UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

DIPLOMOVÁ PRÁCE

Analýza sítí

Vedoucí diplomové práce: **RNDr. Rostislav Vodák, Ph.D.**
Vypracovala: **Bc. Pavla Marešová**
Studijní program: N1101 Matematika
Studijní obor Matematika a její aplikace
Forma studia: prezenční
Rok odevzdání: 2015

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: Pavla Marešová

Název práce: Analýza sítí

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 odolnosti silniční sítě vůči náhodným výpadkům. Analýza silniční sítě byla provedena pomocí principu perkolace. První kapitola se věnuje poznatkům teorie grafů, které jsou důležité k pochopení dané problematiky. Druhá uvádí příklady sítí, na které lze metodu následně aplikovat. Třetí kapitola obsahuje popis potřebných algoritmů a metody perkolace. Ve čtvrté kapitole je aplikována perkolace na konkrétních silničních sítích.

Klíčová slova: Síť, graf, komponenta, perkolace

Počet stran: 57

Počet příloh: 4

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: Pavla Marešová

Title: Networks analysis

Type of thesis: Master's

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 resistance of a road network agains random failure. The analysis of the road network was performed using a percolation. The first chapter deals with the basics of graph theory, which is important for understanding the topic. The second chapter contains examples of various networks, on which can be this method applied. The third chapter introduces used algorithms and principle of percolation. In the fourth chapter we apply percolation on the road network and present the results.

Key words: Network, graph, component, percolation

Number of pages: 57

Number of appendices: 4

Language: Czech

Prohlášení

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

V Olomouci dne 16. dubna 2015

Poděkování

Na tomto místě bych chtěla poděkovat především svému vedoucímu diplomové práce za dostatek trpělivosti, poskytnutí studijních materiálů a za pomoc a rady, které vedly k dokončení této práce. Také bych chtěla poděkovat Centru dopravního výzkumu, v. v. i. za poskytnutí dat silničních sítí v rámci projektu TRISK č. VG20102015057. A dále všem, kteří mi jakýmkoli způsobem pomohli při vzniku této práce.

Obsah

Úvod

Moderní doba 21. století se vyznačuje jedním fenoménem. Všechno je propojené se vším, jak ostatně můžeme vidět v mnoha oblastech lidského života. Dnes si už nedokážeme představit, že bychom se nedostali po silnicích z místa A do místa B, že bychom nemohli být připojeni k internetu, že bychom se kvůli nemocem nemohli stýkat s přáteli a mnoho dalšího. Všechny tyto vazby a propojení lze reprezentovat sítěmi.

V nedávné době jsme mohli zaznamenat, že roste počet teroristických útoků na letadla, metra, silnice a s nimi související místa, jako jsou letiště či důležité budovy. Dojde-li k takovému útoku, ochromí to celou síť. Navíc víme, že u takových útoků nikdo nemůže předpovědět, kdy a kde k nim dojde. Dalším potenciálním nebezpečím jsou přírodní katastrofy v podobě zemětřesení, povodní apod., jejichž počet se změnou klimatických podmínek také roste a jejich předvídání není také dost dobře možné.

I z těchto důvodů je cílem této diplomové práce aplikovat nástroje teorie grafů pro analýzu odolnosti sítí vůči náhodným výpadkům a následně navrhnout jejich vylepšení. K této analýze použijeme proces zvaný perkolace. Konkrétní aplikace bude provedena na silniční síti, ale užitou metodu a algoritmy lze modifikovat i pro jiné druhy sítí.

Práce je rozdělena do čtyř kapitol. První kapitola obsahuje potřebné teoretické poznatky z teorie grafů, které jsou nezbytné k pochopení dané problematiky, protože právě pomocí grafů zjednodušíme síť na model, který budeme moci zadat do počítačového programu. Druhá kapitola ilustruje širokou oblast možných aplikací této teorie. Ve třetí kapitole se dostaneme k samotné analýze odolnosti sítí. Uvedeme zde potřebné algoritmy a metody, které následně v závěrečné čtvrté, kapitole aplikujeme na reálná data silniční sítě.

1. Základní pojmy teorie grafů

Jak jsme zmínili v úvodu, každá analýza sítí stojí na teoretickém základu teorie grafů. Základní pojmy této teorie, mezi které patří definice grafu, a to neorientovaného i orientovaného, definice podgrafu, stupně vrcholu, cesty a podobně, může čtenář nalézt v mnoha publikacích týkajících se výhradně teorie grafů. Proto je v úvodní kapitole této práce pouze stručně připomeneme. Bližší pozornost budeme věnovat pojmu komponenty grafu. Právě komponenty představují zásadní roli ve výpočtu perkolace, a proto se jimi budeme zabývat podrobněji.

Definice v této kapitole jsou převzaty z [6] a jsou také uvedeny v [7] nebo [8].

1.1. Definice grafu, typy grafů

Definice 1.1. *Graf je uspořádaná dvojice* (V, E)*. Množina* V *je množina vrcholů (uzlů), množinu* $E \subseteq V \times V$ *nazýváme množinou hran, kde hrana je definovaná pomocí dvojice koncových vrcholů.*

Graf lze proto zadat obrázkem nebo výčtem vrcholů a hran, tj. $V = \{1, 2, 3, 4, 5\}$ $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{1, 5\}, \{2, 5\}\}$. Prozatím hovoříme o neorientovaném grafu. Hrana {2,1} je tedy totožná s hranou {1,2}.

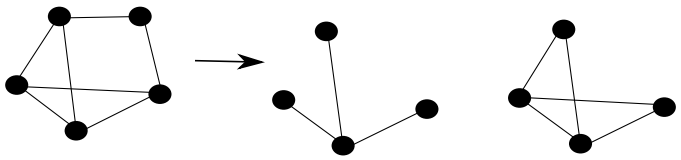
Definice 1.2. *Úplný graf je takový neorientovaný graf, v němž jsou každé dva vrcholy spojené hranou.*

Definice 1.3. *Stupněm vrcholu* v *grafu* G *rozumíme počet hran z něj vycházející. Značíme jej* $d_G(v)$*.*

V úvodní definici jsme zavedli pojem grafu. Mnohdy nás ale nemusí zajímat celý graf. Pokud máme například k dispozici data pro celou silniční síť Evropy, a přitom chceme analýzu provést pouze na silnicích České republiky, z hlediska výpočetního času by bylo nesmyslné zabývat se celou sítí. Proto zavádíme pojem podgrafu.

Definice 1.4. *Podgrafem grafu* G *rozumíme libovolný graf* H *na podmnožině vrcholů* $V(H)$*, kde* $V(H) \subseteq V(G)$*, který má za hrany libovolnou podmnožinu hran*

grafu G majících oba vrcholy ve V(H). Indukovaným podgrafem je podgraf, který obsahuje všechny hrany grafu G mezi dvojicemi vrcholů z V(H).

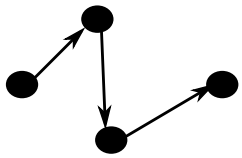


Obrázek 1: Příklad grafu, jeho podgrafu a indukovaného podgrafu

Tato definice nám zaručuje, že nepřevezmeme hrany bez koncových vrcholů.

Doposud jsme hovořili o neorientovaných grafech, ve kterých lze hranu procházet v obou směrech mezi jejími koncovými vrcholy. To ale v mnohých případech není možné. Jako příklad můžeme uvést jednosměrné ulice silniční sítě. Proto vyslovíme definici *orientovaného grafu*, u jehož každé hrany přesně určíme počáteční a koncový vrchol.

Definice 1.5. *Orientovaný graf je uspořádaná dvojice D=(V,E), kde* $E \subseteq V \times V$ *a hrany jsou uspořádané dvojice vrcholů, značíme je* (u, v)*, tj. hrana začíná ve vrcholu* u *a končí ve vrcholu* v*.*



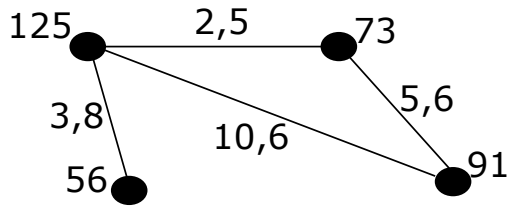
Obrázek 2: Orientovaný graf

Orientované grafy už nemusí být symetrické, tedy hrana (v, u) nemusí existovat.

V mnoha aplikacích si s doposud zavedenými grafy nevystačíme. Potřebujeme totiž, aby graf sebou kromě své struktury nesl i informace o svých vrcholech a hranách. U silnic potřebujeme vědět, jak je která cesta dlouhá nebo kolik žije ve

vrcholech (tj. městech) obyvatel. Ohodnocení tak může reprezentovat kapacitu, vzdálenost a mnoho dalšího. Proto na závěr této podkapitoly uvedeme definici ohodnoceného grafu.

Definice 1.6. *Hranově ohodnocený graf je graf G společně s ohodnocením hran w, kde $w : E(G) \to \mathbb{R}$. Vrcholově ohodnocený graf je graf G společně s ohodnocením vrcholů f, kde $f : V(G) \to \mathbb{R}$. Pokud pro všechny hrany (resp. vrcholy) platí, že $w(e) > 0$ ($f(e) > 0$), hovoříme o kladně ohodnoceném grafu.*



Obrázek 3: Hranově i vrcholově ohodnocený graf

1.2. Procházení grafu

V první podkapitole jsme zavedli pojem grafu a jeho různých typů. Další věcí, kterou o grafem potřebujeme znát, abychom s nimi mohli reálně pracovat, je jak se v grafu pohybovat. Prvním pojmem, který pro pohyb v grafech potřebujeme, je *sled*.

Definice 1.7. *Sledem délky n v grafu G rozumíme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, \ldots, e_n, v_n$, ve které má hrana e_i koncové vrcholy v_{i-1}, v_i. Jestliže $v_0 = v_n$, je sled uzavřený.*

Definice 1.8. *Cyklem rozumíme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, \ldots, e_t, v_t = v_0$, kde vrcholy $v_0, ..., v_{t-1}$ jsou navzájem různé vrcholy grafu G a pro každé i = 1,2,...,t je $e_i = \{v_{i-1}, v_i\} \subset E(G)$.*

Sled, ve kterém se neopakuje žádný vrchol, nazýváme *cestou*. Tím se dostáváme k problematice stanovení délky cesty. U neohodnocených grafů je délka

cesty rovna počtu hran, které na cestě projdeme. U ohodnocených grafů sečteme ohodnocení navštívených hran. V mnoha grafech mezi vybranou dvojicí vrcholů neexistuje právě jedna cesta a my si můžeme vybrat, po které se do cílového vrcholu vydáme. Potom je logická myšlenka, ptát se po *nejkratší cestě*. Pro určení nejkratší cesty u ohodnocených grafů je nejdůležitějším aparátem Dijkstrův algoritmus. U neohodnocených si vystačíme s prohledáváním grafu do šířky. Princip obou metod může čtenář nalézt v [9].

1.3. Souvislost a komponenty souvislosti

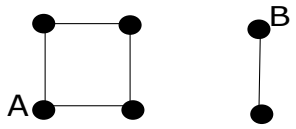
Jestliže graf reprezentuje nějakou síť, je jednou ze základních otázek, zda máme možnost dostat se z každého vrcholu do libovolného jiného vrcholu. Pokud vždy existuje alespoň jedna cesta mezi všemi dvojicemi vrcholů, takový graf nazýváme *souvislý* a síť, kterou představuje, *souvislou*. Naopak graf na obrázku 4 je rozdělen do dvou částí, mezi nimiž neexistuje žádné spojení. Například není žádná cesta z vrcholu A do vrcholu B. Takový graf nazýváme *nesouvislý* a síť *nesouvislou*. Nesouvislý graf se skládá z několika souvislých částí, které nazýváme *komponenty*.

1.3.1. Komponenty neorientovaného grafu

Definice 1.9. *Komponenta souvislosti grafu je taková množina vrcholů, že*
(i) existuje vždy nejméně jedna cesta z každého vrcholu této množiny do všech ostatních vrcholů této množiny
(ii) nejde přidat další vrchol grafu tak, aby byla dodržena vlastnost (i).

Pomocí předchozí definice lze tedy definovat souvislý a nesouvislý graf, a to takto:

Definice 1.10. *Graf je souvislý, pokud je tvořený nejvýše jednou komponentou souvislosti. V opačném případě je graf nesouvislý.*



Obrázek 4: Neorientovaný graf obsahující dvě komponenty souvislosti

Podíváme-li se nyní na obrázek 4, je patrné, že graf je složen ze dvou komponent souvislosti, z nichž jedna je tvořena čtyřmi vrcholy a druhá dvěma. Za komponentu nelze označit například množinu vrcholu A a jeho dvou sousedů (následovníků), protože by sice splňovala vlastnost (i), ale už nikoli vlastnost (ii). Jestliže graf obsahuje izolovaný vrchol, je považován za komponentu velikosti jedna.

Je tedy zřejmé, že každý vrchol patří právě do jedné komponenty souvislosti.

1.3.2. Komponenty orientovaného grafu

V předchozí podkapitole jsme definovali komponenty souvislosti u grafů, u kterých lze hrany procházet v obou směrech. V praxi se ovšem setkáme i s grafy, které mají hrany orientované. Pro názorný příklad nemusíme chodit daleko, každý ho potkává v denním životě a nazývá se World Wide Web. Hrany v internetové síti představují odkazy, které přesměrují uživatele z jedné stránky na druhou, ale pouze v jednom směru. Stane se tak, když například klikneme na internetovou reklamu nebo na nadpis článku na úvodní zpravodajské stránce. Mnohdy ale neexistuje odkaz, který by nás dostal zase zpět. Proto je potřeba zavést definici komponenty i pro takové, orientované, grafy.

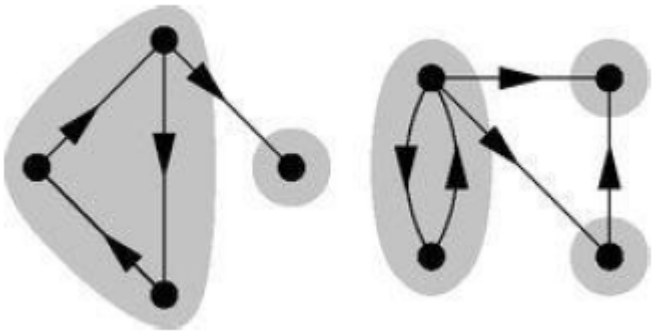
Při stanovení definice narazíme na jeden problém. Pokud existuje cesta z vrcholu A do vrcholu B, ale nikoli z B do A, považujeme vrcholy A, B za spojené? A patří tedy do stejné komponenty?

