Univerzita Pardubice

Dopravní fakulta Jana Pernera

Implementace optimalizačních úloh v jazyku C#

Daniel Krolop

Bakalářská práce

2013

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Daniel Krolop**
Osobní číslo:	**D09012**
Studijní program:	**B3709 Dopravní technologie a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Implementace optimalizačních úloh teorie grafů v jazyku C#**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

Cílem bakalářské práce je počítačová implementace vybraných úloh diskrétní optimalizace na dopravních sítích. Práce bude vycházet z prohlubujícího studia metod a algoritmů teorie grafů v oblasti významných cest na grafech a konstrukčních úloh na grafech typu eulerovské tahy a hamiltonovské kružnice. Tyto budou potom využity při návrhu algoritmů řešení dopravní obsluhy vybraného regionu ČR. Práce bude obsahovat kapitoly věnované teoretickému popisu metod, metodám zpracování dat a ideovému návrhu vlastního postupu řešení vybraných optimalizačních úloh dopravní obsluhy. Kromě toho budou v práci podrobně popsány navržené algoritmy, potřebné datové struktury, včetně programové implementace v C#. Funkčnost navržených algoritmů bude ověřena na konkrétních příkladech. Práce bude v rozsahu 40-60 stran včetně obrázků, tabulek a klíčových úseků programové dokumentace. Výsledky řešení budou uvedeny v příloze. Práce bude obsahovat i kapitolu zhodnocení úspor dopravní práce a časovou náročnost řešení. Přílohou práce bude CD s funkčním programem.

Rozsah grafických prací:

Rozsah pracovní zprávy: **minimálně 40 normostran**

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

Seznam odborné literatury:

1. **VOLEK, Josef. Operační výzkum I. Pardubice : Univerzita Pardubice, 2002. 111 s. ISBN 80-7194-410-6.**
2. **SEDLÁČEK, Jiří. Kombinatorika v teorii i praxi : úvod do teorie grafů. Praha : Československá akademie věd, 1964. 150 s. 000574264.**
3. **CHRISTOFIDES, Nicos. Graph theory : An algorithmic approach. Orlando, FL, USA : Academic Press, Inc., 1975. 415 s. ISBN 0121743500.**

Vedoucí bakalářské práce: **doc. Ing. Josef Volek, CSc.**
Katedra informatiky v dopravě

Datum zadání bakalářské práce: **6. prosince 2012**

Termín odevzdání bakalářské práce: **31. května 2013**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 6. prosince 2012

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 30. 5. 2013

Daniel Krolop

Poděkování

Tímto rád poděkoval doc. Ing. Josefu Volkovi, CSc. za cenné rady při vytváření této práce a také celé katedře AID za vstřícný přístup.

ANOTACE

Teoretická část bakalářské práce se zaobírá potřebnými základními pojmy teorie grafů, reprezentací grafů a algoritmy sloužící k racionalizaci dopravní obsluhy území (Hamiltonovské kružnice, Eulerovské tahy, přiřazovací úlohy, lokační a alokační úlohy). Další část popisuje fungování zdravotnické záchranné služby se zasazením do kontextu integrovaného záchranného systému a její konkrétní podobu v Pardubickém kraji z hlediska řešení lokačně-alokační úlohy. Dále je popsána samotná implementace úlohy včetně popisu použitých datových struktur a zhodnocení složitosti algoritmu.

Implementační část práce je věnována vytvoření editoru schopného sestrojit grafickou podobu grafu. Editor umožňuje nastavovat základní vlastnosti grafu a řešit lokačně-alokační úlohu dle zadaných parametrů.

KLÍČOVÁ SLOVA

graf, editor, teorie grafů, zdravotnická záchranná služba, lokační úloha, alokační úloha

TITLE

Implementation of optimization problems of graph theory in C# language.

ANNOTATION

Theoretical part of this bachelor work is focused on basic terms of graph theory, graph representation and alghoritms for design of Hamiltonian circles and Euler's paths. Next part describes implementation of task itself. The next part describes the functioning of the emergency medical service in the context of the integrated rescue system and its concrete form in the Region of Pardubice in terms of solving location-allocation-task.

Implementation part of work is dedicated to creation of editor capable of constructing graphical representation of graph. Editor allows setting up basic properties of the graph and solve location-allocation problem with given parameters.

KEYWORDS

graph, editor, graph theory, location task, allocation task, emergency medical service

Obsah

Seznam obrázků a tabulek

Obrázky

Tabulky

Použité zkratky

ZZS	zdravotnická záchranná služba
IZS	integrovaný záchranný systém
PNP	přednemocniční neodkladná péče
EKG	elektrokardiogram
RLP	rychlá lékařská pomoc
RZP	rychlá zdravotnická pomoc
RV	systém Rendez-Vous
SUV	sport utility vehicle
LZS	letecká záchranná služba
KZOS	krajské zdravotnické operační středisko
GPS	global positioning system
UML	unified modeling language

1 Úvod

Cílem této bakalářské práce je vytvoření editoru úloh teorie grafů, plně schopného definovat a řešit diskrétní lokační úlohu. Jako dostačující bude editor schopný pracovat s obyčejným grafem, tedy grafem bez smyček a násobných hran. Důležitá vlastnost editoru je schopnost vložit na pozadí kreslícího plátna libovolný rastrový obrázek, podle něhož můžeme zadat graf silniční sítě. Jako praktický příklad lokační úlohy byla zvolena optimalizace umístění výjezdových středisek zdravotnické záchranné služby v Pardubickém kraji. Tento příklad jsem si vybral, protože se, dle mého názoru, týká velice zajímavého a komplexního tématu racionalizace fungování zdravotnické záchranné služby. Toto téma je navíc v oblasti veřejného zájmu, protože je zdravotnická záchranná služba financována státem (tedy z peněz občanů) a především na jejím správném fungování závisí životy pacientů.

Teoretickou částí práce zaměřené na teorii grafů zavádím pojmy, se kterými budeme v práci pracovat. Dále uvádím jednotlivé typy grafů a stručně jejich možnou reprezentaci. Další část kapitoly teorie grafů se zabývá lokační úlohou. Zavádí pojmy potřebné k pochopení fungování iterativního algoritmu pro určení vrcholově optimální lokace *k* dep na síti. Tento algoritmus využívá záměnné heuristiky a byl vybrán pro svoji implementační jednoduchost. V editoru je také potřeba najít způsob jak hledat minimální cestu mezi vrcholy grafu. K tomuto slouží Dijkstrův a Floydoův-Warshallův algoritmus, každý se svým specifickým účelem použití.

S praktickým příkladem souvisí kapitola zaměřená na vysvětlení základní problematiky zdravotnické záchranné služby. V této části popisujeme účel zdravotnické záchranné služby, jaké používá vozidla a také koncepci operačních středisek, výjezdových stanovišť a různých typů výjezdových skupin. Dále se zaměřujeme na fungování zdravotnické záchranné služby v Pardubickém kraji a způsobu získání dat potřebných pro výpočet praktického příkladu.

Program vytvořený jako součást této práce je popsán v další kapitole a to pomocí UML diagramů vytvořených v programu Enterprise Architect. Možnosti uživatele při práci s programem modelují diagramy případů užití. Samotná implementace je popsána po jednotlivých třídách, vždy s přiloženým UML diagramem. Tento popis obsahuje chování nejdůležitějších metod, způsob jejich implementace a chování v kontextu třídy.

V závěrečné části práce popisuji sestavení praktického příkladu a analýzu jeho řešení. Dále hodnotím implementaci použitého lokačního algoritmu z hlediska časové složitosti a analyzuji limity jeho použitelnosti.

Jako programovací jazyk byl zvolen C#. Důvod této volby je má obeznámenost s tímto programovacím jazykem a vývojovým prostředím Microsoft Visual Studio 2010, na které vlastním studentskou licenci poskytnutou Univerzitou Pardubice.

2 Pojmy teorie grafů

Práce předpokládá od čtenáře základní znalost teorie grafů, proto se zde uvádí a definují pouze pojmy, se kterými se bude v práci přímo operovat.

2.1 Základní pojmy

Množina vrcholů
Jako V označujeme množinu všech vrcholů grafu G.

Množina hran
Symbolem X rozumíme množinu všech hran grafu G.

Incidence
Incidence p přiřazuje každé hraně, v případě neorientovaného grafu (viz kapitola Typy grafů), neuspořádanou dvojici vrcholů. Pro hranu $h \in X$, která je ohraničená vrcholy grafu $u, v \in$ V tedy platí: p(h) = (u, v). Tento vztah říká, že hrana h inciduje s vrcholem u a v. Vrcholy u a v budeme nazývat krajními vrcholy hrany h a zároveň můžeme tyto vrcholy označit jako sousedící. Také hrany, které spolu sdílejí krajní bod označujeme jako sousedící.

Smyčka
Smyčka je hrana $h \in X$, pro kterou platí: p(h) = (u, u). Smyčkou je tedy hrana vedoucí z vrcholu u zpět do vrcholu u.

Násobná hrana
Pokud v grafu G existuje hrana $h \in X$ s krajními body $u,v \in V$ a ve stejném grafu G existuje i další hrana $h_i \in X$ pro i = 1,..., n (kde n je počet hran v množině X) se stejnými krajními body u a v, hovoříme o těchto hranách jako o hranách násobných. Platí tedy: p(h) = p(h_i) = (u, v). Tyto hrany se také často označují jako rovnoběžné, či jako multihrany.

Izolovaný vrchol
Izolovaných vrchol $v \in$ V je takový vrchol, se kterým neinciduje žádná hrana $h \in X$.

Stupeň vrcholu
Stupněm vrcholu st(v) rozumíme počet hran incidujících s vrcholem $v \in V$.

Ohodnocení
Ohodnocení přiřazuje hraně či vrcholu nezáporné číslo vyjadřující určitou vlastnost. Značíme ho jako o(h) resp. o(v). Ohodnocení hrany typicky představuje délku této hrany ať už v bezrozměrné jednotce nebo například v kilometrech, pokud graf představuje silniční síť. Ohodnocení nemusí vždy představovat délkovou jednotku, ale může vyjadřovat i pravděpodobnost úspěšného průchodu hranou či čas potřebný pro průchod

hranou. Ohodnocení vrcholů má kupříkladu význam v lokační analýze, kde představuje počet požadavků na obsluhu dané sítě za určité časové údobí.

Sled

Sledem S mezi dvojicí vrcholů $u, v \in V$ nazýváme takovou střídavou posloupnost vrcholů a hran grafu $S = \{u_0, h_1, u_1, h_2, u_2,\ldots,u_{n-1}, h_n, u_n\}$, pro kterou platí:

$$h_i \in X, p(h_i) = (u_{i-1}, u_i) \text{ pro } i = 1,\ldots, n,$$

$$u_i \in V \text{ pro } i = 1,\ldots, n,$$

$$u_0 = u, u_n = v,$$

Vrcholy u a v považujeme za krajní vrcholy sledu nebo počáteční a koncový vrchol, ostatní vrcholy označujeme jako vnitřní. Proměnná n představuje délku sledu. Pokud se krajní vrcholy u_0 a u_n rovnají, hovoříme o uzavřeném sledu neboli kružnici. Tato terminologie je použita analogicky u tahu a cesty.

Tah

Tah je takový sled, ve kterém se neopakuje žádná z hran grafu.

Cesta

Cesta mezi vrcholy $u, v \in V$ se značí $m(u,v)$ a definujeme jí jako tah, ve kterém se neopakuje žádný vrchol.

Délka cesty

Délka cesty v hranově ohodnoceném grafu mezi vrcholy $u, v \in V$ je definována jako:

$$\left|m(u,v)\right| = d(m(u,v)) = \sum_{h \in m(u,v)} o(h)$$

Vzdálenost vrcholů

Vzdálenost mezi dvěma vrcholy $u, v \in V$ v hranově ohodnoceném grafu G je definována tímto vztahem:

$$d(u,v) = \min_{m(u,v) \in M} \left\{ \sum_{h \in m(u,v)} o(h) \right\}$$

2.2 Typy grafů

Souvislý graf

Graf G označujeme jako souvislý, pokud mezi libovolnou dvojicí vrcholů $u, v \in V$ existuje alespoň jedna cesta $m(u,v)$. Graf tvořený pouze jedním izolovaným vrcholem považujeme za souvislý.

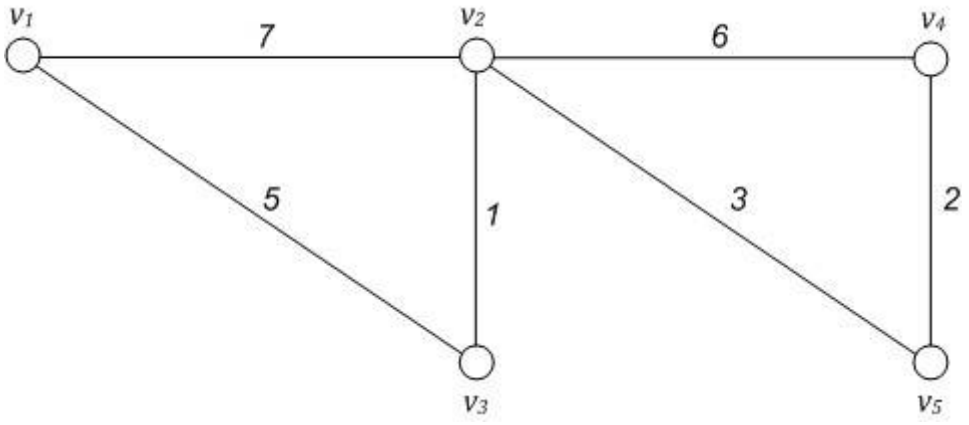
Neorientovaný graf
Neorientovaný graf G definujeme jako upořádanou trojici $G = (V, X, p)$. Z incidence p neorientovaného grafu G vyplývá, že hrany nemají žádnou specifickou orientaci a tudíž je můžeme, na rozdíl od hran orientovaného grafu, procházet libovolně v obou směrech.

Obyčejný graf
Obyčejný graf je takový graf G, který neobsahuje smyčky ani násobné hrany.

Prostý graf
Prostým grafem se nazývá graf G, který neobsahuje násobné hrany a zároveň obsahuje smyčku či smyčky.

Multigraf
O grafu G hovoříme jako o multigrafu, pokud obsahuje násobné hrany. Multigraf může také obsahovat jednu nebo více smyček.

Ohodnocený graf
Pokud má každá hrana $h \in X$ grafu G ohodnocení $o(h)$, označujeme tento graf G jako hranově ohodnocený. Analogicky definujeme vrcholově ohodnocený graf. Neohodnocený graf G si můžeme snadno představit jako hranově ohodnocený, pokud každé jeho hraně $h \in X$ přiřadíme shodné ohodnocení, tedy například $o(h) = 1$.



Obrázek 1 - hranově ohodnocený, souvislý, neorientovaný, obyčejný graf

2.3 Reprezentace grafu

2.3.1 Maticová reprezentace

U následujících maticových reprezentací považujeme graf G za hranově ohodnocený, neorientovaný, obyčejný graf.

Matice sousednosti
Matice sousednosti je čtvercová matice grafu G, která má rozměr n, kde n je počet vrcholů grafu G. Každý řádek a sloupec matice odpovídá jednomu vrcholu $v \in V$ grafu G.

