativní a na jejich základech nejsou postaveny žádné další funkcionality programu. Slouží pouze pro ověření správnosti vstupních dat.

Databáze vlaku je lokalizována v informačním panelu a po volbě této položky se vypíší vlaky ze všech zjištěných vstupních hodnot reálného kolejiště seskupených podle čísla vlaku. Informativně lze vyčíst, zda-li dispečer vlakové dopravy přidělil v pozorovaném období témuž vlaku stejnou nástupištní kolej, nebo jak se liší časy příjezdů a odjezdů jednoho vlaku ve zkoumaném období.

Statistika datum, má opět jako předešlá pouze informativní charakter, a informuje o tom, kolika vlakům za jeden den byla přidělena jiná nástupištní kolej než kterou jí předepisuje grafikon vlakové dopravy přidružený k nádraží. Zpětně se dá snadno dopátrat například, zda-li je víkendový provoz přesnější na dodržování předepsaných kolejí či naopak.

6.5 Wizard tabulky

Průvodce – neboli wizard – tabulky je prvním krokem, který mění či doplňuje námi zadaná vstupní data. Cílem prvního wizardu je vytvořit tabulku Tab. 2, Tab. 4 a tabulku Tab. 6. V několika po sobě následujících krocích, které lze libovolně dle možností parametrizovat, program uloží do databáze výsledky zmíněných tabulek.

Ke spuštění wizardu tabulky postačí v panelu hlavního menu zvolit tuto možnost. Před spuštěním kteréhokoliv průvodce jsou provedeny kontrolní operace, které mají zamezit tomu, aby nedošlo k porušení některé ze vstupních podmínek. Pokud dojde při kontrole ke zjištění, že tyto podmínky nejsou splněny, průvodce nedovolí přistoupit k dalšímu kroku.

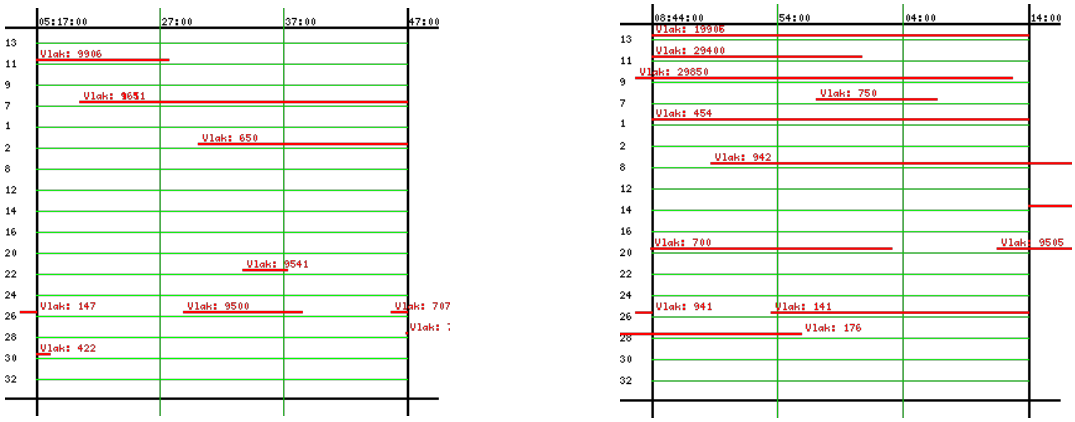
U tabulek je kontrola zaměřena na vstupní databázi GVD a databázi s vlaky z reálného zkoumání. Pokud některá databáze neobsahuje žádný záznam, nelze pokračovat dalším krokem

a je třeba tyto údaje doplnit přes nabídku v hlavním menu „Práce s databází“.

Ve druhém kroku průvodce je na základě již popsaných algoritmů – časového okénka, výpočtech hodnot K_B, K_V, K_D, K_P, Suma a pomocné tabulky pro dílčí výpočty hodnocení přípojného vlaku – vypočtena Tabulka Tab. 2 a Tab. 4. Průvodce nabízí snadnou parametrizaci tohoto kroku a lze si vybrat, zda-li proběhne pouze výpočet na pozadí, nebo zda-li budou výsledky výpočtů publikovány v aktivní ploše.

Program v tomto kroku také umožňuje vypsat shodnost volby koleje, která byla vybrána za nejvhodnější a k ní kolej, jež byla vybrána vlakovým dispečerem. Může také nastat situace, že program vybere více jak jednu kolej za optimální a v tom případě vypíše všechny optimální varianty.

Další zajímavou možností, kterou skýtá program, je volba grafického znázornění situace v kolejišti. Jedná se o schématický obrázek časového okénka, který dokresluje celou situaci. Díky tomuto schématu lze také snadno provést kontrolu výsledků.



Obr. 26 – Schématický náčrt nástupišť v programu GBP

Pokud byla zvolena možnost „zobrazit všechny údaje, které jsou dostupné“, u každého vlaku přijíždějícího na jinou kolej, než která mu byla přidělena grafikonem vlakové dopravy máme k dispozici tabulky Tab. 2, Tab. 4 a grafické znázornění kolejiště. Důležitým upozorněním pro uživatele je fakt, že pokud probíhá výpočet, neměl by zasahovat do programu a měl by vyčkat, dokud algoritmy neskončí svou práci. V opačném případě se může stát, že nedojde ke zpracování všech údajů o všech vlacích a získané výsledky budou pouze částečné.

S naším vstupním vzorkem dat bylo vypsáno 439 tabulek typu Tab. 2, stejné množství tabulek typu Tab. 4 a 439 grafických znázornění kolejí. Pro snadnější orientaci v tomto množství mezivýsledků jsou tabulky sbaleny do ikony čísla vlaku a základních údajů o něm. Pokud si přejeme tyto data zobrazit, postačí kliknout na příslušnou ikonu vlaku.

Krok 3 převede tabulku Tab. 2 do tabulky Tab. 6 a provede všechny potřebné translace k výpočtu pravé části tabulky Tab. 6. Také se provede načtení hraničních hodnot funkcí příslušnosti ze souboru a zkonstruují se třídy příslušnosti a provede se ohodnocení jednotlivých řádků.

Po ukončení kroku 3 je do databáze uložena celá tabulka Tab. 6. Parametrizací následujícího kroku lze provést uložení levé části tabulky Tab. 6 do souboru, jež je potřebný k dalšímu zpracování v programovém prostředí Matlab.

4. krok průvodce vypíše celou tabulku Tab. 6. Je zobrazeno tedy celkem 439 tabulek.

Dokončením průvodce tabulky jsme získali obraz všech potřebných data v databázi pro jejich další zpracování. Trvání výpočtu úzce závislé na velikosti vstupního souboru a počtu operací, které jsou se souborem prováděny. Teoreticky neexistuje žádná hranice, která by vytyčila množství těchto dat. Prakticky se jedná o dostupnou velikost operační paměti databázového stroje, omezení velikosti databáze a například také omezení dávky času (časového kvanta), které je určeno pro zpracování jednotlivých kroků. Ale vše se dá ovlivnit.

6.6 Wizard fuzzifikace

Průvodce fuzzifikací má jednoduchou filosofii. Jeho úkolem je doplnit pravidla z reálného zkoumání na stav, kdy se v souboru dat objeví všechny možné kombinace pravidel, které mohou nastat a setřídit je. Projde tedy všechny možné kombinace, vždy se dotáže databáze, zda-li již takové pravidlo existuje. Pokud ano, sečte výskyt tohoto pravidla a uloží jej včetně jeho četnosti do databáze. Pokud ne vloží pravidlo do databáze s četností 0.

Při použití wizardu fuzzifikace je opět provedena kontrola vstupní a výstupní databáze. Vstupní databáze by měla obsahovat všechny vzniklé tabulky typu Tab. 6. To je zajištěno v kroku, kdy je na zpracování vstupních dat použit průvodce tabulky. Výstupní databáze by měla být prázdná. Průvodce na tyto skutečnosti sám upozorní a pokud není vše, jak by mělo být, nedovolí pokračovat dalším krokem.