Na tuto otázku existují dvě odpovědi. Pokud ano, pak nerozlišujeme mezi orientovanou a neorientovanou hranou, a takovou komponentu nazveme *slabě souvislou komponentou*.

V opačném případě považujeme vrcholy A a B za spojené právě tehdy, když

existuje orientovaná cesta z vrcholu A do B a zároveň z vrcholu B do A. Taková dvojice vrcholů patří do *silně souvislé komponenty*.

Definice 1.11. *Silně souvislá komponenta je takový maximální podgraf orientovaného grafu, v němž pro každou dvojici jeho vrcholů u, v existuje sled.*



Obrázek 5: Silně souvislé komponenty orientovaného grafu (převzato z [8])

Graf na obrázku 5 tvoří dvě slabě souvislé komponenty a pět silně souvislých. Poznamenejme, že silně souvislou komponentu může tvořit pouze jeden vrchol a stejně jako u neorientovaných grafů (a tedy slabě souvislých komponent) patří každý vrchol právě do jedné silně souvislé komponenty. Poznamenejme také, že každá silně souvislá komponenta obsahující více než jeden vrchol, musí obsahovat alespoň jeden cyklus. Protože z definice musí existovat cesta z vrcholu do všech ostatních a zároveň z těchto vrcholů zpět do výchozího, tvoří spojení těchto dvou cest cyklus.

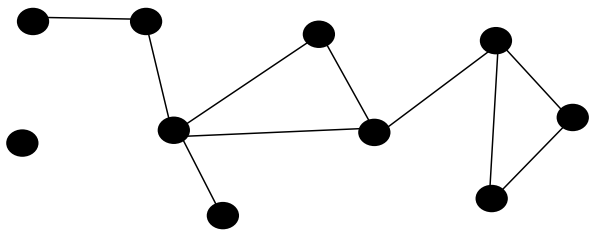
1.4. Rozdělení četnosti stupně vrcholů

V této kapitole se podíváme na jednu ze základních vlastností sítě, tj. na rozdělení četnosti stupně vrcholů.

Ze základních definic víme, že stupeň vrcholu je počet hran, které z něj vychází. Nejprve se zaměříme na neorientované grafy.

Definujeme p_k jako podíl vrcholů grafu, které mají stupeň k.

Uvažujme například síť reprezentovanou grafem na obrázku 6.



Obrázek 6: Neorientovaný graf o deseti vrcholech

Graf se sestává z $n = 10$ vrcholů. Kde jeden má stupeň 0, dva mají stupeň 1, čtyři mají stupeň 2, dva mají stupeň 3 a jeden má stupeň 4. Takže hodnoty p_k pro $k = 0, ..., 4$ jsou

$$p_0 = \frac{1}{10}, p_1 = \frac{2}{10}, p_2 = \frac{4}{10}, p_3 = \frac{2}{10}, p_4 = \frac{1}{10}, \tag{1}$$

a $p_k = 0$ pro $k > 4$. Veličiny p_k reprezentují *rozdělení četnosti stupně vrcholů* daného grafu.

Hodnoty p_k lze také chápat ve smyslu pravděpodobnosti. Je to pravděpodobnost, že náhodně vybraný vrchol grafu má stupeň k. Mnohdy nechceme znát pouze podíl vrcholů daného stupně k, ale jejich celkový počet. Takový výpočet je zřejmý, jejich počet je np_k, kde n je počet vrcholů sítě.

Jiná konstrukce, která dává v podstatě tutéž informaci, je konstrukce *posloupnosti stupňů*, což je množina stupňů všech vrcholů $K = \{k_1, k_2, k_3, ..., k_n\}$. Například pro námi uvedený graf je posloupnost stupňů $K = \{0, 1, 1, 2, 2, 2, 2, 3, 3, 4\}$. Posloupnost nemusí být seřazena vzestupně jako v tomto případě. Její uspořádání může být dáno například očíslováním vrcholů, a tedy na i-té pozici je stupeň vrcholu i.

Podotkněme ještě jednu zřejmou skutečnost. Znalost rozdělení četnosti stupňů (popř. posloupnosti stupňů) neudává jednoznačně kompletní strukturu sítě. Pokud pomineme speciální případy jako $K = \{0\}$, která reprezentuje vždy izolovaný

vrchol, a zvolíme určitou kombinaci hodnot stupňů, pak existuje více než jeden graf, který mohou reprezentovat. Například dva rozdílné grafy na obrázku 7 mají stejnou posloupnost $K = \{1, 2, 2, 2, 1\}$.

Obrázek 7: Grafy se stejnou posloupností stupňů

Rozdělení četnosti stupňů vrcholů můžeme také určovat u orientovaných grafů. U takových grafů rozlišujeme dva různé stupně jednoho vrcholu, tzv. *vstupní a výstupní stupeň*, které definujeme následovně

$$deg^+(u) = |\{e \in E : \exists v \in V : e = (v, u)\}| \tag{2}$$

$$deg^-(u) = |\{e \in E : \exists v \in V : e = (u, v)\}|. \tag{3}$$

A tedy můžeme intuitivně definovat dvě různá rozdělení, zvlášť pro stupně vstupní a zvlášť pro výstupní.

Můžeme ale taky postupovat jinak. Předpokládejme, že rozdělení četnosti orientované sítě je sdružené rozdělení vstupních a výstupních stupňů. Definujeme p_{jk} jako podíl vrcholů, které mají vstupní stupeň j a současně výstupní k. Pokud použijeme toto dvojrozměrné rozdělení, můžeme počítat s tím, že veličiny mohou být korelované. Například že vrcholy s vysokých vstupním stupněm mají vysoký i stupeň výstupní, což se odrazí ve velkých hodnotách p_{jk} pokud j i k jsou velká. Pokud známe pouze oddělená rozdělení, takovou souvislost neuvidíme.

2. Sítě reálného světa

Využití teorie grafů je velmi široké a můžeme jej nalézt v různých aspektech našeho života. I když se v praktické části této práce budeme zabývat pouze silniční sítí a její analýzou, použitou metodu lze zobecnit na všechny následující sítě.

Technické sítě

První velkou oblastí jsou sítě technické. Sem můžeme zařadit internet, ve kterém vrcholy představují počítače nebo routery a hrany fyzické propojení mezi nimi. Více abstraktní sítí je World Wide Web. I když se pojmy "internet"a "web"často zaměňují, web je síť webových stránek a hypertextových odkazů mezi nimi. Do technických sítí můžeme zařadit také velkou skupinu sítí dopravních, do které spadá silniční, letecká, železniční či potrubní síť.

Sociální sítě

Opustíme-li technický svět, jsou další nepominutelnou skupinou sociální sítě. V těch vrcholy představují vždy lidi a hrany jejich vzájemnou interakci. Taková síť může být malá a reprezentovat například členy klubu, zaměstnance firmy apod., nebo naopak celosvětová, jako například facebook.

Biologické sítě

Třetím odvětvím, ve kterém hrají podstatnou roli sítě, je biologie. V té už není jasně určeno, co vrcholy a hrany reprezentují. Mohou to být živočichové a predátoři v potravních řetězcích, můžou to být neurony a jejich propojení v neuronové síti mozku nebo jakákoli biochemická síť látek a jejich vzájemných reakcí, jako například metabolismus či navazování molekul.

3. Užité algoritmy a perkolace

V této práci využíváme několik algoritmů a metod, které v této kapitole vysvětlíme. Pro analýzu odolnosti silniční sítě potřebujeme určit počet komponent, na které se graf rozpadne po zneprůjezdnění určitého počtu hran. Z toho důvodu se budeme zabývat algoritmem, který ze zadání grafu určí počet jeho komponent. A následně jako stěžejní metodu představíme proces perkolace.

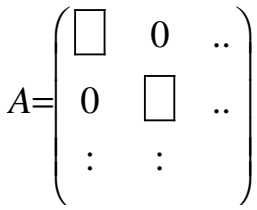
3.1. Implementace grafu v programech

Než se pustíme do představení algoritmu, musíme vědět, jak graf do počítače vůbec zadat. Doposud jsme graf znázorňovali pouze obrázkem. Jestliže potřebujeme s grafy pracovat v počítačových programech, musíme mít k dispozici jinou formu, z které by počítač jednoznačně poznal, jak graf vypadá. Takových metod je k dispozici celá řada a liší se především složitostí zadávání nebo univerzálností. U aplikací také záleží na podstatě sítě, která může rozhodnout o zvolené metodě implementace. My uvedeme pouze dvě nejpoužívanější metody.

3.1.1. Matice sousednosti

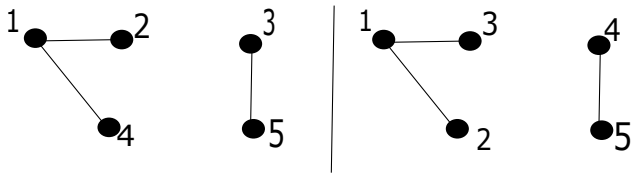
Tato metoda slouží k zadání neohodnoceného grafu, neorientovaného i orientovaného. Matice M je typu $n \times n$ obsahující pouze jedničky a nuly, kde $M(i, j) = 1$ znamená, že existuje hrana mezi vrcholy i a j, hodnota $M(i, j) = 0$, že hrana mezi vrcholy neexistuje. Pokud je graf neorientovaný, tato matice je symetrická. Obdobou matice sousednosti pro ohodnocené grafy je *matice vzdáleností*. Je rovněž typu $n \times n$ a ukládají se do ní ohodnocení jednotlivých hran a znak toho, že hrana neexistuje. Pro graf s kladným ohodnocením hran se jako symbol neexistující hrany používá 0 nebo -1.

Matice sousednosti nesouvislého grafu má následující vlastnost. Lze ji zapsat jako blokovou diagonální matici. Tedy tak, že její nenulové prvky jsou seskupeny do čtvercových bloků podél hlavní diagonály a ostatní prvky jsou nulové. Zápis takové matice je znázorněn na obrázku 8.



Obrázek 8: Matice sousednosti nesouvislého grafu

Poznamenejme ale, že k dosažení tohoto tvaru matice musíme zvolit vhodné očíslování vrcholů. I když očíslování nemá na strukturu grafu vliv, výběr jiného očíslování může mít za následek jiný tvar matice sousednosti, který už nemusí být blokový.



Obrázek 9: Volba očíslování grafu

Podívejme se konkrétně na dvě různá očíslování téže sítě na obrázku 9. Zatímco první síť je reprezentována pouze symetrickou maticí

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

druhou reprezentuje blokově diagonální matice

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

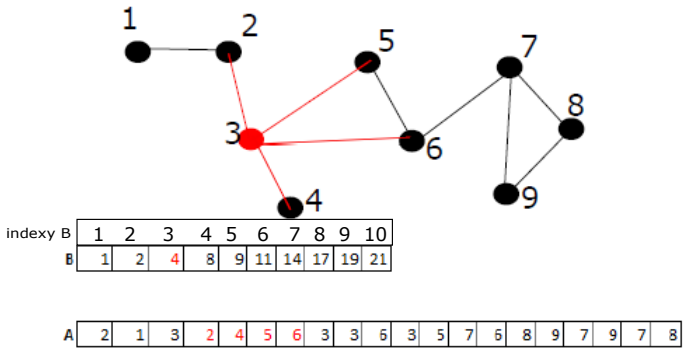
Tedy nemusí být z matice hned zřejmé, zda má graf více komponent. Existuje ale řada algoritmů, které ze zadané sítě s libovolným očíslováním rychle určí její komponenty. Jednomu z nich se budeme věnovat později.

Výhodou této metody je jednoduchost jejího zadání. Na druhou stranu reprezentace pomocí matice sousednosti je výhodná pouze pro úplné grafy nebo grafy jim blízké. V opačném případě, jakým je třeba silniční síť, kde z každého vrcholu vychází malé množství hran, by se matice sestávala z velkého množství nul a výpočet by byl zbytečně paměťově i časově náročný. Pokud si představíme například obec v horách, vede do ní často jenom jedna nebo dvě silnice. Počítáme-li s ní ale do celorepublikové sítě, kde za vrcholy vezmeme jednotlivé obce, byla by to matice přibližně typu 6000×6000. Řádek v matici sousednosti příslušný této vesnici by obsahoval pouze jedno nebo dvě nenulová čísla a zbytek nul. Pro takové sítě je tedy vhodnější metoda *seznamu následovníků*.

3.1.2. Seznam následovníků

Seznam následovníků jednotlivých vrcholů je úspornější reprezentace grafu. V grafu jsou vrcholy číslovány od 1 vzestupně. Ke každému vrcholu si pamatujeme čísla vrcholů, do kterých z něj vede hrana, tedy jeho následovníků. Ukládat následovníky každého vrcholu do samostatného pole by bylo paměťově náročné, proto se používají pro uložení celého grafu dvě jednorozměrná pole: v poli A (následovníci) jsou uložena čísla všech vrcholů, do kterých vedou hrany a v poli B (ukazatelé) jsou ukazatelé, které určují, kde v poli A začínají následovníci konkrétního vrcholu, jehož číslo je reprezentováno indexem v poli B. Vrchol grafu s číslem u má tedy své následovníky v poli A začínající na pozici s indexem $B(u)$. Na pozici $B(u+1)$ už začínají následovníci dalšího vrcholu, a proto poslední následovník vrcholu u je na pozici $B(u+1)-1$. Následovníci u jsou tedy vrcholy s čísly $A(B(u)), A(B(u)+1), \ldots, A(B(u+1)-1)$. Velikost obou polí lze určit předem. V poli A jsou všechny hrany grafu. Hrany obousměrné jsou započítány dvakrát. Pole B by intuitivně mělo obsahovat pouze tolik prvků, kolik je vrcholů. Jako poslední prvek pole B je z technických důvodů (tj. abychom mohli procházet

následovníky i posledního vrcholu) ukazatel na první volnou pozici pole A, proto je jeho hodnota o jedno větší než velikost pole A.
Příklad reprezentace pomocí seznamu následovníků je na obrázku 10.



Obrázek 10: Reprezentace grafu pomocí seznamu následovníků

3.2. Algoritmus počtu komponent

Základním vstupem procesu perkolace je znalost počtu komponent námi studovaného grafu. K jeho zjištění existuje spousta algoritmů, vyskytujících se v různých obměnách. Jejich princip je ale stále stejný, a to procházení grafu do šířky.