Její prvky p mohou nabývat dvou hodnot:

- $p_{ij} = 0$, pokud spolu vrcholy v_i a v_j nesousedí
- $p_{ij} = 1$, pokud spolu vrcholy v_i a v_j sousedí
    - prvek p_{ij} nacházející se na diagonále může mít také hodnotu 1, pokud je na příslušném vrcholu smyčka.

Z této matice lze snadno zjistit, zda mezi vrcholy v_i a v_j existuje hrana. Dále lze vypočítat stupeň libovolného vrcholu, podle vztahů:

$$st(v_i) = \sum_{j=1}^{n} p_{ij} \text{, kde } j = 1, ..., n$$

$$st(v_j) = \sum_{i=1}^{n} p_{ij} \text{, kde } i = 1, ..., n$$

Matice incidence

Matice sousednosti grafu G je matice, kde každý řádek odpovídá jednomu vrcholu $v \in V$ a každý sloupec jedné hraně $h \in X$ grafu G.

Prvky p matice incidence mohou mít tyto hodnoty:

- $p_{ij} = 0$, pokud spolu vrcholy v_i a v_j nesousedí
- $p_{ij} = 1$, pokud spolu vrcholy v_i a v_j sousedí

Z matice incidence můžeme zjistit, stejně jako u matice sousednosti, mezi kterými vrcholy existuje hrana a vypočítat stupeň libovolného vrcholu. Výhodou matice incidence je, že u orientovaných grafů snadno odvodíme orientaci hrany. Jistou nevýhodou je nemožnost zápisu smyček.

Matice přímých vzdáleností

Matice přímých vzdáleností hranově ohodnoceného grafu G je čtvercová matice o rozměru n, kde n je počet vrcholů grafu G. Každý řádek a sloupec matice odpovídá jednomu vrcholu $v \in V$ grafu G. Matice přímých vzdáleností je u neorientovaného grafu symetrická podle hlavní diagonály.

Hodnoty prvků p matice se řídí těmito vztahy:

- $p_{ij} = o(h)$, pokud mezi vrcholy v_i, v_j existuje hrana $h \in X$
- $p_{ij} = \infty$, pokud mezi vrcholy v_i, v_j neexistuje hrana $h \in X$
- $p_{ij} = 0$, pokud se prvek nachází na hlavní diagonále ($i = j$)

Z matice již snadno vyčteme mezi kterými vrcholy je hrana a jaké má tato hrana ohodnocení. Z této matice můžeme také jednoznačně určit a sestavit jakýkoli hranově ohodnocený, obyčejný graf.

0	7	5	∞	∞
7	0	1	6	3
5	1	0	∞	∞
∞	6	∞	0	2
∞	3	∞	2	0

Obrázek 2 - matice přímých vzdáleností grafu na obrázku: Obrázek 1

2.3.2 Datová struktura

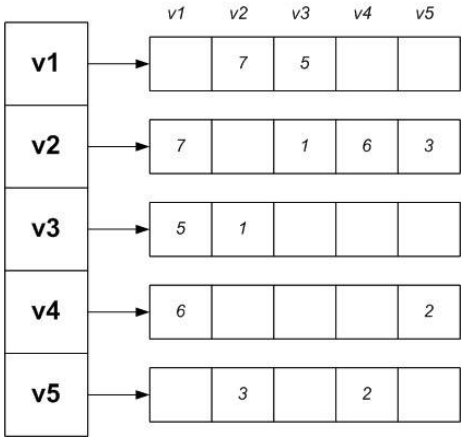
Vrcholově orientovaná - seznam sousedů

Vstupním prvkem struktury je vrchol, ke kterému evidujeme informace o jeho sousedních vrcholech a incidentních hranách.

Jedna z možných implementací je hvězda. Hvězda si v primární struktuře uchovává vrcholy a v druhotné struktuře relace těchto vrcholů s ostatními vrcholy grafu. Možné volby primárních - sekundárních struktur:

- pole - pole
- pole - tabulka
- tabulka - pole
- tabulka - tabulka

Na obrázku č. 3 vidíme příklad reprezentace typu pole - pole. Princip ostatních typů je analogický, přebírají pouze vlastnosti implementovaných datových struktur.



Obrázek 3 - příklad hvězdy typu pole-pole představující graf na obrázku: Obrázek 1 - hranově ohodnocený, souvislý, neorientovaný, obyčejný graf

Další implementací je křížová reprezentace. Ta spočívá v zavedení dvou primárních struktur, kdy každá slouží pro průchod grafu jedním směrem.

- pole - tabulka x pole - tabulka
- tabulka - tabulka x tabulka - tabulka

Hranově orientovaná
Vstupním prvek struktury je hrana, vrcholy jsou brány až jako sekundární informace. Využívá se pro rychlé hledání v hranách.

Možné implementace jsou:

- tabulka hran - ukládáme dvojici incidentních vrcholů
- pole hran - ukládáme hrany s unikátním klíčem a seznamem sousednosti

2.4 Lokačně-alokační problém

2.4.1 Pojmy

Depo
Depem rozumíme místo na síti, kde je umístěno středisko obsluhy

Množina dep
Množinu dep označíme D_k, počet dep značíme - k = | D_k |. Pro k platí: $1 \leq k \leq p$, kde p = |V|.

Atrakční obvod
Atrakčním obvodem depa $v \in D_k$ v širším smyslu slova rozumíme množinu vrcholů $u \in V$ a hran $h \in X$ sítě které jsou obsluhovány z depa v.

Atrakčním obvodem - označeným A(v) depa $v \in D_k$ v užším smyslu slova rozumíme množinu vrcholů $u \in V$ a hran $h \in X$ sítě, pro které platí:

$u \in A(v)$, pokud neexistuje depo $w \in D_k$, pro které $d(w, u) < d(v, u)$,

$h \in A(v)$, pokud neexistuje depo $w \in D_k$, pro které $d(w, h) < d(v, h)$.

Vzdálenost vrcholu od depa
Vzdálenost vrcholu $u \in V$ od depa $v \in D_k$ je definována jako délka minimální cesty

$$d(u, v) = \min_{m(u,v) \in M} \left\{ \sum_{h \in m(u,v)} o(h) \right\}$$, *kde M je množina všech cest mezi u a v.*

Prvotní atrakční obvod
Prvotním atrakčním obvodem – označeným A/(v) depa $v \in D_k$ rozumíme množinu vrcholů $u \in V$ a hran $h \in X$ sítě, pro které platí:

$u \in A/(v)$, pokud neexistuje depo $w \in D_k$, pro které $d(w, u) \leq d(v, u)$,

$h \in A^{/}(v)$, pokud neexistuje depo $w \in D_k$, pro které $d(w, h) \leq d(v, h)$.

Přidělený atrakční obvod

Přiděleným atrakčním obvodem - označeným A(v) depa $v \in D_k$ rozumíme množinu vrcholů $u \in V$ a hran $h \in X$ sítě splňující následující vztahy:*

$$A^{/}(v) \subseteq A^*(v) \subseteq A(v) \text{ pro každé depo } v \in D_k$$

$$\bigcup_{v \in D_k} A^*(v) = X \cup V,$$

$$A^*(v) \cap A^*(u) = \varnothing \text{ pro } u \neq v;\ u, v \in D_k.$$

Vrcholově optimální umístění

Množinu dep – D_k ($|D_k| = k$) nazveme vrcholově optimálním umístěním k dep na síti G = (V, X), když pro ni platí:

$$f(D_k) = \min_{D_k^{/}} \left\{ f(D_k^{/}) \right\},$$

$$\text{kde } f(D_k^{/}) = \sum_{v \in D_k^{/}} \sum_{u \in A^*(v)} 2 \times d(u, v) \times w(u).$$

$D_k^{/}$ jsou všechny k - prvkové podmnožiny V.[1]

Množina cest

Množina cest *M* mezi vrcholy *u*, *v* ∈ *V*, jest množinou obsahující všechny možné cesty m(*u*,*v*) mezi těmito vrcholy.

Dopravní práce depa

Dopravní práci atrakčního obvodu depa vypočteme podle vztahu:

$$\sum_{u \in A^*(v)} 2 \times d(u, v) \times w(u)$$

Kde *v* je vrchol ve kterém je umístěno depo a *w*(*u*) znázorňuje ohodnocení vrcholu *u*.

2.5 Použité optimalizační algoritmy

2.5.1 Dijkstrův algoritmus

Dijkstrův algoritmus hledá nejkratší cestu mezi zadaným uzlem a všemi ostatními uzly na grafu s nezáporným ohodnocením hran. Nejčastější použití tohoto algoritmu je vyhledání a rekonstrukce nejkratší cesty mezi dvěma zadanými vrcholy grafu. Dijkstrův algoritmus má relativně malou časovou složitost, která může být dále snížena použitím

[1] VOLEK, Josef; LINDA, Bohdan. Teorie grafů - aplikace v dopravě a veřejné správě. Dopravní fakulta Jana Pernera ; Pardubice : Univerzita Pardubice ; 2012. 192 s. ISBN 978-80-7395-225-9

specializovaných datových struktur při implementaci použité prioritní fronty. Princip algoritmu spočívá v postupném průchodu grafu (podobně jako při prohledávání do hloubky), kdy postupně zpřesňujeme hodnotu minimální cesty každého vrcholu od vrcholu počátečního.

Složitost Dijkstrova algoritmu je v našem případě $O(|V|^2 + |X|)$. Složitost se dá dále snižovat vhodnou implementací prioritní fronty.

Postup algoritmu

Pro vysvětlení postupu algoritmu použijeme pseudokód obsahující následující proměnné:

- jako vrchol v_p označujeme počáteční vrchol hledané minimální cesty
- T je množina definitivně ohodnocených vrcholů grafu (tedy vrcholů, do kterých již známe minimální cestu).
- hodnota *delka*[*v*] udává délku minimální cesty z počátečního vrcholu do vrcholu $v \in V$ vedoucí pouze přes vrcholy patřící do T
- pole *odkud*[*v*] udává pro vrchol $v \in V$, vrchol předcházející tomuto vrcholu v minimální cestě z počátečního vrcholu do vrcholu *v*
- vrcholem *u* rozumíme sousedící vrchol vrcholu *v*. Platí tedy: $p(h) = (u, v)$, pro $h \in X$
- P představuje prioritní frontu

$T := \varnothing$

delka[v_p] := 0 a $\forall\, v \in V \setminus \{ v_p \}$: delka[*v*] := ∞

$\forall\ v \in V$: odkud[*v*] := nil

vytvoř prioritní frontu P z vrcholů množiny V, s prioritami delka[*v*]

while fronta P není prázdná **do**

    v := Odeber_Min(P) {vyber netrvalý vrchol s nejmenším delka[v]}

    $T := T \cup \{v\}$

    for each $uv \in X$ **do**

        if delka[*u*] > delka[*v*] + d(*uv*) **then**

            delka[*u*] := delka[*v*] + d(*uv*)

            odkud[*u*] := *v*

            Vloz_na_prvni(P, *u*) {aktualizujeme frontu}[2]

Pro rekonstrukci cesty máme k dispozici výčet vrcholů odkud[].

[2] Pseudokód je přepsán do terminologie zavedené v této práci z publikace **Základní grafové algoritmy**, volně dostupné na http://kam.mff.cuni.cz/~kuba/ka/

2.5.2 Floydův-Warshallův algoritmus

Algoritmus spočívá v postupné úpravě matice přímých vzdáleností (viz kapitola 2.3.1) v distanční matici. Je také možné pomocí distanční matice a matice přímých vzdálenosti zrekonstruovat cestu mezi požadovanými vrcholy. Tento algoritmus upřednostňujeme před Dijkstrovým, pokud potřebujeme znát vzdálenost mezi všemi vrcholy, protože Floydův algoritmus proběhne rychleji než Dijkstrův algoritmus použitý na všechny vrcholy grafu. Jistou výhodou tohoto algoritmu je i výrazně snazší implementace oproti Dijkstrovu algoritmu.

Složitost algoritmu je $O(|V|^3)$.

Distanční matice

Distanční matice hranově ohodnoceného grafu G je čtvercová matice o rozměru *n*, kde *n* je počet vrcholů grafu G. Každý řádek a sloupec matice odpovídá jednomu vrcholu $v \in V$ grafu *G*. Tato matice nám umožňuje získat délku minimální cesty mezi libovolnými vrcholy grafu G.

Pro hledanou vzdálenost vrcholů stačí nalézt odpovídající prvek *p* distanční matice podle vztahu $d(v_i, v_j) = p_{ij}$.

0	6	5	11	9
6	0	1	5	3
5	1	0	6	4
11	5	6	0	2
9	3	4	2	0

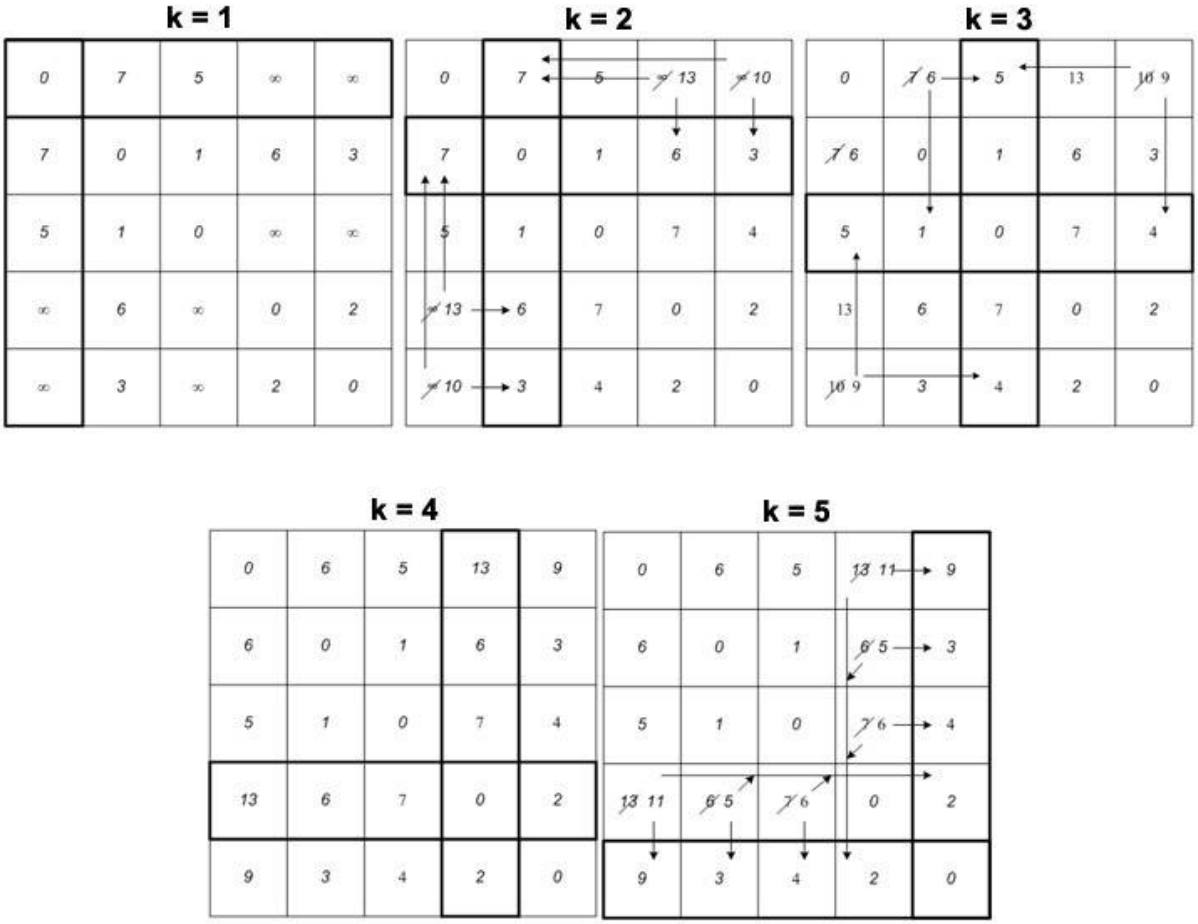
Obrázek 4 - distanční matice grafu na obrázku: Obrázek 1

Postup algoritmu

Algoritmus probíhá v *k* iteracích, kde *k* je počet vrcholů nebo rozměr matice vzdáleností, chcete-li. V každé iteraci *k*, označíme *k*-tý řádek a *k*-tý sloupec matice (jak vidíme na obrázku: Obrázek 5 - postup Floydova-Warshallova algoritmu na grafu z obrázku: Obrázek 1). Po označení řádku a sloupce procházíme každý prvek matice, kromě prvků na hlavní diagonále a prvků v označeném řádku resp. sloupci. Procházený prvek si označíme jako d_{ij}, kde *i* je řádek matice a *j* sloupec matice. Prvek d_{ik} je prvek v označeném sloupci, který je na stejném řádku jako procházený prvek. Podobně je d_{kj} prvkem v označeném řádku a ve stejném sloupci jako procházený prvek. Nyní pro každý procházený prvek provedeme:

Pokud platí, že: $d_{ij} > \mathrm{d}_{\mathrm{ik}} + d_{kj}$, položíme $d_{ij} = \mathrm{d}_{\mathrm{ik}} + d_{kj}$

Ve chvíli kdy projdeme všechny přípustné prvky, položíme k = k + 1 (přejdeme do další iterace) a celý proces opakujeme. Po dokončení všech iterací máme z matice přímých vzdáleností distanční matici.



