UNIVERZITA PALACKÉHO V OLOMOUCI

PŘÍRODOVĚDECKÁ FAKULTA

KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

DIPLOMOVÁ PRÁCE

ANALÝZA ZRANITELNOSTI SILNIČNÍ SÍTĚ

Vedoucí diplomové práce: RNDr. Rostislav Vodák, Ph.D.

Vypracovala: Bc. Pavla Doleželová

Studijní program: Aplikovaná matematika

Studijní obor: Aplikace matematiky v ekonomii

Forma studia: prezenční

Rok odevzdání: 2015

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Bc. Pavla Doleželová

Název práce: Analýza zranitelnosti silniční sítě

Typ práce: Diplomová práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: RNDr. Rostislav Vodák, Ph.D.

Rok obhajoby práce: 2015

Abstrakt: Diplomová práce se zabývá analýzou zranitelnosti silniční sítě. Analýza zranitelnosti silniční sítě byla provedena pomocí pěti modifikací algoritmu simulovaného žíhání. První kapitola pojednává o poznatcích z teorie grafů, které jsou důležité k pochopení ostatních kapitol. Druhá kapitola obsahuje popis algoritmů a metod nutných pro sestavení algoritmu simulovaného žíhání. Ve třetí kapitole jsou uvedeny všechny modifikace simulovaného žíhání. Čtvrtá kapitola obsahuje výsledky aplikace modifikací.

Klíčová slova: simulované žíhání, normovaný betweenness index, teorie grafů, silniční sítě, nejkratší cesta, komponenty.

Počet stran: 49

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Bc. Pavla Doleželová

Title: Analysis of vulnerability of a road network

Type of thesis: Diploma thesis

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: RNDr. Rostislav Vodák, Ph.D.

The year of presentation: 2015

Abstract: This thesis deals with analysis of vulnerability of a road network. The analysis of vulnerability of the road network was performed using a modification of the five simulated annealing algorithm. The first chapter deals with the knowledge of graph theory, which is important for understanding other chapters. The second chapter contains a description of methods and algorithms necessary to compile a simulated annealing algorithm. In the third chapter there are all modifications simulated annealing. The fourth chapter contains the results of the application modifications.

Key words: simulated annealing, normalized betweenness index, graph theory, the road network, shortest path and components.

Number of pages: 49

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracovala samostatně pod vedením pana RNDr. Rostislava Vodáka, Ph.D. s použitím uvedené literatury.

V Olomouci dne ……

.....

Bc. Pavla Doleželová

Poděkování

Na tomto místě bych chtěla poděkovat především svému vedoucímu diplomové práce, panu RNDr. Rostislavu Vodákovi, Ph.D., za odborné vedení, cenné rady a čas, který mi věnoval. Také bych chtěla poděkovat Centru dopravního výzkumu, v. v. i. za konzultace a poskytnutí dat silničních sítí v rámci projektu TRISK č. VG20102015057.

OBSAH

ÚVOD

Budování silniční sítě a její obnova. Téma dnešní moderní společnosti, která si již ve většině případů nedovede představit, že by neměla možnost přepravit se z jednoho místa na místo jiné. Dalo by se říct, že jsme se na silničních sítích stali závislí. Měli bychom tedy hledat i jinou alternativu dopravy. V případě silniční sítě se tato alternativa hledá ale jen velmi obtížně. Jak tedy zabezpečit, aby nedošlo k dlouhodobému výpadku možnosti přepravy z jednoho místa na druhé? Řešením by mohlo být zajistit, aby silniční síť byla odolná a jen velmi málo zranitelná. V prvé řadě musíme stanovit, kde má daná silniční síť slabá místa. Právě k tomu slouží analýza zranitelnosti silniční sítě.

Silniční síť může být postihnuta výpadky, které mají různé příčiny. Mohou to být výpadky plánované (např. oprava částí dálnice D1), ale i výpadky neplánované. Může to být i kombinace obou druhů výpadků, např. nehoda v omezeném provozu na dálnici D1. V této práci se budeme zabývat výpadky v síti bez zkoumání jejich příčiny.

Cílem této diplomové práce je zkoumat zranitelnost silniční sítě s pomocí aplikace simulovaného žíhání. Simulované žíhání je jedna z metod pro minimalizaci účelové funkce a jeho hlavní výhodou je, že připouští i dočasné zhoršení hodnot účelové funkce. Celkem si představíme pět verzí simulovaného žíhání a získané výsledky mezi sebou porovnáme.

Práci jsme rozdělili do čtyř kapitol. V první kapitole uvedeme některé poznatky z oblasti teorie grafů. I když se jedná pouze o teoretické poznatky, je velmi důležitá pro pochopení ostatního textu a pro vytvoření samotných modifikací simulovaného žíhání.

Ve druhé kapitole si představíme algoritmy a metody, které tvoří stěžejní část této práce. Jelikož uvažujeme, že lidé při používání silničních sítí při přepravě z místa A na místo B využívají nejkratších cest, potřebujeme tyto nejkratší cesty stanovit. Pro vyřešení tohoto problému jsme zvolili Dijkstrův algoritmus. Pro stanovení, jak je daná síť zranitelná, potřebujeme určit i komponenty souvislosti. Hlavní částí této kapitoly bude představení samotného algoritmu simulovaného žíhání. Zvolili jsme právě tento algoritmus, jelikož se řadí mezi stochastické a připouští i dočasné přijetí zhoršujícího řešení. Díky této vlastnosti se můžeme vyhnout uváznutí v lokálním extrému.

Ve třetí kapitole se již dostaneme k samotné analýze zranitelnosti silniční sítě. Nejprve si uvedeme samotnou definici zranitelnosti, tak jak ji budeme chápat my, a ukážeme reálná data, se kterými budeme pracovat. Dále představíme účelovou funkci, pomocí které budeme hodnotit výsledky všech silničních sítí. V této kapitole také

představíme a popíšeme jednotlivé modifikace simulovaného žíhání. Těchto modifikací je celkem pět.

V poslední, čtvrté kapitole uvedeme výsledky všech modifikací simulovaného žíhání.

Všechny algoritmy jsme naprogramovali v prostředí Scilab ve verzi 5.4.0. Vytvořené programy jsou ve stručnosti popsány v příloze a jejich přesné znění jsme uložili na přiloženém CD. Veškerá data i obrázky sítí, které jsme v práci uvedli, nám poskytlo Centrum dopravního výzkumu, v. v. i..

1 ZÁKLADNÍ POJMY Z TEORIE GRAFŮ

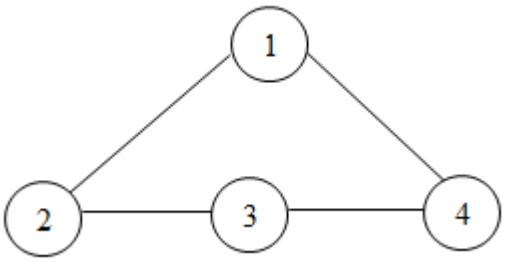
V diplomové práci budeme zkoumat zranitelnost silniční sítě. Abychom mohli se silniční sítí lépe pracovat, znázorníme si ji do grafu a pro práci s ní využijeme některých poznatků z teorie grafů. Jedná se především o samotnou definici grafu, typy grafů, cestu a souvislost v grafu. V této kapitole tyto pojmy uvedeme a pro jejich lepší pochopení využijeme také grafického znázornění. V této kapitole budeme čerpat z knih [3], [4] a [5].

1.1 Definice grafu a typy grafů

V této diplomové práci se nebudeme zabývat grafy užívanými např. ve statistice ani grafickým znázorněním průběhu funkce. V našem případě využijeme grafy ke zjednodušení reálných stavů. Reálný stav zaznačíme pomocí uzlů a čar, které tyto uzly navzájem spojují a vyjadřují tak jejich vzájemné vazby. V teorii grafů uzly můžeme nazvat i jako vrcholy a čáry, které je spojují, pak nazýváme hranami.

Graf G je tedy definován jako uspořádaná dvojice *(V, E)*, kde *V* představuje množinu vrcholů a *E* množinu hran. Množina vrcholů *V* grafu *G* je neprázdná množina a označujeme ji *V(G)*. Množina hran *E* grafu *G* je množina dvouprvkových podmnožin množiny *V* a označujeme ji *E(G)*. Graf s množinou vrcholů *V* a s množinou hran *E* zapíšeme jako *G(V, E)*.

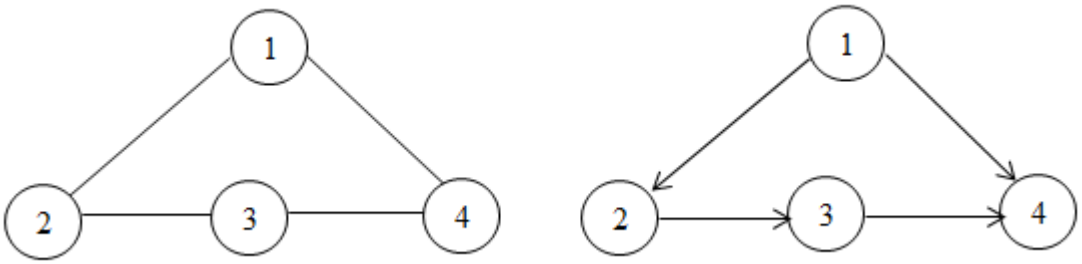
Graf často znázorňujeme pomocí diagramu, ve kterém jsou vrcholy značeny jako kroužky a hrany jako křivky spojující dvojice vrcholů. Můžeme jej také zadat výčtem vrcholů a hran. Na níže uvedeném obrázku vidíme jednoduchý graf se čtyřmi vrcholy a se čtyřmi hranami. Množina vrcholů V obsahuje vrcholy 1, 2, 3 a 4 tj. $V = \{1, 2, 3, 4\}$. Množina hran E vypadá následovně: $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$.



Obrázek 1.1 Neorientovaný graf

K vrcholu může patřit právě jedna hrana nebo více různých hran, ale může nastat i situace, kdy k vrcholu nenáleží žádná hrana. O hraně spojující dva vrcholy říkáme, že je s těmito vrcholy *incidentní*. Vrcholy, které jsou spojeny hranou, pak nazýváme *sousedními* vrcholy.

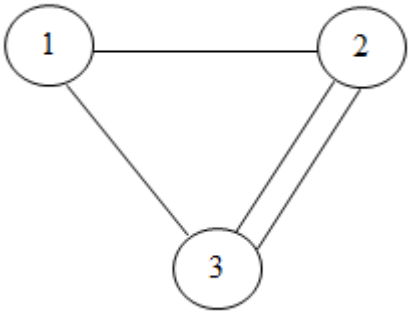
Rovněž ještě můžeme rozlišovat grafy *orientované* a *neorientované*. Orientovaný graf je takový graf, u nějž přesně určíme počáteční a koncový vrchol a říkáme, že hrana vede z počátečního do koncového vrcholu. Hrany v orientovaném grafu představují uspořádanou dvojici vrcholů. Graficky se taková hrana znázorňuje šipkou od počátečního do koncového uzlu. Definice orientovaného grafu je následující. Orientovaný graf *D* je graf *D(V, A)*, kde *V* je množina vrcholů a *A* je množina uspořádaných dvojic vrcholů (tj. dvouprvkových uspořádaných podmnožin množiny *V*). Hranám *A* orientovaného grafu *D* říkáme orientované hrany. Na níže uvedeném obrázku je znázorněný rozdíl mezi orientovaným a neorientovaným grafem.



Obrázek 1.2 Neorientovaný a orientovaný graf

Neorientovaný graf můžeme rovněž převést na orientovaný a to tak, že místo jedné hrany budeme používat hrany dvě s navzájem opačnou orientací.

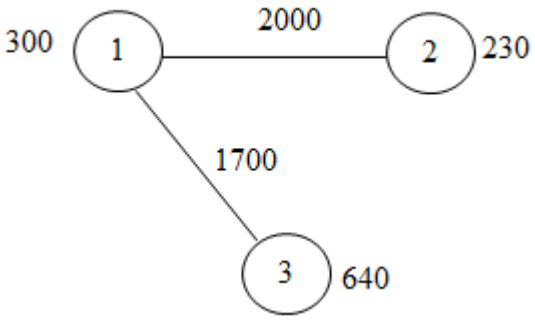
Někdy se také můžeme setkat s tzv. *multigrafem*. V multigrafu připouštíme více hran mezi dvojicí uzlů. Takové hrany označujeme jako *násobné*. Pokud jsou v grafu jak násobné hrany, tak i hrany incidentní pouze s jedním vrcholem, hovoříme o *pseudografu*. Hrany, které mají počáteční i koncový uzel totožný, nazýváme *smyčkou*. Opak pseudografu je obyčejný graf (viz. obrázek 1.1), který neobsahuje násobné hrany ani smyčky. S multigrafem ani pseudografem se v diplomové práci nesetkáme, ale pro názornost je níže uvedený obrázek, jak takový multigraf vypadá.