Uvažujme nejprve neohodnocený neorientovaný graf s počtem vrcholů N. *(Stanovení silných a slabých komponent orientovaného grafu se věnovat nebudeme.)*

Algoritmus probíhá následovně:

1. Deklarujeme všechny vrcholy jako zatím nenavštívené.
2. Vybereme výchozí vrchol s označením i=1 a označíme jej za navštívený.
3. Najdeme jeho následovníky, uložíme je do fronty a označíme je za navštívené.
4. Odebereme z fronty následovníků vrchol a najdeme jeho následovníky. Do fronty zařazujeme pouze následovníky, kteří jsou dosud nenavštíveni a označíme je za navštívené.
5. Pokud není fronta prázdná, vrátíme se ke kroku 4. Tím najdeme všechny

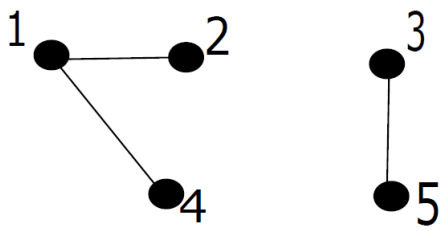
vrcholy patřící do stejné komponenty s vrcholem i.

6. Vezmeme vrchol i=i+1.

7. Je-li dosud nenavštíven, pokračujeme krokem 2. Je-li navštíven, pokračujeme krokem 6.

8. Po projití všech N vrcholů algoritmus končí.

Výstupem algoritmu je počet komponent grafu, a díky jeho procházení a postupnému ukládání, i data o jejich složení, tj. kolik má každá komponenta vrcholů a jaké je jejich označení. Výstupy pro příklad na obrázku 11 jsou $pocetKomp = 2$, $velikost = \{3\ 2\}$, $cleny = \{[1\ 2\ 4]\ [3\ 5]\}$



Obrázek 11: Graf se dvěma komponentami souvislosti

V předchozím algoritmu jsme předpokládali neohodnocený graf. U ohodnocených grafů do algoritmu přidáme podmínku na ohodnocení. Hrany s ohodnocením nekonečno bude algoritmus považovat za neexistující.

3.3. Perkolace

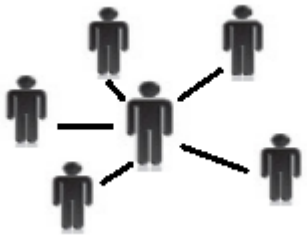
V této kapitole se budeme zabývat jedním z procesů odehrávajícím se na sítích, perkolací, a jejím použitím pro model odolnosti sítě. Obecně procesy na sítích, včetně perkolace, se zabývají publikace [1], [4] a [5].

Mějme libovolnou síť a představme si, že z ní odstraníme nějaké vrcholy společně s hranami, které k těmto vrcholům patří. Například selhání internetových routerů muže být chápáno tak, že jsme ze sítě představující internet odstranili jim odpovídající vrcholy a společně s nimi spojení do zbytku sítě. Přibližně 3 % routerů jsou v kterékoli denní době nefunkční, ať už z jakéhokoli důvodu, a je tedy

přirozené se zajímat o to, jaký to má dopad na funkčnost celé sítě. Dalším příkladem je očkování jedinců proti nemocem. Jak jsme uvedli v kapitole sociálních sítí, šíření nemoci v populaci lze popsat množinou jednotlivců a jejich vzájemným kontaktem. Jenže je-li jedinec očkovaný, a tudíž nemůže nemoc chytit, pak k šíření nemoci nepřispívá. I když je samozřejmě jedinec v síti stále přítomen, z hlediska šíření nemoci chybí. A tedy proces očkování můžeme opět formálně reprezentovat odstraňováním vrcholů grafu.

Proces postupného odstraňování vrcholů ze sítě nazýváme *perkolace* a lze jej využít k modelování jevů reálného světa. Jedním z cílů teorie perkolace je porozumět právě tomu, jak odstraňování nebo selhání vybraných vrcholů ovlivňuje síť jako celek.

Všimněme si jedné významné výhody perkolace. Očkování jedince v populaci nejenže chrání proti nemoci jeho samotného, ale také zabraňuje v šíření a nakažení dalších. Tento efekt má za následek, že v některých případech naočkování relativně malé části populace způsobí efektivní prevenci proti šíření nemoci v celé populaci. Triviální příklad takového efektu je ilustrován na obrázku 12, kde vidíme graf reprezentující jednoho chlapce a jeho pět kamarádů. Pokud se nechá očkovat právě dotyčný chlapec, není už možnost, jak by se nemoc mezi nimi šířila, a přitom jsou náklady na očkování šestinové. Tento příklad je pouze ilustrační, protože je více než pravděpodobné, že by jeho kamarádi přišli do kontaktu i mezi sebou.



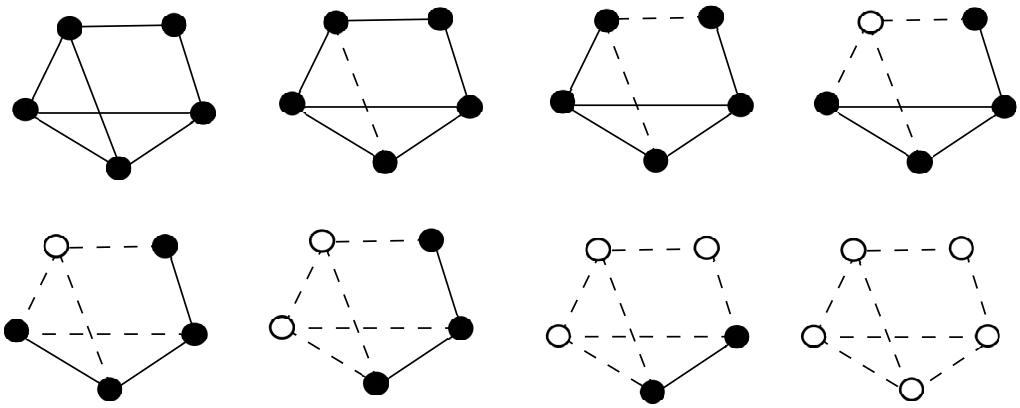
Obrázek 12: Populační síť

V sítích ovšem nemusíme odstraňovat pouze vrcholy. Někdy je tím, co selže

hrana nebo skupina hran. Například v silničních sítích je málo pravděpodobné, že by z ní vypadlo přímo město. Mnohem pravděpodobnější je uzavření silnice mezi dvěma vrcholy, a tedy zneprůjezdněná hrana, se kterou od té chvíle můžeme počítat, jako by v síti vůbec nebyla. Takový jev modelujeme mírně odlišnou perkolací, a to takovým procesem, ve kterém postupně odstraňujeme hrany. Pokud je potřeba tyto procesy rozlišovat, nazýváme je *vrcholová perkolace (site percolation)* a *hranová perkolace (bond percolation)*.

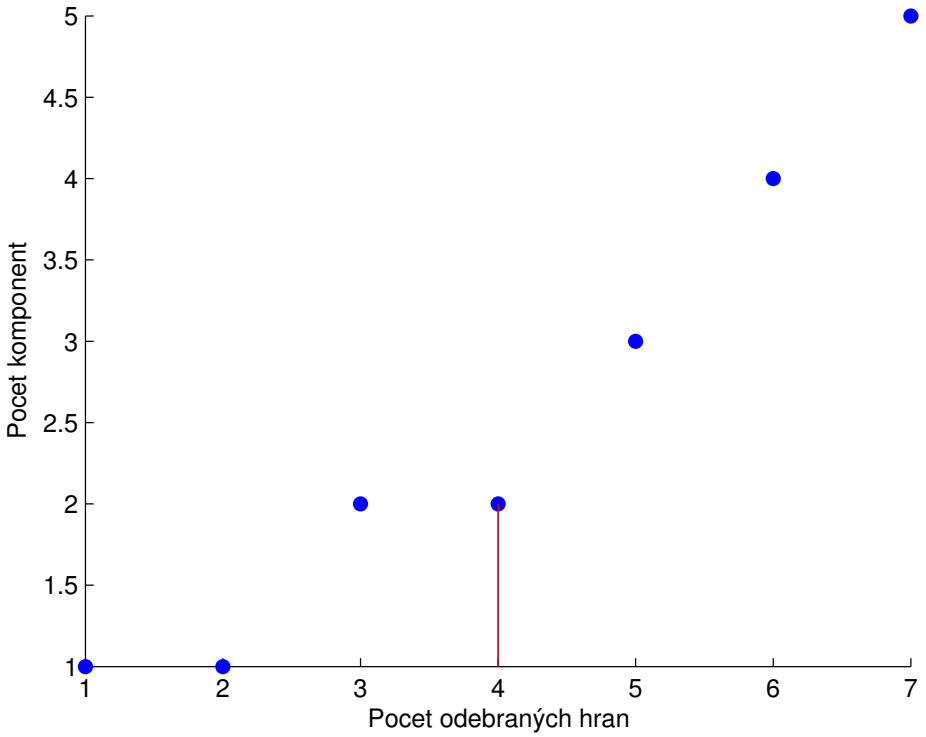
Při postupném odebírání nemusíme postupovat jedním způsobem a nabízí se otázka, v jakém pořadí odstraňování provádět. Principů, kterými se řídit, je více a v každé konkrétní situaci je efektivnější jiný. Nejvíce používaným, a také s pojmem perkolace nejvíce spojovaným, je vypouštění zcela náhodné. Další alternativou je odstraňování v předem daném pořadí. Například u problému šíření nemocí bychom postupovali sestupně od vrcholů nejvyššího stupně k nižším, a tedy očkovali lidi, kteří mají nejvíce propojení na ostatní, a tedy je u nich očkování nejefektivnější. U silniční sítě bychom mohli hrany odebírat sestupně, například podle jejich vytíženosti.

Nadále už se ale budeme zabývat pouze hranovou perkolací s náhodným pořadím vypouštěných hran, kterou jsme zvolili proto, že chceme studovat náhodné a nepředvídatelné výpadky. Než přistoupíme k aplikaci na data reálné sítě, ukažme si princip hranové perkolace na jednodušším příkladě na obrázku 13. Zde je znázorněna souvislá síť o pěti vrcholech a sedmi obousměrných hranách. Vygenerovali jsme náhodné pořadí, v jakém budeme hrany zneprůjezdňovat. Vidíme, že po vypuštění jedné hrany nenastala žádná změna a graf je stále souvislý. Stejný stav je i po odebrání druhé hrany. Při vypuštění třetí hrany se ale graf rozpadá na dvě komponenty a jeden vrchol zůstává izolovaný. Stejné komponenty zůstávají i po odebrání čtyř hran. Zásadní zlom nastává s pátou hranou. S každou další odebranou hranou už totiž automaticky vzniká další komponenta a dochází ke kaskádovitému selhání. Hodnotu, od které počet komponent lineárně závisí na počtu odebraných hran, nazýváme *perkolační práh*. Proces vždy končí grafem složeným z komponent velikosti jedna (izolovaných vrcholů).



Obrázek 13: Princip hranové perkolace

Závislost počtu komponent souvislosti na počtu odebraných hran znázorňuje graf na obrázku 14, ve kterém je svislou čarou vyznačen perkolační práh.



Obrázek 14: Princip hranové perkolace

Uvědomme si, že závislost počtu komponent na odebíraných hranách je silně určena pořadím hran. Při jiném vygenerovaném náhodném pořadí odebíraných hran bude perkolační proces vypadat odlišně. Například tři komponenty mohou vzniknout už při odebrání čtyř hran, ne až s pátou jako v námi uvedeném případě. Proto musíme pořadí generovat vícekrát, tedy i vícekrát provést perkolaci, a zjistit průměrné hodnoty počtu komponent, které už budou mít vypovídající hodnotu o dané síti.

4. Aplikace na reálné sítě

V závěrečné kapitole této práce přistoupíme k aplikaci teorie perkolace na reálné silniční sítě. Ještě před dosaženými výsledky uvedeme, odkud byla vstupní data poskytnuta a jakou mají formu.

4.1. Vstupní data

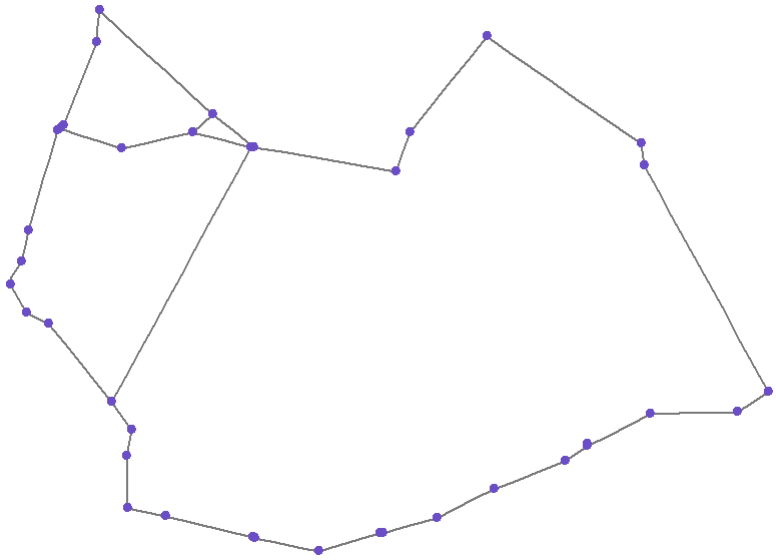
Veškerá data o sítích, se kterými v práci pracujeme, poskytlo Centrum dopravního výzkumu, v. v. i. (zkráceně CDV). Data jsou uložena v textovém souboru. Řádek začíná kódem vrcholu a počtem obyvatel v tomto vrcholu. Následují všichni jeho následovníci s počty obyvatel, kódem hrany a vzdáleností uvedenou nejdříve v metrech a poté v sekundách. Vrchol s jedním následovníkem je reprezentován takto: $2514A037; 327\ 2514A038; 102; 2514A037\ 2514A038; 118; 9$

Souřadnice vrcholů máme uložené v dalším textovém souboru, který na řádku obsahuje kód vrcholu a jeho souřadnice.