Obrázek 5 - postup Floydova-Warshallova algoritmu na grafu z obrázku: Obrázek 1

2.5.3 Iterativní algoritmus pro určení vrcholově optimální lokace *k* dep na síti

Vzhledem k povaze hledání řešení lokační úlohy, je nutné tuto úlohu řešit pomocí heuristických metod. Důvodem je obrovský počet přípustných kombinací, daných například velkým počtem vrcholů u reálného příkladu. Algoritmus využívá iterativní metody, kdy se snažíme o postupné hledání řešení ve stále se zužující oblasti možných řešení. Určitou formou heuristiky je i to, že jako výchozí množinu dep volíme vrcholy s největším ohodnocením resp. nejvyšším stupněm vrcholu. Obecnou nevýhodou tohoto typu algoritmu je, že zpravidla vrací sub-optimální řešení pro úlohy většího rozsahu.

Postup algoritmu

1. krok:

- zvolíme výchozí množinu dep $D_k \subset V$, $|D_k| = k$,

- určíme množinu neprozkoumaných vrcholů $N = V \setminus D_k$ (rozdíl množin V a D_k),

- položíme z = 0

- určíme $f(D_k)$ resp. $g(D_k)$.

2. krok:

Zjistíme, zda je množina neprozkoumaných vrcholů prázdná

2a) *je-li $N = \emptyset$, pokračujeme krokem 4,*

2b) *je-li $N \neq \emptyset$, vybereme libovolný $v \in N$ a vytvoříme množiny*

$D_k^{v_j} = D_k - \{v_j\} + \{v\}$, $j = 1, 2, ..., k$, *dále*

- vypočteme $f(D_k^{v_j})$, *resp.* $g(D_k^{v_j})$,

- určíme $f(D_k^{v_r}) = \min_{v_j}\left\{f(D_k^{v_j})\right\}$, *resp.* $g(D_k^{v_r}) = \min_{v_j}\left\{g(D_k^{v_j})\right\}$

3. krok:

Porovnáme hodnoty kritéria

3a) *když* $f(D_k^{v_r}) \geq f(D_k)$, *resp.* $g(D_k^{v_r}) \geq g(D_k)$ *položíme N = N - {v} a pokračujeme krokem 2.*

3b) *když* $f(D_k^{v_r}) < f(D_k)$, *resp.* $g(D_k^{v_r}) < g(D_k)$ *vytvoříme novou množinu dep* $D_k = D_k - \{v_r\} + \{v\}$, *položíme z = z + 1 a pokračujeme krokem 2.*

4. krok:

4a) *je-li z = 0 pokračujeme krokem 5.*

4b) *je-li z > 0, položíme znovu z = 0, určíme novou množinu $N = V \setminus D_k$ a pokračujeme krokem 2.*

5. krok:

Množina D_k, představuje vrcholově (hranově) optimální rozmístění k dep na síti. Hodnota $f(D_k)$ resp. $g(D_k)$ je minimální hodnotou kriteriální funkce, která může být dosažena tímto algoritmem při zadané počáteční množině dep.[3]

[3] VOLEK, Josef; LINDA, Bohdan. Teorie grafů - aplikace v dopravě a veřejné správě. Dopravní fakulta Jana Pernera ; Pardubice : Univerzita Pardubice ; 2012. 192 s. ISBN 978-80-7395-225-9

3 Zdravotnická záchranná služba v ČR

3.1 Integrovaný záchranný systém

Integrovaný záchranný systém (dále jen IZS) není instituce, jak by se mohlo na první pohled zdát, nýbrž systém vazeb, způsobu spolupráce a koordinace jeho jednotlivých složek (případně i státní správy a samosprávy a dalších subjektů) na řešení mimořádných událostí a provádění likvidačních a záchranných operací. IZS v současné podobě existuje v České republice od roku 2001. Hlavním koordinátorem IZS v České republice je Hasičský záchranný sbor České republiky

Složky IZS se dělí na:

Základní složky

Základními složkami integrovaného záchranného systému jsou Hasičský záchranný sbor České republiky (dále jen "hasičský záchranný sbor"), jednotky požární ochrany zařazené do plošného pokrytí kraje jednotkami požární ochrany, zdravotnická záchranná služba a Policie České republiky.

Ostatní složky

Vyčleněné síly a prostředky ozbrojených sil, ostatní ozbrojené bezpečnostní sbory, ostatní záchranné sbory, orgány ochrany veřejného zdraví, havarijní, pohotovostní, odborné a jiné služby, zařízení civilní ochrany, neziskové organizace a sdružení občanů, která lze využít k záchranným a likvidačním pracím. Ostatní složky integrovaného záchranného systému poskytují při záchranných a likvidačních pracích plánovanou pomoc na vyžádání.[4]

3.2 Popis zdravotnické záchranné služby

Zdravotnická záchranná služba (dále jen ZZS) je nedílnou součástí IZS. ZZS je zajišťována příspěvkovými organizacemi spadající pod krajské úřady. Jejím hlavním úkolem je zajištění kvalitní přednemocniční neodkladné péče (dále jen PNP) podle zákona 374/2011 Sb. o zdravotnické záchranné službě.

Přednemocniční neodkladná péče

Za PNP můžeme považovat odborný zásah přímo na místě úrazu, náhlého onemocnění či zhoršení stavu již nemocného člověka. A to včetně ošetření v průběhu transportu a při předávání do péče kompetentního zdravotnického zařízení. Opravdová nutnost PNP nastává zejména při ohrožení základních životních funkcí pacienta na místě mimo přímý dosah kvalifikovaného zdravotnického zařízení. Takovými případy jsou typicky projevy náhlého onemocnění srdce a plic, neurologické potíže, vážné úrazy, otravy, stejně jako

[4] Zákon č. 239/2000 Sb., o integrovaném záchranném systému a o změně některých zákonů

všechny ostatní případy, při nichž dochází k nepředvídatelnému zhoršení zdravotního stavu.

3.3 Ambulance

Ambulance je označení pro dopravní prostředek provozovaný ZZS, který je používán k lékařským účelům. Kromě dobře známých sanitních vozidel a helikoptér se může jednat o motocykly, terénní vozidla, lodě či dokonce letadla.

3.3.1 Sanitní vozidla

Sanitní vozidlo neboli sanitka je silniční motorové vozidlo používané k lékařským účelům. Sanitní vozidlo nemusí být nutně provozováno pouze ZZS. Může se jednat také o vozidlo provozované přímo nemocnicí, soukromím subjektem zajišťujícím dopravní zdravotní službu (doprava raněných, nemocných a rodiček), armádou, atd. Tyto pak mohou také sloužit například k přesunům pacientů mezi nemocnicemi či transportu tělesných orgánů.

V současnosti rozlišujeme 4 třídy sanitních vozidel:

Třída A

Jedná se o vozidla umožňující přepravu jednoho ležícího/sedícího pacienta či jednoho ležícího pacienta a většího počtu sedících pacientů. Jsou obvykle provozovány nemocnicí či soukromým subjektem pro zajištění dopravní zdravotní služby. Se svým minimálním lékařským vybavením slouží typicky k přepravě neakutních pacientů mezi jednotlivými nemocnicemi nebo nemocnicí a domovem pacienta. V případě nouze mohou být použity i k účelům PNP.

Zpravidla se jedná o menší dodávky bez střešní nástavby. Oproti obvyklému žlutému zbarvení sanitek třídy B a C jsou lakovány bílou barvou, ale není to pravidlem. Přestože nejsou tato vozidla určena pro akutní převozy, jsou vozidla vybavena zvláštním výstražným světlem modré barvy a sirénou.

Třída B

Vozidla používaná převážně pro akutní výjezdy ZZS. Tyto vozidla obsahují výbavu nutnou ke stabilizaci a převoz akutních pacientů. Vedle běžného lékařského vybavení obsahují například defibrilátor, speciální nosítka, plicní ventilátor, vakuové matrace, monitor EKG, základní vyprošťovací prostředky nebo kyslíkové láhve.

Jedná se o větší dodávku s výkonným motorem (ideálně zážehovým), žlutým nátěrem a dobře viditelným označením reflexními materiály (pruhy, battenburská šachovnice). Vybavení zvláštním výstražným světlem modré barvy a sirénou je samozřejmostí.

Třída C
Tato vozidla jsou často označována jako resuscitační vůz. Vozidla obsahují kromě vybavení třídy B i prostředky potřebné k resuscitaci, připojení na umělou plicní ventilaci a následnému monitorování stavu pacienta.

Vzhledem k potřebě prostoru jak pro vybavení, tak pro zdravotnický personál se v této třídě nejčastěji využívají vozidla se skříňovou nástavbou. Ostatní parametry odpovídají třídě B.

Speciální sanitní vozidla
Mezi speciální sanitní vozidla např. patří:

- motocykly
- jízdní kola
- osobní automobily
- terénní automobily
- vozidla určená pro hromadná neštěstí
- vozidla pro přepravu orgánů či krve

Jejich technické a lékařské vybavení odpovídá účelu jejich použití.

3.4 Výjezdové skupiny

Výjezdovou skupinou se označuje odpovídající sanitní vozidlo s pevně určeným složením posádky.

3.4.1 Dělení výjezdových skupin

V tomto přehledu uvádím pouze minimální složení osádky, protože jejich složení se může u jednotlivých výjezdových stanovišť nebo případů měnit. Posádku často doplňuje další nelékařský zdravotnický pracovník či nižší/pomocný zdravotnický pracovník (ošetřovatel). Nelékařským zdravotnickým pracovníkem rozumíme zdravotní sestru s postgraduálním vzděláním zaměřeným na akutní péči nebo zdravotnického záchranáře.

Výjezdové skupiny se dělí do těchto standardizovaných kategorií:

Rychlá lékařská pomoc (RLP)
Tato skupina pracuje ve tříčlenném složení:

- řidič - zdravotnický záchranář
- nelékařský zdravotnický pracovník
- lékař

Sanitní vůz se skupinou RLP vyjíždí k pacientům s bezprostředním ohrožením životních funkcí. Jedná se o vážné úrazy, dopravní nehody, zástavy dechu, poruchy vědomí, atp.

Tato skupina má díky přítomnému lékaři plné možnosti v medikaci a aplikaci terapeutických postupů ještě v rámci PNP.

Výjezdová skupina RLP používá plně vybavené sanitní vozy třídy B nebo C.

Rychlá zdravotnická pomoc (RZP)
Vyjíždí k případům pouze ve složení:

- řidič - zdravotnický záchranář
- nelékařský zdravotnický pracovník

Vzhledem k absenci lékaře tato skupina vyjíždí pouze k nekomplikovaným úrazům a obecně stavům, u kterých se nepředpokládá potřeba okamžité diagnózy a léčby. Může rovněž asistovat skupině RLP u větších dopravních nehod či provádět převozy akutních pacientů z ordinací praktických lékařů. Primárním úkolem této skupiny je zajistit pacienta a urychleně ho předat do lékařské péče. I přes absenci lékaře se jedná o dostatečně kvalifikovaný personál pro zajištění kvalitní PNP. Navíc si mohou v případě potřeby přivolat na místo lékaře nebo sním konzultovat případ na dálku.

Výjezdová skupina RZP používá totožné vozy jako RLP.

Obrázek 6 - skupina RZP vyjíždí z výjezdového stanoviště; foto: autor

Rychlá lékařská pomoc v systému Rendez-Vous (RV)
Posádka se skládá z:

- řidič - zdravotnický záchranář
- lékař

Klasickým případem použití je vyslání RZP a zároveň RV k jednomu případu. Po dokončení lékařským úkonů lékařem z RV se posádka RZP postará a transport pacienta. Tím umožní například alokaci lékaře u dalšího zásahu a celkově tím snižuje vytíženost lékaře. Posádka RV se rovněž může použít v případech kde je sice zapotřebí lékař, ale pravděpodobně nebude nutný převoz do zdravotnického zařízení (epileptické nebo astmatické záchvaty).

Jedná se o speciální sanitní vozidlo, většinou osobní automobil typu SUV. Toto vozidlo je výbavou srovnatelné se sanitním vozidlem třídy B, s výjimkou možnosti transportu pacienta.

Obrázek 7 - sanitní vozidlo výjezdové skupiny v systému RV; foto: autor

Letecká záchranná služba (LZS)

Pracuje minimálně ve dvoučlenném složení:

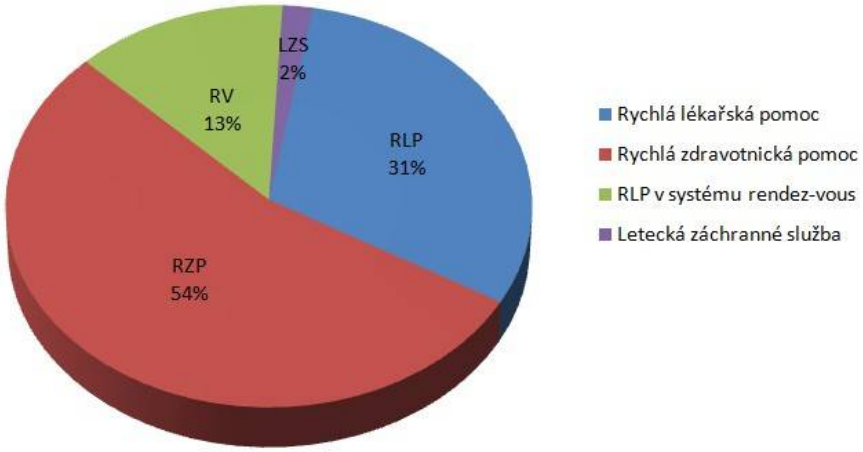
- pilot - zdravotnický záchranář
- lékař

V našich podmínkách je použití LZS jako reakce na tísňovou výzvu omezeno na rychlý převoz pacientů v kritickém stavu do specializovaných zdravotnických zařízení (například rozsáhlé popáleniny). Nebo záchraně z místa, které je buď špatně dosažitelné z hlediska terénu či dosahu jiných složek ZZS. LZS může být použit také při živelných pohromách a pátracích akcích. LZS nabývá na významu v přímořských nebo horských oblastech. V dalších případech může jít o převoz pacientů či orgánů do jiných zdravotnických zařízení.