Vymazání výstupní databáze se provede z menu „Práce s databází“ → „Vyprázdnění databáze s vlaky po průchodu shlukovacím algoritmem“.

Pokud jsou splněny všechny vstupní podmínky, lze pokračovat prvním krokem. V prvním kroku se použije důmyslný algoritmus pro výpočet všech kombinací pravidel a provedou se všechny operace uvedené v prvním odstavci této podkapitoly. Průchod tímto algoritmem – jak již bylo řečeno – je velice zdlouhavý a je třeba počkat na jeho dokončení. Jinak opět může dojít ke ztrátě dat, nebo k jejich neúplnosti. Za podmínek nastavených při testování celého programu trval průchod přibližně 90 vteřin.

Druhý krok je o mnoho náročnější. Skládá se ze sedmi dílčích částí. Snadno se dá odvodit proč. V každém dílčím kroku třídíme pravidla dle proměnných v prvním až sedmém sloupečku a to následujícím způsobem.

Při prvním průchodu si spočítáme všechny varianty, které mohou nastat v prvním sloupečku. Tedy v první proměnné „blízko“. Jsou to tři. Takže projdeme 3× soubor a v každém kroku ohodnotíme novou pomocnou proměnnou v řádku krokem procházení. Krok procházení je zvolen na základě četnosti první proměnné a je dle ní také činěn výběr, který řádek ohodnotíme jakou proměnnou. Zjednodušeně řečeno, zjistili jsme, že první proměnná může nabývat tří hodnot. A to blízko, středně, daleko. Projdeme celý soubor dat a vybereme pouze ty řádky, kde je první proměnná blízko. Ty ohodnotíme do nově zavedeného atributu řádku číslicí 1. Při druhém průchodu vybereme všechny řádky, kde je první proměnná „středně“, a ohodnotíme je 2. A v třetím kroku zvolíme zbývající proměnnou „daleko“ a ty řádky, jež disponují touto hodnotou v prvním sloupci ohodnotíme trojkou.

Nyní jsme schopni vybrat jednoduše libovolnou skupinu pouze na základě výběru mezi hodnotami 1, 2 a 3. V dalším postupu máme za úkol třídit dle druhé proměnné, ale nesmíme opomenout fakt, že již máme celý soubor rozdělen do tří utříděných skupin. Musíme tedy utřídit každou z těchto skupin zvlášť dle hodnoty její druhé proměnné.

První krok je stejný jako v předchozím případě, je třeba zjistit, kolik variant může nastat ve druhé proměnné. Jsou to 4. Musíme tedy projít všechny řádky, které jsou v pomocné proměnné ohodnoceny jedničkou a setřídit je do čtyř skupin. Postup je následující. O skupině víme, že její pomocná proměnná – říkejme jí „tmp“ – je ohodnocena 1. Procházíme tedy 4× celý soubor, kde je takto ohodnocena proměnná tmp a vybírám jen ty řádky, které mají ve druhém sloupečku literární proměnnou „dlouho“. U takovýchto řádků nastavím proměnnou tmp na 4. (maximální ohodnocení z předchozího třídění je 3 a nyní poprvé procházím druhým tříděním, tedy + 1 = 4). Následně procházím znovu celý soubor, kde je pomocná proměnná tmp rovna 1 a jsou vybírány řádky s proměnnou ve druhém sloupečku „středně“. Tmp u řádků vyhovující této podmínce je ohodnoceno na 5 (3 + 2). Následně je vybrána proměnná „krátce“, zde je tmp nastaveno na 6 a v posledním kroku je tmp nastaveno na 7. Ale stále ještě zbývají řádky po prvním třídění, jejichž pomocná proměnná je ohodnocena 2 a 3. Tím samým způsobem je postupováno dále pro druhou a třetí skupinu. Po dokončení třídění dle druhého sloupečku bude pomocná tmp nabývat hodnoty 14.

Popis dalšího třídění je naprosto totožný pouze s dalšími iteracemi pomocné proměnné tmp a obměnou tříděných jazykových proměnných. Snad je předpis třídění dostatečně vysvětlen, aby mohl sloužit pro další pochopení operací, jež jsou zde pouze naznačeny. Zajímavostí může být, že po dokončení třídění dle sedmého sloupce proměnná tmp nabude maximální hodnoty 3092 a minimální 1365. Když se posuneme dále, k časové náročnosti algoritmu, trvalo zpracování celého vstupního vzorku ve druhém kroku wizardu fuzzy 7 minut a 38 sekund.

6.7 Wizard třídění pravidel

Tento krok zde jen zběžně popíšeme. Algoritmy v něm použité již byly vysvětleny v teoretické části práce a proto není nutné se k nim vracet. Vhodným doplněním by bylo upřesnění, který krok se zabývá jakou částí algoritmu.

I u průvodce třídění pravidel je třeba dodržet vstupní podmínky. Podmínkou je dokončený předchozí krok a nezačnutý následující krok. Podmětně řečeno, pravidla musí být setříděna a nesmí být nad nimi provedeny žádné další operace. Pokud tomu tak není, lze se podívat do „Práce s databází“ a vymazat databázi pro průchod tříděním pravidel. Následně lze pokračovat dalším krokem.

0. krok provede výběr nejčetnějšího pravidla. Návod na něj je popsán v kapitole „výběr nejčetnějšího pravidla“. Za podmínek nastavených při testování programu trval tento výběr 2 minuty 42 sekund.

1. krok lze nazvat také spárování z násadou. Jednotlivé řádky matice jsou konfrontovány s násadou, která určuje logickou návaznost řádků. V souboru dat je nově přidán sloupeček nazvaný „pozice v matici“. Do „pozice v matici“ je uložena číselná hodnota z násady. Krok trval přibližně 2 minuty 9 sekund.

Aby bylo možné vypočítat procento četnosti pravidel před shlukováním a po něm, je třeba získat hodnoty na počátku výpočtu. Než jsou tedy započaty jakékoliv operace se souborem dat, je spočítána četnost jednotlivých třicetišestic a uchována pro pozdější vyjádření. O to se stará krok 2.

Skutečnou první operaci nad daty provádíme ve 3. kroku. U prvních třech čtveřic z třicetišestice zajišťuje to, aby hodnoty výsledku druhého a třetího řádku byly stejné. Operace trvá několik málo okamžiků a její trvání je závislé pouze na tom, u kolika čtveřic se vyskytne situace, kdy výsledná proměnná vhodnost ve druhém a třetím řádku není shodná.

Ve čtvrtém kroku jsou procházeny zbylé řádky ve třicetišestici, tedy 3× osmici a algoritmus se pokouší zajistit, aby první dvojice řádků z osmice měla stejnou výstupní proměnnou.

V pátém kroku je kopírován postup z kroku čtverého s tím rozdílem, že se pokouším sladit osmé proměnná čtveřice z osmice.

6. krok je shlukování. Shlukování prvních třech čtveřic z třicetišestice. Podmínky pro shlukování jsou dány stejnou hodnotou výsledné proměnné vhodnost. Pokud tedy čtveřice disponuje stejnou výslednou proměnnou je nad ní provedena akce shlukování. Shlukování čtveřic je časově o něco náročnější než předchozí operace – měníme více řádků v databázi – ale za daných podmínek by nemělo trvat déle jak půl minuty.

Sedmý krok ve třídění pravidel eliminuje náhodnost výsledných osmých proměnných s nulovou četností. Pokud tedy pravidlo disponuje četností 0, tento krok se pokouší vhodnou kombinací výsledné proměnné u takovýchto řádků v prvních třech čtveřicích připravit skupinu na shlukování a 8. krok toto shlukovaní provede.