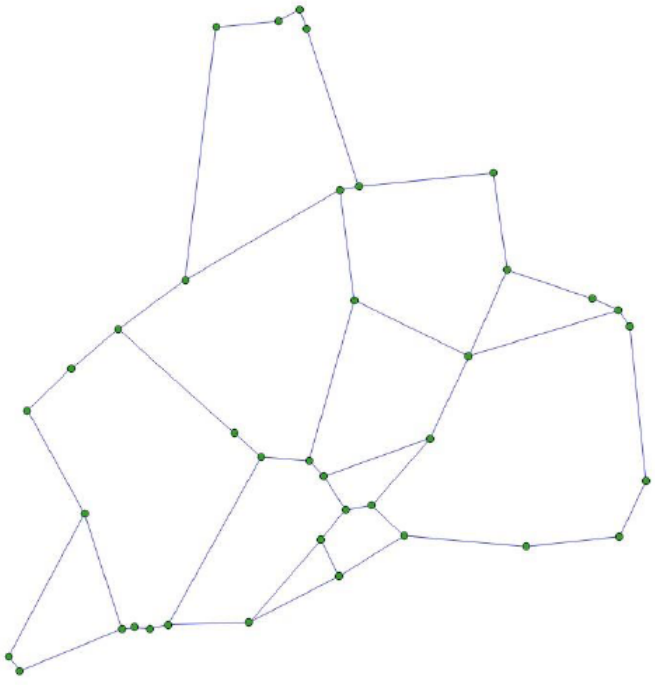
Např. 2514A037 -497929.610 -1145678.360.

Pro dekódování takových souborů použijeme jednoduchý program v Matlabu, který tento soubor bude procházet jako textový řetězec, vyhodnocovat a ukládat informaci, kterou zjistil, pokud narazil na mezeru.

Pro účely této práce jsme využili data ze dvou částí Zlínského kraje. Obě sítě mají podobný počet vrcholů i hran. První síť má 37 vrcholů a 40 hran, druhá 38 vrcholů a 49 hran. Z grafického znázornění na obrázcích 15 a 16 je zřejmé, že obě sítě, i když mají podobné rozměry, jsou naprosto odlišné.



Obrázek 15: Síť 1



Obrázek 16: Síť 2

4.2. Perkolace na silniční síti

V podkapitole věnované principu perkolace jsme uvedli základy tohoto procesu. Nyní upřesníme, jak perkolace vypadá na silniční síti.

Někdo se může ptát, proč vůbec k analýze odolnosti silniční sítě volit náhodnou perkolaci, když nejkritičtější cesty lze určit z méně náhodných metod, které počítají s vytížeností cesty, počty obyvatel ve městech nebo průjezdností cesty v jednotlivých denních hodinách (viz [2], [3]). Jenže i náhodnost perkolace má reálný význam. V každodenních situacích na silnicích můžeme pozorovat jednu skutečnost. V každém městě se v ranní a odpolední špičce zneprůjezdní každý den ty stejné cesty. Důvod je zřejmý, každý volí při cestě do práce nebo z práce stále tu nejrychlejší cestu, což bývá většinou obchvat města. V analýzách zabývajících se touto skutečností, by byl považován za nejkritičtější hranu, a zjišťovali bychom, jak síť vylepšit, aby obchvat nebyl každodenně tak vytížen. Jenže co když se v síti odehraje něco nepředvídatelného? Může to být teroristický útok, přírodní katastrofa v podobě záplav či zemětřesení a mnoho dalšího. V takových situacích nikdo předem neví, jak se zachová. Zda vůbec bude volit cestu autem nebo jestli se spolehne na jinou formu dopravy. V tu chvíli se predikce obvyklé průjezdnosti mění a v síti můžou selhávat naprosto jiné hrany. Bylo by ale dobré, aby silniční síť s takovou situací počítala a byla odolná i v takovém případě. Tím se dostáváme k perkolaci, kde volbou náhodných zneprůjezdnění zjistíme odolnost sítě vůči všem možnostem náhodných selhání.

Jak jsme uvedli v minulé kapitole, u silniční sítě už se budeme zabývat pouze hranovou perkolací. Dále musíme určit, jak informaci o zneprůjezdnění do grafu zadáme. Intuitivní myšlenka určitě je, že koncové vrcholy hrany odstraníme ze seznamu následovníků a budeme počítat s novým grafem, který tuto hranu nebude vůbec obsahovat. Tento přístup je ale z hlediska výpočetního času dosti nevhodný. Obzvláště když si uvědomíme, že ze vstupních dat máme přístup ke vzdálenostem mezi vrcholy, které jsou v původním grafu vždy konečné. Takže hranu nemusíme přímo odstraňovat, pouze jí přiřadíme ohodnocení nekonečno a do algoritmu počtu komponent přidáme podmínku, že následovník existuje pouze

v případě, že do něj vede hrana s konečnou vzdáleností. Jelikož se u silniční sítě můžeme často setkávat se slepými vrcholy, tj. s vrcholy, do kterých vede pouze jedna hrana, při odstranění této hrany sice narůstá počet komponent, ale přitom je izolování jednoho vrcholu z hlediska celé sítě zanedbatelné. Pokud například v síti o 40 vrcholech odebráním jedné hrany rozdělíme síť na komponentu 39 vrcholů a jednoho izolovaného vrcholu, nemá to takový důsledek, jako když odebráním jedné hrany rozdělíme síť na 20 a 20 vrcholů. Proto v procesu perkolace zároveň s počtem komponent sledujeme i velikost největší komponenty, kterou máme jako jeden z výstupů algoritmu počtu komponent.

Jak tedy probíhá algoritmus perkolace na silniční síti:

1. Zjistíme kolik hran graf obsahuje a vygenerujeme jejich náhodnou permutaci P.
2. Deklarujeme pole x a y délky length(P).
3. Pomocnou proměnnou nastavíme na i=1.
4. Délku hrany na pozici P(i) nastavíme na nekonečno (zneprůjezdníme).
5. Pomocí algoritmu počtu komponent spočítáme počet komponent a velikost největší komponenty upraveného grafu.
6. Na pozici x(i) uložíme počet komponent po zneprůjezdnění i hran.
7. Na pozici y(i) uložíme velikost největší komponenty po zneprůjezdnění i hran.
8. Vypustíme další hranu, tedy nastavíme i=i+1 a pokračujeme krokem 4.

Pomocí tohoto algoritmu zjistíme tendenci rozpadu sítě pro jedno náhodně vygenerované pořadí hran. To ale není pro celou síť dostatečně vypovídající, protože, jak jsme ukázali i na jednoduchém příkladu v kapitole 3.3, rozpad pro jiné pořadí může být značně odlišný. Například při odstranění první hrany se ve většině případů nestane nic a graf má stále jednu komponentu. Pokud ale zrovna jako první trefíme hranu, která byla jedinou přístupovou cestou do některého vrcholu, po jejím zneprůjezdnění máme hned dvě komponenty. Z tohoto jasně vyplývá, že algoritmus musíme nechat proběhnout vícekrát, uložit si všechny výsledky a následně z nich určit průměrnou hodnotu.

Kolikrát by měl algoritmus proběhnout není přesně definováno. V práci jsme stanovili pevný počet opakování na 500. Výstupem jsou tedy dvě matice *(zn. N500, M500)*, obsahující 500 řádků, z nichž na každém je uložen jeden průběh předchozího algoritmu. Jedna obsahuje průběh v počtu komponent (řádky jsou pole x), druhá průběh ve velikosti největší komponenty (řádky jsou pole y). Abychom měli jistotu, že je tento počet dostatečný, pokračujeme ve výpočtu dále a sledujeme, jak moc další permutace změní dosavadní průměrné hodnoty. Určíme tedy průměr sloupců matice N500 a N501. Jestliže maximum jejich rozdílu nepřesáhne 0.001, považujeme počet opakování za dostatečný. Pokud je rozdíl větší, tak program pokračuje ve výpočtu dalšího řádku matic M a N.

4.3. Možnosti optimalizace

Známe-li původní podobu sítě a z doposud uvedených metod zjistíme, že je oproti náhodným výpadkům málo odolná, potřebujeme ji nějakým způsobem vylepšit. Protože zkoumáme rozpad na komponenty, je zřejmé, že síť, která bude hustší, propojenější, bude proti takovému rozpadu odolnější. Jednoznačnou optimalizací je tedy přidání hrany, tedy postavení nové silnice. Jak vybrat tu nejlepší variantu, si ukážeme v této podkapitole.

4.3.1. Přidání jedné hrany

Nejjednodušší optimalizací silniční sítě je přidání pouze jedné hrany. Takže jaké hrany máme pro danou síť k dispozici? Budeme-li uvažovat čistě teoreticky, pak do sítě můžeme přidat jakoukoli hranu, která v ní doposud neexistovala, a podívat se, jak právě tato hrana síť vylepšila. Obsahuje-li síť například 100 vrcholů a my víme, že vrchol 1 má následovníky vrcholy 4, 8 a 12, pak máme možnost přidávat všechny hrany, které budou vrchol 1 spojovat se všemi ostatními 96 vrcholy. Tedy bychom museli minimálně 500x96-krát provést algoritmus perkolace, a to jsme teprv u vrcholu 1. Kdyby podobně vypadaly i ostatní vrcholy, výpočet by byl velmi časově náročný, ale ne nemožný.

Nám ovšem v této práci jde o reálnou aplikaci na silnicích a tedy se nabízí

možnost všechny tyto teoretické kombinace nějak zúžit. Je totiž dosti nereálné navrhovat jako optimální vylepšení hranu, která bude spojovat vrcholy ležící na opačných koncích sítě, a tedy by její skutečná výstavba zahrnovala několik přemostění stávajících cest. Jako přípustné budeme tedy volit pouze hrany, které ani jednou nekříží stávající hrany. K tomuto zjištění použijeme ze vstupních dat souřadnice koncových vrcholů a pomocí analytického výpočtu zjistíme, zda zvolená hrana má společný průsečík s nějakou stávající hranou. Tento výpočet vypadá následovně:

1. Chceme přidat hranu [A B], mezi vrcholy se souřadnicemi $A = (a_1, a_2), B = (b_1, b_2)$.

2. Pomocnou proměnnou i nastavíme na jedničku.

3. Vezmeme hranu s číslem i, která má koncové vrcholy $E = (e_1, e_2), F = (f_1, f_2)$.

4. Zjistíme existenci průsečíku hran [A B] a [E F]. Stanovíme tedy matici

$$L = \begin{pmatrix} a_1 - b_1 & f_1 - e_1 \\ a_2 - b_2 & f_2 - e_2 \end{pmatrix} \tag{4}$$

a vektor

$$q = \begin{pmatrix} f_1 - b_1 \\ f_2 - b_2 \end{pmatrix}. \tag{5}$$

5. Je-li hodnost matice $rank(L) \neq 1$, zároveň $rank(L) = rank([L\ g])$ a řešení λ soustavy $\lambda * L = q \in (0, 1)$, pak průsečík existuje a leží uvnitř sítě. Tedy hranu [A B] nemůžeme přidat a algoritmus končí.

6. Pokud není splněna některá podmínka kroku 5, průsečík s hranou [E F] neexistuje a pokračujeme krokem 3 s hranou i=i+1.

7. Pokud projdeme všechny hrany grafu, aniž by se výpočet ukončil, lze hranu [A B] přidat.

U optimalizace jednou hranou máme z hlediska výpočtu dvě reálné možnosti. Buď si ze zadané sítě spočítáme všechny kandidáty na přidání, které si uložíme do matice obsahující dva sloupce s počátečním a koncovým vrcholem přípustné hrany. Takovou matici bychom potom po řádcích procházeli a prošli tak všechny

možnosti optimalizace a vybrali tu nejlepší. Při tomto deterministickém přístupu máme zaručeno, že na závěr vybereme optimální vylepšení, protože jsme prošli všechna přípustná vylepšení. Nebo do sítě přidáme náhodně vygenerovanou hranu, zjistíme míru jejího vylepšení a to budeme porovnávat s další náhodnou hranou. Z každé takové dvojice si zapamatujeme tu lepší variantu. Pokud toto provedeme dostatečně krát, dojdeme i tímto přístupem k optimalizaci. První přístup je vhodný pro malé sítě, na kterých je čas výpočtu perkolace krátký a všech přípustných kandidátů není mnoho. U rozsáhlejších sítí bychom přistoupili k druhé variantě.

Zbývá nám stanovit kritérium, podle kterého budeme rozhodovat o tom, které vylepšení je optimální. Pokud si vykreslíme průměrné hodnoty matice N pro původní síť (tyto hodnoty jsou uloženy v poli $meanN_{puv}$ délky m) a pro síť s přidanou hranou (tj. pole $meanN_{new}$), graf vylepšené sítě bude ležet pod původní sítí. Nejlepší vylepšení potom tedy leží "nejníže"pod grafem původní sítě, hledáme tedy minimum účelové funkce f

$$f(new) = \sum_{i=1}^{m} meanN_{new}(i) - \sum_{i=1}^{m} meanN_{puv}(i) \tag{6}$$

Postupujeme tedy takto:

1. Chceme zjistit hodnoty pole $meanN_{best}$, tedy průměr sloupců matice N pro nejlepší vylepšení.
2. Spočítáme průměr matice N pro původní síť ($meanN_{puv}$) a zjistíme jeho délku $m = length(meanN_{puv})$.
3. Zatím jsme nepřidali žádnou hranu a tedy do proměnné $porS$ (tj. původní hodnota účelové funkce) uložíme nulu.
4. Do sítě přidáme vybranou hranu a spočítáme průměr matice N pro upravenou síť ($meanN_{new}$).
5. Spočítáme hodnotu účelové funkce

$$porN = f(new) = \sum_{i=1}^{m} meanN_{new}(i) - \sum_{i=1}^{m} meanN_{puv}(i) \tag{7}$$

6. Je-li hodnota $porN < porS$, je nová varianta vylepšením, a tedy si ji uložíme $meanN_{best} = meanN_{new}$ a $porS = porN$.

7. Vrátíme se k původní podobě sítě a pokračujeme krokem 4 pro jinou přidanou hranu.

6. Po projití všech (popř. dostatečného množství) možných kandidátů na přidání máme v poli $meanN_{best}$ uloženou nejlepší optimalizaci.

V tomto algoritmu pomocí proměnných $porS$ a $porN$ porovnáváme nové sítě stále s původní sítí a zjišťujeme, která leží "níže"pod grafem původní sítě. Neporovnáváme dvě nové sítě pouze mezi sebou, protože nemáme zaručeno, že se jejich vykreslení nekříží. Pozor si také musíme dát na krok 5. Pole $meanN_{new}$ a $meanN_{puv}$ nemají stejnou délku! Jelikož nová síť má hranu navíc, je i pole $meanN_{new}$ o jedno delší. Poslední hodnotu tedy nemáme s čím porovnávat a kdybychom ji zahrnuli do součtu, nemělo by porovnání smysl. Proto sčítáme jen hodnoty do délky $meanN_{puv}$.