Zpravidla se jedná o lehčí typy vrtulníků.

3.4.2 Rozložení výjezdových skupin

Procentuální rozložení jednotlivých skupin vidíme na obrázku: Obrázek 8 - procentuelní rozložení jednotlivých výjezdových skupin.



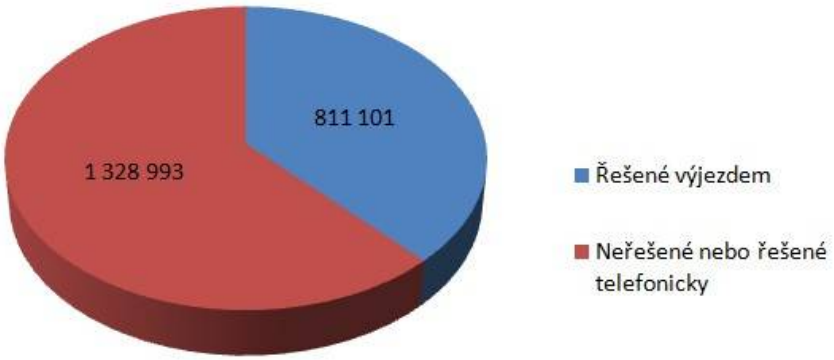
Obrázek 8 - procentuelní rozložení jednotlivých výjezdových skupin k roku 201; zdroj: výroční zpráva ZZS

3.5 Operační střediska

Fungování ZZS v ČR je organizačně rozděleno mezi operační střediska jednotlivých krajů. Krajské zdravotnické operační středisko (dále jen KZOS) je organizační a řídící centrum, které řídí provoz a činnost ZZS v daném kraji a spolupracuje s ostatními středisky. Hlavním úkolem KZOS je přijímat tísňová volání a adekvátně na ně reagovat. Součástí reakce může být i součinnost s jinými složkami IZS. O obsluhu tísňových linek se starají operátoři KZOS.

Operátoři

Operátoři tísňových linek jsou speciálně vyškolení zdravotničtí pracovníci s praxí v urgentních oborech medicíny a nezřídka také se zkušenostmi s prací ve výjezdových skupinách. Musíme si uvědomit, že ne každý hovor na linku 155 znamená výjezdovou akci. Je bezpodmínečně nutné, aby operátor příchozí volání správně vyhodnotil. Na obrázku: Obrázek 9 - absolutní počty tísňových volání na linku 155 vidíme, že většina tísňových volání není řešena výjezdem.



Obrázek 9 - absolutní počty tísňových volání na linku 155 k roku 201; zdroj: výroční zpráva ZZS

U neřešených volání se může jednat například o smyšlenou událost, událost která lze vyřešit telefonicky nebo si volající na tísňové lince chce objednat převoz z nemocnice. Pokud se operátor rozhodne, že jde o událost u které je nutné zásah výjezdové skupiny, tak tuto událost elektronicky zpracuje a předá jí dispečerům. Operátor může nadále zůstat ve spojení s volajícím a radit mu jak má postupovat do příjezdu záchranné služby.

Dispečeři

Dispečeři mají na starost organizaci práce jednotlivých výjezdových stanovišť. Předanou událost vyhodnotí, zvolí vhodnou výjezdovou skupinu a zajistí její vyslání. Do příslušného výjezdového stanoviště pošle údaje o přidělené akci, zároveň do GPS systému vybraného vozidla načte údaje o místě zásahu. S výjezdovou skupinou je možno být nepřetržitě v kontaktu pomocí radiového spojení a sledovat ji v reálném čase v informačním systému operačního střediska.

Obrázek 10 - vybavení sanitního hlášení stavů výjezdu dispečinku (vpravo dole); foto: autor

Přehled operačních středisek

Kraj	Krajské operační středisko	Počet stanovišť	Počet výjezdových skupin
hlavní město Praha	Praha	18	25
Středočeský	Kladno	43	76
Jihočeský	České Budějovice	26	47
Plzeňský	Plzeň	23	36
Karlovarský	Karlovy Vary	11	17
Ústecký	Ústí nad Labem	19	37
Liberecký	Liberec	14	30
Královéhradecký	Hradec Králové	14	27
Pardubický	Pardubice	15	25
Vysočina	Jihlavy	20	26
Jihomoravský	Brno	23	42
Olomoucký	Olomouc	15	26
Zlínský	Zlín	13	27
Moravskoslezský	Ostrava	30	60

Tabulka 1 - přehled operačních středisek, výjezdových stanovišť a skupin v jednotlivých krajích k roku 2011; zdroj: výroční zpráva ZZS

3.6 Výjezdová stanoviště

Výjezdové stanoviště je zázemím pro personál a techniku potřebnou pro perfektní fungování přidělených výjezdových skupin. Vzhledem k tomu, že většina stanovišť funguje non-stop, je třeba vytvořit kvalitní zázemí pro personál. Tím bývá kuchyňka, jídelna, pokoje, převlékárny a sociální zařízení. Nutností jsou minimální zásoby zdravotnického materiálu pro doplnění vybavení sanitního vozidla po výjezdu. Často bývá ke stanovišti přidružen sklad materiálu, který se rozváží do jiných výjezdových stanovišť nebo je použit při případných hromadných neštěstích. Pro sanitní vozidla jsou vybudovány obvykle vytápěné garáže se zabezpečenou příjezdovou cestou. Ve větších výjezdových stanovištích může být také umístěna část KZOS nebo nadstandardní vybavení jako je například posilovna. Technickým zázemím jsou PC spojená pomocí informačního systému s KZOS.

Počty stanovišť v jednotlivých krajích najdeme v tabulce: Tabulka 1

3.7 ZZS Pardubického kraje

ZZS Pardubického kraje pokrývá oblast o rozloze 4500 km^2 a více než 510000 obyvatel. V současné podobě funguje od 1.1.2003 a je financována z rozpočtu kraje.

3.7.1 Krajské operační středisko

KZOS Pardubického kraje vzniklo v roce 2008 sloučením okresních dispečinků záchranné služby a nachází se v městské části Pardubičky. Organizačně je kraj rozdělen do čtyř územních oblastí odpovídajícím hranicím bývalých okresů.

Zvláštností pardubického KZOS je záložní krizový dispečink v Chrudimi, který může v případě potřeby převzít úlohu KZOS v Pardubicích. Za zmínku stojí také využívání technologií jako je sledování výjezdových vozidel a identifikace polohy volajícího.

Sledování vozidel

Zajišťuje plnou integraci informačního systému S.O.S. se systémem pro sledování vozidel tak, aby bylo možné přímo z informačního systému S.O.S. provádět níže uvedené operace.

- *K dané akci ZZS zobrazit místo akce na mapě, na požádání vyhledat nejbližší prostředky ZZS k místu akce*
- *Identifikovat na mapě libovolný výjezd z S.O.S*
- *Možnost automatického přebírání stavových informací z vozidel s časy jednotlivých fází výjezdů a jejich automatické promítnutí do dat výjezdů S.O.S.*
- *Při výjezdu vozidla ZZS možnost z S.O.S. automaticky zasílat textové informace o adrese spolu se souřadnicemi místa akce do vozidla tak, aby navigační jednotka ve vozidle umožnila navigaci vozidla na místo akce*

Identifikace polohy mobilního telefonu
Umožňuje při příjmu volání ze sítě mobilních operátorů lokalizace přibližné polohy volajícího a její zobrazení pro potřeby operátorů KZOS v mapových podkladech propojených s informačním systémem S.O.S.[5]

3.7.2 Výjezdové skupiny

V Pardubickém kraji funguje v nepřetržitém provozu 25 výjezdových skupin. Jedná se o skupiny rychlé lékařské pomoci, rychlé zdravotnické pomoci a rychlé lékařské pomoci v systému rendez-vous. Systém RV se začal v Pardubickém kraji využívat až na sklonku roku 2010, nicméně se nyní uvažuje o plošném zavedení tohoto systému po vzoru Libereckého kraje. Vlastní letecká záchranná služba na území kraje nefunguje, je však vyjednáno pokrytí LZS provozovanou Královéhradeckým krajem.

Vozový park sanitních vozidel třídy B a C zahrnuje většinou vozy Volkswagen Transporter T5/T4 a Mercedes-Benz Sprinter. Dále osobní vozy Škoda Octavia Combi a Škoda Yeti, určených pro systém RV.

3.7.3 Výjezdová stanoviště

ZZS Pardubického kraje provozuje v současné době 16 výjezdových stanovišť. V dlouhodobém horizontu se plánuje výstavba na 4 dalších místech (Seč, Nasavrky, Choceň, Jevíčko).

Současné umístění výjezdových stanovišť v příslušných územních oblastech je následující:

- Pardubice
  - Pardubice - Pardubičky
  - Pardubice - Dukla
  - Holice
  - Přelouč
- Chrudim
  - Chrudim
  - Hlinsko
  - Skuteč
- Svitavy
  - Svitavy
  - Litomyšl
  - Moravská Třebová
  - Polička
- Ústí nad Orlicí
  - Ústí nad Orlicí
  - Červená Voda
  - Lanškroun

[5] Zdroj: http://www.zzspak.cz/ - webová prezentace ZZS Pardubického kraje

- Žamberk
- Vysoké Mýto

Na obrázku 7 najdeme mapu s rozmístěním jednotlivých výjezdových středisek a výjezdových skupin, které mají tato střediska k dispozici.

Obrázek 11 - koncepce výjezdových skupin v Pardubickém kraji k roku 2011-2012; zdroj: webová prezentace ZZS Pardubického kraje

3.7.4 Hledisko lokačně-alokační úlohy

Pro výpočet diskrétní lokačně-alokační pro vrcholově optimální umístění středisek obsluhy - výjezdových stanovišť ZZS budeme potřebovat několik údajů:

- průměrnou rychlost vozidel ZZS při zásahu
- dojezdový čas – doba, za kterou musí být výjezd na místě určení od přijetí výzvy
- aktivační čas - za jak dlouho po obdržení výzvy vyrazí posádka na místo určení
- ohodnocení hran - délky jednotlivých úseků
- ohodnocení vrcholů - počet požadavků na zásah ZZS v určitém časovém období

Délku komunikací a jednotlivých úseků můžeme zjistit z veřejně dostupných údajů o silniční síti ČR.

Statistiku počtu výjezdů do jednotlivých měst, případně na místa na pozemní komunikaci se pro práci nepodařilo získat. Z dokumentů asociace ZZS ČR[6] můžeme zjistit, že v roce 2012 vyjela posádka ZZS k zásahu 42 527 krát. Stejný zdroj uvádí, že ZZS eviduje v Pardubickém kraji 511 627 obyvatel. To znamená, že na jednoho obyvatele Pardubického kraje připadá přibližně 0,08312 výjezdu za rok. Pomocí tohoto údaje a počtu obyvatel města - představeného vrcholem, získáme ohodnocení vrcholů.

Dojezdový čas je pevně daný zákonem č. 374/2011 Sb. o zdravotnické záchranné službě. V současné době je tato doba stanovená na 20 minut, včetně času potřebného pro aktivaci posádky.

Pro zjištění průměrné rychlosti a aktivačního času byly použity záznamy z několika desítek výjezdů výjezdového stanoviště Doksy v Libereckém kraji. V následující tabulce je uveden výsek několika reprezentativních údajů ze souboru dat získaných z těchto záznamů. Bližší informace o výjezdech jsou záměrně vynechány.

Výzva	Výjezd	Čas aktivace	Dojezd na místo	Čas cesty [min]	Vzdálenost [km]	Rychlost [km/h]
7:16	7:17	0:01	7:29	0:12	15.6	78.00
18:35	18:37	0:02	18:48	0:11	15.1	82.36
16:04	16:05	0:01	16:21	0:16	21.4	80.25
10:49	10:51	0:02	11:05	0:14	18.6	79.71
1:48	1:49	0:01	2:02	0:13	12.3	56.77
8:58	9:00	0:02	9:11	0:11	16.1	87.82
19:18	19:19	0:01	19:31	0:12	14.8	74.00
13:12	13:13	0:01	13:21	0:08	9.8	73.50
9:54	9:56	0:02	10:01	0:05	5.7	68.40
5:09	5:11	0:02	5:22	0:11	8.6	46.91
2:58	2:59	0:01	3:13	0:14	14.7	63.00
Průměr:		**0:01:27**		**0:11:33**	**13.88**	**71.88**

Tabulka 2 - výběr dat z výjezdů stanoviště Doksy - Liberecký kraj

Maximální doba aktivačního času je opět stanovena zákonem č. 374/2011 Sb. o zdravotnické záchranné službě. Během dne je maximální aktivační doba 3 minuty, v nočních hodinách 5 minut. I když bylo pozorováním zjištěno, že aktivační doba dosahuje v průměru 1,5 minuty, použijeme vzhledem k možnému zkreslení dobu 3 minuty.

Průměrná rychlost se může mírně lišit u jednotlivých výjezdových stanovišť z důvodů umístění či "rychlostní politiky" krajského střediska. Data pochází z výjezdového stanoviště v menším městě. Ve větším městě může průměrná rychlost dále klesat, tím jak se vozidlo musí nejprve dostat ze zastavěného území. Na druhou stranu je u velkého města pravděpodobnější výskyt silnic vyšší třídy, kde může sanitní vozidlo dosáhnout bezpečně vyšší rychlosti. Průměrná rychlost také klesá, pokud se zásah nachází ve stejné aglomeraci jako výjezdové středisko, což v této prací zanedbáváme. Pro účely této práce budeme považovat za průměrnou rychlost vozidel ZZS rychlost 70km/h.