9. krok provede shlukování zbylých třech osmic. Podmínky jsou stejné jako u čtveřic. Výsledná proměnná „vhodnost“ musí být v osmici stejná.

10. a 11. krok provede stejnou operaci jako 6. a 7. krok. Rozdíl je pouze ve skupinách, kdy se 10. a 11. krok zajímá o zbylé tři osmice.

Posledním krokem je výpis upravených a shluknutých pravidel. Výpis lze jednoduše parametrizovat a aplikovat nad množinou výsledků filtry. Program dovoluje vypisovat pravidla od různých četností. Aby byl výpis očištěn od pravidel, jejichž četnost je nula, postačí aplikovat filtr se zadáním, aby četnost vypisovaných pravidel byla větší či rovna 1.

Výsledky lze také, jak již bylo řečena na počátku této kapitoly, exportovat do CSV souboru. Tento export je navržen tak, aby výsledná pravidla šla snadno číst a libovolně upravovat ve zvoleném tabulkovém procesoru.

7 Závěrečné shrnutí

Cílem práce měl být počítačový program (aplikace) jež bude generovat bázi fuzzy pravidel. Tento program byl vytvořen a umožňuje všechny požadované nastavení. Uživatel je schopen sám nastavit funkce příslušnosti. I když je tato možnost omezena pouze na hranice hodnot proměnných, je pro řešenou problematiku dostačující. Následné vytvoření báze pravidel typu *if – then* zajišťuje aplikace také včetně uložení do souboru vhodného formátu.

Při řešení problematiky nastalo několik situací, které daný problém dosti zkomplikovaly a bylo nutné upustit od detailnějších nastavení programu. Jednalo se především o problém automatizovaného sběru provozních dat. Nakonec se problém ukázal jako neřešitelný a jednotlivé dny musely být analyzovány ručně, což do značné míry pozdrželo další řešenou problematiku.

Zejména muselo být upuštěno od detailnějšího nastavení tvaru a počtu proměnných u funkcí příslušnosti. Zadané hranice a jazykové proměnné u ohodnocení by bylo vhodné dále doplnit o možnost vkládání vlastních názvů těchto proměnných a i jejich počtu. Například výsledná proměnný „vhodnost“ by šla rozšířit takto.

Nevhodná <0; 1,8), méně vhodná <1,8; 2,5), středně <2,5; 3,0), vhodnější <3,0; 3,5), nejvhodnější <3,5; 4> . Změnili jsme počet proměnných na 5 a lze tak podrobněji ještě roztřídit výstupní, popřípadě vstupní data.

Program GBP je schopen po zadání plánu obsazení kolejí a provozních dat obsazení kolejí vygenerovat bázi pravidel fuzzy, jež je předpisem pro výběr náhradní nástupištní koleje v rozhodovacím problému přiřazení koleje přijíždějícímu osobnímu vlaku. Tento předpis lze dále zpracovat za pomocí software Matlab a FuzzyToolboxu.

Celý program a jeho algoritmy jsou spíše směřovány do reálného světa a postupy algoritmů kopírují postupy výpočtů. Zajímavým námětem by byla adaptace programu, aby byl výhradně zaměřen na optimalitu výpočtů. Vznikla by tak aplikace, která by pocházela ze světa čistě programátorského. Nevýhodou by byl fakt, že takto vytvořená aplikace by byla velice špatně rozšiřitelná či upravitelná z hlediska filozofie problému, ale výpočty by prováděla za minimum programových nároků a to nejen hardwarových ale i časových.

Při tvorbě diplomové práce byla její část – převážně výsledky z provozních dat a levá část tabulky Tab. 6 – použity pro článek do odborné publikace týkající se problému hledání vhodné nástupištní koleje užívaného při dílčím rozhodovacím úkolu. Článek byl publikován pod názvem „Architektury a techniky simulačních modelů dopravních systémů“ výzkumného záměru teorie dopravních systémů. Díky funkcionalitě exportu pro Matlab bylo snadné použít mezivýsledky z programu a doposud zjištěných historických dat, jako podkladů k tomuto článku.

Soupis bibliografických citací

[1] JURA, P. Základy fuzzy logiky pro řízení a modelování. Brno : *VUTIUM*, 2003. ISBN 80-214-2261-0

[2] ZÁHOROVÁ, Věra. Konstrukce pravidel pro určení nástupištní koleje přijíždějícího vlaku na základě provozních dat . *Infotrans 2007 : konference*. 2007, [cit. 2008-12-08], s. 1-7. Dostupný z WWW: <http://infotrans.upce.cz/>.

[3] JANAL, Petr. Užití Fuzzy modelu jako analizátoru stupně ohrožení za povodňové situace. Brno 2007, [cit. 2008-12-08], s. 1-7. Dostupný z WWW: <http://www.fce.vutbr.cz/veda/JUNIORSTAV2007/Sekce_3/Janal_Petr_CL.pdf />.

[4] TYKAL, J., DOKULIL, J. Fuzzy SQL . 2007, [cit. 2008-12-08], PowerPoint prezentace, Dostupný z WWW: <www.ksi.mff.cuni.cz/~pokorny/dj/prezentace/2_50.ppt>.

[5] ŠOTEK, K. a spol. Tvorba jízdních řádů pomocí výpočetní techniky na českých drahách. 2001, [cit. 2008-12-08], Dostupný z WWW: <http://spz.logout.cz>