Na závěr kapitoly věnované této variantě optimalizace poznamenejme ještě jednu věc. I když přistoupíme k deterministickému postupu výpočtu a projdeme tedy všechny přípustné hrany, nestačí algoritmus provést jenom jednou. Stále je totiž algoritmus náhodný v hodnotách pole $meanN_{puv}$, se kterými hrany porovnáváme a které jsou s novým spuštěním lehce odlišné, protože se vygeneruje jiných 500 permutací. Proto pro kontrolu výsledků spouštíme algoritmus optimalizace alespoň dvakrát a očekáváme, že hrana, která byla nejlepší při prvním spuštění, se při druhém objeví alespoň v první desítce nejlepších hran. Proto je užitečné, si v algoritmu kromě nejlepší hrany ukládat i pořadí ostatních hran.

4.3.2. Přidání více hran

Pokud optimalizace pomocí jedné hrany není dostatečná, nebo pokud máme dostatek finančních prostředků na výstavbu více silnic, můžeme se zabývat optimalizací pomocí většího počtu přidaných hran. V takovém případě získáváme hned několik možností, jak postupovat. I když se přidávání dvou, tří, čtyř nebo více hran v principu neliší, hned na úvod si musíme stanovit, kolik hran budeme

přidávat. Asi nás hned napadne, že čím víc hran přidáme, tím lépe. Což je sice pravda, ale s touto myšlenkou musíme být opatrní. Mohli bychom totiž narazit na problém, že příliš velký počet hran do grafu ani přidat nelze. Pořád totiž zůstává platná podmínka, že nové hrany nesmí křížit stávající, a navíc se nám problém komplikuje tím, že ani přidané hrany se nesmí křížit mezi sebou. Pokud bychom analyzovali graf s velkým počtem hran, napadne nás, že jedna přidaná hrana jej moc nezlepší a stanovili bychom si počet nových hran třeba na deset. Jenže právě graf s velkým počtem hran nám nedává moc příležitostí na přidání tak vysokého počtu, aniž by se žádné dvě nekřížili a my nenajdeme ani jednu kombinaci deseti, natož tak abychom porovnávali různé varianty.

U optimalizace pomocí jedné hrany bylo přípustné vybírat ze dvou možností výpočtů. U více hran ale už musíme přístup pomocí matice kandidátů zavrhnout. S ohledem na výpočetní rychlost nemá smysl stanovovat celou matici. Spočítali bychom ji totiž pro původní síť, pak z ní vybrali jednu hranu, tu přidali a ostatní kandidáti už by mohli novou hranu křížit. Počítali bychom ji tedy znovu pro novou síť už s jednou hranou navíc, abychom z celkového počtu vybrali opět jen jednu. Takových matic bychom tedy počítali tolik, kolik přidáváme hran, a přitom bychom většinu výsledku nikdy nepoužili.

Proto volíme přístup pomocí náhodného generování hran. Tedy vygenerujeme čísla dvou vrcholů grafu a pouze ověříme, jestli hrana mezi těmito vrcholy neexistuje (prohledání pole následovníků) a pokud ne, tak jestli ji lze s ohledem na počet průsečíků přidat. Výpočetní čas těchto dvou podmínek je daleko nižší než pro stanovení matice. Takto vygenerujeme i další hrany, ovšem s každou další ověřujeme podmínky pro již změněnou síť. Jakmile tímto postupem získáme přípustnou kombinaci stanoveného počtu hran, přidáme do sítě všechny a dále už postupujeme jako u optimalizace jednou hranou. Zjistíme průměrný rozpad původní a nové sítě a zlepšení posuzujeme podle kritéria z předchozí kapitoly. Toto je jeden z možných přístupů optimalizace větším počtem hran. (Výsledky této metody jsou uvedeny v kapitole 4.4.3). Nevýhoda tohoto přístupu je, že s každým dalším pokusem přidáváme naprosto odlišnou kombinaci hran. Z toho důvodu si

ukážeme ještě jednu metodu, která je určitou obdobou algoritmu ***simulovaného žíhání***.

Simulované žíhání se řadí mezi stochastické optimalizační algoritmy pro hledání globálního minima funkce. Jeho hlavní předností je, že připouští i dočasné zhoršení hodnot účelové funkce. Tím se můžeme vyhnout uváznutí v lokálním minimu. Hledáme tedy globální minimum funkce $f(x)$. Máme počáteční stav x. Stav systému x se změní na stav x'. O pravděpodobnosti přijetí změněného stavu rozhoduje tzv. *Metropolisovo kritérium*

$$P(x \to x') = \begin{cases} 1, & f(x') < f(x) \\ e^{-\frac{f(x')-f(x)}{T}}, & f(x') \geqq f(x). \end{cases}$$

Metropolisovo kritérium nám tedy říká, že pokud má nový stav menší funkční hodnotu než stav původní, pak nový stav nahradí původní. V opačném případě je nový stav přijat s pravděpodobností P. Hodnota této pravděpodobnosti závisí nejen na tom, o kolik je nový stav horší, ale také na parametru T. Jelikož má simulované žíhání základ ve fyzice, značí T teplotu. Zvolíme-li vhodně počáteční teplotu T_0, metoda simulovaného žíhání prohledává prostor nejprve silně stochasticky a často přijímá stavy s horší funkční hodnotou. Počáteční teplota by měla být zvolena tak, aby zhruba polovina stavů s horší funkční hodnotou byla Metropolisovým kritériem přijata. Se snižováním teploty Metropolisovo kritérium připouští už jen takové stavy, které vedou ke zlepšení funkční hodnoty.

Obecný algoritmus simulovaného žíhání probíhá následovně:

1. Stanovíme počáteční teplotu T_0 a koncovou teplotu T_{konec}. Vygenerujeme počáteční stav x a spočítáme $f(x)$.

2. Provedeme Metropolisovo kritérium:

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 vygenerujeme náhodné číslo r z intervalu (0,1), vypočítáme mez Metropolisova kritéria, $mez = e^{\frac{-\delta}{T}}$, jestliže $r < mez$, pak nový stav přijmeme.

3. Provedeme snížení teploty T.

4. Pokud $T \geq T_{konec}$ pokračujeme krokem 2, v opačném případě je algoritmus u konce.

Při užití této metody pro optimalizaci silniční sítě, znamenají stavy x a x' jiné kombinace přidaných hran, a jelikož v práci hledáme největší rozdíl oproti původní síti, vede nás to k hledání minima účelové funkce f

$$f(new) = \sum_{i=1}^{m} meanN_{new}(i) - \sum_{i=1}^{m} meanN_{puv}(i). \tag{8}$$

Zbývá nám rozhodnout, jak budeme generovat nový stav x', tedy novou kombinaci hran, které do grafu přidáme. Zvolili jsme přístup, ve kterém budeme počet nahrazených hran postupně snižovat. Nejprve vyměňujeme počáteční počet hran (celou čtveřici, pětici,...). Po dosažení určité teploty počet vyměňovaných hran o jeden snížíme, dokud nedosáhneme nahrazování pouze jedné hrany. Musíme tedy stanovit, ve které teplotě T snížíme počet vyměňovaných hran. Teplotní rozmezí, ve kterém bude počet vyměňovaných hran konstantní, určíme podle následujícího vztahu:

$$rozmezi = \frac{T_0}{pridavano}, \tag{9}$$

kde proměnná *pridavano* udává počet původně přidávaných hran. Demonstrujme si to na příkladu. Předpokládejme, že chceme přidat celkově tři hrany a počáteční teplota $T_0 = 30$. Podle výše uvedeného vztahu spočítáme teplotní rozmezí, které činí 10 stupňů. V tomto rozmezí budeme vyměňovat konstantní počet hran. V teplotním rozmezí 30 – 21 budeme vyměňovat všechny tři hrany. V rozmezí teplot 20 – 11 budeme nahrazovat o hranu méně, tedy dvě hrany. Od teploty T = 10 nahrazujeme pouze jednu hranu. Pokud rozmezí nevyjde celočíselně, ale je to číslo desetinné, zaokrouhlíme jej na nejbližší vyšší celé číslo.

Jelikož je simulované žíhání stochastický algoritmus, musíme jej spustit několikrát, abychom měli zaručenou vyšší pravděpodobnost lepšího výsledku. Algoritmus totiž začne prohledávat jinou část sítě.

4.4. Výsledky

Všechny uvedené algoritmy jsme naprogramovali v programu Matlab a v této podkapitole uvedeme výsledky, kterých jsme dosáhli. Pro lepší orientaci doporučujeme nahlédnout do přílohy 4, kde jsou sítě zobrazeny i s čísly vrcholů.

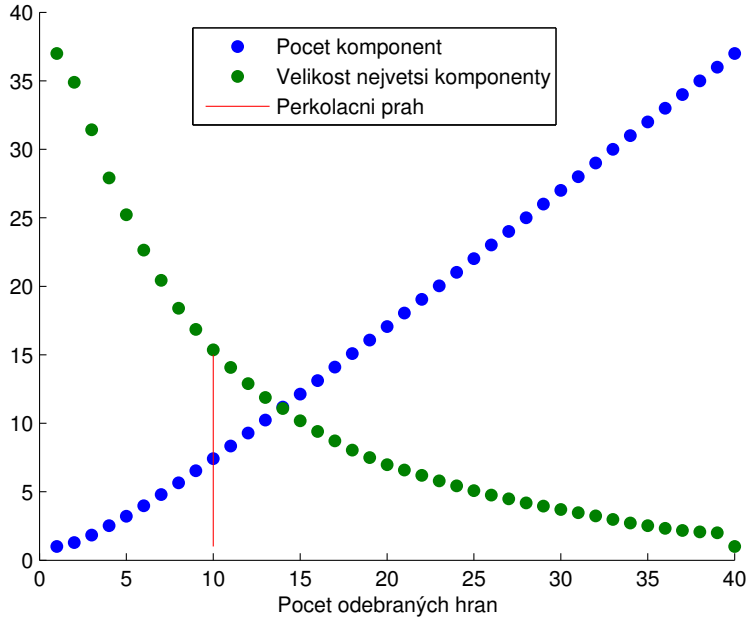
4.4.1. Původní sítě

Předtím než se budeme zabývat optimalizací sítí a navrhovat tedy jejich vylepšení, podíváme se, jak je z hlediska náhodných výpadků odolná stávající podoba silniční sítě.

Síť 1

Ze vstupních dat víme, že tato část sítě obsahuje 37 vrcholů a 40 hran. Při zneprůjezdňování těchto hran budou tedy výsledné matice *N, M* obsahovat 40 sloupců. Počet řádků je dán množstvím opakování algoritmu perkolace. U této sítě se pro odchylku 0.001 napočítalo několik permutací navíc nad zvolených 500 opakování. Matice jsou tedy rozměru okolo 540×40. Maximální hodnota v těchto maticích je vždy rovna počtu vrcholů, v tomto případě tedy 37, a minimální je vždy 1. Matice N obsahuje maximum vždy v posledním sloupci, protože řádek udává počet komponent grafu, a v posledním sloupci máme situaci, kdy jsme už zneprůjezdnili všechny hrany a graf se tedy sestává pouze z izolovaných vrcholů, kterých je samozřejmě 37. Naopak minimum má tato matice v prvním sloupci, kdy začínáme s jednou komponentou. Na druhou stranu matice M udává velikost největší komponenty, a tedy její řádky mají klesající tendenci od 37 (souvislý celý graf) až po 1 (největší komponentou je izolovaný vrchol).

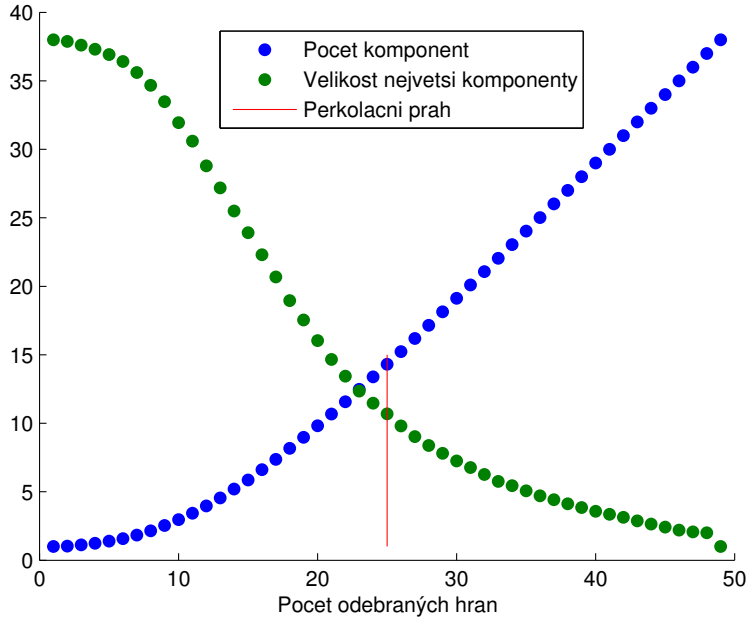
Průměrné hodnoty sloupců jsou vykresleny na obrázku 17. Na základě tohoto výsledku můžeme pro tuto síť označit za perkolační práh 10 odebraných hran, což je čtvrtina z celkového počtu hran. Příčinu tak nízkého čísla můžeme hledat také v tom, že síť 1 je podle obrázku 15 docela řídká.



Obrázek 17: Průměrné hodnoty pro síť 1

Síť 2

Jak jsme zmínili v kapitole o vstupních datech, druhá síť je rozměrově velmi podobná té první. Podstatně se ale liší stupni vrcholů, které jsou v ní obsaženy. Zatímco síť 1 byla řídká a vrcholy v ní měly většinou stupeň dva, síť 2 je navzájem více propojená a tedy lze očekávat, že rozpad na komponenty nebude tak prudký. Toto očekávání nám potvrdil výsledek po provedení dostatečného množství opakování algoritmu perkolace. Průměrný průběh rozpadu je znázorněn na obrázku 18. Z něj vidíme, že perkolační práh pro tuto síť je mnohem dál. Lineární závislost počtu komponent na odebraných hranách je patrná až u 25. hrany, což je po odebrání poloviny hran.