[6] http://www.azzs.cz/dokumenty/zzs-cr-v-cislech/

4 Implementace

4.1 Modelování případů užití

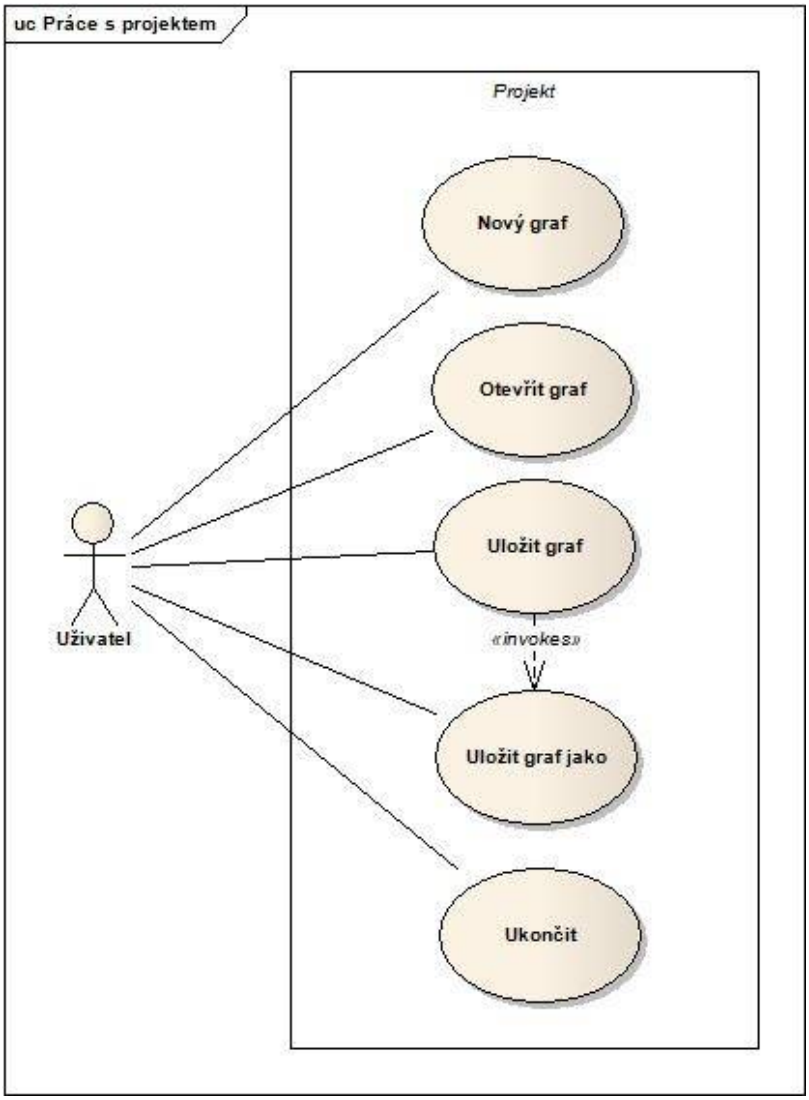
Diagram případů užití neboli use-case diagram je jedním z UML diagramů, který znázorňuje vnější pohled na modelovaný systém. Tedy v našem případě na vypracovaný program a jeho uživatelské rozhraní. Mimo jiné zachycuje, jakým způsobem mohou jednotlivý aktéři (uživatel, administrátor) komunikovat s modelovaným systémem a také vztahy mezi poskytovanými službami systému. Je určen k znázornění chování systému jak pro vývojáře, tak pro uživatele, aniž by popisoval vnitřní fungování systému.

Pro vytvoření diagramů užití byl použit program Enterprise Architect. Použité vztahy mezi jednotlivými službami systému jsou následující:

- *invokes* - zdrojový element za určitých podmínek spustí činnost cílového elementu
- *extend* - cílový element rozšiřuje určité chování zdrojového elementu
- *precedes* - cílový element nemůže zahájit svojí činnost, dokud není spuštěna a dokončena činnost zdrojového elementu
- *include* - zdrojový element obsahuje část funkcionality cílového elementu

Práce s projektem

Popis uživatelských možnosti při práci s projektem - grafem.

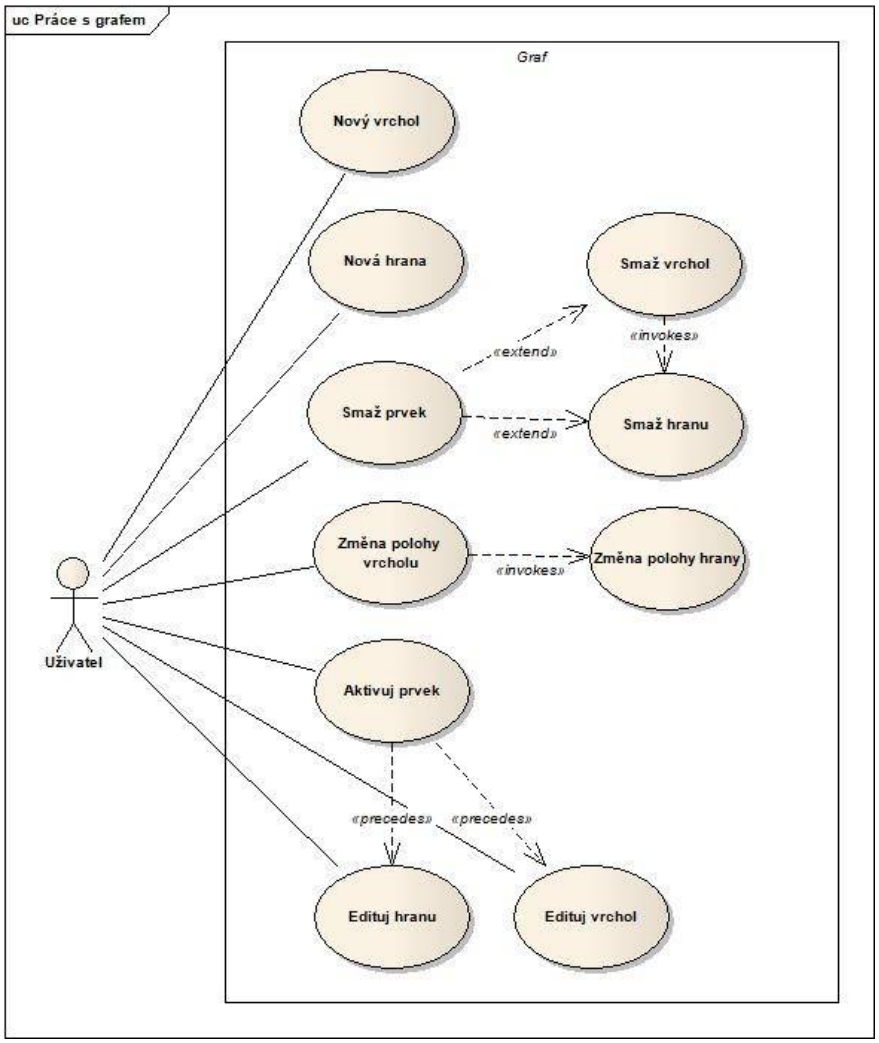


Obrázek 12 - use-case diagram práce s projektem

- Nový graf - otevře dialogové okno pro specifikaci vytvoření nového grafu
- Otevřít graf - načte graf z vybraného souboru
- Uložit graf - uloží graf do asociovaného souboru, pokud je graf nový (nemá asociovaný soubor) tak odkazuje na "*Uložit graf jako"*
- Uložit graf jako - uloží graf do zadaného souboru
- Ukončit - ukončí program

Tvorba grafu

Popisuje práci s grafem na kreslícím plátně.

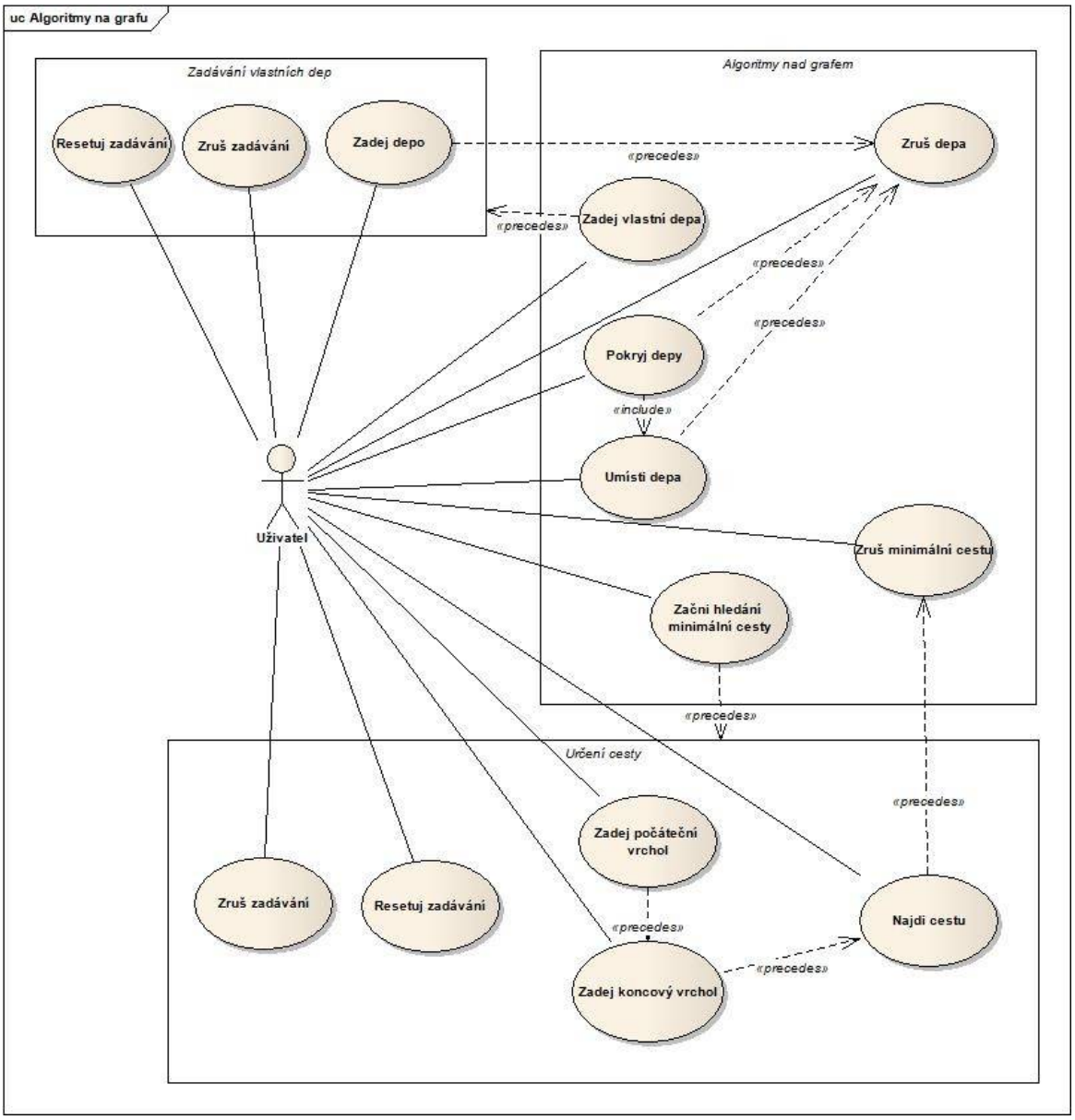


Obrázek 13 - use-case diagram práce s grafem

- Nový vrchol - vloží do grafu nový vrchol,
- Nová hrana - vytvoří novou hranu mezi dvěma zadanými vrcholy
- Smaž prvek - vymaže vybraný prvek z grafu
    - Smaž hranu - odstraní vybranou hranu
    - Smaž vrchol - odstraní vybraný vrchol, případně všechny jeho incidující hrany
- Změna polohy vrcholu - změní polohu vybraného vrcholu
    - Změna polohy hrany - změní polohu hran, které incidují s vrcholem, kterému měníme polohu
- Aktivuj prvek - označí vybraný prvek grafu jako aktivní a tím umožní prohlížení a editaci jeho vlastností
    - Edituj hranu - umožní editovat vlastnosti aktivní hrany
    - Edituj vrchol - umožní editovat vlastnosti aktivního vrcholu

Algoritmy nad grafem

Tento diagram popisuje ovládání algoritmů, které můžeme nad grafem provádět.



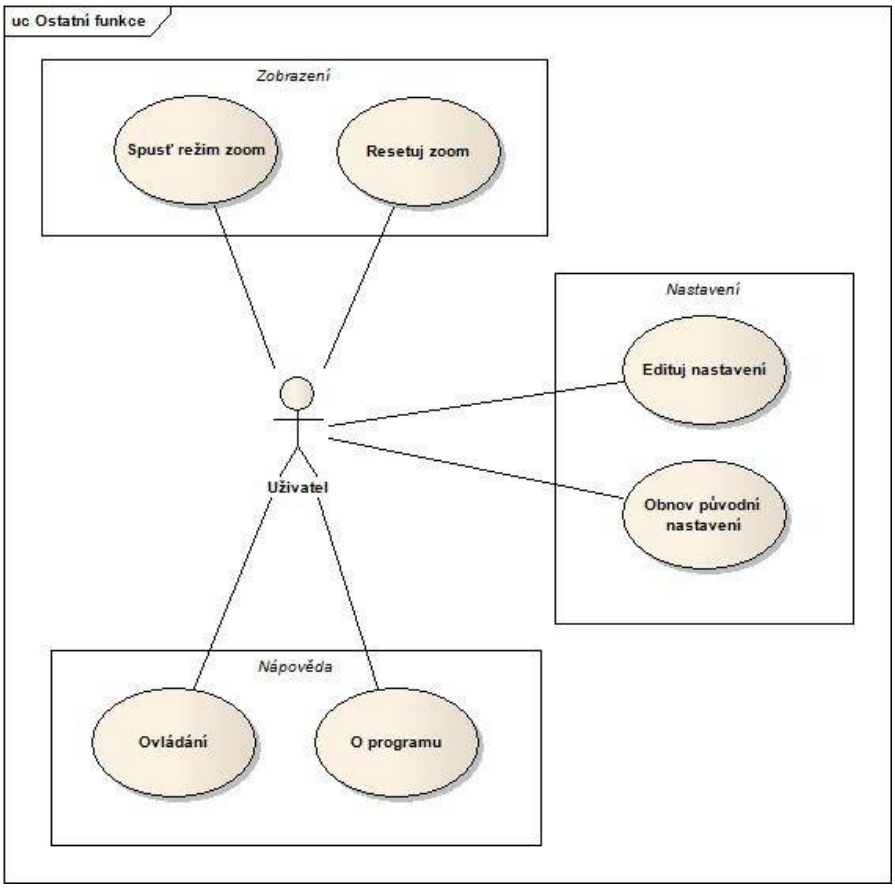
Obrázek 14 - use-case diagram ovládání algoritmů nad grafem

- Pokryj depy - podle parametrů zadaných v nastavení vytvoří takový počet optimálně umístěných dep, aby tato depa dokázala svými atrakčními obvody pokrýt celý graf
- Umísti depa - vytvoří zadaný počet dep, které vrcholově optimálně umístí v grafu
- Zadej vlastní depa - zpřístupní systém sloužící k zadávání vlastních dep na vybrané umístění
- Resetuj zadávání dep - odznačí již zadaná depa a umožní zadávání dep opakovat
- Zruš zadávání dep - odznačí již zadaná depa a znepřístupní systém
- Zruš depa - pokud v grafu existují depa, vymaže je

- Začni hledání minimální cesty - zpřístupní systém sloužící k zadání a potvrzení hledání minimální cesty v grafu
- Zadej počáteční vrchol - označí zadaný vrchol grafu jako počáteční vrchol hledané minimální cesty
- Zadej koncový vrchol - označí zadaný vrchol grafu jako koncový vrchol hledané minimální cesty
- Resetuj zadávání cesty - odznačí zadaný počáteční/koncový vrchol hledané minimální cesty a umožní zadávání opakovat
- Zruš zadávání cesty - odznačí zadaný počáteční/koncový vrchol hledané minimální cesty a zruší zadávání, čímž znepřístupní systém
- Najdi cestu - vyhledá minimální cestu mezi zadaným počátečním a koncovým vrcholem
- Zruš minimální cestu - pokud v grafu existuje minimální cesta, zruší ji

Nastavení a vedlejší funkce

Popisuje manipulaci s nastavením, režimem zobrazení a práci se sekcí nápovědy.