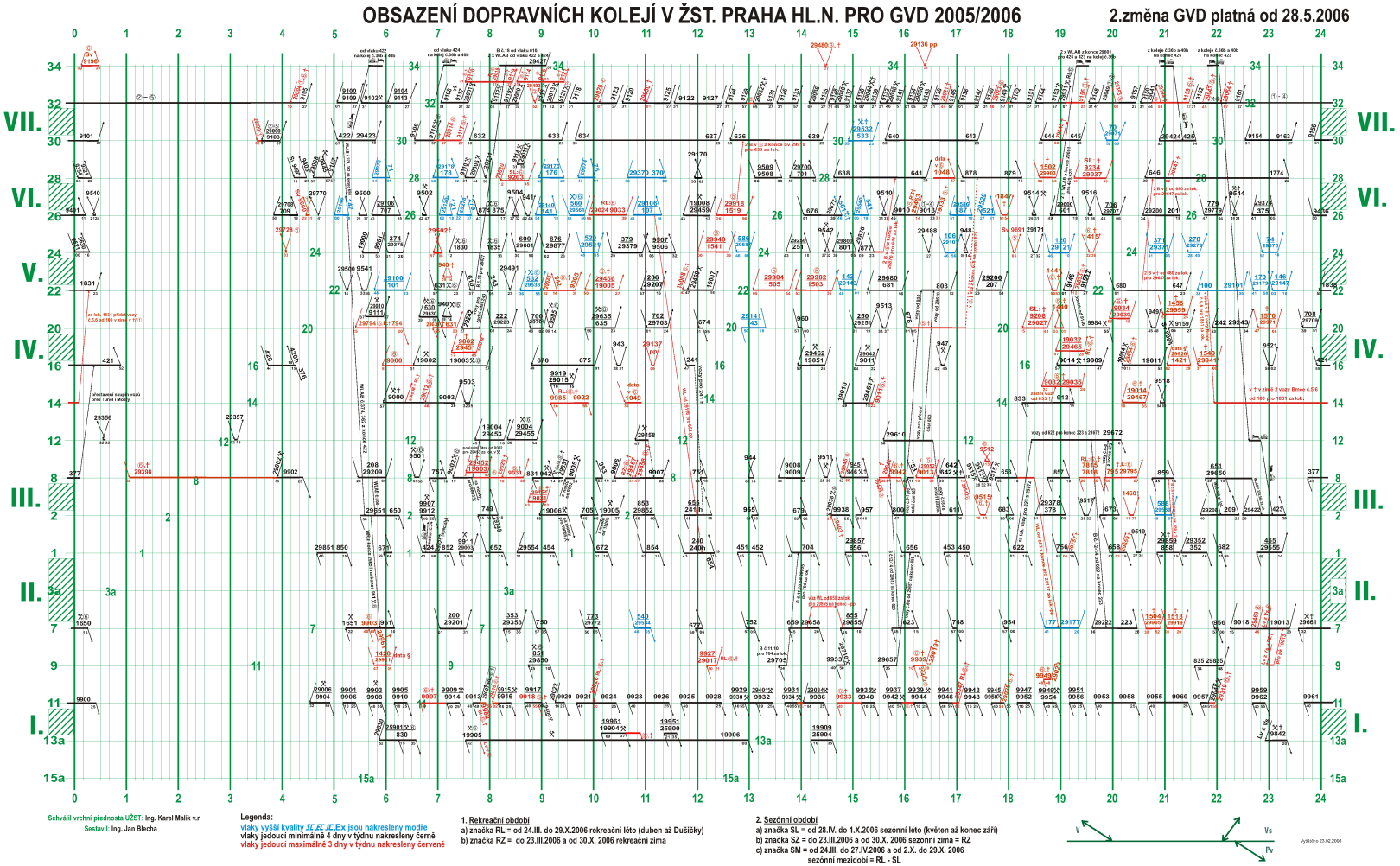
Přílohy

Seznam příloh

Příloha A	Plán obsazení kolejí v žst. Praha hl. n.
Příloha B	Vstupní soubor dat vypracovaný na základě plánu obsazení dopravních kolejí
Příloha C	Ilustrativní vstupní soubor vlaků z 1.8.2006 – 4.8.2006
Příloha D	Část nejzajímavějších výsledných *if–then* pravidel
Příloha E	Datový nosič s doprovodnými materiály. Popis adresářové struktury a obsahu datového média je uveden v souboru *readme.txt*, který je umístěn v kořenovém adresáři

Příloha A

Obsazení dopravních kolejí v železniční stanici Praha hlavní nádraží pro grafikon vlakové dopravy 2005/2006 platném po 2. změně od 28.5.2006



Příloha B

Vstupní soubor dat vypracovaný na základě plánu obsazení dopravních kolejí v žst. Praha hl.n. Pro GVD 2005/2006 2.změna

Číslo vlaku	Příjez	Odjezd	Kolej
377	00:00:00	00:08:00	8
9401	00:00:00	00:11:00	26
29356	00:32:00	00:33:00	12
1650	00:00:00	00:15:00	7
9540	00:21:00	00:24:00	26
9900	00:00:00	00:25:00	11
421	00:00:00	00:52:00	16
420	03:44:00	04:15:00	16
29770	04:41:00	04:41:00	26
29851	04:40:00	05:15:00	1
9904	04:32:00	04:55:00	11
9445	04:37:00	04:57:00	28
422	05:07:00	05:20:00	30
29500	05:14:00	05:14:00	22
850	04:40:00	05:15:00	1
147	05:00:00	05:18:00	26
1651	05:22:00	05:40:00	7
9906	05:10:00	05:28:00	11
208	05:28:00	06:00:00	8
29651	04:40:00	05:15:00	1
9541	05:33:00	05:36:00	22
9500	05:29:00	05:37:00	26
71	05:44:00	06:01:00	28
671	06:59:00	06:06:00	1
961	05:22:00	06:10:00	7
9905	06:10:00	06:25:00	11
707	05:43:00	06:15:00	26
650	05:30:00	06:15:00	2
9910	06:10:00	06:25:00	11
830	05:52:00	06:35:00	13
424	06:42:00	07:15:00	1
9501	06:33:00	06:36:00	8
9907	06:40:00	07:10:00	11
200	07:01:00	07:31:00	7
631	06:57:00	06:23:00	22
940	07:09:00	07:14:00	20
121	07:02:00	07:18:00	26
9909	07:10:00	07:25:00	11
178	06:56:00	07:22:00	28
757	06:50:00	07:10:00	8
852	06:42:00	07:15:00	1
19905	07:32:00	13:00:00	13
610	07:37:00	07:45:00	22
279	07:22:00	07:37:00	26
874	07:47:00	08:00:00	26

Číslo vlaku	Příjez	Odjezd	Kolej
9913	07:40:00	07:49:00	11
9503	07:32:00	07:35:00	14
222	08:01:00	08:24:00	20
243	07:37:00	08:05:00	22
652	07:57:00	08:15:00	1
29491	08:23:00	08:23:00	22
9504	08:27:00	08:32:00	26
941	08:42:00	08:45:00	26
353	08:15:00	08:35:00	7
700	08:46:00	09:02:00	20
29850	08:45:00	09:10:00	9
750	08:57:00	09:05:00	7
29400	08:40:00	09:00:00	11
942	08:50:00	09:14:00	8
454	08:31:00	09:15:00	1
141	08:54:00	09:18:00	26
176	08:56:00	08:22:00	28
9920	09:14:00	09:25:00	14
9505	09:09:00	09:14:00	20
29635	09:50:00	10:23:00	20
672	10:01:00	10:15:00	1
75	09:43:00	10:01:00	28
953	10:04:00	10:10:00	8
705	09:45:00	10:05:00	2
9924	10:00:00	10:25:00	11
19904	10:09:00	10:37:00	13
702	11:01:00	11:24:00	20
379	10:20:00	10:58:00	24
943	10:28:00	10:31:00	16
854	10:57:00	11:15:00	1
107	10:43:00	11:18:00	26
370	10:36:00	11:22:00	28
9926	10:43:00	11:25:00	11
29852	10:45:00	11:10:00	2
25900	11:21:00	11:35:00	13
9506	11:05:00	11:32:00	24
655	11:45:00	12:00:00	2
241	11:47:00	11:58:00	16
677	11:59:00	12:06:00	7
674	12:01:00	12:05:00	20
751	12:04:00	12:10:00	8
9928	11:40:00	12:25:00	11
29141	12:50:00	13:18:00	20
19906	07:32:00	13:00:00	13
955	13:04:00	13:10:00	2

Číslo vlaku	Příjez	Odjezd	Kolej
752	12:57:00	13:05:00	7
944	13:09:00	13:14:00	8
452	12:45:00	13:15:00	1
9932	12:40:00	13:25:00	11
9508	13:06:00	13:38:00	28
9931	13:40:00	14:25:00	11
659	13:45:00	15:00:00	7
679	13:59:00	14:06:00	2
676	14:01:00	14:46:00	26
29658	14:00:00	14:20:00	7
960	13:57:00	14:00:00	20
704	13:48:00	14:15:00	1
251	13:43:00	14:10:00	24
701	13:48:00	14:15:00	28
9936	14:04:00	14:25:00	11
9542	14:27:00	14:32:00	24
25904	14:10:00	14:35:00	13
855	14:45:00	15:10:00	7
29710	14:40:00	14:50:00	9
957	15:04:00	15:10:00	2
945	14:42:00	15:14:00	8
801	14:39:00	15:05:00	24
29657	15:35:00	15:52:00	9
541	15:00:00	15:18:00	26
856	14:50:00	15:15:00	1
946	14:42:00	15:14:00	8
9940	15:10:00	15:25:00	11
877	15:09:00	15:23:00	24
9510	15:31:00	15:37:00	26
800	15:47:00	16:00:00	2
678	16:01:00	16:05:00	20
656	15:57:00	16:15:00	1
623	15:45:00	16:15:00	7
9513	15:34:00	15:37:00	20
29488	16:27:00	16:27:00	24
9944	15:40:00	16:25:00	11
611	15:47:00	16:06:00	2
748	16:57:00	17:00:00	7
450	16:45:00	17:15:00	1
947	17:42:00	17:45:00	16
587	16:51:00	17:18:00	26
642	16:39:00	17:10:00	8
9948	17:10:00	17:25:00	11
9515	17:28:00	17:32:00	2
9512	17:31:00	17:35:00	8