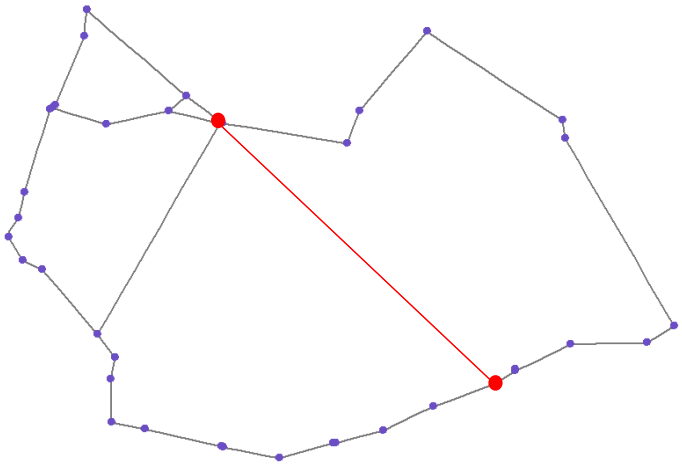
Obrázek 18: Průměrné hodnoty pro síť 2

4.4.2. Přidání jedné hrany

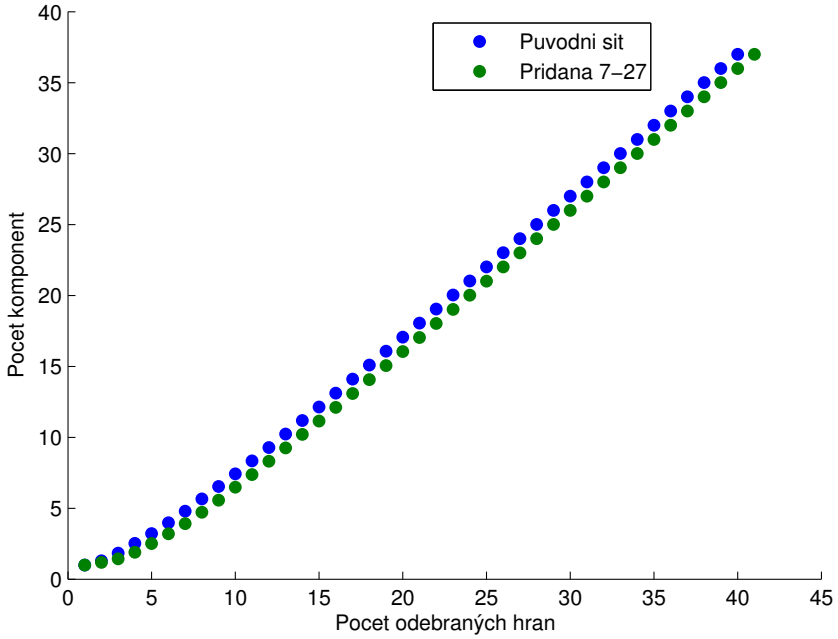
V předešlém textu jsme uvedli, jak do sítě obecně přidávat jednu hranu. Nyní se podíváme, jak tato varianta optimalizace vypadá na sítích 1 a 2. Protože se jedná o relativně menší sítě a výpočetní čas perkolace není dlouhý, prověřili jsme postupně všechny možné kandidáty na přidání, takže jsme zvolili deterministický přístup.

Síť 1

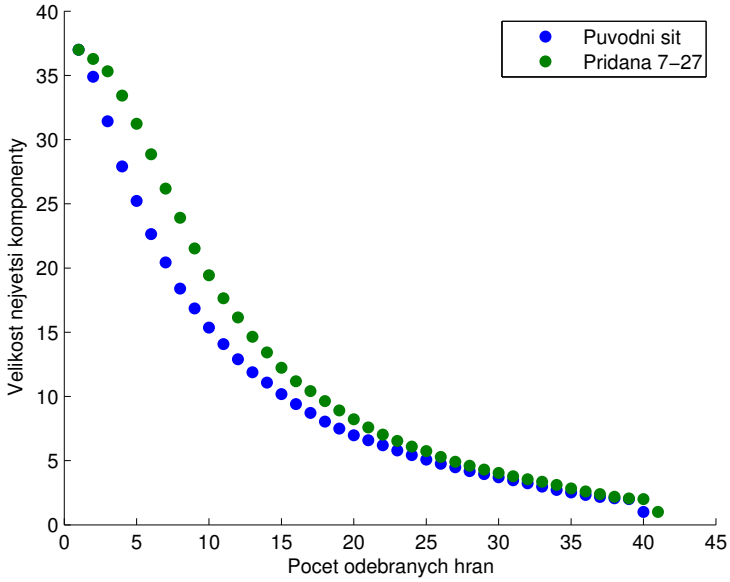
Jak jsme už vícekrát zmínili, je tato síť dosti řídká, a proto je možných kandidátů na přidání dost, protože nové hrany málokdy překříží stávající. Konkrétně máme k dispozici 267 nových hran. Podle kritéria, které jsme stanovili v kapitole optimalizace, jsme rozhodli, že do této sítě je nejlepší přidat hranu [7 27] (viz obr. 19). Výsledky této optimalizace jsou znázorněny na obrázku 20 a 21. Že je toto vylepšení nejlepší, dokazují obrázky 22 a 23, kde jsme původní a nejlepší stav porovnali s jinou, libovolně zvolenou, přidanou hranou.



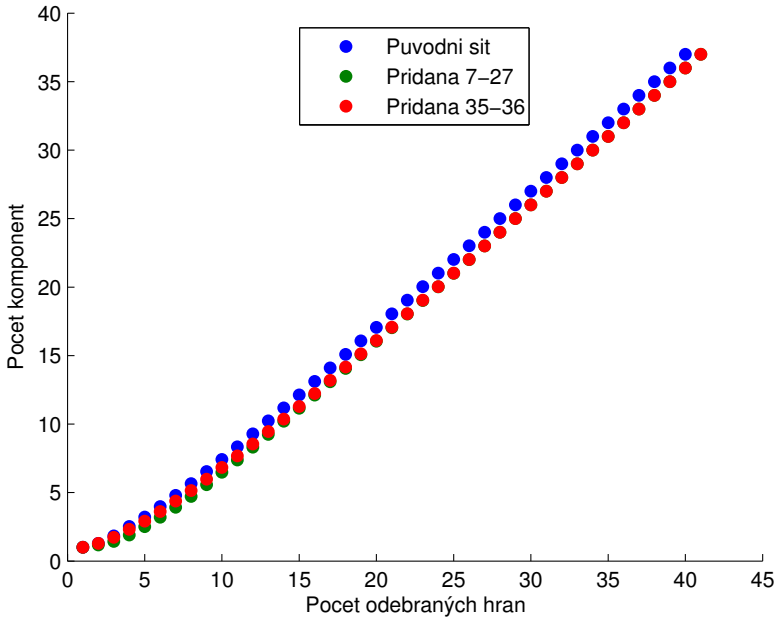
Obrázek 19: Optimalizace sítě 1 pomocí jedné hrany



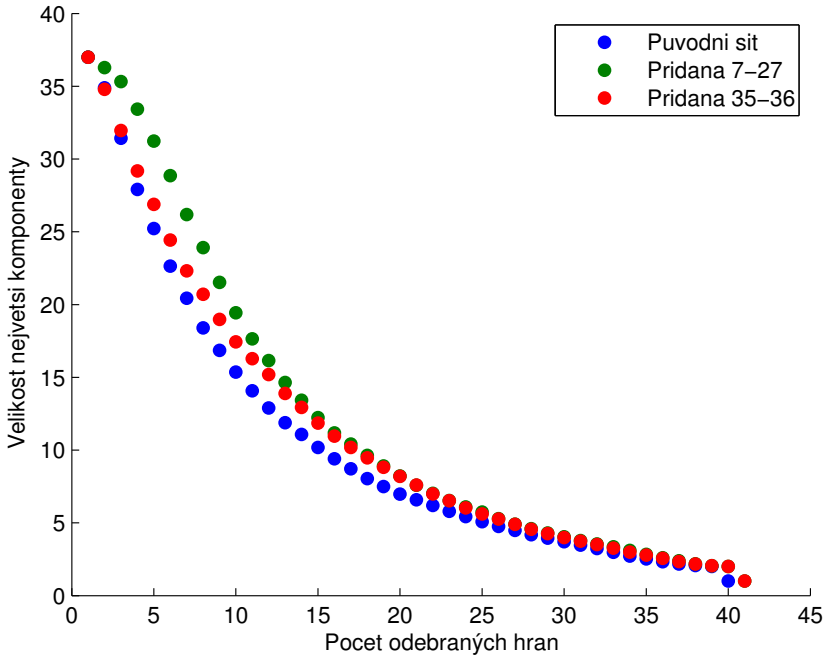
Obrázek 20: Porovnání průměrných hodnot počtu komponent pro síť 1



Obrázek 21: Porovnání průměrných hodnot velikosti největší komponenty sítě 1



Obrázek 22: Porovnání průměrných hodnot počtu komponent pro síť 1



Obrázek 23: Porovnání průměrných hodnot velikosti největší komponenty sítě 1

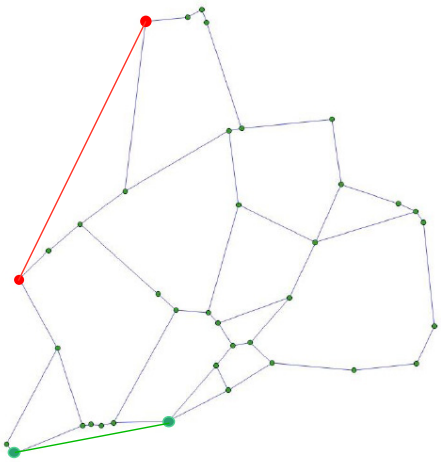
Jak jsme již dříve uvedli, i při projití všech přípustných hran je část algoritmu náhodná. Proto jsme výsledek ověřili opětovným spuštěním algoritmu optimalizace. Porovnání výsledků obou průběhů je v tabulce 1.

	Průběh 1			Průběh 2	
Pořadí	Čísla uzlů	Hodnota	Pořadí	Čísla uzlů	Hodnota
1	7 27	-36.8448	1	7 27	-36.8727
2	9 37	-36.6274	2	5 26	-36.8470
3	5 18	-36.6119	3	10 32	-36.6912
4	3 20	-36.5151	4	22 26	-36.6400
5	10 32	-36.5000	5	8 30	-36.5560
6	7 30	-36.4913	6	6 20	-36.5469
7	11 37	-36.4494	7	5 18	-36.5385
8	23 31	-36.4410	8	11 37	-36.5357
9	9 28	-36.4284	9	3 20	-36.5341
10	7 18	-36.3887	10	23 26	-36.5308

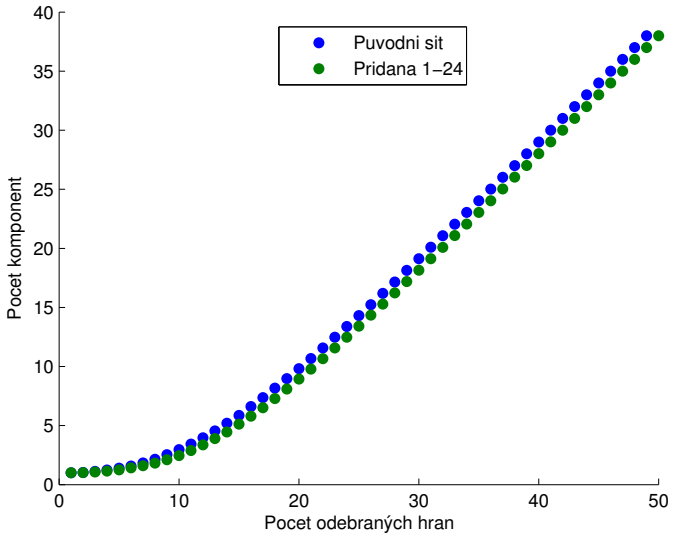
Tabulka 1: Porovnání dvou průběhů optimalizace jednou hranou

Síť 2

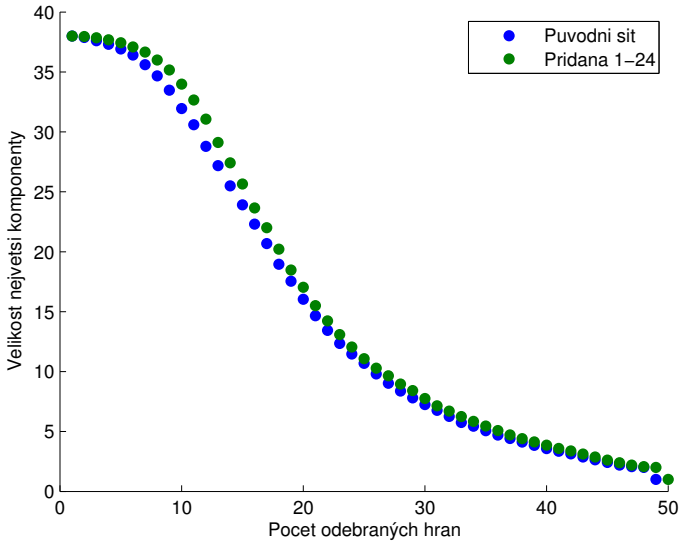
Pro druhou část sítě jsme postupovali totožně. Jelikož je tato síť propojenější, možných kandidátů na přidání je méně, konkrétně 136. Nejlepší optimalizací se ukázala být hrana [1 24] (viz obr. 24 červeně). Vykreslení výsledku je na obrázcích 25 a 26.



Obrázek 24: Optimalizace sítě 2 pomocí jedné hrany



Obrázek 25: Porovnání průměrných hodnot počtu komponent pro síť 2



Obrázek 26: Porovnání průměrných hodnot velikosti největší komponenty sítě 2

Že je tato hrana opravdu optimální, jsme opět ověřili druhým spuštěním algoritmu. Porovnání těchto dvou průběhů je v tabulce 2, ze které je také vidět, že za úvahu stojí i hrana [35 38] (viz obr. 24 zeleně).