Obrázek 1.3 Multigraf

V mnoha reálných aplikacích si nevystačíme pouze s grafem, který je dán výčtem uzlů a hran nebo znázorněn obrázkem. Také potřebujeme, aby graf s sebou nesl určité informace. Proto vrcholům i hranám přiřazujeme různá data, která reprezentují pro nás podstatné informace. V našem případě budeme zkoumat silniční síť, vrcholy mohou např. představovat obce a každý vrchol má přiřazený počet obyvatel žijících v dané obci. Hranami můžeme znázornit silnice mezi obcemi. Hrany obsahují informaci např. o délce silnice v metrech nebo o době průjezdu v sekundách.

Graf skládající se z takovýchto vrcholů a hran označujeme jako *vrcholově,* resp. *hranově ohodnocený*. Neohodnocený graf je speciálním případem ohodnoceného grafu, ve kterém mají všechny hrany ohodnocení jedna. Na níže uvedeném obrázku 1.4 je znázorněno, jak vypadá graf ohodnocený hranově i vrcholově. Právě takové grafy budeme v diplomové práci využívat.

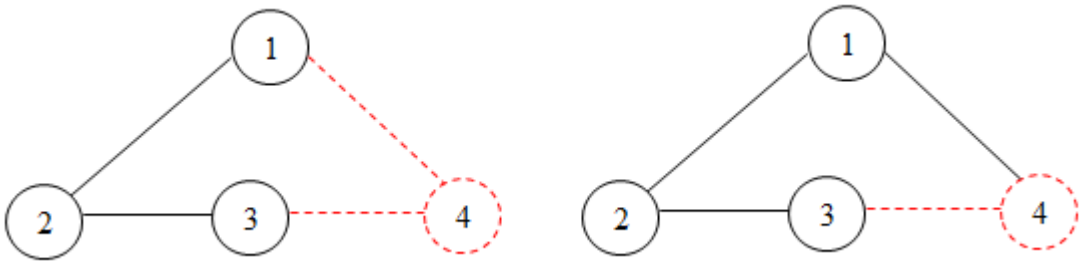


Obrázek 1.4 Hranově i vrcholově ohodnocený graf

1.2 Podgraf, cesta, souvislost

Vynecháváním některých vrcholů a hran z původního grafu *G(V, E)* získáme nový graf *G´(V, E)*, o kterém říkáme, že je *podgrafem* grafu *G*. Pro podgraf *G´* platí $V(G') \subseteq V(G)$ a zároveň $E(G') \subseteq E(G)$.

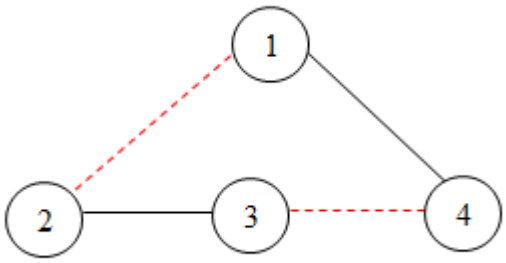
Abychom vytvořili podgraf, vypustíme z původního grafu *G* některé vrcholy a hrany. Zároveň musíme dávat pozor, abychom v grafu neponechali hranu, jejíž vrchol jsme vypustili. Porušili bychom definici grafu, která říká, že hrana je dvouprvková podmnožina množiny vrcholů. Na obrázku 1.5 jsme znázornili, jak se takový podgraf tvoří. V prvním grafu jsme vynechali jeden vrchol (označený červenou přerušovanou čárou) a odebrali jsme také obě hrany incidentní s tímto vrcholem. Takto upravený graf splňuje definice podgrafu. V případě druhého grafu jsme také odebrali jeden vrchol, ale pouze jednu hranu s ním incidentní. Zůstane nám tedy jedna hrana, která nevede do žádného vrcholu. Touto úpravou nevytvoříme podgraf, ale ještě navíc získaný výsledek není ani grafem.



Obrázek 1.5 Správná a špatná tvorba podgrafu

Jestliže z grafu *G* vynecháme některé vrcholy a pouze ty hrany, které jsou s těmi vrcholy incidentní, vznikne *indukovaný podgraf G´* grafu *G*. Na obrázku 1.5 je levý graf právě indukovaný podgraf.

Pokud z grafu *G* vypustíme pouze některé hrany, přičemž množina vrcholů zůstane nezměněna, získáme *faktor G´´* grafu *G*. Pro faktor G´´ grafu G platí: $V(G'') = V(G)$ a $E(G'') \subseteq E(G)$. Na obrázku 1.6 jsme znázornili, jak takový faktor může vypadat.



Obrázek 1.6 Faktor grafu

Abychom mohli s grafem pracovat, potřebujeme vědět, jak se v něm můžeme pohybovat. Nemůžeme „skákat“ z jednoho vrcholu do druhého, jak nás napadne. Musíme dodržovat určitá pravidla.

Prvním pojmem je *sled* neboli procházka. Sled chápeme jako posloupnost vrcholů a hran

$$W = (v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n),$$

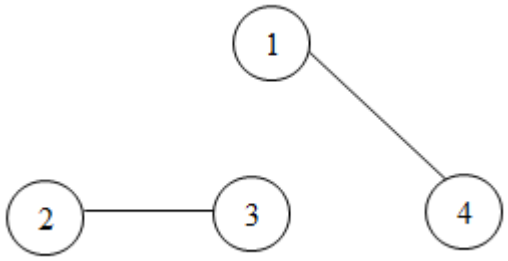
kde $v_j \in V(G)$ pro $j = 0, 1, \ldots, n$ a $e_i \in E(G)$ pro $i = 1, 2, \ldots, n$, přičemž hrana e_i spojuje vrcholy v_{i-1} a v_i. Vrchol v_0 je počáteční a vrchol v_n je koncový vrchol. O takovémto sledu říkáme, že vede z vrcholu v_0 do vrcholu v_n.

Jestliže je počáteční i koncový vrchol totožný, $v_0 = v_n$, jedná se o *uzavřený sled*. V opačném případě hovoříme o *otevřeném sledu*. Otevřený sled, ve kterém se neopakuje žádný vrchol, nazýváme *cestou*.

V ohodnoceném i neohodnoceném grafu jsme schopni určit *délku cesty*. Jestliže máme hranově ohodnocený graf, sečteme ohodnocení daných hran. V hranově neohodnoceném grafu uvažujeme, že každá hrana má délku jedna. Délka cesty v takovém grafu je rovna počtu hran, které se na cestě nachází. Kromě samotné cesty nás mnohdy zajímá i *délka nejkratší cesty* mezi vrcholy *u* a *v*. Hledání nejkratší cesty je věnována kapitola 2.1.

Jestliže máme graf modelující silniční síť, zajímá nás, jestli se z vrcholu *u* dostaneme do vrcholu *v*. Graf, ve kterém pro každé dva vrcholy *u* a *v* existuje alespoň jedna cesta z *u* do *v*, nazýváme *souvislým*. V opačném případě se jedná o graf *nesouvislý*. Nesouvislý graf se skládá z několika souvislých částí, které označujeme jako *komponenty*. Komponentu tvoří každý podgraf, jenž je souvislý a zároveň obsahuje maximální počet

vrcholů a hran původního grafu. Na obrázku 1.7 je uvedený nesouvislý graf se dvěma komponentami.



Obrázek 1.7 Nesouvislý graf

1.3 Způsoby zadávání grafů do počítače

Doposud jsme pro znázornění grafů využívali pouze obrázků. V případě rozsáhlejších sítí, nebo pokud chceme s grafy pracovat na počítači, si nevystačíme jen s pouhým obrázkem. Existuje několik možností, které se liší ve složitosti zadávání i v univerzálnosti použití. Mezi způsoby zadávání grafů patří např. seznam vrcholů a hran, matice sousednosti, incidenční matice a seznam následníků.

V této diplomové práci budeme graf zadávat právě pomocí seznamu následníků. Tuto metodu tedy blíže popíšeme a znázorníme na jednoduchém příkladu. O zbylých metodách se pouze stručně zmíníme.

1.3.1 Seznam vrcholů a hran

V této metodě je množina vrcholů popsána seznamem prvků a množina hran počátečním a koncovým vrcholem dané hrany. Mezi hlavní přednosti tohoto způsobu patří univerzálnost a relativní úspornost. Není ale vhodný pro rozsáhlejší grafy. Např. graf na obrázku 1.1 můžeme přepsat pomocí seznamu vrcholů a hran následovně: Množina vrcholů V obsahuje vrcholy 1, 2, 3 a 4, $V = \{1, 2, 3, 4\}$. Množina hran E vypadá takto: $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$.

1.3.2 Matice sousednosti

Matice sousednosti obsahuje informace, které vrcholy jsou sousední a které nikoliv. Matice sousednosti jednoduchého grafu $G = (V, E)$ je čtvercová matice $M = (m_{ij})_{i,j=1}^{n}$ definována předpisem

$$m_{ij} = \begin{cases} 1, & pro\ \{v_i, v_j\} \in E \\ 0, & jinak. \end{cases}$$

Matice je tvořená pouze jedničkami a nulami, a pokud v grafu nejsou smyčky, na diagonále se vyskytují jenom nuly. Součet *i*-tého řádku nebo *i*-tého sloupce udává stupeň vrcholu v_i, který značí, s kolika hranami je daný vrchol incidentní (nebo také počet sousedních vrcholů). Výhodou tohoto způsobu je jeho jednoduchá reprezentace. Pro grafy s malým počtem hran ale obsahuje velké množství nul a vyhledávání je časově náročné.

Graf na obrázku 1.1 zapíšeme pomocí matice sousednosti následovně:

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

1.3.3 Incidenční matice

Podobně jako matice sousednosti se tvoří incidenční matice. Prvky matice jsou také pouze 1 a 0, ale nehledáme sousední vrcholy, nýbrž hrany, které jsou s daným vrcholem incidentní.

Incidenční matice B grafu *G = (V, E)* s *n* vrcholy $v_1, v_2, \ldots, v_n$ a *m* hranami $e_1, e_2, \ldots, e_m$ je matice typu $n \times m$ s prvky b_{ij}, i = 1, 2, …, n; j = 1, 2, …, m; kde

$$b_{ij} = \begin{cases} 1, & pokud\ je\ vrchol\ v_i\ incidentní\ s\ hranou\ e_j \\ 0, & jinak. \end{cases}$$

Součet prvků *i*-tého řádku udává stupeň vrcholu v_i. Součet prvků j-tého sloupce je vždy 2. Oproti matici sousednosti má incidenční matice na diagonále jedničky. Výhodou je opět jednoduchá reprezentace grafu. Nevýhodou je rozsáhlost formy zápisu pro grafy s vyšším počtem hran.

Graf z obrázku 1.1 zapíšeme pomocí incidenční matice následovně:

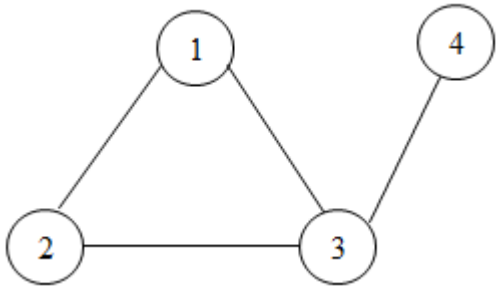
$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

1.3.4 Seznam následníků

V diplomové práci budeme využívat právě seznam následníků. Jedná se o jednoduchou a úspornou reprezentaci grafu. Ke každému vrcholu v grafu si pamatujeme seznam čísel sousedních vrcholů. Při aplikaci této metody používáme dvě jednorozměrná pole, budeme je značit UKAZ a NASL. Do pole NASL ukládáme čísla všech následníků,

do pole UKAZ pak indexy určující, kde v poli NASL jsou uloženi následníci daného vrcholu.

Jak se vytvoří seznam následníků z daného grafu, si ukážeme na následujícím příkladu. Mějme graf zadaný nejprve obrázkem 1.8.



Obrázek 1.8 Graf pro tvorbu seznamu následníků

Následně graf přepíšeme právě pomocí uzlů a následníků do dvou polí. Jejich tvorbu popíšeme níže.

Číslo vrcholu	1	2	3	4	5
UKAZ	1	3	5	8	9

Tabulka 1.1 Pole UKAZ

Index	1	2	3	4	5	6	7	8
NASL	2	3	1	3	1	2	4	3

Tabulka 1.2 Pole NASL

Pole UKAZ má rozměr $n + 1$ a pole NASL $2n$ (v případě neorientovaného grafu). Pole UKAZ má rozměr o jedničku větší, než je počet vrcholů jen z technických důvodů, aby bylo možné procházet následníky i posledního n-tého uzlu.

Pole UKAZ zaznamenává, na kterém místě v poli NASL najdeme následníky (sousední vrcholy) vrcholů. Např. následníci uzlu 3 začínají na 5. pozici a končí na pozici 7. Konkrétně se jedná o uzly 1, 2 a 4.

Nyní si popíšeme, jak taková pole vytvoříme a jak je použijeme. Předpokládejme, že máme graf G s vrcholy očíslovanými 1, 2, …, n. Do pole UKAZ napíšeme na první pozici jedničku, která značí, na které pozici najdeme v poli NASL prvního následníka vrcholu v_1. Do pole NASL, začínáme v prvním políčku, napíšeme čísla jeho následníků. Opět se vrátíme do pole UKAZ a zaznačíme do něj číslo pozice, na které se bude nacházet

první následník dalšího uzlu. Takto pokračujeme i pro další uzly. Na konec napíšeme číslo uzlu, $n + 1$.