Číslo vlaku	Příjez	Odjezd	Kolej
521	17:27:00	17:41:00	26
948	17:09:00	17:14:00	24
29171	18:27:00	18:31:00	24
653	17:45:00	18:10:00	8
683	17:59:00	18:06:00	2
954	17:57:00	18:00:00	7
622	18:01:00	18:15:00	1
879	17:06:00	18:15:00	28
833	18:14:00	18:15:00	14
9952	18:10:00	18:25:00	11
857	18:45:00	19:10:00	8
378	18:31:00	19:05:00	2
912	18:15:00	19:14:00	14
9956	19:10:00	19:25:00	11
601	18:49:00	19:18:00	26
756	18:57:00	19:15:00	1
9516	19:31:00	19:36:00	26
9517	19:28:00	19:32:00	2
673	19:45:00	20:06:00	2
658	19:57:00	20:15:00	1
29672	18:25:00	20:10:00	12
9958	19:40:00	20:25:00	11
223	19:36:00	20:28:00	7
9519	20:27:00	20:31:00	1
859	20:45:00	21:10:00	8
949	20:45:00	20:48:00	20
9518	20:54:00	21:00:00	14
858	20:50:00	21:15:00	1
201	20:34:00	21:18:00	26
9960	20:40:00	21:25:00	11
425	20:53:00	21:34:00	30
835	21:34:00	22:08:00	9
352	21:19:00	21:46:00	1
956	21:57:00	22:00:00	7
682	22:01:00	22:05:00	1
209	21:40:00	22:25:00	2
29650	21:45:00	22:10:00	8
9962	22:40:00	22:55:00	11
9521	22:59:00	23:02:00	16
1774	20:39:00	23:22:00	28
375	22:41:00	23:05:00	26
377	23:45:00	00:00:00	8
421	23:55:00	00:00:00	16

Příloha C

Ilustrativní vstupní soubor vlaků z 1.8.2006 – 4.8.2006 přijíždějících na jinou nástupištní kolej, než která byla přidělena plánem obsazení kolejí.

Datum	Číslo vlaku	Předzvěst	Příjezd	Odjezd	Kolej
2006-08-01	9401	00:01:00	00:11:00	00:17:00	24
2006-08-01	29356	00:05:00	00:10:00	00:11:00	14
2006-08-01	1650	00:09:00	00:13:00	00:16:00	1
2006-08-01	29770	04:31:00	04:36:00	04:37:00	20
2006-08-01	9904	04:22:00	04:23:00	04:56:00	2
2006-08-01	9445	04:30:00	04:30:00	04:57:00	22
2006-08-01	29500	04:59:00	05:05:00	05:07:00	2
2006-08-01	1651	05:17:00	05:21:00	05:24:00	1
2006-08-01	208	05:17:00	05:31:00	05:32:00	4
2006-08-01	29651	05:17:00	05:28:00	05:30:00	2
2006-08-01	9541	05:24:00	05:33:00	05:38:00	24
2006-08-01	671	05:49:00	06:00:00	06:06:00	7
2006-08-01	961	05:24:00	05:24:00	06:11:00	1
2006-08-01	9503	07:41:00	07:49:00	07:52:00	8
2006-08-01	29491	08:20:00	08:22:00	08:26:00	14
2006-08-01	9920	09:09:00	09:11:00	09:26:00	2
2006-08-01	75	09:40:00	09:47:00	10:02:00	26
2006-08-01	943	10:56:00	11:03:00	11:05:00	22
2006-08-01	677	11:49:00	11:58:00	12:06:00	20
2006-08-01	674	11:51:00	12:01:00	12:05:00	7
2006-08-01	955	12:54:00	13:04:00	13:11:00	16
2006-08-01	944	13:01:00	13:10:00	13:16:00	22
2006-08-01	9508	13:35:00	13:36:00	13:40:00	26
2006-08-01	679	13:42:00	13:59:00	14:16:00	16
2006-08-01	676	13:50:00	14:00:00	14:47:00	20
2006-08-01	960	14:06:00	14:10:00	14:17:00	2
2006-08-01	251	13:48:00	14:15:00	14:18:00	22
2006-08-01	701	13:44:00	13:46:00	14:22:00	26
2006-08-01	9542	14:17:00	14:29:00	14:33:00	22
2006-08-01	957	14:55:00	15:04:00	15:22:00	16
2006-08-01	945	14:59:00	15:09:00	15:11:00	12
2006-08-01	678	15:56:00	16:01:00	16:07:00	16
2006-08-01	29488	15:58:00	16:11:00	16:12:00	14
2006-08-01	9515	17:31:00	17:36:00	17:41:00	20
2006-08-01	9512	17:20:00	17:21:00	17:39:00	24
2006-08-01	948	17:39:00	17:46:00	17:47:00	16
2006-08-01	29171	17:52:00	17:58:00	18:04:00	14
2006-08-01	833	18:07:00	18:17:00	18:18:00	16
2006-08-01	912	19:22:00	19:24:00	19:25:00	16
2006-08-01	9516	19:29:00	19:37:00	19:40:00	24
2006-08-01	673	19:49:00	19:57:00	20:08:00	16
2006-08-01	29672	19:41:00	19:41:00	20:08:00	14
2006-08-01	223	19:35:00	19:35:00	20:29:00	26
2006-08-01	9518	21:00:00	21:06:00	21:09:00	24
2006-08-01	9519	20:38:00	20:46:00	20:48:00	16
2006-08-01	949	21:01:00	21:08:00	21:12:00	16
2006-08-01	835	21:48:00	21:56:00	21:57:00	16
2006-08-01	956	21:57:00	22:03:00	22:17:00	1
2006-08-01	682	22:03:00	22:10:00	22:20:00	9
2006-08-01	9521	23:12:00	23:18:00	23:20:00	8

Datum	Číslo vlaku	Předzvěst	Příjezd	Odjezd	Kolej
2006-08-02	9401	00:00:00	00:00:00	00:11:00	24
2006-08-02	1650	00:00:00	00:00:00	00:15:00	9
2006-08-02	9900	00:00:00	00:18:00	00:26:00	1
2006-08-02	29770	04:36:00	04:41:00	04:42:00	14
2006-08-02	9904	04:29:00	04:30:00	04:56:00	8
2006-08-02	9445	04:44:00	04:53:00	04:58:00	24
2006-08-02	29500	04:55:00	04:58:00	05:08:00	12
2006-08-02	1651	05:11:00	05:17:00	05:19:00	1
2006-08-02	9906	05:05:00	05:09:00	05:26:00	7
2006-08-02	29651	05:17:00	05:26:00	05:27:00	2
2006-08-02	9500	04:54:00	05:07:00	05:08:00	4
2006-08-02	71	05:39:00	05:41:00	06:03:00	26
2006-08-02	961	05:17:00	05:19:00	06:10:00	1
2006-08-02	707	05:41:00	05:43:00	06:17:00	28
2006-08-02	9501	06:52:00	06:57:00	06:59:00	24
2006-08-02	9907	06:26:00	06:34:00	06:39:00	2
2006-08-02	9909	07:02:00	07:08:00	07:10:00	9
2006-08-02	9503	07:47:00	07:53:00	07:56:00	20
2006-08-02	29491	08:43:00	08:48:00	08:51:00	2
2006-08-02	941	09:11:00	09:17:00	09:22:00	22
2006-08-02	750	09:29:00	09:34:00	09:41:00	9
2006-08-02	9920	08:59:00	09:01:00	09:37:00	11
2006-08-02	9505	09:33:00	09:38:00	09:45:00	2
2006-08-02	75	09:44:00	09:59:00	10:03:00	26
2006-08-02	379	10:29:00	10:33:00	10:39:00	22
2006-08-02	9508	13:27:00	19:29:00	13:42:00	26
2006-08-02	679	13:59:00	14:08:00	14:14:00	20
2006-08-02	676	13:54:00	13:55:00	13:59:00	22
2006-08-02	960	14:02:00	14:08:00	14:21:00	2
2006-08-02	855	14:42:00	14:49:00	14:51:00	20
2006-08-02	945	15:10:00	15:14:00	15:15:00	4
2006-08-02	9513	15:57:00	16:02:00	16:05:00	16
2006-08-02	29488	16:21:00	16:28:00	16:30:00	14
2006-08-02	947	17:22:00	17:26:00	17:29:00	2
2006-08-02	9515	17:28:00	17:35:00	17:36:00	16
2006-08-02	9512	17:17:00	17:18:00	17:36:00	20
2006-08-02	948	17:43:00	17:46:00	17:52:00	26
2006-08-02	29171	18:23:00	18:28:00	18:29:00	12
2006-08-02	833	18:15:00	18:22:00	18:23:00	16
2006-08-02	912	19:15:00	19:19:00	19:20:00	16
2006-08-02	29672	20:12:00	20:20:00	20:21:00	14
2006-08-02	223	19:42:00	19:42:00	20:29:00	24
2006-08-02	9519	20:31:00	20:41:00	20:43:00	8
2006-08-02	949	20:45:00	20:56:00	20:59:00	2
2006-08-02	9518	20:44:00	20:54:00	21:05:00	20
2006-08-02	835	21:25:00	21:33:00	21:34:00	8
2006-08-02	956	22:01:00	22:03:00	22:17:00	1
2006-08-02	209	21:42:00	21:42:00	22:27:00	22
2006-08-02	29650	21:57:00	22:02:00	22:35:00	20
2006-08-02	375	22:38:00	22:40:00	23:06:00	7