Průběh 1			Průběh 2		
Pořadí	Čísla uzlů	Hodnota	Pořadí	Čísla uzlů	Hodnota
1	1 24	-38.3224	1	35 38	-37.3814
2	1 37	-37.8282	2	1 24	-37.3292
3	3 36	-37.8232	3	4 23	-37.1027
4	35 38	-37.6993	4	20 34	-36.8527
5	20 37	-37.5351	5	4 21	-36.6949
6	20 34	-37.5110	6	3 36	-36.6714
7	3 23	-37.2393	7	4 20	-36.6652
8	4 23	-37.1904	8	21 34	-36.5769
9	4 21	-37.1279	9	3 37	-36.3642
10	1 4	-36.8481	10	4 36	-36.3432

Tabulka 2: Porovnání dvou průběhů optimalizace jednou hranou

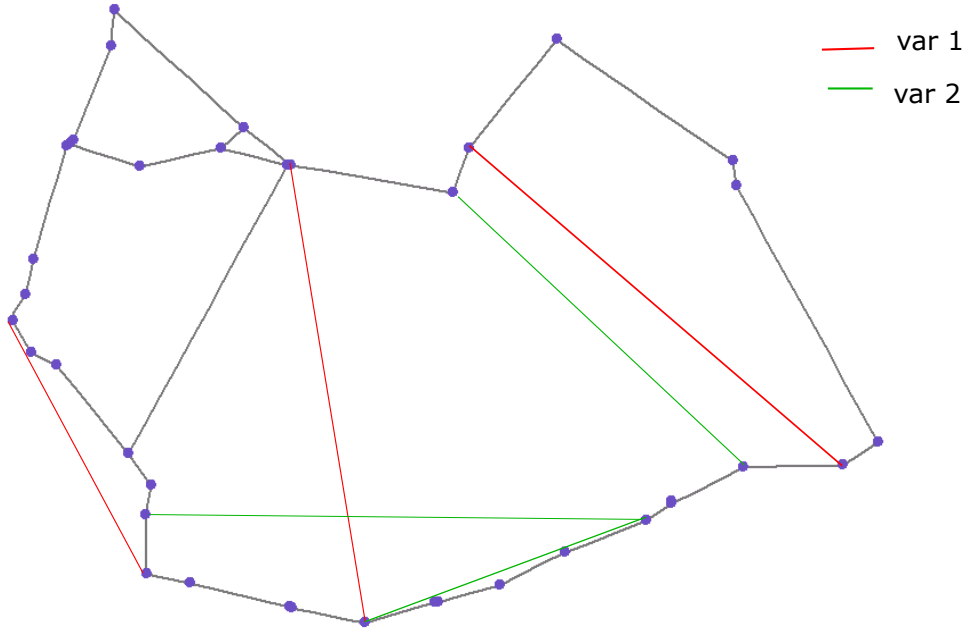
4.4.3. Přidání tří hran - náhodně

U optimalizace vyšším počtem hran jsme u námi vybraných sítí zvolili přidávání trojic. Algoritmus tedy vybere přípustnou trojici hran, které se nekříží mezi sebou a ani žádnou již existující hranu. Nezjišťujeme kolik takových trojic celkem existuje, ale náhodnou trojici stanovíme 200krát. Z této dvoustovky algoritmus vybere nejlepší s ohledem na hodnotu účelové funkce (vztah (6)). Protože to zdaleka nejsou všechny možnosti, může být při spuštění další dvoustovky optimální trojice jiná. Algoritmus například nalezne trojici, kterou v jiném spuštění nevygeneroval, a která je zrovna ještě lepší. Vždy ale detekuje řídkou část sítě a do ní hrany přidává. Proto jsme v této práci program pro dvě stě trojic spustili dvakrát a kromě nejlepší hrany si pamatujeme i další pořadí ostatních trojic. Porovnání těchto dvou průběhů už není tak jednoznačné jako u optimalizace jednou hranou. Je totiž málo pravděpodobné, že by algoritmus vygeneroval naprosto stejnou trojici hran v obou pokusech a my tak mohli pořadí k této trojici přesně porovnat. Pokud se ale blíže podíváme na výsledky a strukturu dané sítě, zjistíme, že i když se čísla hran liší, jsou některé kombinace velmi blízké.

Protože neznáme hlubší souvislosti reálné výstavby námi stanovených nejlepších trojic, které mohou bránit různé okolnosti, navrhujeme pro každou část sítě dvě lišící se varianty.

Síť 1

Pro tuto část sítě vybíráme dvě různé varianty vylepšení znázorněné na obrázku 27. Varianta 1 je kombinace trojice hran [4 37 / 11 29 / 16 20]. Ta vyšla nejlepší při druhém průběhu. V prvním očekávaně tato kombinace není, ale je v něm na prvním místě jí blízká kombinace [4 27 / 8 29 / 16 20]. Druhá varianta vylepšení je kombinace [5 28 / 7 11 / 7 18], která je v druhém průběhu čtvrtá nejlepší a v prvním je obdobná kombinace desátá. Kompletní přehled nejlepší dvacítky hran obou pokusů je uveden v příloze 2, její část je v tabulce 3.



Obrázek 27: Optimalizace sítě 1 pomocí tří hran

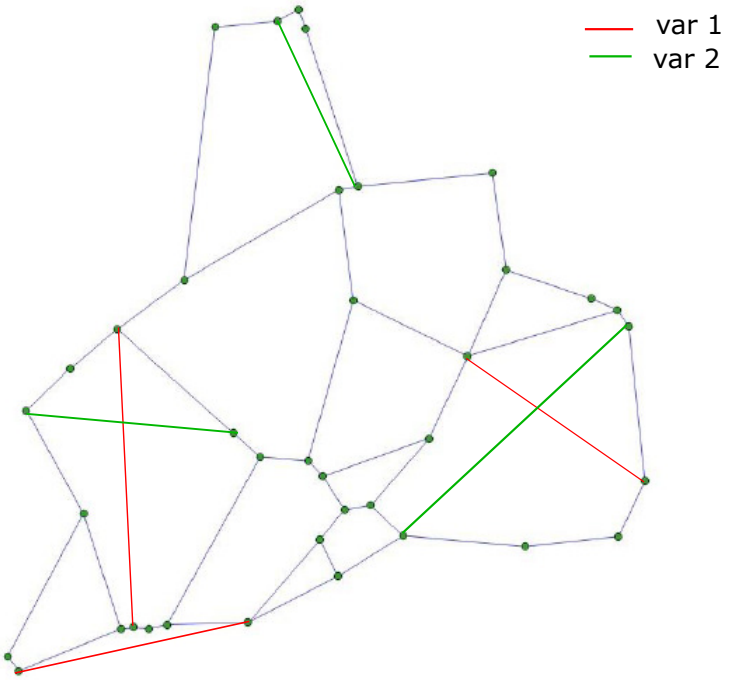
	Průběh 1		Průběh 2	
P.	Čísla uzlů	Hodnota	Čísla uzlů	Hodnota
1	4 27/ 8 29/ 16 20	-102.5738	4 37/ 11 29/ 16 20	-101.4504
2	22 25/ 10 23/ 17 20	-102.0518	23 31/ 8 18/ 22 23	-100.9054
3	5 31/ 21 30/ 11 31	-101.5389	4 28/ 9 29/ 11 18	-100.8780
4	11 30/ 6 11/ 5 27	-101.2275	5 28/ 7 11/ 7 18	-100.8412
5	7 29/ 5 37/ 6 25	-101.0008	7 32/ 20 28/ 7 20	-100.7071
6	26 29/ 11 23/ 26 32	-100.9736	22 31/ 6 18/ 17 35	-100.6671
7	5 29/ 6 31/ 17 18	-100.8535	7 25/ 25 29/ 2 30	-100.4771
8	3 18/ 18 28/ 6 24	-100.8331	22 23/ 18 24/ 7 30	-100.3857
9	21 31/ 7 31/ 19 32	-100.7352	23 28/ 6 26/ 18 36	-100.2545
10	7 20/ 7 28/ 9 20	-100.6214	4 29/ 19 37/ 10 31	-100.1859

Tabulka 3: Deset nejlepších trojic v obou pokusem pro síť 1

Síť 2

Pro druhou síť jsou navrhované optimalizace zakresleny na obrázku 28. Varianta 1, tj. kombinace [2 23 / 16 32 / 35 38] byla nejlepší v prvním průběhu a jí obdobná kombinace se ukázala jako nejlepší i při průběhu druhém. Varianta 2, reprezentována kombinací [8 12 / 1 4 / 18 30], byla při prvním průběhu pátá

a při druhém sedmá. Kompletní přehled nejlepší dvacítky hran obou pokusů je uveden v příloze 3, její část je v tabulce 4.



Obrázek 28: Optimalizace sítě 2 pomocí tří hran

	Průběh 1		Průběh 2	
P.	Čísla uzlů	Hodnota	Čísla uzlů	Hodnota
1	2 23/ 16 32/ 35 38	-104.9418	4 20/ 28 32/ 35 36	-104.0100
2	3 24/ 35 38/ 2 15	-103.8950	4 20/ 18 31/ 7 18	-103.7833
3	2 15/ 35 38/ 7 13	-103.7373	3 24/ 2 37/ 19 33	-103.7485
4	21 34/ 35 36/ 5 12	-103.3855	18 28/ 13 24/ 3 36	-103.6812
5	8 12/ 1 4/ 18 30	-103.1092	19 33/ 1 24/ 23 37	-103.4047
6	3 23/ 12 14/ 16 33	-103.0615	2 21/ 36 38/ 18 30	-103.2897
7	20 34/ 16 31/ 11 33	-103.0382	16 31/ 3 23/ 8 24	-103.1405
8	1 37/ 7 10/ 33 38	-102.9954	17 31/ 5 12/ 21 34	-103.1317
9	13 25/ 20 37/ 31 32	-102.7295	13 25/ 4 36/ 35 36	-102.9724
10	4 23/ 11 33/ 2 15	-102.6406	1 24/ 34 36/ 18 28	-102.8286

Tabulka 4: Deset nejlepších trojic v obou pokusech pro síť 2

4.4.4. Přidání tří hran - simulované žíhání

Obecný princip algoritmu simulovaného žíhání s výměnou klesajícího počtu hran jsme uvedli v kapitole 4.3.2. Nyní uvedeme jaké parametry jsme volili a jaké jsme získali výsledky. Nejprve k problematice volby počáteční teploty. Jak jsme uvedli dříve, měla by být zvolena tak, aby na začátku algoritmus přijímal horší stavy s 50% pravděpodobností. Z předchozí varianty optimalizace jsme vypozorovali, že u náhodně generovaných stavů se hodnota účelové funkce liší asi o 2, což tedy považujeme za počáteční δ. Ze vztahu pro mez algoritmu $mez = e^{\frac{-\delta}{T}}$ zjistíme, že bychom museli začínat na 3 stupních a poté krokovat po desetinách. Čistě z technických důvodů proto δ vynásobíme vždy deseti a počáteční teplotu zvolíme $T_0 = 30$, kterou snižujeme o jeden stupeň.

V každém průběhu tedy algoritmus porovná 30 různých trojic. A to tak, že prvních deset generuje náhodně, u dalších deseti už mění v přijatém stavu jen dvě hrany a v posledních deseti krocích vyměňuje už jen jednu hranu. Po skončení algoritmu si pamatujeme z této třicítky tu nejlepší možnou.

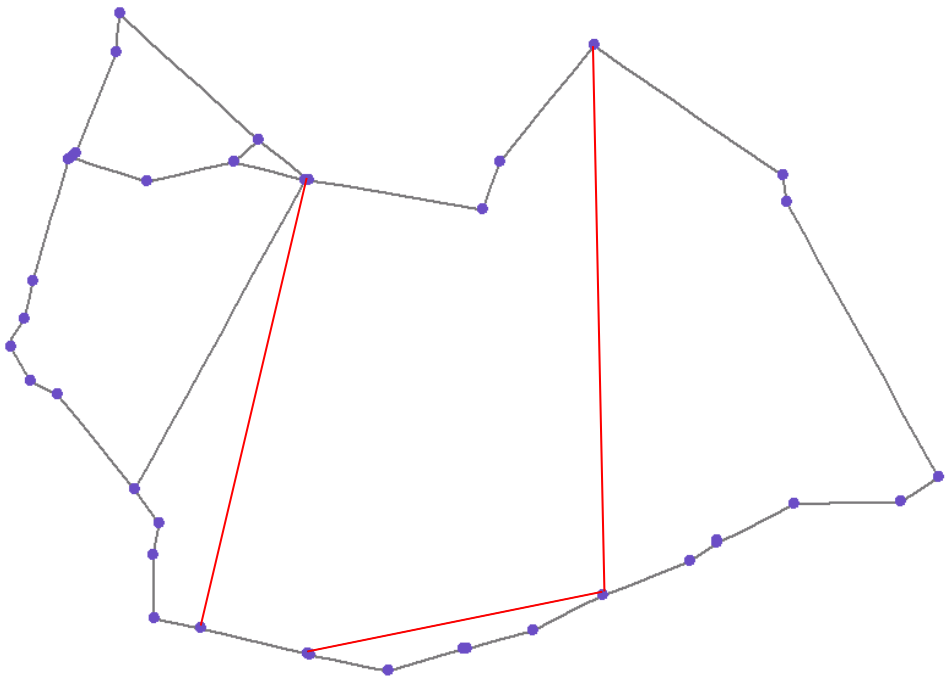
Jelikož s každým spuštěním začne algoritmus v jiné části sítě, pustili jsme jej desetkrát. Abychom těchto deset výsledků mohli navzájem porovnat, zjišťujeme jejich vylepšení oproti stále stejnému rozpadu původní sítě. Jedině tak má smysl hodnoty účelové funkce seřadit vzestupně.

Síť 1

Výsledky deseti opakování algoritmu simulovaného žíhání jsou uvedeny v tabulce 5 a nejlepší varianta zakreslena na obrázku 29.