Následníky vrcholu v_i zjistíme tak, že v poli UKAZ(i) najdeme odpovídající pozici v poli NASL. Následníci dalšího vrcholu v_{i+1} začínající na pozici UKAZ($i + 1$), takže následníci vrcholu v_i končí na pozici UKAZ($i + 1$) – 1. Jestliže uzel nemá žádné následníky, položíme UKAZ(i) = UKAZ($i + 1$).

2 Použité algoritmy a metody

V této práci využíváme několik základních algoritmů a metod, které si nyní vysvětlíme. Pro analýzu zranitelnosti silniční sítě potřebujeme znát nejkratší cesty mezi všemi uzly. Ukážeme si tedy algoritmus, kterým budeme tyto cesty hledat. Kromě zjištění nejkratších cest budeme potřebovat určit počet komponent, na které se graf rozpadne po zneprůjezdnění určitého počtu hran. Jako stěžejní metodu pro analýzu zranitelnosti silniční sítě jsme zvolili simulované žíhání. Jedná se o jednu z metod pro minimalizaci účelové funkce, která měří zranitelnost sítě. Vysvětlíme, jak můžeme simulované žíhání aplikovat na náš problém a také představíme jeho algoritmus. Budeme čerpat z knih [7], [9] a [10].

2.1 Dijkstrův algoritmus

Jedním z prvních kroků při práci se silniční sítí bývá většinou nalezení nejkratších cest. Ani my nebudeme výjimkou a budeme hledat nejkratší cesty ze všech vrcholů do všech ostatních. Nalezení těchto nejkratších cest nám pomůže při identifikaci, které hrany jsou pravděpodobně nejvíce využívané.

Jelikož máme ohodnocený graf, jehož ohodnocení je nezáporné, zvolíme si pro výpočet nejkratších cest *Dijkstrův algoritmus*. Kromě samotné délky nejkratší cesty pomocí tohoto algoritmu také zjistíme, které hrany nejkratší cestu tvoří.

Dijkstrův algoritmus začíná ve zvoleném výchozím vrcholu a postupně prochází všechny ostatní dostupné vrcholy. Výpočet probíhá v několika krocích. Nejprve ale musíme výchozímu vrcholu přiřadit dočasnou vzdálenost nula a všem ostatním vrcholům dočasnou hodnotu nekonečno, jelikož do nich prozatím žádné kratší cesty neznáme. Tyto hodnoty se budou v průběhu výpočtu měnit, dokud cílovému vrcholu, nebo všem dostupným vrcholům, nepřiřadíme trvalou hodnotu. Trvalá hodnota znamená, že jsme již nalezli nejkratší cestu do daného vrcholu. Naopak u dočasné hodnoty je ještě možnost, že se hodnota sníží. Samotný algoritmus probíhá následovně:

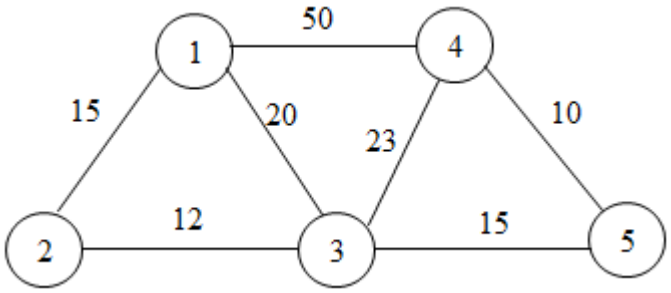
1. Vybereme vrchol x s nejmenší dočasnou hodnotou.
2. Tuto dočasnou hodnotu změníme na trvalou.
3. Všem jeho následníkům s dočasnou hodnotou přepočítáme jejich vzdálenosti:

$$h_i = min(h_i, h_x + d_{xi}),$$

kde i označuje číslo následníka vrcholu x, h_i je délka nejkratší cesty do i-tého vrcholu, d_{xi} je vzdálenost mezi vrcholy x a i.

Tento postup je konečný, protože v každém kroku je jednomu vrcholu změněna hodnota z dočasné na trvalou. Jestliže graf obsahuje *N* vrcholů, výpočet skončí nejvýše po *N* krocích. Správnost algoritmu vyplývá z volby vrcholu, jenž má v aktuálním okamžiku nejmenší dočasnou hodnotou. Tato dočasná hodnota totiž vede jen přes vrcholy, které mají přiřazenu trvalou hodnotu, a ta se již nemůže snížit.

Nyní si na příkladu (graf na Obrázku 2.1) předvedeme, jak hledání nejkratší cesty pomocí Dijkstrova algoritmu probíhá. Budeme hledat nejkratší cestu pouze z prvního vrcholu do všech ostatních. Výpočet budeme realizovat pomocí tabulky s pěti řádky. Do prvního řádku jsme napsali čísla uzlů. Do druhého řádku ukládáme délku nejkratší cesty, do třetího, jestli je délka již konečná (1) nebo pouze dočasná (0). Do čtvrtého řádku ukládáme vrchol, který leží na nejkratší cestě před daným vrcholem. A do pátého řádku zaznamenáváme číslo hrany ležící mezi daným vrcholem a jeho předchůdcem. Nejkratší cestu poté čteme od cílového vrcholu k vrcholu výchozímu (čili pozpátku).



Obrázek 2.1 Graf pro ukázku Dijkstrova algoritmu

Nejprve si tedy vytvoříme tabulku se vstupními údaji:

1	2	3	4	5
0	Inf	Inf	Inf	Inf
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Hodnoty v tabulce budeme v průběhu výpočtu měnit.

Krok 1:

1	2	3	4	5
0	15	20	50	Inf
1	0	0	0	0
0	1	1	1	0
0	(1, 2)	(1, 3)	(1, 4)	0

Krok 2:

1	2	3	4	5
0	15	20	50	Inf
1	1	0	0	0
0	1	1	1	0
0	(1, 2)	(1, 3)	(1, 4)	0

Krok 3:

1	2	3	4	5
0	15	20	43	35
1	1	1	0	0
0	1	1	3	3
0	(1, 2)	(1, 3)	(3, 4)	(3, 5)

Krok 4:

1	2	3	4	5
0	15	20	43	35
1	1	1	0	1
0	1	1	3	3
0	(1, 2)	(1, 3)	(3, 4)	(3, 5)

Krok 5:

1	2	3	4	5
0	15	20	43	35
1	1	1	1	1
0	1	1	3	3
0	(1, 2)	(1, 3)	(3, 4)	(3, 5)

Poslední tabulka nám říká, že nejkratší cesta z vrcholu 1 do vrcholu 2 vede přes vrcholy 1 -2. Nejkratší cesta z vrcholu 1 do vrcholu 3 vede přes vrcholy 1 – 3, do vrcholu 4 jdeme přes vrcholy 1 – 3 – 4. A nejkratší cesta do vrcholu 5 je přes vrcholy 1 – 3 – 5. To samé můžeme udělat pomocí hran. Např. nejkratší cesta z vrcholu 1 do vrcholu 5 vede přes hrany (1, 3), (3, 5). Z tabulky také můžeme vyčíst, že cesta (1, 4) není součástí žádné nejkratší cesty.

Pokud bychom chtěli zjistit nejkratší cesty ze všech vrcholů, postupujeme stejně. Opět vytvoříme výchozí tabulku a ve výpočtu pokračujeme obdobným způsobem s tím rozdílem, že vzdálenost rovnu nule nastavíme postupně na pozici 2, 3, 4 až 5.

2.2 Komponenty souvislosti grafu

Základním vstupem účelové funkce, jejíž minimum hledáme pomocí simulovaného žíhání (viz následující podkapitola), jsou údaje založené na znalosti komponent příslušného grafu.

Definici souvislosti grafu jsme si již uváděli v 1. kapitole. Jen pro připomenutí, o grafu řekneme, že je souvislý, pokud se můžeme z libovolného vrcholu dostat do všech ostatních. Jestliže graf je nesouvislý, hledáme co největší souvislé části (komponenty souvislosti).

Budeme uvažovat graf, který je neorientovaný a neohodnocený. Při ověřování souvislosti, potažmo určování komponent souvislosti, vycházíme ze skutečnosti, že mezi kterýmikoliv vrcholy ležícími ve stejné komponentě existuje cesta. Oproti tomu mezi vrcholy z různých komponent cesta neexistuje.

Samotný algoritmus probíhá následovně:

1. Vybereme výchozí vrchol a přiřadíme mu číslici jedna.
2. Najdeme všechny následníky výchozího vrcholu.
3. Pokud jsme je již nezařadili do některé komponenty, přiřadíme jim stejnou číslici, jakou má výchozí vrchol.
4. Pokud jsme číslici přiřadili všem vrcholům v grafu, algoritmus končí.
5. V opačném případě algoritmus pokračuje dále. Vybereme jeden z vrcholů, kterým jsme dosud nepřiřadili číslici a přiřadíme mu číslici zvýšenou o jedničku. Označíme jej jako výchozí vrchol. V algoritmu dále pokračujeme krokem 2.

Jestliže jsme všem vrcholům v grafu přiřadili pouze číslici jedna, graf je souvislý a skládá se tedy pouze z jedné komponenty. Pokud existuje alespoň jeden vrchol s jinou číslicí, graf je nesouvislý. Počet komponent stanovíme podle přiřazených číslic.

Doposud jsme předpokládali, že máme neohodnocený graf. Pokud ale pracujeme s grafem, ve kterém úmyslně vynecháme některou z hran, musíme tento fakt uvést i v grafu. Hranu z grafu ale ve skutečnosti neodstraňujeme, pouze jí přiřadíme ohodnocení nekonečno a do algoritmu přidáme podmínku na ohodnocení. Hrany s ohodnocením nekonečno bude algoritmus považovat za neexistující.

2.3 Simulované žíhání

Nyní se blíže podíváme na algoritmus, pomocí kterého budeme hledat minimum naší účelové funkce. Jak jsme se již zmiňovali, stěžejní částí naší analýzy zranitelnosti silniční sítě bude aplikace simulovaného žíhání. Zvolili jsme právě tento algoritmus, protože se řadí mezi stochastické optimalizační algoritmy a jeho hlavní předností je, že připouští i dočasné zhoršení hodnot účelové funkce. Tím se můžeme vyhnout uváznutí v lokálním minimu.

Simulované žíhání (Simulated Annealing, SA) se řadí mezi stochastické optimalizační algoritmy, které mají svůj základ ve fyzice. Autoři tohoto algoritmu se při problému hledání globálního minima inspirovali právě procesem žíhání tuhého tělesa.

Fyzikálně bychom postup žíhání kovů mohli popsat následovně: Těleso se umístí do pece, která je vyhřátá na vysokou teplotu. Těleso se zahřeje na žíhací teplotu a je v ní určitý čas ponecháno. Postupně se teplota začne snižovat a dochází tak k odstraňování vnitřních defektů v tělese. Při vysoké teplotě je těleso roztavené a jeho částice jsou tedy náhodně rozptýleny. Jak se teplota postupně snižuje, dochází k ochlazování tělesa a částice zaujímají rovnovážnou polohu. Požadavkem je, aby se teplota snižovala velmi pomalu a těleso tak mohlo získat požadovanou kvalitu a neobsahovalo žádné vnitřní defekty ani pnutí.

Ve fyzikálním modelu je těleso představeno vektorem $x = (x_1, x_2, \ldots, x_n)$. Jemu můžeme přiřadit funkční hodnotu (energii) $y = f(x)$. V simulovaném žíhání se minimalizuje právě funkce $f(x)$. Vektor x je náhodně přeměněn na nový vektor x'.

Jestliže je ochlazování prováděno dostatečně pomalu, žíhané těleso je za každé teploty T v rovnovážném stavu. Rovnovážný stav je popsán Boltzmannovým rozdělením pravděpodobnosti, že při teplotě T je žíhané těleso ve stavu i s energií E_i

$$W_T(E_i) = \frac{1}{Q(T)} e^{\frac{-E_i}{kT}},$$

kde k je Boltzmannova konstanta a funkce $Q(T)$ je definována:

$$Q(T) = \sum_i e^{\frac{-E_i}{kT}},$$

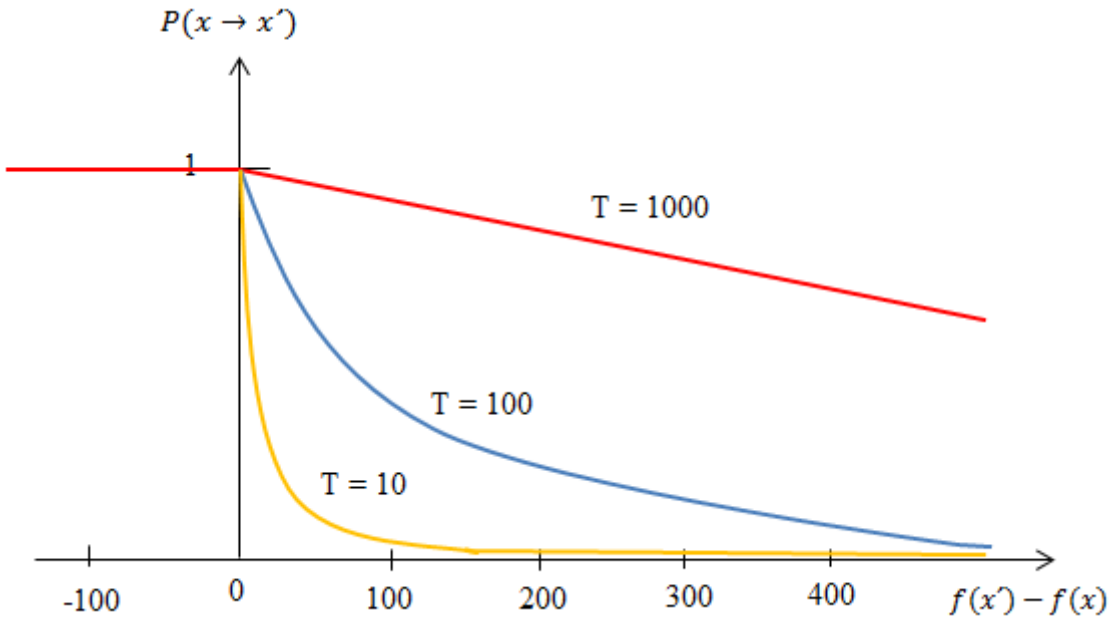
kde i obsahuje všechny stavy tělesa.