Datum	Číslo vlaku	Předzvěst	Příjezd	Odjezd	Kolej
2008-08-03	9401	00:00:00	00:00:00	00:22:00	24
2008-08-03	9900	00:13:00	00:24:00	00:25:00	9
2008-08-03	29770	04:35:00	04:41:00	04:42:00	2
2008-08-03	9904	04:18:00	04:23:00	04:56:00	7
2008-08-03	9445	04:34:00	04:34:00	04:58:00	22
2008-08-03	29500	05:06:00	05:12:00	05:13:00	12
2008-08-03	1651	05:11:00	05:20:00	05:25:00	1
2008-08-03	9906	05:16:00	05:16:00	05:27:00	7
2008-08-03	29651	05:03:00	05:04:00	05:10:00	2
2008-08-03	71	05:46:00	05:46:00	06:02:00	26
2008-08-03	671	05:50:00	05:59:00	06:06:00	7
2008-08-03	961	05:26:00	05:26:00	06:11:00	1
2008-08-03	9905	06:25:00	06:34:00	06:35:00	7
2008-08-03	707	05:37:00	05:37:00	06:16:00	28
2008-08-03	610	07:30:00	07:33:00	07:36:00	24
2008-08-03	9503	07:41:00	07:46:00	07:49:00	2
2008-08-03	222	07:57:00	08:01:00	08:03:00	22
2008-08-03	243	07:36:00	07:37:00	08:08:00	24
2008-08-03	29491	08:44:00	08:53:00	09:08:00	14
2008-08-03	700	08:42:00	08:43:00	08:45:00	26
2008-08-03	29850	08:58:00	08:58:00	09:20:00	2
2008-08-03	750	08:59:00	09:07:00	09:11:00	9
2008-08-03	29400	09:00:00	09:00:00	09:30:00	12
2008-08-03	176	09:00:00	09:02:00	09:23:00	20
2008-08-03	9920	09:18:00	09:18:00	09:26:00	11
2008-08-03	75	10:14:00	10:14:00	10:20:00	26
2008-08-03	9924	10:36:00	10:36:00	10:42:00	7
2008-08-03	379	10:30:00	10:32:00	10:33:00	22
2008-08-03	9508	13:24:00	13:28:00	13:42:00	26
2008-08-03	676	13:52:00	13:54:00	13:56:00	20
2008-08-03	960	14:11:00	14:16:00	14:23:00	7
2008-08-03	701	13:50:00	13:54:00	14:20:00	26
2008-08-03	855	14:52:00	14:58:00	15:00:00	20
2008-08-03	945	14:40:00	14:47:00	14:48:00	14
2008-08-03	678	15:58:00	16:02:00	16:08:00	16
2008-08-03	9513	15:33:00	15:37:00	15:39:00	8
2008-08-03	29488	15:57:00	16:02:00	16:09:00	14
2008-08-03	9515	17:27:00	17:31:00	17:33:00	24
2008-08-03	9512	17:34:00	17:34:00	17:37:00	24
2008-08-03	948	17:01:00	17:09:00	17:17:00	22
2008-08-03	29171	18:37:00	18:41:00	18:42:00	12
2008-08-03	833	18:28:00	18:31:00	18:32:00	16
2008-08-03	912	19:14:00	19:17:00	19:18:00	16
2008-08-03	9516	19:24:00	19:32:00	19:38:00	24
2008-08-03	9517	19:31:00	19:36:00	19:39:00	8
2008-08-03	29672	21:01:00	21:01:00	21:04:00	14
2008-08-03	223	19:36:00	20:25:00	20:29:00	22
2008-08-03	9519	20:19:00	20:25:00	20:28:00	8
2008-08-03	949	20:41:00	20:54:00	20:58:00	22
2008-08-03	9518	20:45:00	20:53:00	21:08:00	2
2008-08-03	835	21:25:00	21:28:00	21:36:00	8
2008-08-03	209	21:40:00	21:40:00	22:28:00	24
2008-08-03	29650	22:13:00	22:13:00	22:19:00	20

Datum	Číslo vlaku	Předzvěst	Příjezd	Odjezd	Kolej
2008-08-04	1650	00:00:00	00:00:00	00:17:00	2
2008-08-04	9900	00:14:00	00:20:00	00:26:00	9
2008-08-04	29770	04:32:00	04:40:00	04:41:00	22
2008-08-04	9904	04:17:00	04:18:00	04:56:00	7
2008-08-04	9445	04:32:00	04:47:00	04:57:00	24
2008-08-04	29500	05:02:00	05:08:00	05:09:00	2
2008-08-04	1651	05:11:00	05:20:00	05:21:00	1
2008-08-04	9906	05:18:00	05:18:00	05:25:00	7
2008-08-04	29651	05:16:00	05:27:00	05:28:00	2
2008-08-04	71	05:42:00	05:44:00	06:02:00	24
2008-08-04	671	05:48:00	05:59:00	06:06:00	7
2008-08-04	961	05:23:00	05:23:00	06:11:00	1
2008-08-04	9905	06:03:00	06:20:00	06:21:00	7
2008-08-04	707	05:36:00	05:38:00	06:16:00	28
2008-08-04	940	07:01:00	07:09:00	07:15:00	2
2008-08-04	9909	07:35:00	07:44:00	07:45:00	7
2008-08-04	19905	07:32:00	07:40:00	07:42:00	2
2008-08-04	9913	07:40:00	07:49:00	07:50:00	9
2008-08-04	9503	07:30:00	07:38:00	07:40:00	8
2008-08-04	29491	08:25:00	08:32:00	08:39:00	14
2008-08-04	941	08:41:00	08:45:00	08:48:00	2
2008-08-04	9920	09:12:00	09:12:00	09:35:00	11
2008-08-04	75	09:41:00	09:41:00	10:02:00	26
2008-08-04	379	10:19:00	10:21:00	10:26:00	22
2008-08-04	9508	13:01:00	13:01:00	13:38:00	26
2008-08-04	676	13:54:00	14:00:00	14:02:00	28
2008-08-04	701	13:52:00	13:52:00	14:25:00	26
2008-08-04	855	14:46:00	14:56:00	14:57:00	20
2008-08-04	9510	15:23:00	15:32:00	15:42:00	24
2008-08-04	29488	16:05:00	16:16:00	16:17:00	14
2008-08-04	9515	17:25:00	17:30:00	17:35:00	24
2008-08-04	9512	17:30:00	17:30:00	17:45:00	24
2008-08-04	29171	17:58:00	18:04:00	18:05:00	26
2008-08-04	879	18:18:00	18:18:00	18:22:00	30
2008-08-04	833	18:08:00	18:15:00	18:16:00	16
2008-08-04	912	19:10:00	19:10:00	19:15:00	16
2008-08-04	9516	19:21:00	19:30:00	19:44:00	24
2008-08-04	29672	20:34:00	20:34:00	20:35:00	14
2008-08-04	9519	20:37:00	20:39:00	20:41:00	8
2008-08-04	9518	20:45:00	20:53:00	21:11:00	24
2008-08-04	425	20:50:00	20:50:00	21:36:00	32
2008-08-04	835	22:04:00	22:12:00	22:13:00	14
2008-08-04	682	22:01:00	22:14:00	22:18:00	9
2008-08-04	9962	22:45:00	22:54:00	23:01:00	9
2008-08-04	9521	22:54:00	23:01:00	23:06:00	8