Obrázek 15 - use-case diagram ostatních funkcí grafu

- Spusť režim zoom - aktivuje/deaktivuje režim zoom, který umožňuje přibližovat a oddalovat kreslící plátno s grafem
- Resetuj zoom - vrátí plátno do původní polohy a vypne režim zoom
- Edituj nastavení - vyvolá dialogové okno s nastavením projektu

- Obnov původní nastavení - vrátí nastavení projektu na defaultní hodnoty
- Ovládání - vyvolá okno s popisem ovládání programu
- O programu - vyvolá okno s údaji o programu

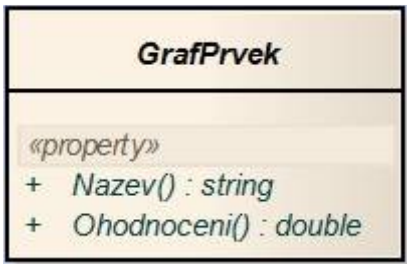
4.2 Popis jednotlivých tříd

Tato sekce popisuje účel, funkce a fungování jednotlivých tříd v projektu. Pro grafické znázornění tříd použijeme UML diagramy vytvořené v programu Enterprise Architect. Značení diagramů je následující:

- název třídy kurzívou - abstraktní třída
- červený text - atributy třídy
- modro-zelený text - funkce nebo vlastnosti třídy
- podtržený text - statický prvek třídy
- znaménko plus - veřejný prvek třídy
- znaménko minus - soukromý prvek třídy

GrafPrvek

Je abstraktní třída, která je předkem prvků grafu.



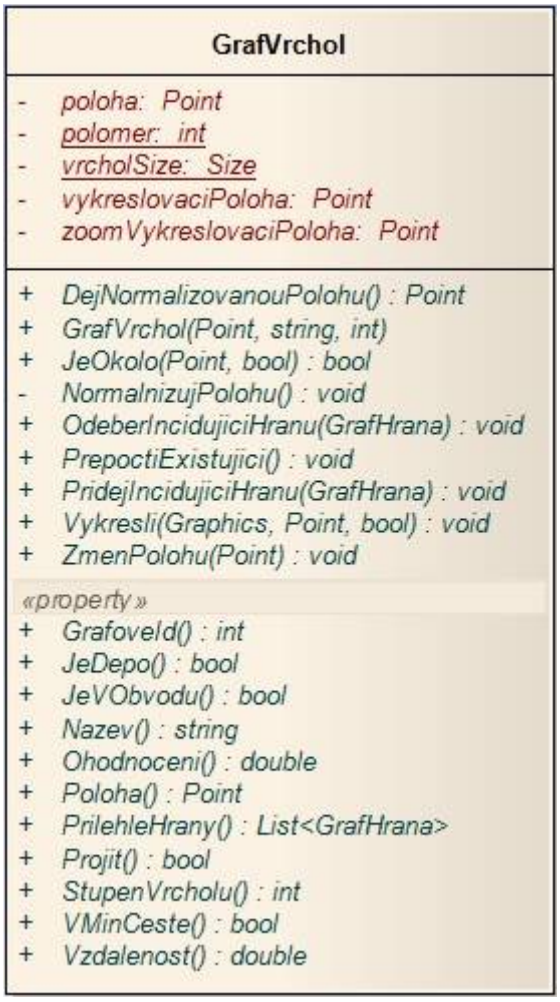
Obrázek 16 - UML diagram třídy GrafPrvek

Proměnná této třídy se používá, pokud chceme předat jiné třídě prvek grafu.

Obsahuje pouze vlastnosti *Nazev()* a *Ohodnoceni()* popisující základní vlastnosti prvků grafu.

GrafVrchol

GrafVrchol je třída představující vrchol grafu odvozený od třídy *GrafPrvek*.



Obrázek 17 - UML diagram třídy GrafVrchol

Vrchol grafu si v sobě uchovává 3 druhy poloh:

- *poloha* - reálná poloha v souřadnicovém systému
- *vykreslovacíPoloha* - poloha pro vykreslení na kreslící plátno mimo režim zoom
- *zoomVykreslovacíPoloha* - poloha pro vykreslení na kreslící plátno v režimu zoom

Nutnost ukládat si reálnou polohu je zřejmá. I přes to, že by se ostatní polohy dali v případě potřeby lehce přepočítat z reálné polohy, je vhodnější je mít uložené ve vlastní datové složce a přepočítávat je pouze v případě změny. To vychází ze skutečnosti, že se graf překresluje velice často, mimo jiné při každém pohybu myši nad kreslící plochou. Vykreslovací poloha je nutná z důvodu, že pokud program potřebuje nakreslit kruh - vrchol tak si nejprve vytvoří čtverec, umístěný levým horním rohem v bodu ve kterém vrchol vykreslujeme, do kterého poté vepíše kružnici. Vykreslovací poloha je tedy poloha posunutá tak, aby se při vykreslení zdálo, že souřadnice bodu ve kterém

vykreslujeme kruh odpovídá středu kruhu - vrcholu. Poloha *zoomVykreslovaciPoloha* je poloha pro vykreslení při přiblížení či oddálení plátna v režimu zoom. S tím souvisí metoda *PrepoctiExistujici()*, která přepočte polohy (kromě reálné polohy která se nemění) na souřadnice odpovídající změněnému měřítku kreslící plochy při přiblížení či oddálení plátna. Metoda *NormalizujPolohu()* vypočítá reálnou polohu, pokud přidáváme nebo přesouváme vrchol v režimu zoom. Jediný způsob jak změnit reálnou polohu je přemístěním vrcholu na jiné souřadnice pomocí metody *ZmenPolohu().*

Pomocí vlastností ukládáme základní atributy vrcholu, jako je název, ohodnocení nebo unikátní grafové id. Vlastnosti *Nazev()* a *Ohodnoceni()* jsou zděděné z třídy *GrafPrvek*.

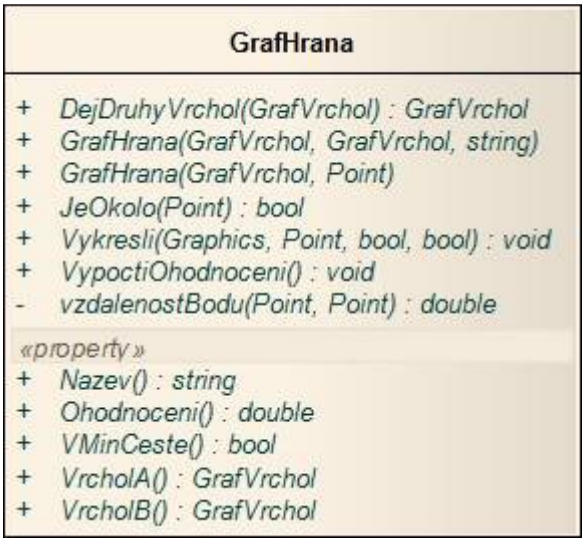
Vrchol dále eviduje sadu logických hodnot značících status vrcholu (v minimální cestě, v atrakčním obvodu, součástí depa), pro potřeby vykreslování na kreslící plochu funkcí *Vykresli().* Další možné stavy jsou předávány parametry při volání metody. Každý tento stav se projevuje na kreslící ploše jinou barvou vrcholu, dle současného nastavení. Tato sada logických hodnot by se mohla nahradit výčtovým typem s příznaky. Jelikož výčtový typ nepřináší žádnou výraznou přidanou hodnotu a je složitější na implementaci, bylo upřednostněno stávající řešení. Pokud by počet možných stavů měl nadále stoupat, tak by byla implementace výčtového typu vhodnější.

Ze stejného důvodu jako si ukládáme 3 různé polohy, máme jako statickou datovou složku uloženou instanci struktury *Size*, uchovávající velikost vrcholu. Při každé změně pozice kurzoru se volá v každém vrcholu metoda *JeOkolo()*, která nám oznámí, zda je kurzor v blízkosti vrcholu. Pokud ano, označí se tento vrchol jako vrchol pod kurzorem a je možné ho smazat, editovat, či přesunout. Metoda spočítá rovinnou vzdálenost mezi vrcholem a současnou polohou kurzoru a podle ní rozhodne, zda je kurzor nad vrcholem či ne.

V poslední řadě si uchováváme pole ukazatelů na hrany, které incidují s vrcholem. To nám usnadní práci při některých operacích jako je například mazání vrcholu, kdy nemusíme v seznamu hran hledat, která hrana inciduje s kterým vrcholem.

GrafHrana

Třída *GrafHrana* představuje hranu v grafu. Tato třída je odvozena od třídy *GrafPrvek*.



Obrázek 18 - UML diagram třídy GrafHrana

Existence hrany je úzce spjata s existencí vrcholu. Ve vlastnostech *VrcholA* a *VrcholB* uchováváme ukazatele na krajní vrcholy hrany. Pokud zanikne některý z těchto vrcholů, zanikne automaticky i hrana. Jelikož pracujeme s neorientovaným grafem, tak na pořadí vrcholů nijak nezáleží. Při vytvoření hrany se spočítá ohodnocení hrany pomocí vzdálenosti v rovině mezi krajními vrcholy. O tento výpočet se stará metoda *VypoctiOhodnoceni()*.

Vlastnosti hrany a metoda *Vykresli()* jsou řešeny obdobně jako u třídy *GrafVrchol*.

Metoda *JeOkolo()* určí, zda je poloha kurzoru nad hranou či ne. Nejdříve vypočte vzdálenost polohy kurzoru od hrany představené přímkou. Jelikož je hrana úsečkou a ne přímkou musíme vytvořit pomyslnou kružnici ve středu hrany, která má průměr odpovídající délce hrany. Průsečíky této kružnice a vypočítané přímky jsou krajními body hrany. Pokud je kurzor v minimální vzdálenosti od přímky hrany a zároveň není vně pomyslné kružnice, můžeme říci, že se kurzor nachází nad hranou.

Pro účely procházení grafu a hledání minimální cesty implementuje metodu *DejDruhyVrchol()*, která vrátí ukazatel na opačný vrchol hrany.

Depo

Třída depo představuje depo - středisko obsluhy lokační úlohy.

Obrázek 19 - UML diagram třídy Depo

Tato třída je vytvořena pro lepší práci s algoritmy lokační úlohy. V grafu se nepoužívá jako objekt schopný vykreslení sebe sama na kreslící plátno.

Depo si uchovává ukazatel na vrchol, ve kterém je umístěno. Dále obsahuje pole ukazatelů na vrcholy, které se nacházejí v atrakčním obvodu depa. Vlastnost *DopravniPrace()* uchovává celkovou dopravní práci pro obsloužení atrakčního obvodu depa.

Nastaveni

Je statická třída uchovávající nastavení projektu.

Obrázek 20 - UML diagram třídy Nastaveni

Zřejmou výhodou uchovávání nastavení ve statické třídě je, že můžeme k jejím hodnotám bezinstančně přistupovat z jakékoli třídy.

Datové složky metody z větší části uchovávají instance třídy *Pen*. Ty jsou potřeba pro vykreslování prvků grafu, kde pro každý možný stav prvku grafu máme různou instanci třídy *Pen*.

Dále ukládáme nastavení týkající se lokační úlohy. *PrumernaRychlost* představuje technickou rychlost mezi střediskem obsluhy (depo) a místem dopravní obsluhy (vrchol). *DojezdovyCas* udává, za jakou dobu od vytvoření požadavku na obsluhu musí středisko obsluhy tento požadavek obsloužit. Z těchto dvou parametrů můžeme jednoduchým výpočtem zjistit do jaké vzdálenosti je schopno středisko obsloužit požadavek na obsluhu.

Další část datových složek je potřeba pro práci režimu zoom.

- *RezimZoom* - logická hodnota která udává, zda je plátno v režimu zoom
- *PuvodniDelka* - počáteční délka plátna, tedy původní stav v nezměněném měřítku
- *PuvodniSirka* - počáteční šířka plátna, analogicky k původní délce plátna
- *PocetKroku* - udává kolik kroků zoomu bylo provedeno, kde přiblížení jsou kladné hodnoty a oddálení záporné
- *KrokZoomu* - udává o kolik se má změnit měřítko plátna při oddálení/přiblížení
- *Zoom* - měřítko kreslícího plátna, původní měřítko má hodnotu 1

Změněné měřítko získáme tak, že hodnotu *Zoom* vynásobíme či vydělíme hodnotou *KrokZoomu*, podle toho zda chceme přiblížit čí oddálit kreslící plátno. Poté stačí podle následujícího vztahu změnit velikost plátna.

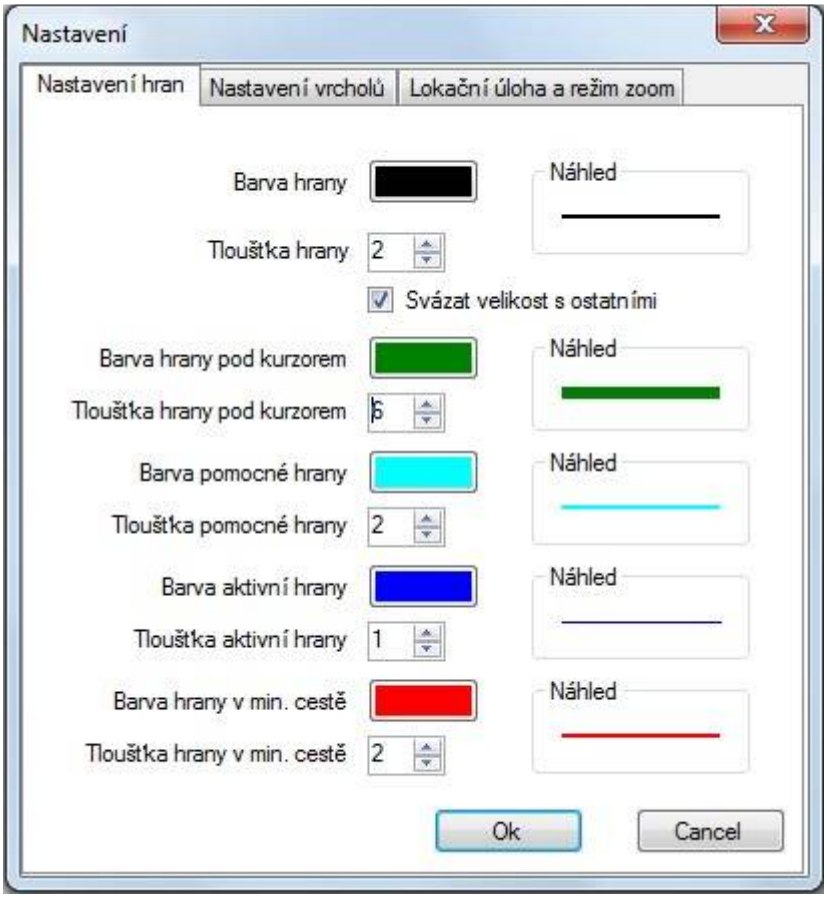
nová šířka = *PuvodniSirka* * *Zoom*

nová délka = *PuvodniDelka* * *Zoom*

Všechny datové složky kromě složek týkajících se režimu zoom je možno načíst či uložit pomocí textového souboru. Této služby využíváme při každém spuštění, respektive ukončení programu. Pokud nastane při nahrávání nastavení ze souboru chyba, načteme defaultní nastavení. Barvu pera ukládáme v barevném modelu RGBA, kde:

- R - červená složka barvy
- G - zelená složka barvy
- B - modrá složka barvy
- A - udává informaci o průhlednosti (0 - průhledná, 255 - neprůhledná)

Například plně červenou barvu ukládáme v modelu RGBA jako: (255,0,0, 255). Knihovna .NET používá pro ukládání barev formát ARGB, nicméně pro účely uložení do textového souboru bude vhodnější použít všeobecně známý barevný model RGBA.