Jak se teplota snižuje, boltzmannovská distribuce dává přednost stavům s menší energií. Pro dostatečně malé teploty se pravděpodobnost výskytu stavu s minimální energií blíží k hodnotě jedna.

Při aplikaci simulovaného žíhání můžeme s určitou pravděpodobností přijmout i řešení s horší hodnotou účelové funkce. Při hledání globálního minima se díky této možnosti vyhneme uvíznutí v lokálním minimu. Pravděpodobnost přijetí horšího řešení je dána následujícím principem. Máme aktuální stav systému, který je určený polohou částic tělesa. Následně dojde k mírnému posunu částic. Stav systému x se změní na stav $x´$. Nový stav $x´$ není automaticky přijat, ale posuzuje se na základě *Metropolisova kritéria*. Toto kritérium stanovuje pravděpodobnost přijetí nového stavu za stav starý:

$$P(x \to x´) = \begin{cases} 1 & pro\ f(x´) < f(x) \\ e^{\frac{-(f(x)-f(x´))}{T}} & pro\ f(x´) \geq f(x) \end{cases}$$

Metropolisovo kritérium nám tedy říká, že pokud má nový stav menší funkční hodnotu než stav původní, pak nový stav nahradí původní stav. V opačném případě je nový stav přijat s pravděpodobností $P(x \to x´)$. Na tuto pravděpodobnost má vliv nejen to, o kolik je nový stav horší než původní, ale především pak teplota. Pro velké hodnoty teploty T se pravděpodobnost přijetí nového stavu blíží jedné. Oproti tomu, čím víc se teplota T blíží k nule, tím se pravděpodobnost přijetí podstatně snižuje. Tuto situaci jsme znázornili na následujícím obrázku.



Obrázek 2.2 Metropolisovo rozložení

Pokud vhodně nastavíme počáteční teplotu T_0, metoda simulovaného žíhání prohledává prostor nejprve silně stochasticky a přijímá stavy s horší funkční hodnotou. Tato vlastnost se se snižováním teploty zmenšuje. Pro malé teploty T Metropolisovo kritérium připouští už jen takové stavy, které vedou ke zlepšení funkční hodnoty.

Princip simulovaného žíhání můžeme znázornit následovně:

1. Stanovíme počáteční teplotu T_0 a koncovou teplotu T_{konec}. Vygenerujeme počáteční stav *x* a spočítáme *f(x)*.
2. Provedeme *k*-krát Metropolisův algoritmus:
   2.1 Vytvoříme nový stav *x´*, spočítáme *f(x´)* a $\delta = f(x´) - f(x)$.
   2.2 Pokud $\delta < 0$ nový stav *x´* přijmeme.
   2.3 Jestliže je $\delta \geq 0$, pak
      2.3.1 vygenerujeme náhodné číslo *r* z intervalu (0, 1),
      2.3.2 vypočítáme mez Metropolisova kritéria, *mez* = $e^{\frac{-\delta}{T}}$,
      2.3.3 jestliže *r* < *mez*, pak nový stav přijmeme.
3. Provedeme snížení teploty *T*.
4. Pokud $T \geq T_{konec}$ vracíme se na krok 2, v opačném případě je algoritmus u konce.

V uvedeném algoritmu ještě zbývá vyřešit, jak nastavit počáteční teplotu T_0, snižování teploty a počet iterací. Počáteční teplota by měla být zvolena tak, aby zhruba polovina stavů s horší funkční hodnotou byla Metropolisovým algoritmem přijata. Pro snižování teploty můžeme využít dva přístupy. Buď můžeme teplotu měnit skokově, nebo kontinuálně. V prvním případě stanovíme hodnotu, o kterou budeme teplotu snižovat (tzv. krok).

$$T = T - krok$$

Nebo můžeme zvolit multiplikátor α, $\alpha \epsilon (0, 1)$, a novou teplotu *T* vypočítáme podle vztahu:

$$T = T \times \alpha$$

Počet iterací není striktně určen, záleží na průběhu samotného algoritmu. Můžeme pracovat s více verzemi počtu iterací a posléze se rozhodnout, jaký počet iterací bude pro nás optimální.

3 Analýza zranitelnosti silniční sítě

Analýza zranitelnosti silniční sítě nám pomáhá identifikovat kriticky důležité úseky, jejichž selhání by mělo za následek značné negativní následky. Součástí analýzy zranitelnosti bývá mnohdy i navrhnutí nápravného opatření, které by vedlo ke zvýšení odolnosti silniční sítě.

Definicí zranitelnosti existuje celá řada. Vždy záleží, v jakém kontextu zranitelnost používáme. V našem případě budeme zranitelnost chápat tak, jak ji definoval Berdica [1]. Zranitelnost silniční sítě je náchylnost k nepříjemným událostem, které mohou zapříčinit výrazné snížení provozuschopnosti dané silniční sítě.

V této kapitole si ukážeme jeden z možných způsobů, jak identifikovat kritické úseky v silniční síti. Silniční síť budeme reprezentovat pomocí grafu, který je neorientovaný, souvislý, konečný, bez smyček a ohodnocený hranově i vrcholově. Tento graf je tvořen vrcholy a hranami. Vrcholy představují obce a křižovatky, hrany znázorňují silnice. Hranové ohodnocení představuje délku hrany v metrech nebo v sekundách. Ohodnocení vrcholu může být buď nula, tzn., že se silnice kříží mimo obec a vrchol je neobydlený. Pokud se silnice kříží v obci, vrchol je ohodnocený počtem obyvatel žijících v dané obci. V této kapitole budeme čerpat z knih [2], [6] a [8].

3.1 Základní pojmy

Ve většině případů se zranitelnost silniční sítě zkoumá bez ohledu na pravděpodobnost zneprůjezdnění hrany, protože mnoho situací můžeme předvídat jen velmi obtížně (např. sabotáž nebo přírodní katastrofu). V našem případě ale pravděpodobnost při analýze zranitelnosti silniční sítě zohledníme. Při určování pravděpodobnosti nám pomůže *normovaný betweenness index* (*NBC*).

Normovaný betweenness index řadíme mezi centrality. Centrality nám pomáhají určit, které hrany nebo uzly (záleží, co zkoumáme) jsou v síti nejdůležitější neboli také centrální. Rozlišujeme tři nejpoužívanější centrality: *stupeň vrcholu*, *blízkost polohy ve středu* a již zmiňovaný betweenness index. Při zkoumání zranitelnosti silniční sítě budeme využívat jen poslední uvedenou centralitu.

Betweenness centrality (zkráceně pouze BC) používáme na vyjádření potřebnosti hrany k propojení vrcholů. BC dané hrany *i* je počet nejkratších cest, jejichž součástí je právě hrana *i*. Můžeme ji definovat vztahem:

$$\mathrm{BC}(i) = \sum_{st} n_{st}^{i},$$

kde *BC(i)* je betweenness centrality dané hrany *i*, *s* je počáteční vrchol, *t* je koncový vrchol a n_{st}^{i} nabývá hodnoty nula nebo jedna. Hodnotu jedna má, pokud hrana *i* leží na nejkratší cestě z vrcholu *s* do vrcholu *t*. Hodnotu nula, jestliže hrana *i* na nejkratší cestě neleží, nebo pokud taková cesta neexistuje. Hrany s vysokou hodnotou BC označujeme jako mosty nebo zprostředkovatele.

V diplomové práci budeme pracovat s *normovaným betweenness indexem*, který vypočítáme podle následujícího vztahu:

$$NBC(i) = \frac{BC(i)}{\sum_i BC(i)}$$

Právě pomocí NBC budeme stanovovat pravděpodobnost, s jakou dojde ke zneprůjezdnění hrany *i*. Čím bude hodnota *NBC* pro hranu *i* vyšší, tím bude pravděpodobnější, že ji zneprůjezdníme. Hodnotu *NBC(i)* budeme považovat za pravděpodobnost zneprůjezdnění hrany *i*.

3.2 Popis dat

Veškerá data, se kterými budeme dále pracovat, nám byla poskytnuta od Centra dopravního výzkumu, v. v. i. (zkráceně CDV). Informace o uzlech a hranách jsou uloženy v textovém souboru. Každý řádek obsahuje informace o konkrétním vrcholu a dále také seznam všech jeho následníků a další informace o nich. Také je zaznamenána hrana spojující tyto vrcholy a údaje o hraně. Data na řádku jsou napsána v tomto formátu:

název_vrcholu;počet_obyvatel název_následníka_1;počet_obyvatel_v_následníku_1;
název_hrany;délka_hrany_v_metrech;délka_hrany_v_sekundách
název_následníka2;počet_obyvatel_v_následníku_2;název_hrany;délka_hrany_v_metrech;
délka_hrany_v_sekundách název_následníka_3;…

Vrchol s jedním následníkem je zapsán např. takto:

1223A266;11 1223A249;237;1223A249071223A26603;4495;124

3.3 Účelová funkce

Při zkoumání zranitelnosti silniční sítě chceme nalézt takovou kombinaci hran, jejíž vypuštění z dané silniční sítě na ni bude mít největší negativní dopad. Za negativní dopad můžeme považovat více skutečností. Může to být počet komponent, na které se síť rozpadne. Také to může být prodloužení doby dojezdu v rámci celé sítě. V této diplomové práci budeme negativní dopad vyjadřovat pomocí směrodatné odchylky počtu obyvatel žijících v jednotlivých komponentách, na které se graf rozpadl po zneprůjezdnění dané kombinace hran. Tato směrodatná odchylka bude naší účelovou funkcí a naším cílem je tuto účelovou funkci minimalizovat.

Směrodatnou odchylku budeme počítat podle následujícího vztahu:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(počet_i - \bar{x})^2},$$

kde n je počet komponent, na které se síť rozpadla. $Počet_i$ představuje počet obyvatel žijících v i-té komponentě a $\bar{x}$ je průměrný počet obyvatel žijících v dané síti.

Vycházíme z předpokladu, že čím více obyvatel bude žít v jednotlivých komponentách, tím to bude pro všechny horší. Domníváme se, že se lidé budou chtít přemístit z dané komponenty do jiné části sítě. Také např. při zásobování nebo evakuaci bude těžší rozhodnout, kterou komponentou se máme zabývat jako první. Nejhorší situace nastane tehdy, když se síť rozpadne na komponenty, ve kterých bude žít zhruba stejný počet obyvatel. Směrodatná odchylka se bude blížit k hodnotě jedna. Situace je tedy tím horší, čím je směrodatná odchylka nižší.

Rovněž předpokládáme, že počet komponent bude vždy o jednu větší, než počet vynechaných hran. Vycházíme z toho, že při vynechání např. dvou hran se daný graf může rozpadnout nejvýše na tři komponenty. Může nastat i situace, kdy se graf rozpadne pouze na dvě komponenty. V takovém případě počítáme také se třemi komponentami, ale s tím, že té třetí přiřadíme nulový počet obyvatel.

Uvažujme nyní situaci, kdy se síť rozpadne na dvě komponenty. Nejdříve v jedné komponentě bude počet obyvatel roven 10 000 a ve druhé komponentě bude 500 obyvatel. V tomto případě není rozhodování příliš obtížné. Naproti tomu rozhodnout se v situaci,

kdy bude počet obyvatel v komponentách 5 200 a 5 300, již není vůbec jednoduché. Kombinaci hran, jejichž vynechání způsobí druhou zmiňovanou situaci, budeme chtít najít.

3.4 Aplikace algoritmu simulovaného žíhání

Nyní si podrobněji popíšeme aplikaci simulovaného žíhání při zjišťování zranitelnosti silniční sítě. Simulované žíhání jsme zvolili z toho důvodu, že patří mezi stochastické algoritmy a připouští i dočasné zhoršení účelové funkce. Kostru algoritmu simulovaného žíhání jsme popsali v předchozí kapitole. Z tohoto popisu budeme dále vycházet.