Příloha D

Výsledná fuzzy pravidla nejzajímavějších částí souboru. V levé části tabulky jsou pravidla po výběru nejčetnější z trojice a v pravé části jsou pravidla po celém průchodu třídícím algoritmem.

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
0	středně	středně	středně	blízko	brzy	blízko	brzy	vhodná
1	středně	středně	středně	blízko	brzy	blízko	dlouho	nevhodná
1	středně	středně	středně	blízko	dlouho	blízko	brzy	středně
0	středně	středně	středně	blízko	dlouho	blízko	dlouho	vhodná
0	středně	středně	středně	středně	brzy	středně	brzy	vhodná
1	středně	středně	středně	středně	brzy	středně	dlouho	středně
0	středně	středně	středně	středně	dlouho	středně	brzy	vhodná
0	středně	středně	středně	středně	dlouho	středně	dlouho	vhodná
0	středně	středně	středně	daleko	brzy	daleko	brzy	vhodná
0	středně	středně	středně	daleko	brzy	daleko	dlouho	vhodná
0	středně	středně	středně	daleko	dlouho	daleko	brzy	vhodná
0	středně	středně	středně	daleko	dlouho	daleko	dlouho	vhodná
0	středně	středně	středně	blízko	brzy	středně	brzy	vhodná
0	středně	středně	středně	středně	brzy	blízko	brzy	vhodná
0	středně	středně	středně	blízko	dlouho	středně	dlouho	vhodná
0	středně	středně	středně	středně	dlouho	blízko	dlouho	vhodná
0	středně	středně	středně	středně	brzy	blízko	dlouho	vhodná
1	středně	středně	středně	středně	dlouho	blízko	brzy	středně
0	středně	středně	středně	blízko	brzy	středně	dlouho	vhodná
1	středně	středně	středně	blízko	dlouho	středně	brzy	nevhodná
0	středně	středně	středně	blízko	brzy	daleko	brzy	vhodná
0	středně	středně	středně	daleko	brzy	blízko	brzy	vhodná
0	středně	středně	středně	blízko	dlouho	daleko	dlouho	vhodná
0	středně	středně	středně	daleko	dlouho	blízko	dlouho	vhodná
0	středně	středně	středně	blízko	brzy	daleko	dlouho	vhodná
0	středně	středně	středně	blízko	dlouho	daleko	brzy	vhodná
0	středně	středně	středně	daleko	brzy	blízko	dlouho	vhodná
0	středně	středně	středně	daleko	dlouho	blízko	brzy	vhodná
0	středně	středně	středně	daleko	brzy	středně	brzy	vhodná
0	středně	středně	středně	středně	brzy	daleko	brzy	vhodná
0	středně	středně	středně	daleko	dlouho	středně	dlouho	vhodná
0	středně	středně	středně	středně	dlouho	daleko	dlouho	vhodná
0	středně	středně	středně	daleko	brzy	středně	dlouho	vhodná
0	středně	středně	středně	daleko	dlouho	středně	brzy	vhodná
0	středně	středně	středně	středně	brzy	daleko	dlouho	vhodná
0	středně	středně	středně	středně	dlouho	daleko	brzy	vhodná

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
0	středně	středně	středně	blízko	x	blízko	x	x
0	středně	středně	středně	středně	x	středně	x	x
0	středně	středně	středně	daleko	x	daleko	x	x
0	středně	středně	středně	blízko	x	středně	x	x
0	středně	středně	středně	středně	x	blízko	x	x
0	středně	středně	středně	blízko	x	daleko	x	x
0	středně	středně	středně	daleko	x	blízko	x	x
0	středně	středně	středně	daleko	x	středně	x	x
0	středně	středně	středně	středně	x	daleko	x	x

zdroj: Výsledný CSV soubor programu GBP fuzzy

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
73	blízko	dlouho	středně	blízko	brzy	blízko	brzy	nevhodná
5	blízko	dlouho	středně	blízko	brzy	blízko	dlouho	středně
17	blízko	dlouho	středně	blízko	dlouho	blízko	brzy	středně
5	blízko	dlouho	středně	blízko	dlouho	blízko	dlouho	středně
8	blízko	dlouho	středně	středně	brzy	středně	brzy	středně
7	blízko	dlouho	středně	středně	brzy	středně	dlouho	středně
8	blízko	dlouho	středně	středně	dlouho	středně	brzy	nevhodná
4	blízko	dlouho	středně	středně	dlouho	středně	dlouho	nevhodná
0	blízko	dlouho	středně	daleko	brzy	daleko	brzy	vhodná
0	blízko	dlouho	středně	daleko	brzy	daleko	dlouho	vhodná
0	blízko	dlouho	středně	daleko	dlouho	daleko	brzy	vhodná
1	blízko	dlouho	středně	daleko	dlouho	daleko	dlouho	nevhodná
15	blízko	dlouho	středně	blízko	brzy	středně	brzy	nevhodná
20	blízko	dlouho	středně	středně	brzy	blízko	brzy	středně
2	blízko	dlouho	středně	blízko	dlouho	středně	dlouho	středně
5	blízko	dlouho	středně	středně	dlouho	blízko	dlouho	nevhodná
4	blízko	dlouho	středně	středně	brzy	blízko	dlouho	nevhodná
14	blízko	dlouho	středně	středně	dlouho	blízko	brzy	středně
4	blízko	dlouho	středně	blízko	brzy	středně	dlouho	středně
3	blízko	dlouho	středně	blízko	dlouho	středně	brzy	nevhodná
0	blízko	dlouho	středně	blízko	brzy	daleko	brzy	vhodná
4	blízko	dlouho	středně	daleko	brzy	blízko	brzy	nevhodná
0	blízko	dlouho	středně	blízko	dlouho	daleko	dlouho	vhodná
3	blízko	dlouho	středně	daleko	dlouho	blízko	dlouho	nevhodná
4	blízko	dlouho	středně	blízko	brzy	daleko	dlouho	středně
0	blízko	dlouho	středně	blízko	dlouho	daleko	brzy	vhodná
0	blízko	dlouho	středně	daleko	brzy	blízko	dlouho	vhodná
2	blízko	dlouho	středně	daleko	dlouho	blízko	brzy	nevhodná
1	blízko	dlouho	středně	daleko	brzy	středně	brzy	nevhodná
4	blízko	dlouho	středně	středně	brzy	daleko	brzy	nevhodná
1	blízko	dlouho	středně	daleko	dlouho	středně	dlouho	středně
1	blízko	dlouho	středně	středně	dlouho	daleko	dlouho	nevhodná
0	blízko	dlouho	středně	daleko	brzy	středně	dlouho	vhodná
7	blízko	dlouho	středně	daleko	dlouho	středně	brzy	nevhodná
1	blízko	dlouho	středně	středně	brzy	daleko	dlouho	nevhodná
0	blízko	dlouho	středně	středně	dlouho	daleko	brzy	vhodná
100%								