Obrázek 21 - pohled na dialogové okno zajišťující nastavení

Graf

Třída graf představuje strukturu, která pracuje s grafem jako celkem.

Obrázek 22 - UML diagram třídy Graf

Třída graf si uchovává seznam hran a vrcholů v generické struktuře knihovny .NET *List<>().* Jednotlivé relace mezi prvky si uchovávají přímo objekty představující vrchol nebo hranu grafu. Jedná se tedy o vrcholově a zároveň hranově orientovaný způsob uložení grafu. Třídu *List<>()* budeme dále využívat poměrně často, proto jí budu nadále označovat jako "*seznam*", navzdory skutečnosti, že je tato třída interně postavena na poli.

Metody pracující s vrcholy a hranami ve většině případů přijímají z třídy *Editor* současnou polohu kurzoru. Pokud tedy editor volá metodu pracující s prvek grafu, graf si nejprve zjistí, který prvek se na této poloze nachází (a jestli vůbec nějaký) a poté

probíhá vlastní algoritmus požadované akce. Tuto skutečnost nebudu u jednotlivých metod dále zmiňovat.

Metoda *PridejVrchol()* po kontrole cílové polohy, jestli se na ní již něco nenachází, vytvoří novou instanci třídy *GrafVrchol*, kterou uloží do seznamu vrcholů. Vrcholu také přiřadí volné unikátní id a automatický název, který je možné později změnit. Metoda *PridejHranu()* obdrží od editoru dvě polohy, polohu stlačení a polohu kde uživatel pustil tlačítko myši. Pokud se na obou těchto polohách nachází vrchol, vytvoří se mezi těmito vrcholy hrana. Jestliže se nachází vrchol pouze na poloze stlačení tlačítka, vrácí metoda nalezený vrchol editoru. *PridejPomocnouHranu()* vytvoří speciální hranu, který vede z vrcholu do libovolného místa na kreslícím plátně, tu uloží do datové složky *pomocnáHrana* pro účely pozdějšího vykreslení. *AktivujPrvek()* najde na zadané poloze prvek, který uloží do datové složky a následně ho může poskytovat editoru k editaci.

Metoda *ZjistiDelkuMinimalniCesty()* najde v grafu minimální cestu z určeného počátečního a koncového bodu. Tuto cestu ukládá do datové složky *minCesta* jako seznam hran. Dijkstrův algoritmus je rozdělen na dvě části. Část, ve které ohodnotíme všechny vrcholy vzdáleností od vrcholu počátečního a část rekonstrukční, kdy rekonstruujeme vlastní cestu z původního počátečního vrcholu do libovolného vrcholu grafu. Výhoda toho přístupu je, že pokud zůstane počáteční vrchol cesty stejný, nemusíme znovu provádět Dijkstrův algoritmus pro ohodnocení vrcholů a pouze rekonstruujeme.

Lokace dep probíhá pomocí iterativního algoritmu určení vrcholově optimální lokace *k* dep na síti. Pro práci tohoto algoritmu musíme nejdříve spočítat distanční matici pomocí Floydův-Warshallova algoritmu. Nyní můžeme metodou *NajdiDepa()* určit lokaci potřebného počtu dep. Metoda *PokryjVseDepy()* spouští algoritmus lokace dep tak, že postupně inkrementujeme parametr udávající počet dep do té doby, než jsou schopna alokovaná depa svým dostupem (dostup je vypočítán z hodnot ve třídě Nastavení) pokrýt celý graf. Dalším způsobem lokace dep je manuální zadání pomocí metody *PridejDepo()*. Poté co jsme manuálně přidali všechny zamýšlená depa, musíme depům přiřadit atrakční obvody a vypočítat celkovou dopravní práci, což je součet dopravních prací všech dep. Výstupy z algoritmů týkajících se dep ukládáme do datové složky *depa*, jako seznam instancí třídy *Depo*. Před hledáním dep kontrolujeme celistvost grafu.

KresliciPlatno

Tato třída odvozená od třídy panel slouží jako kreslící plátno pro graf.

Obrázek 23 - UML diagram třídy KresliciPlatno

Vlastnost *Picture()* (kterou budeme označovat jako obrázek) uchovává instanci třídy *PictureBox*, která má registrované handlery událostí zaznamenávající akce myši nad její pozicí. Do vlastnosti *BackgroundImage* obrázku ukládáme případné pozadí.

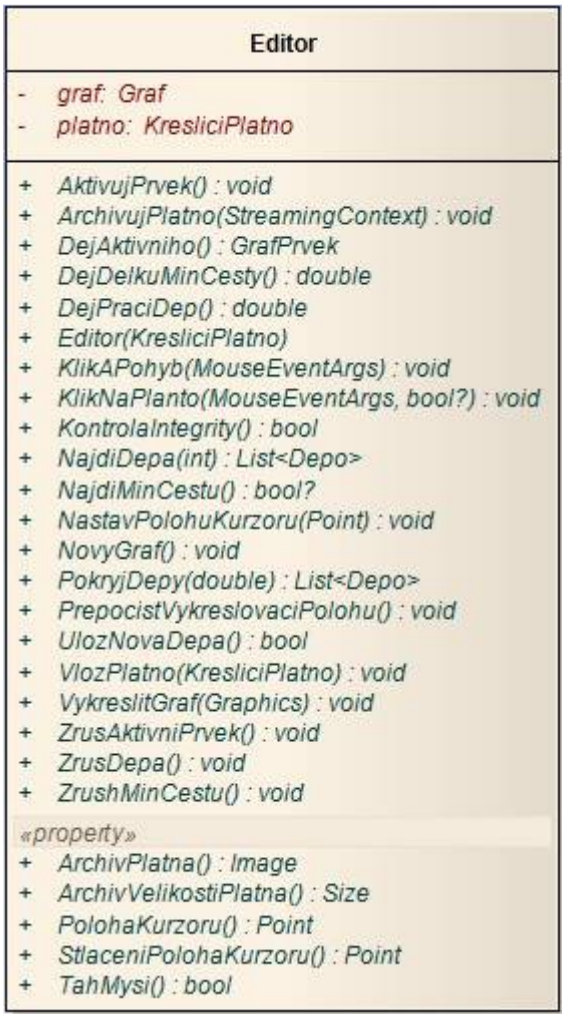
Důvodem proč vykreslujeme potřebné tvary do obrázku a ne přímo do kreslícího plátna - panelu je automatické zajištění scrollování. Samotné kreslící plátno se svou velikostí přizpůsobuje obrázku, až do hodnoty maximální velikosti kreslícího plátna. Pokud je velikost obrázku větší než maximální velikost kreslícího plátna, dochází k automatickému zapnutí scrollovacích panelů tak, abychom si mohli obrázek v menším prostoru kreslícího plátna prohlédnou celý. Pokud za běhu programu potřebujeme vytvořit plátno s jinou velikostí či přidat obrázek pozadí, změníme pouze vlastnosti obrázku pomocí metod *InicializujRozmer()*, respektive *PridejObrazek().*

Obrázek v třídě *KresliciPlatno* má sleduje tyto události:

- *Paint* - překreslení
- *MouseMove* - změna polohy kurzoru
- *MouseDown* - stlačení tlačítka myši
- *MouseUp* - puštění tlačítka myši
- *MouseEnter* - vstup kurzoru nad oblast
- *MouseWheel* - posunutí kolečka myši

Editor

Třída editor slouží jako spojení a interpretátor požadavků mezi třídou graf a uživatelským rozhraním.



Obrázek 24 - UML diagram třídy Editor

O část reakcí na události zde uvedené se stará třída *MainForm* programu.

Editor si jako datové složky drží instance tříd *KresliciPlatno* a *Graf*. Hlavní funkcí editoru je interpretace událostí vyvolaných uživatelem pomocí počítačové myši. Každou změnou polohy kurzoru automaticky ukládáme do vlastnosti *PolohaKurzoru()* pomocí události *MouseMove*. Při vyvolání události *MouseDown* ukládáme polohu kurzoru na místě stlačení tlačítka do vlastnosti *StlaceniPolohaKurzoru().* Při každé další změně polohy voláme metodu *graf.PridejPomocnouHranu().* Je důležité si uvědomit, že se volání pomocné hrany odehrává každou změnou polohy kurzoru, byť o jediný pixel. Výsledkem je dojem, že hrana "sleduje" kurzor v jeho pohybu. Pokud dojde k události *MouseUp*, kontrolujeme, zda se poloha puštění tlačítka liší od polohy stlačení. Pokud se neliší tak víme, že uživatel pouze kliknul a podle stisknutého tlačítka voláme odpovídající metody třídy Graf:

- levé tlačítko - *graf.PridejVrchol()*
- prostřední tlačítko (kolečko) - *graf.Odeber()*
- pravé tlačítko - *graf.AktivujPrvek()*

Jestliže se polohy liší, voláme metodu *graf.PridejHranu()* s parametry obou poloh. Pokud tato metoda vrátí instanci třídy *GrafPrvek* tak víme, že máme tento vrchol přesunout.

Při každém pohybu kurzoru nebo manipulaci s grafem voláme pro kreslící plátno metodu *Invalidate().* Tato metoda nařídí plátnu překreslení. Na překreslení reaguje událost *Paint*. Při zachycení této události nechá editor celý graf kompletně překreslit. V metodách pro vykreslení odesíláme současnou polohu kurzoru pro případnou identifikaci statusu "pod kurzorem" jednotlivých prvků grafu.

Událost *MouseEnter* z volá u kreslícího plátna metodu *Focus().* To je nutné pro správné fungovální scrollingu kolečkem myši jako reakci na událost *MouseWheel*.

Další metody slouží k předání pokynů z uživatelské vrstvy směrem ke třídě graf a předání výsledků zpět.

5 Aplikace na reálném příkladu

5.1 Výsledky algoritmů

Pomocí editoru byl zadán zjednodušený graf měst a silniční sítě Pardubického kraje. Jako mapový podklad posloužila rastrová mapa silniční sítě Pardubického kraje[7].

Postupně byla jako vrcholy grafu přidávána města s největší populací, poté města na některých důležitějších silničních uzlech. Graf má v konečné podobě 82 vrcholů a 198 hran. Počet obyvatel Pardubického kraje je přibližně 516 000, z toho je jich alokováno ve vrcholech tohoto grafu 345 000. Soubor s uloženým grafem je umístěn ve složce Uložené grafy na CD v příloze A.

Nejdříve manuálně zadáme do grafu současný stav rozmístěný výjezdových středisek ZZS. Výsledná celková dopravní práce činí 203 012 km/rok. Tato hodnota ale platí pouze pro předpoklad, že je na každém výjezdovém stanovišti pouze jedna výjezdová skupina. Další nepřesnost je dána postupem algoritmu pro umístění dep. Algoritmus počítá pouze s paprskovitým výjezdem z výjezdového stanoviště na místo určení a zase zpět. V reálně situaci bude pravděpodobně sanita přesouvat pacienta do nejbližší nemocnice. Navíc pokud se jedná o výjezd ve stejném vrcholu - městě počítá se hodnota dopravní práce jako 0. Hodnotu celkové dopravní práce budeme tedy spíše brát jako srovnávací kritérium, než reálně použitelnou hodnotu.

Dále aplikujeme algoritmus pro optimální umístění 15 dep, což je stejná hodnota jako současně umístěný počet výjezdových stanovišť. Výjezdová stanoviště v Pardubicích sloučíme do jednoho, to z toho důvodu, protože hledáme optimální umístění vzhledem k vrcholům a určení počtu stanovišť, ani počtu výjezdových skupin není naším cílem. Celková dopravní práce klesla přemístěním některých stanovišť do větších měst na 188 660 km/rok. Jelikož jde o optimalizaci pouze z hlediska dopravní práce je pravděpodobné, že by v této konfiguraci nastaly problémy s dojezdovou dobou sanitních vozidel. Nevelké snížení dopravní práce nám napovídá, že současné umístění výjezdových stanovišť je z hlediska dopravní práce relativně vyhovující.

Jako poslední aplikujeme na graf algoritmus pro plné pokrytí grafu podle zadaných parametrů. Jako průměrnou rychlost vozidel ZZS volíme 70 km/h a dojezdovou dobu 20 minut, z čehož jsou vyhrazeny 3 minuty na aktivaci posádky. Algoritmus tedy umístí výjezdová střediska ZZS vrcholově optimálně tak, aby výjezdové skupiny stihly obsloužit požadavek kdekoli na grafu do 20 minut od zpracování výzvy. Výsledkem algoritmu je 24 alokovaných dep v následujících umístěních:

[7] volně dostupné na adrese: http://www.rsd.cz/Mapy/Soubor-map---kraje

- Pardubice
    - Pardubice
    - Holice
    - Přelouč
    - Lázně Bohdaněč
    - Chvaletice
- Chrudim
    - Chrudim
    - Třemošnice
    - Trhová Kamenice
    - Skuteč
    - Heřmanův Městec
- Svitavy
    - Svitavy
    - Litomyšl
    - Moravská Třebová
    - Polička
    - Březová nad Svitavou
    - Jevíčko
- Ústí nad Orlicí
    - Ústí nad Orlicí
    - Lanškroun
    - Žamberk
    - Vysoké Mýto
    - Choceň
    - Česká Třebová
    - Králíky
    - Letohrad

Algoritmus ke stávajícímu stavu přidal 8 dalších výjezdových středisek. Za zmínku stojí, že nově plánovaná střediska v Chocni a Jevíčku jsou součástí vypočítaných středisek. Další nová výjezdová stanoviště umístěná algoritmem v Trhové Kamenici a Třemošnicích poměrně blízko sousedí s umístěním dalších nově plánovaných středisek na Seči a v Nasavrkách. Celková dopravní práce klesla na polovinu a to na hodnotu 95 103. Nabízí se otázka, jak je možné, že za současného stavu rozmístění výjezdových středisek dokážou sanity dodržet zákonem stanovenou lhůtu 20 minut. Přesné statistiky k dispozici nemáme, je však obecně známo, že k nedodržení tohoto limitu nezřídka dochází. Jedná se ale spíše o odlehlejší nebo okrajové oblasti. Obsluha okrajových oblastí kraje se dá řešit mezikrajovou spoluprácí, k níž relativně často také dochází. Soubor s grafem, na kterém jsou alokována tato depa je uložen ve složce Uložené grafy na CD v příloze A.

Ukázku nového umístění dep pomocí tohoto algoritmu najdete v příloze B.

5.2 Časová složitost řešení

Hlavní algoritmus pro pokrytí depy podle zadaných parametrů se skládá z několika dílčích operací.

- vypočtení matice vzdáleností
- vypočtení distanční matice
- iterativní algoritmus pro určení vrcholově optimální lokace *k* dep na síti (dále jen iterativní algoritmus)
- kontrola pokrytí

Dále uvedené složitosti jsou vždy horním odhadem asymptotické složitosti daných algoritmu, tedy takovým nejhorším možným scénářem z hlediska složitosti. Proměnná *n* udává počet vrcholů grafu.