Nejprve si nastavíme počáteční parametry, se kterými budeme dále pracovat. Potřebujeme znát počáteční T_0 a konečnou teplotu T_{konec}. Dále také počet hran, které z grafu vynecháme a počet opakování Metropolisova algoritmu.

1. Pomocí Dijkstrova algoritmu vypočítáme nejkratší cesty mezi všemi uzly.
2. Spočítáme normovaný betweenness index pro všechny hrany (vycházíme z toho, kolikrát se daná hrana vyskytla na nejkratších cestách).
3. S ohledem na NBC vygenerujeme množinu hran *x*, kterou z grafu vypustíme.
4. Zavoláme funkci pro zjištění počtu komponent, na které se graf rozpadl, a určíme, z kolika komponent se graf skládá.
5. Vypočítáme směrodatnou odchylku počtu obyvatel žijících v jednotlivých komponentách a označíme ji *f(x)*.

Prostřednictvím kroků 1-5 stanovíme výchozí hodnotu účelové funkce (směrodatná odchylka počtu obyvatel žijících v komponentách). Pomocí algoritmu simulovaného žíhání se budeme snažit tuto funkci minimalizovat.

Tím, že jsme z grafu vypustili určitou množinu hran, došlo k jeho změně. Pro tento nový graf opět provedeme kroky 1-5 a získáme novou směrodatnou odchylku, *f(x´)*. Způsob generování nové množiny neprůjezdných hran si podrobněji vysvětlíme v další části diplomové práce.

Získali jsme tak dvě hodnoty účelové funkce a nyní je mezi sebou porovnáme pomocí Metropolisova algoritmu:

6. Provedeme *k*-krát Metropolisův algoritmus:

6.1 Spočítáme $\delta = f(x') - f(x)$.

6.2 Pokud $\delta < 0$ nový stav x' přijmeme.

6.3 Jestliže je $\delta \geq 0$, pak

6.3.1 vygenerujeme náhodné číslo r z intervalu (0, 1),

6.3.2 vypočítáme mez Metropolisova kritéria, $mez = e^{\frac{-\delta}{T}}$,

6.3.3 jestliže $r < mez$, pak nový stav přijmeme.

Novou přijatou směrodatnou odchylku a novou množinu neprůjezdných hran použijeme jako výchozí údaje pro další výpočty. Pokud jsme směrodatnou odchylku nepřijali, pokračujeme ve výpočtu s výchozí hodnotou směrodatné odchylky.

7. Provedeme snížení teploty *T*.
8. Pokud $T \geq T_{konec}$ vracíme se na krok 2, v opačném případě je algoritmus u konce.

Výstupem algoritmu je nejmenší směrodatná odchylka, jí odpovídající množina neprůjezdných hran a počet komponent, na které se síť rozpadla.

V průběhu algoritmu se nám s každou iterací mění množina průjezdných a neprůjezdných hran. Tím se nám také mění nejkratší cesty mezi všemi uzly. Musíme tedy opakovaně provádět výpočet *NBC*. Jak jsme se již zmiňovali, *NBC* budeme považovat za pravděpodobnosti, s jakými dojde k přerušení (zneprůjezdnění) průjezdných hran. Vycházíme z předpokladu, že frekventovanější hrana je více náchylná na zneprůjezdnění (např. v důsledku nehody). *Z* nových hodnot *NBC* bereme nové pravděpodobnosti zneprůjezdnění hran.

V následující části si představíme jednotlivé verze výměny neprůjezdných hran. Těchto verzí je celkem pět. U jednodušších metod uvedeme pouze slovní popis, jak výměna probíhá. U složitějších výměn ji demonstrujeme i na jednoduchém příkladu.

3.4.1 Simulované žíhání – vyměňování jedné hrany

V této verzi simulovaného žíhání vyměňujeme vždy pouze jednu hranu. Nejprve z množiny průjezdných hran vybereme jednu hranu, kterou zneprůjezdníme. Abychom stanovili, která hrana to bude, potřebujeme vědět *NBC*. Spočítáme jeho kumulované četnosti a vygenerujeme náhodné číslo z intervalu (0, 1). Z *NBC* vybereme první hranu, která má tuto hodnotu větší nebo rovnu vygenerovanému náhodnému číslu. Pro lepší pochopení uvedeme krátký příklad. V níže uvedené tabulce jsou v prvním sloupci názvy

hran, ve druhém jejich NBC a ve třetím sloupci pak kumulované četnosti NBC. Předpokládejme, že vygenerujeme náhodné číslo 0,42. Podíváme se do tabulky do sloupečku s kumulovanými četnostmi a vidíme, že první hodnota větší nebo rovna hodnotě 0,42 je číslo 0,55. Tato hodnota odpovídá hraně 3. Na základě NBC tedy zneprůjezdníme hranu 3.

	NBC	Kumulované četnosti
Hrana 1	0,3	0,3
Hrana 2	0,1	0,4
Hrana 3	0,15	0,55
Hrana 4	0,1	0,65
Hrana 5	0,35	1

Tabulka 3.1 Kumulované četnosti hran

Abychom zachovali počet vynechaných hran, který jsme stanovili na začátku, z množiny neprůjezdných hran náhodně jednu hranu zprůjezdníme.

3.4.2 Simulované žíhání – výměna všech hran

V této modifikaci nahrazujeme vždy celou množinu neprůjezdných hran. Hrany, které zneprůjezdníme, vybíráme podle stejného principu, který jsme uvedli v předchozí modifikaci. Tento výběr je ostatně u všech modifikací stejný. Rozdíl oproti předchozí verzi je v tom, že vyměňujeme všechny neprůjezdné hrany. Následně celou množinu původních neprůjezdných hran změníme na průjezdné.

3.4.3 Simulované žíhání – nahrazování náhodného počtu hran

V této verzi simulovaného žíhání není přesně stanovený počet hran, který budeme v průběhu chodu algoritmu měnit. Před samotnou změnou některých neprůjezdných hran na průjezdné se vygeneruje náhodné číslo. Generování náhodného čísla jsme omezili počtem neprůjezdných hran. Je to z toho důvodu, abychom nepřekročili původní počet neprůjezdných hran a abychom vyměnili alespoň jednu hranu. Toto náhodné číslo nám určí, jaký počet neprůjezdných hran budeme vyměňovat. Dále algoritmus probíhá stejně jako předchozí.

3.4.4 Simulované žíhání – nahrazování klesajícího počtu hran

V této verzi simulovaného žíhání nenahrazujeme konstantní počet hran, ale tento počet postupně snižujeme. V modifikaci jsme se inspirovali principem simulovaného žíhání. Nejprve vyměňujeme počáteční počet neprůjezdných hran. Po dosažení určité teploty počet vyměňovaných hran o jeden snížíme. Jakmile dosáhneme nahrazování pouze jedné hrany v množině neprůjezdných hran, počet hran již dále neměníme. Musíme tedy stanovit, ve které teplotě *T* snížíme počet vyměňovaných hran.

Teplotní rozmezí, ve kterém se bude vyměňovat stejný počet hran, určíme podle následujícího vztahu:

$$teplotní\ rozmezí = \frac{počáteční\ teplota}{vypuštěno}$$

Na jednoduchém příkladu si ukážeme, jak můžeme tuto modifikaci aplikovat. Předpokládejme, že počet vypuštěných hran je pět a počáteční teplota $T_0 = 40$. Podle výše uvedeného vztahu spočítáme teplotní rozmezí, které činí 8 stupňů. V tomto rozmezí budeme vyměňovat konstantní počet hran. V teplotním rozmezí 40 – 33 budeme vyměňovat všech pět hran. Tzn., že s ohledem na NBC zneprůjezdníme 5 hran. V rozmezí teplot 32 – 25 budeme nahrazovat o jednu hranu méně, tedy čtyři hrany. Od teploty 24 do teploty 17 vyměňujeme tři hrany, pro teploty 16 – 9 už jen dvě hrany. Od teploty $T = 8$ nahrazujeme pouze jednu hranu. Počáteční teplotu a počet neprůjezdných hran jsme si zde stanovili tak, aby teplotní rozmezí patřilo do množiny celých čísel. Ne vždy tomu tak musí být. Pokud teplotní rozmezí bude desetinné číslo, zaokrouhlíme jej na nejbližší vyšší celé číslo.

3.4.5 Simulované žíhání – nahrazování počtu hran podle pravděpodobnosti

V poslední modifikaci simulovaného žíhání využijeme při stanovení počtu neprůjezdných hran, které se budou vyměňovat, pravděpodobnost. Tuto verzi simulovaného žíhání si představíme nejprve obecně a posléze pro lepší ilustraci na jednoduchém příkladu.

Předpokládejme, že máme celkem *n* neprůjezdných hran. Z této *n*-tice hran nahradíme jednu hranu s pravděpodobností *P(1)*, dvě hrany s pravděpodobností *P(2)* až všech *n* hran s pravděpodobností *P(n)*. Vytvořené pravděpodobnosti tvoří klesající posloupnost čísel. Naším cílem je, aby program s větší pravděpodobností prohledával

blízké okolí stanovené *n*-tice hran. Vyvstává zde otázka, jak určíme jednotlivé pravděpodobnosti.

Pravděpodobnosti stanovíme podle následujícího postupu. Nejprve sečteme všechny možnosti, kolik můžeme vyměnit neprůjezdných hran:

$$sum = \sum_{i=1}^{n} i.$$

Následně jednotlivé možnosti vydělíme zjištěným součtem „*sum*“:

$$pocet(i) = \frac{i}{sum}.$$

Jelikož chceme, aby se s větší pravděpodobností vyměňoval nižší počet hran, pořadí pravděpodobností otočíme. Tedy pravděpodobnost, že vyměníme pouze jednu hranu, bude ve tvaru:

$$P(1) = pocet(n).$$

Pravděpodobnost, že vyměníme dvě hrany, bude následující:

$$P(2) = pocet(n-1).$$

Takto budeme pokračovat až do počtu *n* hran:

$$P(n) = pocet(1).$$

Nyní předchozí postup aplikujeme na konkrétní příklad. Opět budeme předpokládat, že budeme mít pět neprůjezdných hran. Nejprve spočítáme „*sum*“.

$$sum = \sum_{i=1}^{5} i = 15$$

Pravděpodobnosti výměny určitého počtu hran jsou následující:

$$P(1) = \frac{5}{15}, P(2) = \frac{4}{15}, P(3) = \frac{3}{15}, P(4) = \frac{2}{15}, P(5) = \frac{1}{15}.$$

Výsledné pravděpodobnosti nám říkají, že jedna hrana se v množině neprůjezdných hran bude vyměňovat s pravděpodobností $\frac{5}{15}$, dvě hrany s pravděpodobností $\frac{4}{15}$ až všech pět hran s pravděpodobností $\frac{1}{15}$. Náš stanovený předpoklad, aby se s větší pravděpodobností vyměňoval nižší počet hran, je splněn.

Stanovili jsme pravděpodobnosti, které nám říkají, jaký počet neprůjezdných hran bude vyměněn. Nyní již můžeme přejít k samotnému chodu programu. Stejně jako v předchozích modifikacích po načtení vstupních parametrů nejprve s pomocí NBC zneprůjezdníme stanovený počet hran. Přepočítáme délky nejkratších cest a uložíme hrany, které na nových nejkratších cestách leží. Spočítáme nový NBC (vycházíme z nových vzdáleností). Nyní musíme určit, jaký počet hran v množině neprůjezdných hran vyměníme. Ke stanoveným pravděpodobnostem vytvoříme i jejich kumulativní četnosti. Vygenerujeme náhodné číslo z intervalu $(0, 1)$ a v kumulativní četnosti vybereme první hodnotu, která je stejná, popř. vyšší jako ono náhodné číslo. Takto zjistíme, jaký počet neprůjezdných hran budeme vyměňovat.

Jestliže se vrátíme k našemu příkladu, tak kumulativní četnosti jsou následující:

Pravděpodobnosti	Kumulativní četnost
$\frac{5}{15}$	$\frac{5}{15}$
$\frac{4}{15}$	$\frac{9}{15}$
$\frac{3}{15}$	$\frac{12}{15}$
$\frac{2}{15}$	$\frac{14}{15}$
$\frac{1}{15}$	$\frac{15}{15}$

Tabulka 3.2 Kumulované četnosti

Pokud vygenerujeme např. náhodné číslo 0,4, vybereme pravděpodobnost P(2). Ve sloupci s kumulativní četností je první větší číslo $\frac{9}{15}$, což odpovídá pravděpodobnosti P(2). Z množiny neprůjezdných hran vyměníme tedy pouze dvě hrany. Takto program pracuje tak dlouho, dokud nedosáhne koncové teploty $T = 1$.