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
73	blízko	dlouho	středně	blízko	brzy	blízko	brzy	nevhodná
5	blízko	dlouho	středně	blízko	brzy	blízko	dlouho	středně
17	blízko	dlouho	středně	blízko	dlouho	blízko	brzy	středně
5	blízko	dlouho	středně	blízko	dlouho	blízko	dlouho	středně
8	blízko	dlouho	středně	středně	brzy	středně	brzy	středně
0	blízko	dlouho	středně	středně	brzy	středně	dlouho	nevhodná
8	blízko	dlouho	středně	středně	dlouho	středně	brzy	nevhodná
4	blízko	dlouho	středně	středně	dlouho	středně	dlouho	nevhodná
1	blízko	dlouho	středně	daleko	x	daleko	x	vhodná
0	blízko	dlouho	středně	blízko	brzy	středně	brzy	středně
20	blízko	dlouho	středně	středně	brzy	blízko	brzy	středně
0	blízko	dlouho	středně	blízko	dlouho	středně	dlouho	nevhodná
5	blízko	dlouho	středně	středně	dlouho	blízko	dlouho	nevhodná
0	blízko	dlouho	středně	středně	brzy	blízko	dlouho	středně
14	blízko	dlouho	středně	středně	dlouho	blízko	brzy	středně
4	blízko	dlouho	středně	blízko	brzy	středně	dlouho	středně
0	blízko	dlouho	středně	blízko	dlouho	středně	brzy	středně
0	blízko	dlouho	středně	blízko	x	daleko	x	nevhodná
9	blízko	dlouho	středně	daleko	x	blízko	x	nevhodná
1	blízko	dlouho	středně	daleko	brzy	středně	brzy	nevhodná
4	blízko	dlouho	středně	středně	brzy	daleko	brzy	nevhodná
1	blízko	dlouho	středně	daleko	dlouho	středně	dlouho	středně
0	blízko	dlouho	středně	středně	dlouho	daleko	dlouho	středně
0	blízko	dlouho	středně	daleko	brzy	středně	dlouho	nevhodná
7	blízko	dlouho	středně	daleko	dlouho	středně	brzy	nevhodná
1	blízko	dlouho	středně	středně	brzy	daleko	dlouho	nevhodná
0	blízko	dlouho	středně	středně	dlouho	daleko	brzy	nevhodná
Zohledněno		83.856502242152%						

zdroj: Výsledný CSV soubor programu GBP fuzzy

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
548	blízko	dlouho	dlouho	blízko	brzy	blízko	brzy	středně
25	blízko	dlouho	dlouho	blízko	brzy	blízko	dlouho	vhodná
165	blízko	dlouho	dlouho	blízko	dlouho	blízko	brzy	vhodná
21	blízko	dlouho	dlouho	blízko	dlouho	blízko	dlouho	vhodná
36	blízko	dlouho	dlouho	středně	brzy	středně	brzy	středně
42	blízko	dlouho	dlouho	středně	brzy	středně	dlouho	středně
103	blízko	dlouho	dlouho	středně	dlouho	středně	brzy	středně
56	blízko	dlouho	dlouho	středně	dlouho	středně	dlouho	středně
0	blízko	dlouho	dlouho	daleko	brzy	daleko	brzy	vhodná
6	blízko	dlouho	dlouho	daleko	brzy	daleko	dlouho	středně
2	blízko	dlouho	dlouho	daleko	dlouho	daleko	brzy	středně
3	blízko	dlouho	dlouho	daleko	dlouho	daleko	dlouho	středně
39	blízko	dlouho	dlouho	blízko	brzy	středně	brzy	středně
187	blízko	dlouho	dlouho	středně	brzy	blízko	brzy	středně
14	blízko	dlouho	dlouho	blízko	dlouho	středně	dlouho	vhodná
35	blízko	dlouho	dlouho	středně	dlouho	blízko	dlouho	vhodná
35	blízko	dlouho	dlouho	středně	brzy	blízko	dlouho	středně
192	blízko	dlouho	dlouho	středně	dlouho	blízko	brzy	středně
36	blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	vhodná
19	blízko	dlouho	dlouho	blízko	dlouho	středně	brzy	středně
0	blízko	dlouho	dlouho	blízko	brzy	daleko	brzy	vhodná
45	blízko	dlouho	dlouho	daleko	brzy	blízko	brzy	středně
5	blízko	dlouho	dlouho	blízko	dlouho	daleko	dlouho	středně
2	blízko	dlouho	dlouho	daleko	dlouho	blízko	dlouho	vhodná
8	blízko	dlouho	dlouho	blízko	brzy	daleko	dlouho	vhodná
0	blízko	dlouho	dlouho	blízko	dlouho	daleko	brzy	vhodná
2	blízko	dlouho	dlouho	daleko	brzy	blízko	dlouho	středně
37	blízko	dlouho	dlouho	daleko	dlouho	blízko	brzy	středně
12	blízko	dlouho	dlouho	daleko	brzy	středně	brzy	středně
25	blízko	dlouho	dlouho	středně	brzy	daleko	brzy	středně
20	blízko	dlouho	dlouho	daleko	dlouho	středně	dlouho	středně
4	blízko	dlouho	dlouho	středně	dlouho	daleko	dlouho	středně
10	blízko	dlouho	dlouho	daleko	brzy	středně	dlouho	středně
24	blízko	dlouho	dlouho	daleko	dlouho	středně	brzy	středně
4	blízko	dlouho	dlouho	středně	brzy	daleko	dlouho	středně
0	blízko	dlouho	dlouho	středně	dlouho	daleko	brzy	vhodná
100%								

četnost	blízko	volná před	volná dlouho	přípoj 1 kolej	přípoj 1 odjezd	přípoj 2 kolej	přípoj 2 odjezd	vhodnost
548	blízko	dlouho	dlouho	blízko	brzy	blízko	brzy	středně
25	blízko	dlouho	dlouho	blízko	brzy	blízko	dlouho	vhodná
165	blízko	dlouho	dlouho	blízko	dlouho	blízko	brzy	vhodná
21	blízko	dlouho	dlouho	blízko	dlouho	blízko	dlouho	vhodná
237	blízko	dlouho	dlouho	středně	x	středně	x	středně
11	blízko	dlouho	dlouho	daleko	x	daleko	x	středně
39	blízko	dlouho	dlouho	blízko	brzy	středně	brzy	středně
187	blízko	dlouho	dlouho	středně	brzy	blízko	brzy	středně
14	blízko	dlouho	dlouho	blízko	dlouho	středně	dlouho	vhodná
35	blízko	dlouho	dlouho	středně	dlouho	blízko	dlouho	vhodná
35	blízko	dlouho	dlouho	středně	brzy	blízko	dlouho	středně
192	blízko	dlouho	dlouho	středně	dlouho	blízko	brzy	středně
0	blízko	dlouho	dlouho	blízko	brzy	středně	dlouho	středně
19	blízko	dlouho	dlouho	blízko	dlouho	středně	brzy	středně
5	blízko	dlouho	dlouho	blízko	x	daleko	x	středně
84	blízko	dlouho	dlouho	daleko	x	blízko	x	středně
66	blízko	dlouho	dlouho	daleko	x	středně	x	středně
33	blízko	dlouho	dlouho	středně	x	daleko	x	středně
Zohledněno		97.38933030647%						

zdroj: Výsledný CSV soubor programu GBP fuzzy