Složitost výpočtu distanční matice je $O\left(n^2 + n^3\right)$, pokud musíme spočítat nejprve matici přímých vzdáleností.

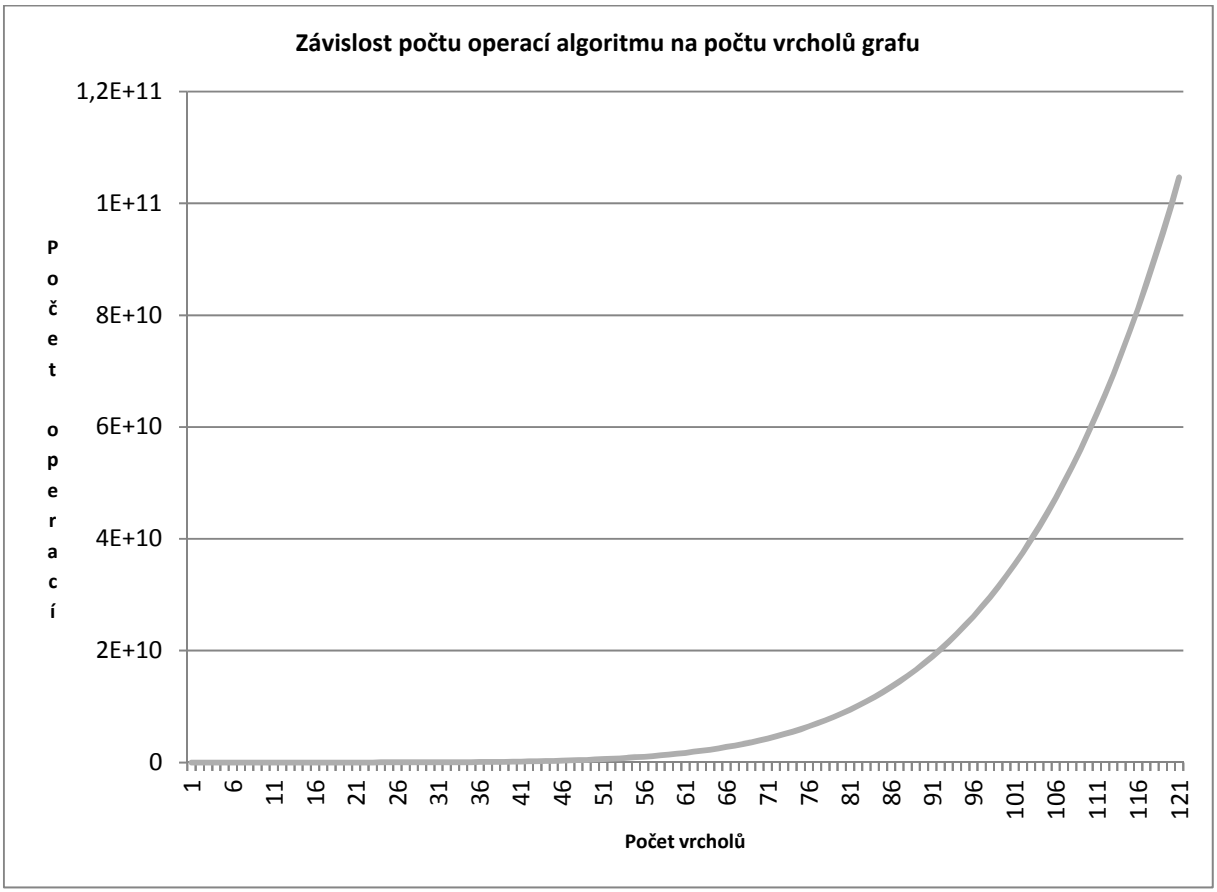
Horní odhad složitosti iterativního algoritmu je velmi nepříznivý. Naštěstí tím, že vybíráme nejprve nejvhodnější kandidáty na umístění depa dojde k výpočtu s touto složitostí s velmi malou pravděpodobností. Složitost iterativního algoritmu je $O\left(2d^2 + d + 2n + d\left(\left(n-d\right)\left(2d + d^3 + n\right)\right)\right)$, kde *d* je počet hledaných dep.

Kontrola pokrytí má složitost pouze $O\left(n\right)$.

Algoritmus na pokrytí dep zkouší "hrubou silou" najít minimální počet dep potřebných k pokrytí grafu. Toho dosáhne tím, že postupně inkrementuje počet hledaných dep iterativním algoritmem do doby, než jsou schopna nalezená depa pokrýt celý graf. V nejhorším případě tedy proběhne iterativní algoritmus *n* krát. Horní odhad složitosti toho řešení včetně přípravy algoritmu a kontroly pokrytí je:

$$O\left(n^2 + n^3 + \sum_{d=1}^{n}\left(2d^2 + d + 2n + d\left(\left(n-d\right)\left(2d + d^3 + n\right)\right) + n\right)\right)$$

Pomocí jednoduchého skriptu pro tabelaci této složitostní funkce sestrojíme graf závislosti počtu operací na počtu vrcholů.



Obrázek 25 - graf závislosti počtu operací algoritmu na počtu vrcholů grafu

Z grafu vyplývá, že tento algoritmus není příliš efektivní. Nad počet 90 vrcholů již stoupá složitost algoritmu do nepřípustných výšek. Pro přiložený graf s 82 vrcholy trvá vykonání algoritmu na průměrném PC přibližně 1 minutu, což je již na hranici použitelnosti. Je zřejmé, že pro aplikaci na grafu reálné silniční sítě, kde by mohl počet vrcholů dosahovat řádu tisíců, je algoritmus nepoužitelný. Na vině není ani tak samotný iterativní algoritmus, ale samotné jeho použití "hrubou silou". V tomto případě je tedy nutná aplikace nějaké formy heuristiky.

6 Závěr

V práci se podařilo splnit podmínky dané zadáním. Použitý aparát teorie grafů je dostatečně popsán, stejně jako problematika záchranné zdravotnické služby v ČR. Aplikace je plně použitelná pro práci s obyčejným grafem, který postačí k definování úlohy diskrétní lokace na grafu, potřebné pro řešení praktického příkladu.

Pro řešení praktického příkladu byl sestaven graf o 82 vrcholech a 198 hranách představující města a silniční síť Pardubického kraje. Jako vrcholy byli vybrány především větší města a obce nacházejících se na důležitějších silničních kříženích. Na tomto grafu jsme aplikoval implementované algoritmy. Z analýzy současného umístění výjezdových stanovišť ZZS vyplývá, že je toto umístění z hlediska optimalizace dopravní práce relativně dobré. Výrazných úspor dopravní práce není možno docílit bez přidání dalších výjezdových stanovišť. Algoritmus pro hledání optimálního umístění dep tak, aby mohla výjezdová stanoviště obsloužit celý Pardubický kraj, přidal ke stávajícím 16 výjezdovým stanovištím 8 dalších. Přidáním těchto výjezdových stanovišť klesá dopravní práce oproti současnému stavu na polovinu. Za zmínku stojí, že ze 4 nově plánovaných umístění se 2 objevili ve vypočítaném výčtu nových umístění. Zbývající 2 plánovaná stanoviště těsně sousedí s lokací nově vypočtených umístění stanovišť.

Vzhledem k počtu nových výjezdových stanovišť vyvstává otázka, jak za současného stavu dokáže ZZS udržet dojezdovou dobu pod zákonem stanovených 20 minutách a to po celém Pardubickém kraji. Poznatky s praxe se shodují s výsledky algoritmu, za současného stavu toto není technicky možné.

Program tedy zadanou úlohu zvládne vyřešit, nicméně vzhledem k mnohým zjednodušením se nedají jeho výsledky považovat za dostatečně průkazné pro praktické využití.

Důležitým vylepšením by mohlo být použití specializovaných datových struktur pro ukládání grafu, čímž by se znatelně snížila složitost použitých algoritmů. Dalším vhodným vylepšením je přechod z diskrétní lokace na lokaci spojitou, protože v našem případě může dojít k požadavkům na obsluhu kdekoli na silniční síti a ne pouze ve vrcholech představující města. Analýzou použitého lokačního algoritmu jsem došel k závěru, že tento algoritmus není v současné podobě vhodný pro úlohy většího rozsahu než je tato. Zde je tedy vhodné aplikovat heuristiku, ke snížení časové složitosti algoritmu.

Práce mne velice obohatila zkušenostmi s programováním aplikace většího rozsahu než krátké seminární práce. Jako velice zajímavou hodnotím osobní návštěvu výjezdového stanoviště ZZS, kterou jsem absolvoval za účelem sběru poznatků pro tuto práci.

7 Použitá literatura a další zdroje

[1] VOLEK, Josef. Operační výzkum I. Pardubice : Univerzita Pardubice, 2002. 111 s. ISBN 80-7194-410-6.

[2] SEDLÁČEK, Jiří. Kombinatorika v teorii i praxi : úvod do teorie grafů. Praha : Československá akademie věd, 1964. 150s. ISBN 000574264.

[3] CHRISTOFIDES, Nicos. Graph theory : An algorithmic approach. Orlando, FL, USA : Academic Press, Inc., 1975. 415 s. ISBN 0121743500.

[4] VOLEK, Josef; LINDA, Bohdan. Teorie grafů - aplikace v dopravě a veřejné správě. Dopravní fakulta Jana Pernera. Pardubice : Univerzita Pardubice, 2012. 192 s. ISBN 978-80-7395-225-9

[5] KANISOVÁ, Hana a Miroslav MÜLLER. UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006. 176 s. ISBN 80-251-1083-4.

[6] KAVIČKA, Antonín. Datové struktury [intranet]. Pardubice, 2010 [cit. 2013-05-30] Dostupné z: Studijní agendy Univerzity Pardubice.

[7] ČERNÝ, Jakub. Základní grafové algoritmy. Praha, 2010 Dostupné z: http://kam.mff.cuni.cz/~kuba/ka/

[8] MVČR: Počty obyvatel v obcích [online]. 2012 [cit. 2013-05-30]. Dostupné z: WWW: <http://www.mvcr.cz/clanek/statistiky-pocty-obyvatel-v-obcich.aspx>.

[9] Silniční a dálniční síť ČR [online]. 2011 [cit. 2013-05-30]. Dostupné z: <http://www.rsd.cz/doprava/silnicni_sit/start.htm>.

[10] FRANĚK, O. Záchranná služba: Nezávislý web o zdravotnické záchranné službě [online]. 2012, [cit. 2013-05-30].Dostupné z:<http://www.zachrannasluzba.cz/>.

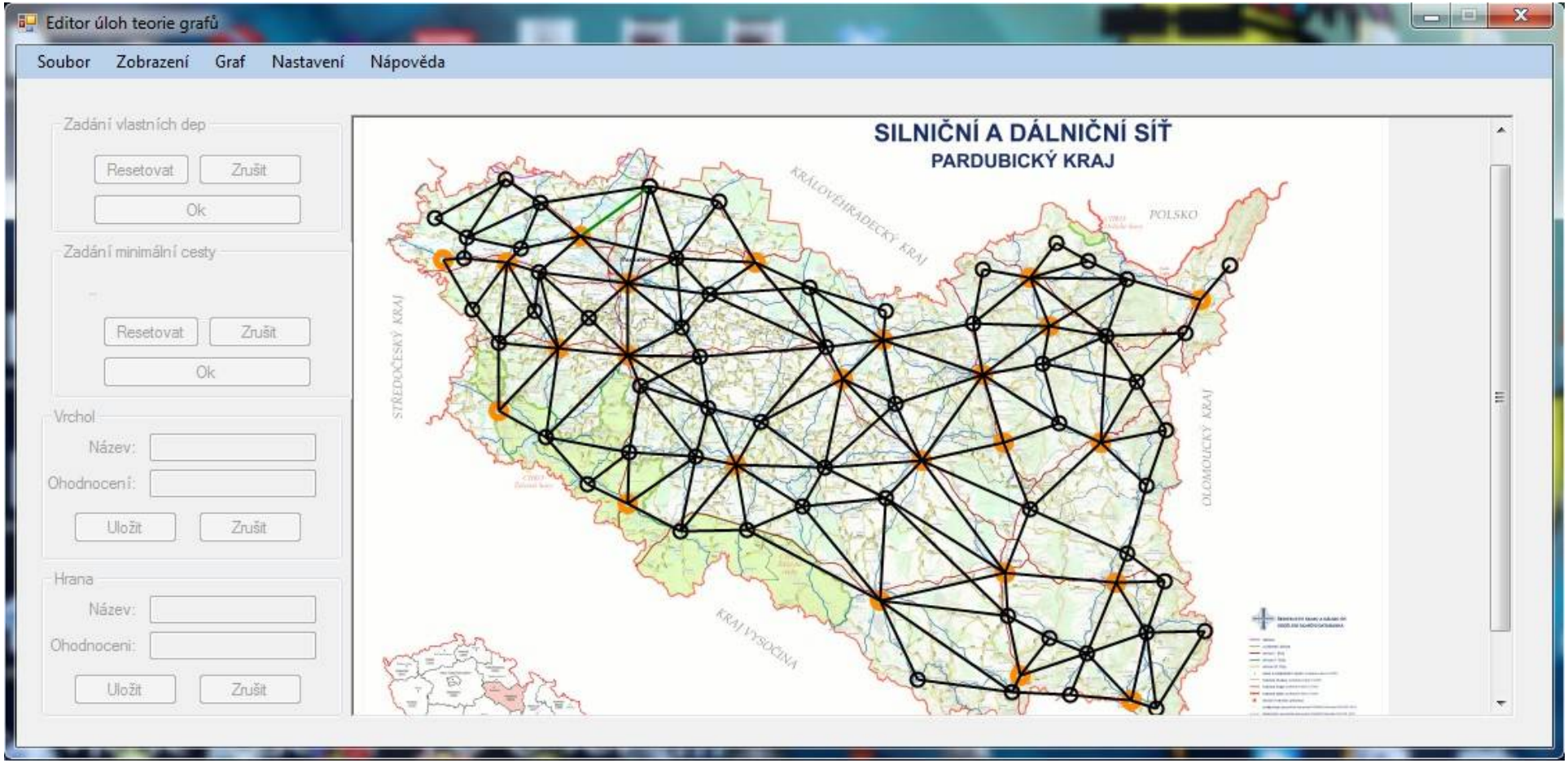
[11] Souhrnné statistiky ÚZIS pro obor ZZS [online]. 2012 [cit. 2013-05-30]. Dostupné z: <http://www.urgmed.cz/uzis/uzis.htm>.

[12] Vybrané ukazatele ZZS ČR [online]. 2012 [cit. 2013-05-30]. Dostupné z: < http://www.azzs.cz/dokumenty/zzs-cr-v-cislech/>.

8 Příloha A

CD s přiloženým programem.

9 Příloha B


Editor úloh teorie grafů
Soubor
Zobrazení
Graf
Nastavení
Nápověda
Zadání vlastních dep
Resetovat
Zrušit
Ok
Zadání minimální cesty
Resetovat
Zrušit
Ok
Vrchol
Název:
Ohodnocení:
Uložit
Zrušit
Hrana
Název:
Ohodnocení:
Uložit
Zrušit
SILNIČNÍ A DÁLNIČNÍ SÍŤ
PARDUBICKÝ KRAJ
KRÁLOVÉHRADECKÝ KRAJ
POLSKO
STŘEDOČESKÝ KRAJ
OLOMOUCKÝ KRAJ
KRAJ VYSOČINA