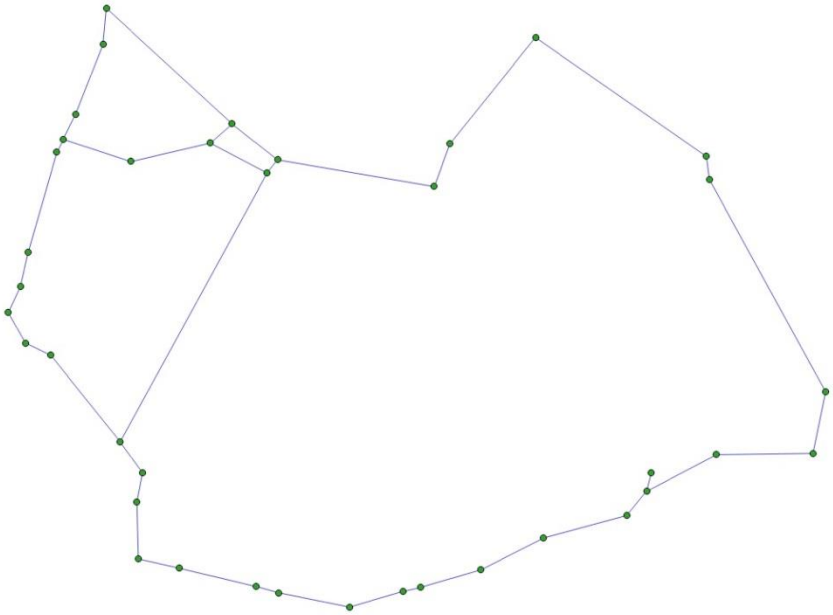
4 Výsledky

V této kapitole aplikujeme všechny uvedené modifikace simulovaného žíhání na čtyři sítě. První dvě sítě jsou menšího rozsahu a jedná se o části Zlínského kraje. Na těchto sítích si názorně ukážeme neprůjezdné hrany, jejichž vynechání bude mít na síť největší negativní dopad. Pro tyto dvě menší sítě jsme zvolili počet opakování Metropolisova algoritmu na 1 000 opakování. Zbylé dvě sítě jsou již větší a z důvodu časové náročnosti při výpočtu jsme zvolili pouze 500 opakování. U všech sítí začínáme s počáteční teplotou T_0 = 40 a koncovou teplotou T_{konec} = 1. Teplotu budeme snižovat skokově, o jedničku. V případě větších sítí zneprůjezdníme sedm hran a u menších sítí hrany tři.

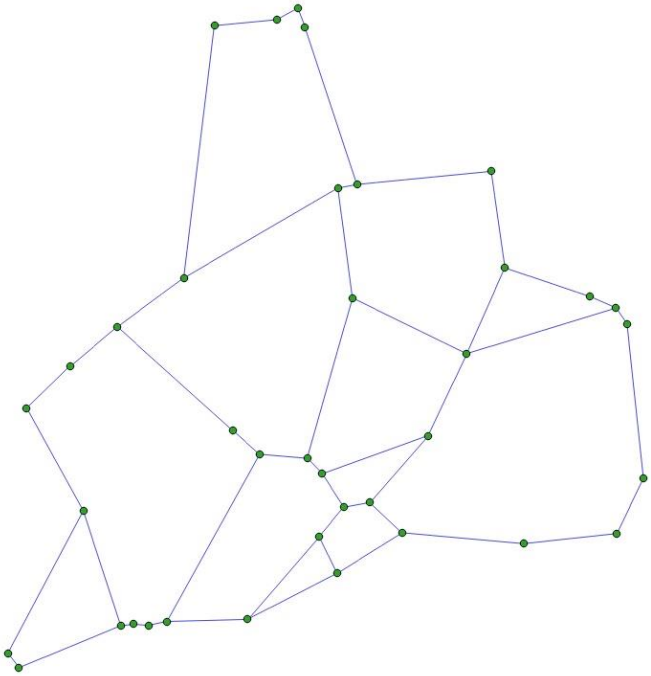
4.1 Výsledky pro menší sítě

Obě části Zlínského kraje mají stejný počet vrcholů, 38. Počet hran je rovněž podobný. Část Zlínského kraje, kterou jsme pojmenovali Síť část 1, má 41 hran. Druhá část pojmenovaná Síť část 2 má 49 hran. Grafickou podobu těchto dvou sítí jsme znázornili na obrázku 4.1 a 4.2. Je z nich zjevné, že obě sítě, ač mají podobné rozměry, jsou naprosto odlišné.

Vstupní parametry jsme nastavili následovně. Počet neprůjezdných hran je tři, počáteční teplota je 40, koncová teplota 1. Počet opakování Metropolisova algoritmu je 1000 opakování.



Obrázek 4.1 Graf Síť část 1



Obrázek 4.2 Graf Síť část 2

Níže v tabulce uvádíme výsledky nejprve pro silniční síť s názvem Síť část 1. V tabulce je napsána verze simulovaného žíhání a počet komponent, na které se síť po zneprůjezdnění tří hran rozpadla. Nejdůležitější je zde výsledná hodnota účelové funkce,

směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách. Nevíme, jestli jsme dosáhli nejhorší (nejmenší) směrodatné odchylky počtu obyvatel, ale výsledky by měly patřit mezi dostatečně špatné. Jednotlivé verze simulovaného žíhání jsme očíslovali následovně:

1 – výměna pouze jedné hrany,

2 – výměna všech hran,

3 – výměna podle pravděpodobností,

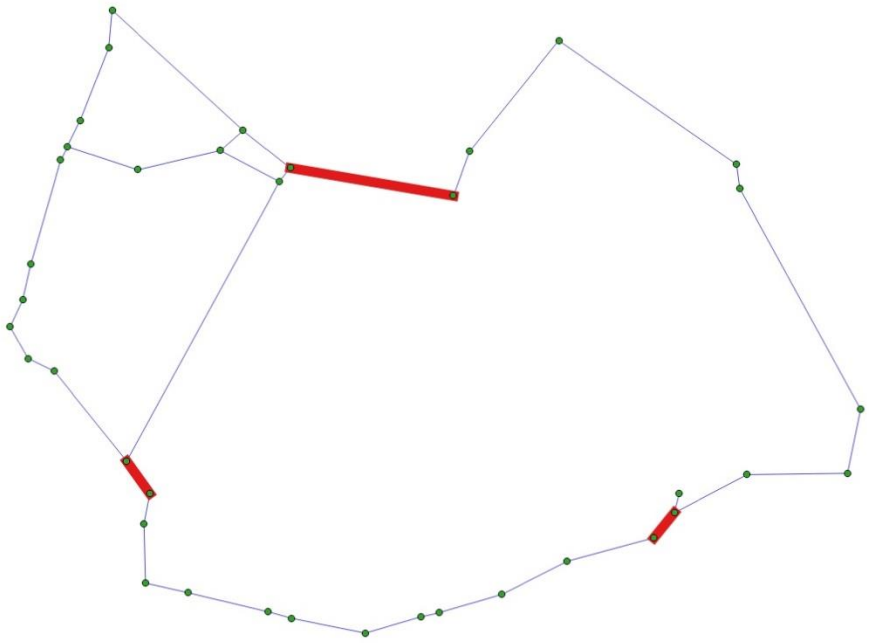
4 – výměna náhodného počtu hran,

5 – výměna snižujícího počtu hran.

Simulované žíhání:	**1**	**2**	**3**	**4**	**5**
Počet komponent:	3	3	3	3	3
Směrodatná odchylka:	8 019,5	8 019,5	8 019,5	8 019,5	8 019,5

Tabulka 4.1 Výsledky všech verzí simulovaného žíhání pro Síť část 1

U menších sítí můžeme očekávat výsledky velmi podobné. V tomto případě jsou výsledky dokonce zcela totožné. Na dalším obrázku si graficky znázorníme, které hrany byly označeny jako neprůjezdné. Z obrázku 4.3 si můžeme snadno ověřit, zda je počet komponent správný.



Obrázek 4.3 Neprůjezdné hrany v Síť část 1

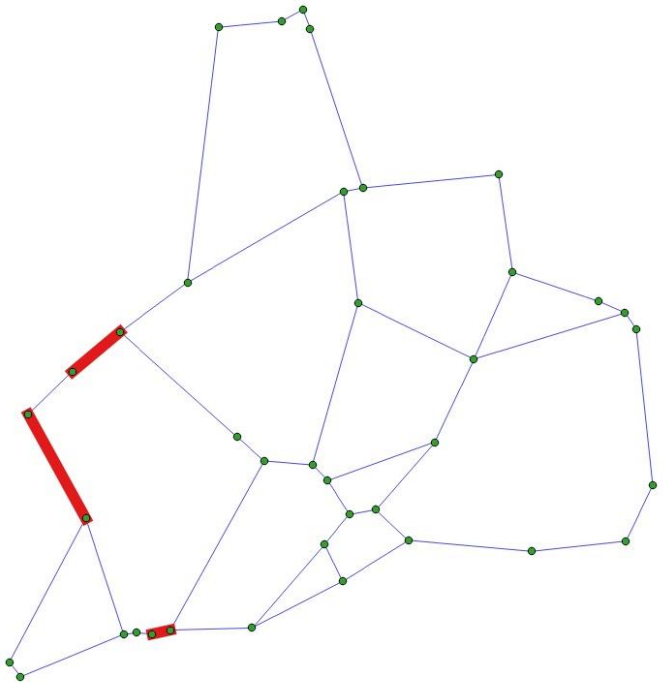
Pro druhou menší síť jsme zjištěné výsledky uvedli v tabulce 4.2. Opět je zde nejdůležitější řádek s hodnotou směrodatné odchylky počtu obyvatel žijících v jednotlivých komponentách. Rovněž předpokládáme výsledky velmi podobné, jelikož se jedná o síť s malým počtem uzlů a hran.

Simulované žíhání:	**1**	**2**	**3**	**4**	**5**
Počet komponent:	3	2	3	3	3
Směrodatná odchylka:	13 630,4	14 390,9	14 081,2	13 630,4	13 630,4

Tabulka 4.2 Výsledky všech verzí simulovaného žíhání pro Síť část 2

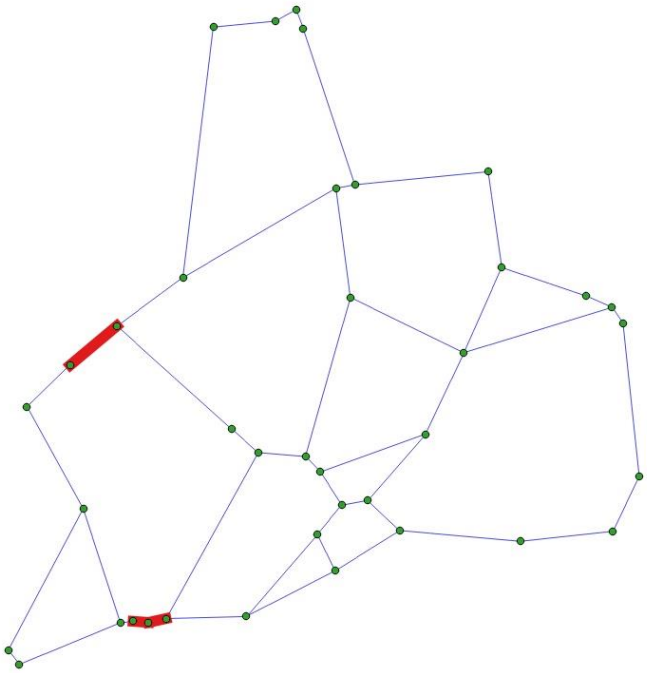
U grafu Síť část 2 jsme neobdrželi shodné výsledky. Ale tak, jak jsme předpokládali, jsou si velmi podobné. Nejmenší směrodatné odchylky jsme dosáhli při použití verzí simulovaného žíhání 1, 2 a 3. Tyto verze odpovídají výměně pouze jedné neprůjezdné hrany, výměně náhodného počtu hran a snižujícího počtu neprůjezdných hran. Pokud bychom modifikace hodnotili z hlediska počtu komponent, rovněž jsou výsledky téměř stejné.

Na obrázku 4.4 jsme znázornili neprůjezdné hrany při použití modifikace simulovaného žíhání, které pracuje s nahrazováním klesajícího počtu hran.



Obrázek 4.4 Neprůjezdné hrany v modifikaci se snižujícím počtem hran pro Síť část 2

Na obrázku 4.5 jsme znázornili neprůjezdné hrany při použití nahrazování náhodného počtu hran. Z obou dvou obrázků (4.4 i 4.5) opět můžeme jednoduše ověřit, že počet komponent je stanovený správně.

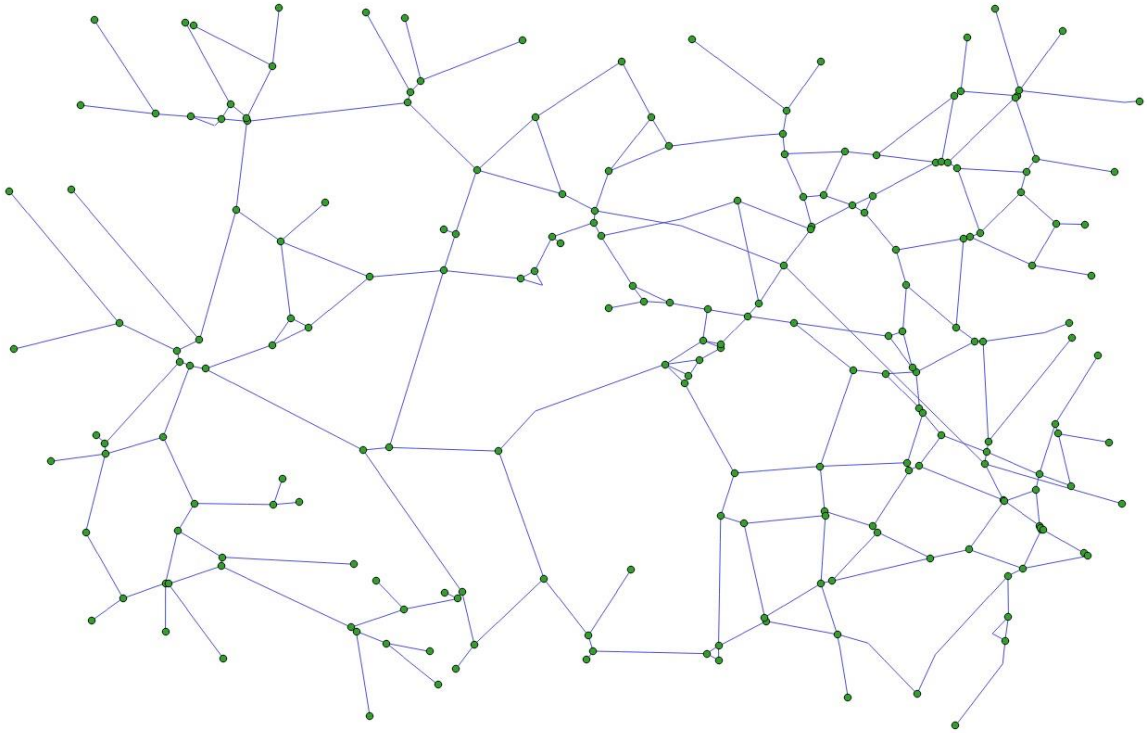


Obrázek 4.5 Neprůjezdné hrany v modifikaci s náhodným počtem hran pro Síť část 2

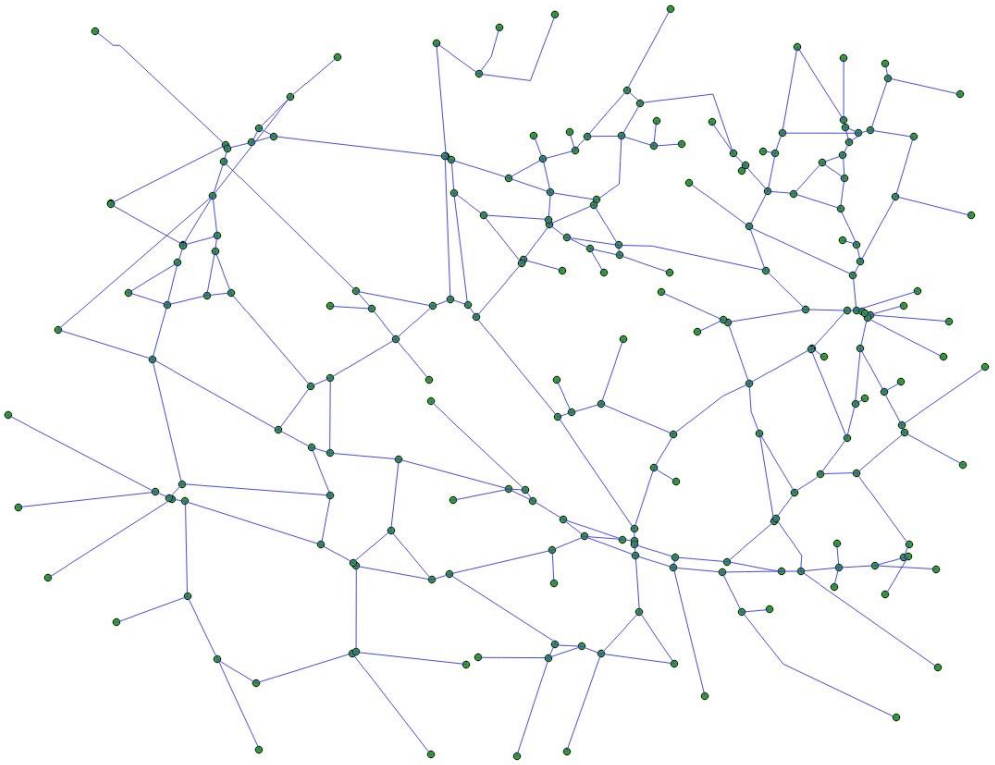
4.2 Výsledky pro větší sítě

První rozsáhlejší síť jsme nazvali Síť 1. Tato silniční síť má 210 vrcholů a 276 hran. Druhé síti jsme dali název Síť 2. Tato síť má 217 vrcholů a 272 hran. Grafy sítí jsme uvedli na následujících dvou obrázcích 4.6 a 4.7.

Vstupní parametry jsme nastavili následovně. Počet neprůjezdných hran je sedm, počáteční teplota je 40, koncová teplota 1. Počet opakování Metropolisova algoritmu je 500 opakování.



Obrázek 4.6 Síť 1



Obrázek 4.7 Síť 2

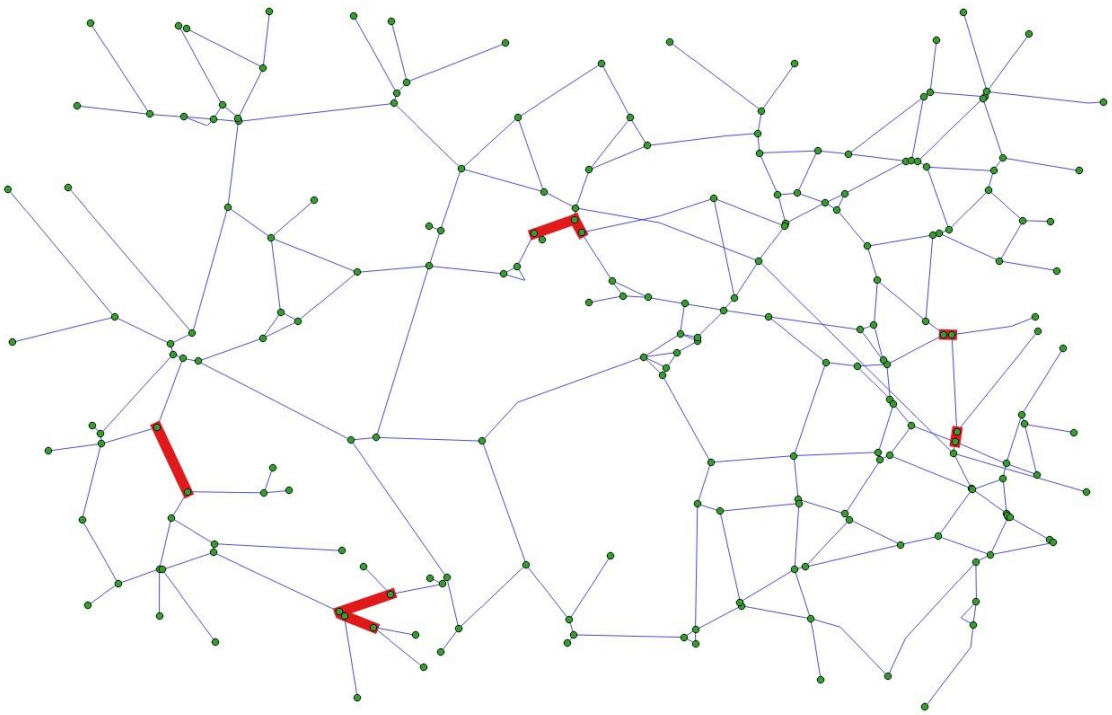
Nejprve si uvedeme výsledky pro silniční síť s názvem Síť 1. Následující tabulka je popsaná stejně jako tabulky u předchozích menších sítí.

Simulované žíhání:	**1**	**2**	**3**	**4**	**5**
Počet komponent:	7	2	3	3	2
Směrodatná odchylka:	31 411,4	38 125,6	40 391,8	38 053,7	38 125,6

Tabulka 4.3 Výsledky všech verzí simulovaného žíhání pro Síť 1

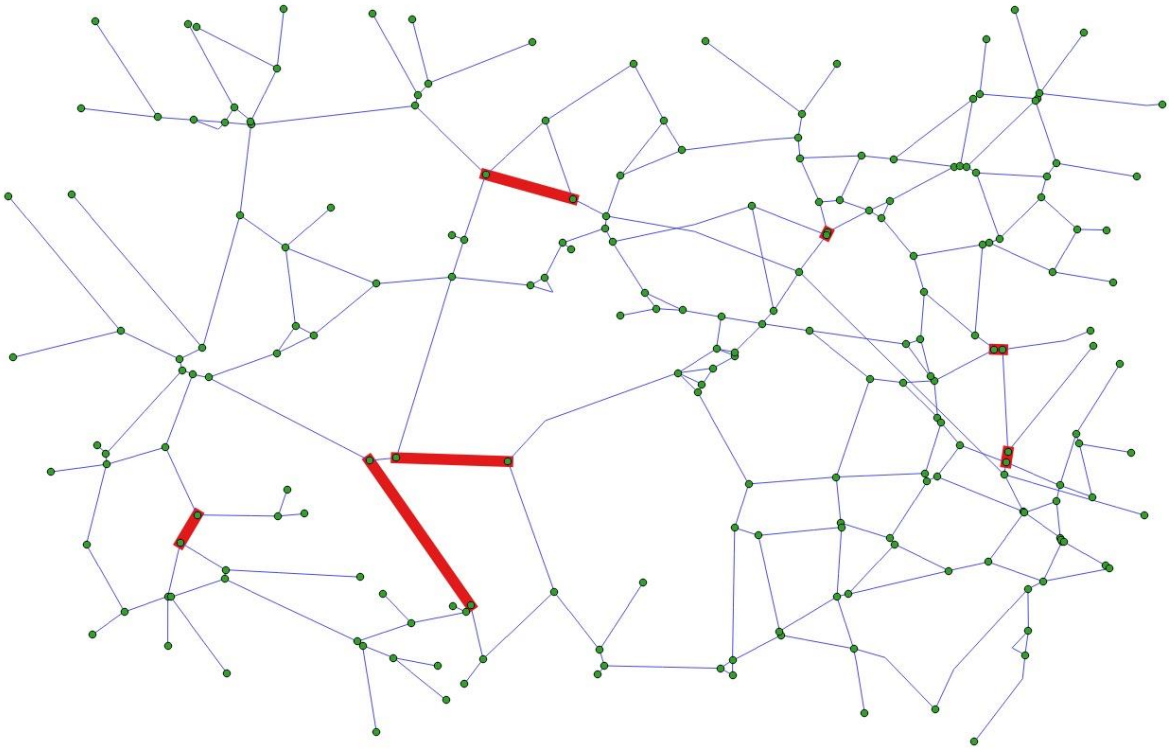
Výsledné směrodatné odchylky počtu obyvatel žijících v jednotlivých komponentách již nejsou téměř shodné. Naopak, rozdíl je zde poměrně velký. Nejmenší hodnoty směrodatné odchylky jsme dosáhli při aplikaci výměny pouze jedné hrany. Nejvyšší hodnotu jsme zjistili při použití výměny podle pravděpodobností. Zbylé tři modifikace mají podobné výsledky.

Z hlediska počtu komponent, na které se graf rozpadl, je nejvyšší počet komponent u modifikace simulovaného žíhání s jednou neprůjezdnou hranou. Jelikož počet komponent je dost vysoký, můžeme usoudit, že program zneprůjezdnil hrany, které vedou do vrcholů incidentních pouze s jednou hranou. Při bližším zkoumání grafu v souboru sit1.txt jsme skutečně zjistili, že se jedná právě o takovéto hrany.



Obrázek 4.8 Neprůjezdné hrany v modifikaci s náhodným počtem hran u grafu Síť 1

Na obrázku 4.8 jsme znázornili, které hrany jsme vynechali při použití modifikace s výměnou náhodného počtu hran. Z obrázku si opět můžeme ověřit počet komponent. Na obrázku 4.9 jsme znázornili neprůjezdné hrany při použití verze simulovaného žíhání s výměnou všech neprůjezdných hran.



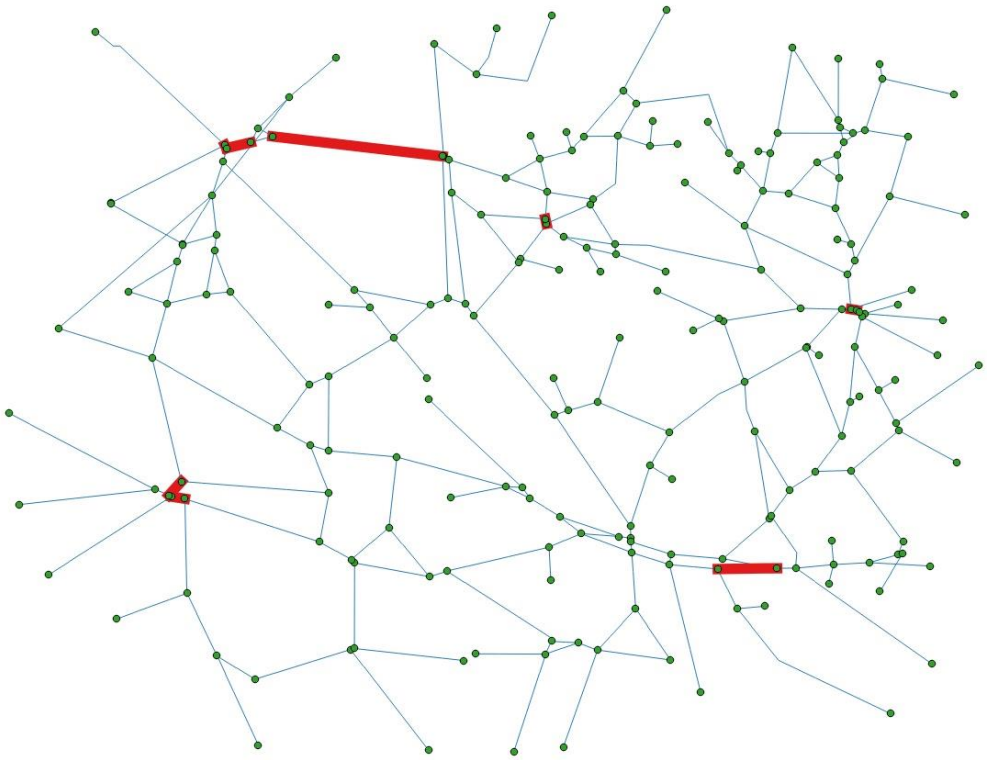
Obrázek 4.9 Neprůjezdné hrany v modifikaci s výměnou všech hran u grafu Síť 1

Pro druhou větší síť jsme výsledky shrnuli do následující tabulky. Vstupní parametry jsou opět stejné jako v předchozím případě. Počet neprůjezdných hran je sedm, počáteční teplota je 40, koncová teplota 1. Počet opakování Metropolisova algoritmu jsme stanovili na 500.

Simulované žíhání:	**1**	**2**	**3**	**4**	**5**
Počet komponent:	7	3	4	5	4
Směrodatná odchylka:	17 150	20 250,3	19 078,8	19 123,7	19 109,5

Tabulka 4.4 Výsledky všech verzí simulovaného žíhání pro Síť 2

Získané výsledky jsou více vyrovnané než u předchozí sítě. Nejmenší směrodatné odchylky počtu obyvatel žijících v jednotlivých komponentách jsme zaznamenali u modifikace, která nahrazuje pouze jednu neprůjezdnou hranu. Tato verze simulovaného žíhání má nejvyšší počet komponent, sedm. Takto vysoké číslo nás opět vede k domněnce, že jsme zneprůjezdnili hrany vedoucí do uzlů incidentních pouze s jednou hranou. Při detailnějším zkoumání grafu v souboru sit2.txt se nám tato domněnka potvrdila.



Obrázek 4.10 Neprůjezdné hrany v modifikaci s výměnou všech hran u grafu Síť 2

4.3 Porovnání modifikací na základě výsledků

Nyní ještě srovnáme jednotlivé modifikace simulovaného žíhání na základě získaných výsledků.

Pokud bychom měli stanovit, která verze simulovaného žíhání by mohla být nejlepší, podle výsledků směrodatné odchylky bychom vybrali modifikaci s výměnou pouze jedné neprůjezdné hrany. Musíme zde brát ale v úvahu i samotnou strukturu silničních sítí. Jestliže se podíváme obrázky 4.6 a 4.7, vidíme, že se jedná o sítě, které mají spoustu uzlů incidentních pouze s jednou hranou. Při jiném tvaru sítí bychom mohli vybrat jinou modifikaci, která by dávala nejmenší směrodatné odchylky. Z pohledu konstrukce výměny neprůjezdných hran bychom se přikláněli spíš k verzi, která pracuje se snižujícím

počtem hran. Vede nás k tomu jistá podobnost se samotným algoritmem simulovaného žíhání. Tento algoritmus nejprve prohledává okolí silně stochasticky, naše modifikace nejdříve vyměňuje všechny neprůjezdné hrany. Se snižující se teplotou dochází k výměně menšího počtu hran, až je vyměňována pouze jedna hrana.

ZÁVĚR

V diplomové práci jsme se zabývali analýzou zranitelnosti silniční sítě. Jako nástroj analýzy jsme si zvolili simulované žíhání. V práci jsme postupně představili pět modifikací algoritmu simulovaného žíhání.

Všechny modifikace jsme testovali na reálných datech na čtyřech sítích. Dvě sítě byly menšího rozsahu a jednalo se o části silniční sítě Zlínského kraje. Zbylé dvě sítě byly větší. Veškerá data i obrázky sítí nám poskytlo Centrum dopravního výzkumu, v. v. i. v rámci projektu TRISK č. VG20102015057.

Vytvořené modifikace simulovaného žíhání se od sebe liší způsobem výměny neprůjezdných hran. Při výběru, které hrany z množiny průjezdných hran změníme na neprůjezdné, jsme vycházeli z předpokladu, že hrana, která je více využívaná, je náchylnější ke zneprůjezdnění. Abychom zjistili, které hrany jsou více využívané, aplikovali jsme nejprve Dijkstrův algoritmus pro zjištění všech nejkratších cest v daném grafu. Výsledky tohoto algoritmu jsme použili jako vstupy pro výpočet normovaného betweenness indexu. Tento *NBC* jsme využívali pro stanovení pravděpodobnosti přerušení dané hrany.

Při analýze zranitelnosti menších silničních sítí jsme zvolili 1 000 opakování Metropolisova algoritmu. Protože v každém kroku Metropolisova algoritmu přepočítáváme NBC pro všechny průjezdné hrany, je časová náročnost všech modifikací značně velká. Z toho důvodu jsme zvolili pro větší sítě počet opakování pouze na 500. Pokud bychom počet opakování navýšili, zejména u větších sítí, mohli bychom získat přesnější výsledky. V práci jsme uvedli výsledky pouze pro tři neprůjezdné hrany v případě menších sítí a pro sedm neprůjezdných hran u větších sítí. V rámci testování jsme počítali i s jiným počtem hran, ale uvedli jsme pouze výsledky pro tyto dvě sady, protože se nám výsledky jevily jako nejzajímavější.

Z provedených výpočtů jsme zjistili, že výsledky závisí na velikosti i struktuře sítě. U menších sítí jsme získali velmi podobné výsledky, oproti tomu u větších sítí se výsledky znatelně lišily. Nejlepších výsledků dosahovala modifikace simulovaného žíhání, která pracovala s výměnou pouze jedné neprůjezdné hrany.

Přínosem diplomové práce je vytvoření algoritmů pro zkoumání zranitelnosti sítě. Tyto algoritmy lze použít i na jiné sítě než jen na silniční.

SEZNAM LITERATURY

[1] BERDICA, K., An introduction to road vulnerability: what has been done, is done and should be done, Transport Policy 9, 117-127, 2002.

[2] BÍL, M., Vodák, R.: Odolnost silniční sítě proti následkům katastrof, Silniční obzor - roč. 74, červenec-srpen 2013, 176–179.

[3] FRONČEK, Dalibor. Úvod do teorie grafů. Opava: Slezská univerzita, 1999, 105 s. ISBN 80-724-8044-8.

[4] HLINĚNÝ, P., Základy teorie grafu [online], dostupné z: http://www.fi.muni.cz/~hlineny/Vyuka/GT/Grafy-text07.pdf.

[5] MATOUŠEK, Jiří a Jaroslav NEŠETŘIL. *Kapitoly z diskrétní matematiky*. 2. upr. vyd. Praha: Karolinum, 2002, 377 s. ISBN 80-246-0084-6.

[6] NEWMAN, M. Networks: an introduction. New York: Oxford University Press, 2010, xi, 772 s. ISBN 978-0-19-920665-0.

[7] POSPÍCHAL, Jiří, Vladimír KVASNIČKA a Peter TIŇO. Evolučné algoritmy. 1. vyd. Bratislava: Slovenská technická univerzita, 2000, 215 s. Edícia vysokoškolských učebníc. ISBN 80-227-1377-5.

[8] SLIVONĚ, M., Několik přístupů k identifikaci kriticky důležitých úseků na dopravní síti [online], dostupné z: http://pernerscontacts.upce.cz/12_2008/slivone.pdf.

[9] SPALL, James C. Introduction to stochastic search and optimization: estimation, simulation, and control. Hoboken, N.J.: Wiley-Interscience, c2003, xx, 595 p. ISBN 04-713-3052-3.

[10] TÖPFER, Pavel. Algoritmy a programovací techniky. 1. vyd. Praha: Prometheus, 1995, 299 s. ISBN 80-858-4983-6.

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

Příloha č. 1:

Seznam přiložených programů:

- betweenness.sce
- hrany_pst.sce
- hrany_sestupne.sce
- hrany1_1.sce
- hranyvsechny.sce
- NACTIDATA.sci
- nahoda_hrany.sce
- prohledavani.sce
- simulovane_zihani1.sce
- simulovane_zihani2.sce
- simulovane_zihani3.sce
- simulovane_zihani4.sce
- simulovane_zihani5.sce
- vynechani_hrany.sce
- zaklad.sce

Popis programů:

- zaklad.sce – výchozí program, ve kterém uživatel zadá vstupní parametry a zvolí si modifikaci simulovaného žíhání. V úvodu programu je nutné přepsat cesty k jednotlivým programům.
- betweenness.sce - funkce spočítá pravděpodobnosti, s jakými budou vybrány jednotlivé hrany v grafu ke zneprůjezdnění. Funkce je součástí všech funkcí simulovaného žíhání.
- hrany_pst.sce – funkce z množiny neprůjezdných hran vyměňuje variabilní počet hran na základě stanovených pravděpodobností výběru. Stejný počet hran se poté zprůjezdní. Hrany, které budou zprůjezdněny, jsou vybírány náhodně z množiny neprůjezdných hran. Funkce je součástí funkce simulovane_zihani3.sce.
- hrany_sestupne.sce – funkce z množiny průjezdných hran náhodně zprůjezdní určitý počet hran. Tento počet hran se v průběhu opakování algoritmu snižuje.

Stejný počet hran se z množiny neprůjezdných hran vymění. Funkce je součástí funkce simulovane_zihani5.sce.

- hrany1_1.sce – funkce z množiny neprůjezdných hran náhodně jednu hranu zprůjezdní. Z množiny neprůjezdných hran vymění pouze jednu hranu. Výměna probíhá s ohledem na normovaný betweenness index. Funkce je součástí funkce simulovane_zihani1.sce.
- hranyvsechny.sce – funkce na základě normovaného betweenness indexu vybere celou novou množinu hran, kterou zneprůjezdní. Funkce je součástí funkce simulovane_zihani2.sce.
- NACTIDATA.sci – funkce načte a zakóduje data. Tato funkce byla poskytnuta vedoucím práce.
- nahoda_hrany.sce – funkce v množině neprůjezdných hran vyměňuje náhodný počet hran. Hrany ke zneprůjezdnění jsou vybírány s ohledem na normovaný betweenness index. Funkce je součástí funkce simulovane_zihani4.sce.
- prohledavani.sce – funkce spočítá počet komponent, na které se graf rozpadl po zneprůjezdnění daného počtu hran. Dále spočítá směrodatnou odchylku počtu obyvatel žijících v jednotlivých komponentách. Funkce je součástí všech funkcí simulovaného žíhání.
- simulovane_zihani1.sce – funkce simulovaného žíhání, která pracuje s výměnou jedné hrany v množině neprůjezdných hran. Výstupem funkce je nejmenší zjištěná směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách a jí odpovídající množina neprůjezdných hran.
- simulovane_zihani2.sce – funkce simulovaného žíhání, která pracuje s výměnou všech hran v množině neprůjezdných hran. Výstupem funkce je nejmenší zjištěná směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách a jí odpovídající množina neprůjezdných hran.
- simulovane_zihani3.sce – funkce simulovaného žíhání, která pracuje s variabilním počtem hran, které se v množině neprůjezdných hran vyměňují. Počet hran je ovlivněn stanovenou pravděpodobností výběru. Výstupem funkce je nejmenší zjištěná směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách a jí odpovídající množina neprůjezdných hran.
- simulovane_zihani4.sce – funkce simulovaného žíhání, která pracuje s výměnou náhodného počtu hran v množině neprůjezdných hran. Výstupem

funkce je nejmenší zjištěná směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách a jí odpovídající množina neprůjezdných hran.

- simulovane_zihani5.sce – funkce simulovaného žíhání, která pracuje se snižujícím počtem hran, které se vyměňují v množině neprůjezdných hran. Výstupem funkce je nejmenší zjištěná směrodatná odchylka počtu obyvatel žijících v jednotlivých komponentách a jí odpovídající množina neprůjezdných hran.
- vynechani_hrany.sce - funkce podle *NBC* vybírá hrany, které se zneprůjezdní. Počet těchto hran je dán volbou modifikace simulovaného žíhání.

Při výpočtech jsme používali data uložená v těchto souborech:

- sit1.txt
- sit2.txt
- zlin_cast1_trb.txt
- zlin_cast2_trb.txt.