Pořadí	Čísla uzlů	Hodnota porovnání
1	6 32/ 6 25/ 26 27	-101.8892
2	6 27/ 4 8/ 6 20	-101.6485
3	7 21/ 21 31/ 10 20	-101.3342
4	7 22/ 11 18/ 8 31	-101.0472
5	9 18/ 9 22/ 9 29	-101.0429
6	10 23/ 23 28/ 10 31	-100.8424
7	6 20/ 16 29/ 22 31	-100.5594
8	35 36/ 5 11/ 24 27	-100.5322
9	15 34/ 6 31/23 29	-100.3129
10	6 28/ 8 26/ 8 18	-99.8753

Tabulka 5: Výsledky simulovaného žíhání pro síť 1



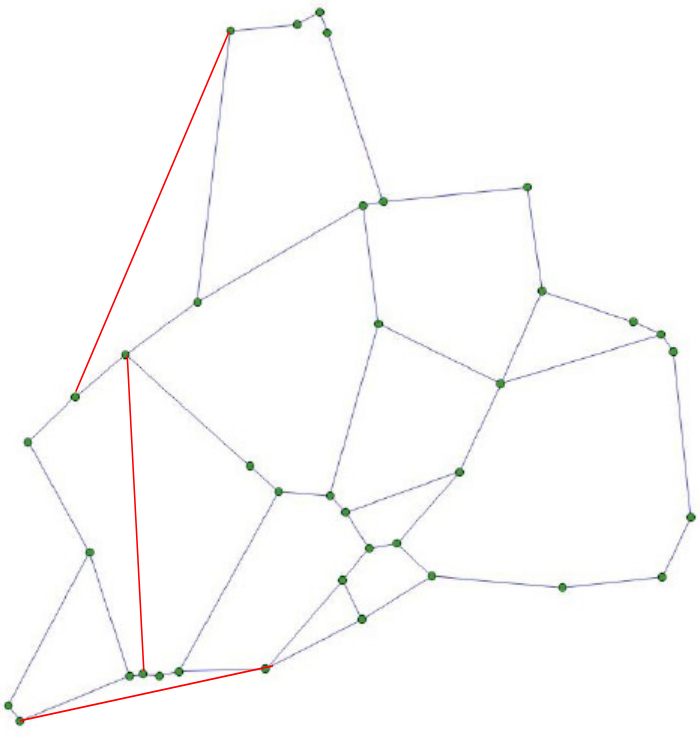
Obrázek 29: Optimalizace sítě 1 pomocí simulovaného žíhání

Síť 2

Výsledky deseti opakování algoritmu simulovaného žíhání jsou uvedeny v tabulce 6 a nejlepší varianta zakreslena na obrázku 30.

Pořadí	Čísla uzlů	Hodnota porovnání
1	2 23/ 35 38/ 3 24	-108.1449
2	35 38/ 3 23/ 7 18	-107.6954
3	1 24/ 17 31/ 3 36	-107.4641
4	1 24/ 19 32/ 2 23	-107.4188
5	1 37/ 7 13/ 7 18	-106.8926
6	35 38/ 31 38/ 7 13	-106.7411
7	35 38/ 1 24/ 20 34	-106.1941
8	3 23/ 30 32/ 8 12	-104.7944
9	25 34/ 2 20/13 25	-103.4808
10	18 31/ 20 23/ 11 33	-102.6066

Tabulka 6: Výsledky simulovaného žíhání pro síť 2



Obrázek 30: Optimalizace sítě 2 pomocí simulovaného žíhání

Závěr

V této diplomové práci jsme se zabývali analýzou odolnosti silniční sítě vůči náhodným výpadkům. Jako nástroj analýzy jsme zvolili proces perkolace. Na základě zjištěných informací o původních sítích jsme uvedli tři možné způsoby optimalizace. Všechny modifikace jsme uplatnili na reálných datech dvou sítí a jednalo se o části Zlínského kraje. Veškerá data společně s obrázky sítí byla poskytnuta Centrem dopravního výzkumu, v. v. i. v rámci projektu TRISK č. VG20102015057.

Pomocí perkolace jsme zjišťovali tendenci rozpadu sítě na komponenty. Jelikož jsme se zabývali odolností sítě vůči náhodným zneprůjezdněním, které nelze predikovat, zneprůjezdňovali jsme hrany v náhodném pořadí. Tedy bez ohledu na jejich vytíženost, délku či jiné kritérium. Určujícím měřítkem této analýzy je počet komponent a také jejich velikost po výpadku určitého množství hran. Ze všech možných pořadí, ve kterých můžeme hrany vypouštět, jich provedeme alespoň 500, abychom zjistili průměrné hodnoty a měli tak vypovídající údaje pro danou síť.

Zřejmou optimalizací je přidání hrany nebo případně skupiny hran. V práci jsme uvedli vylepšení nejprve pomocí jedné hrany, kde jsme prošli všechny přípustné možnosti, a dále potom pomocí trojice hran, kde jsme přistoupili k náhodnému generování přípustných hran. U přidávání trojic jsme postupovali dvěma způsoby, a to nejprve náhodným generováním celé trojice a poté pomocí simulovaného žíhání. I když na námi analyzovaných menších sítích jsou hodnoty účelové funkce obou metod srovnatelné, metoda simulovaného žíhání se oproti náhodné trojici ukázala výpočtově mnohem rychlejší. Také předpokládáme, že na větších sítích by byl znatelnější rozdíl i v hodnotách účelové funkce, a tedy by metoda simulovaného žíhání byla i přesnější. Už u menších sítí jsme se přesvědčili, že program simulovaného žíhání našel kombinace s nižší hodnotou účelové funkce než program náhodného prohledávání.

Jelikož jsme používali stochastické algoritmy a pro smysluplné výsledky jsme museli provést opakované výpočty, projití dostatečného množství přípustných

hran a vybrání té nejlepší kombinace si vyžádalo výpočetní čas v řádu hodin. Z toho důvodu jsme nepřistoupili k analýze větších sítí, ke které by bylo zapotřebí lepší výpočetní techniky.

Přínosem této diplomové práce je vytvoření algoritmů pro zkoumání odolnosti sítě, které lze použít jak na rozsáhlejší silniční sítě, tak na sítě zcela odlišné.

Příloha 1: Seznam a popis přiložených programů

- NACTIDATA4, NACTISOURADNICE, TESTPOLE - programy pro načtení dat o sítích z textových souborů. Poskytnuty vedoucím práce.
- kandidati_na_pridani - funkce, která pro zadaný graf zjistí matici všech možných hran, které lze do něj přidat.
- moznost_pridani_hrany - funkce pro ověření, zda-li zadanou hranu lze přidat či nikoli.
- nahrada_hran - funkce, která ze zadané trojice hran nahradí určený počet a zbytek fixuje. Funkce je součástí programu simulovaného žíhání.
- networkComponents_vzdal - funkce, která pro zadaný ohodnocený graf zjistí počet komponent, jejich velikost a vrcholy, které do nich patří.
- pridani_hrany - funkce pro přidání jedné hrany do grafu a přepsání grafu do nové podoby.
- pridani_tri_hran - obdoba přidaní jedné hrany. Funkce najde možnou kombinaci tří hran a tu přidá.
- PRIDANI_JEDNE_HRANY_POROVNANI - program pro optimalizaci jednou hranou při projití matice kandidátů.
- PRIDANI_TRI_HRAN_POROVNANI - program pro optimalizaci trojicí hran - náhodně.
- PRIDANI_TRI_HRAN_SIMULOVANE_ZIHANI - program pro optimalizaci trojicí hran - pomocí simulovaného žíhání.
- vypocet_jistici_odchylka - výpočet odolnosti sítě pomocí perkolace. Funkce zahrnuje jistící odchylku 0.001.
- zneprujezdneni_hrany - nastavení délky požadované hrany na nekonečno.

Příloha 2: Kompletní výsledky pro síť 1

	Průběh 1		Průběh 2	
P.	Čísla uzlů	Hodnota	Čísla uzlů	Hodnota
1	4 27/ 8 29/ 16 20	-102.5738	4 37/ 11 29/ 16 20	-101.4504
2	22 25/ 10 23/ 17 20	-102.0518	23 31/ 8 18/ 22 23	-100.9054
3	5 31/ 21 30/ 11 31	-101.5389	4 28/ 9 29/ 11 18	-100.8780
4	11 30/ 6 11/ 5 27	-101.2275	5 28/ 7 11/ 7 18	-100.8412
5	7 29/ 5 37/ 6 25	-101.0008	7 32/ 20 28/ 7 20	-100.7071
6	26 29/ 11 23/ 26 32	-100.9736	22 31/ 6 18/ 17 35	-100.6671
7	5 29/ 6 31/ 17 18	-100.8535	7 25/ 25 29/ 2 30	-100.4771
8	3 18/ 18 28/ 6 24	-100.8331	22 23/ 18 24/ 7 30	-100.3857
9	21 31/ 7 31/ 19 32	-100.7352	23 28/ 6 26/ 18 36	-100.2545
10	7 20/ 7 28/ 9 20	-100.6214	4 29/ 19 37/ 10 31	-100.1859
11	15 35/ 6 37/ 26 27	-100.5709	21 23/ 17 20/ 6 25	-100.1415
12	22 30/ 10 22/ 11 18	-100.5249	7 21/ 20 27/ 7 28	-100.1274
13	1 17/ 9 22/ 11 28	-100.4629	23 37/ 14 17/ 11 28	-100.1017
14	6 11/ 17 20/ 4 37	-100.3863	5 24/ 19 37/ 25 29	-99.9255
15	21 31/ 4 7/ 6 20	-100.3280	16 20/ 4 6/ 5 32	-99.8943
16	17 34/ 10 31/ 7 37	-100.1890	6 31/22 30/ 10 25	-99.7641
17	5 30/ 4 29/ 23 24	-100.1757	20 24/ 7 27/ 7 26	-99.5126
18	11 27/ 5 10/ 4 8	-100.0548	10 18/ 5 30/ 6 10	-99.4628
19	18 22/ 17 31/ 10 21	-100.0533	10 20/ 6 30/ 15 34	-99.4448
20	6 30/ 4 37/ 16 34	-100.0256	5 25/ 5 31/ 11 23	-99.3036

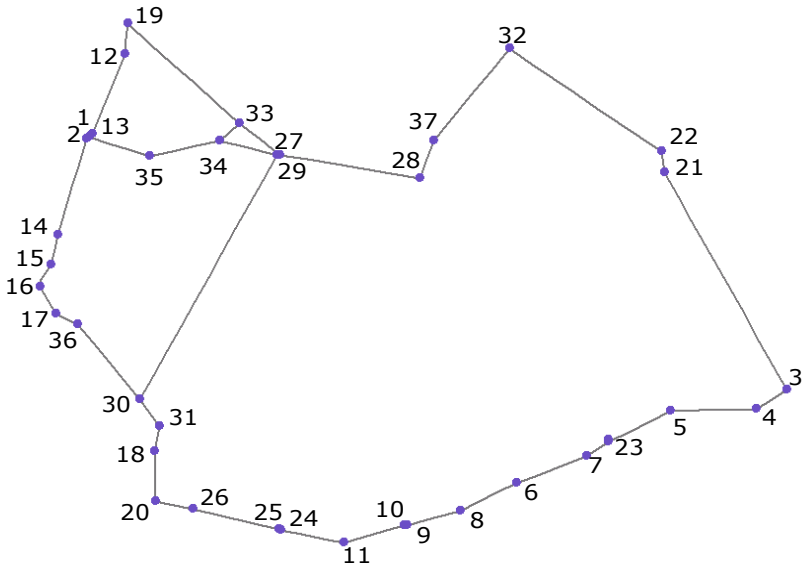
Tabulka 7: 20 nejlepších hran a jim odpovídající hodnota účelové funkce

Příloha 3: Kompletní výsledky pro síť 2

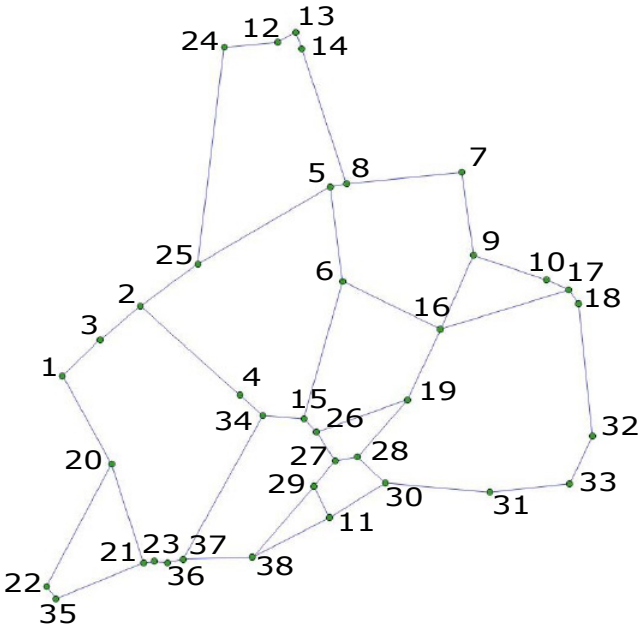
	Průběh 1		Průběh 2	
P.	Čísla uzlů	Hodnota	Čísla uzlů	Hodnota
1	2 23/ 16 32/ 35 38	-104.9418	4 20/ 28 32/ 35 36	-104.0100
2	3 24/ 35 38/ 2 15	-103.8950	4 20/ 18 31/ 7 18	-103.7833
3	2 15/ 35 38/ 7 13	-103.7373	3 24/ 2 37/ 19 33	-103.7485
4	21 34/ 35 36/ 5 12	-103.3855	18 28/ 13 24/ 3 36	-103.6812
5	8 12/ 1 4/ 18 30	-103.1092	19 33/ 1 24/ 23 37	-103.4047
6	3 23/ 12 14/ 16 33	-103.0615	2 21/ 36 38/ 18 30	-103.2897
7	20 34/ 16 31/ 11 33	-103.0382	16 31/ 3 23/ 8 24	-103.1405
8	1 37/ 7 10/ 33 38	-102.9954	17 31/ 5 12/ 21 34	-103.1317
9	13 25/ 20 37/ 31 32	-102.7295	13 25/ 4 36/ 35 36	-102.9724
10	4 23/ 11 33/ 2 15	-102.6406	1 24/ 34 36/ 18 28	-102.8286
11	16 30/ 4 21/ 33 38	-102.4088	2 36/ 1 24/ 8 9	-102.7914
12	8 12/ 1 36/ 29 34	-101.8942	4 6/ 33 38/ 3 24	-102.6982
13	3 24/ 28 31/ 7 13	101.7716	4 23/ 8 24/ 17 28	-102.2510
14	7 10/ 1 34/ 5 12	-101.7333	4 21/ 7 13/ 4 15	-102.1797
15	2 23/ 18 30/ 4 37	-101.6892	14 25/ 28 31/ 2 20	-102.0778
16	19 33/ 5 12/ 34 36	-101.6542	6 7/ 17 33/ 3 23	-101.9793
17	16 32/ 19 31/ 1 37	-101.5805	13 25/ 3 36/ 36 38	-101.8661
18	1 37/ 4 6/ 23 37	-101.5064	5 13/ 35 38/ 17 19	-101.7640
19	20 23/ 11 33/ 3 36	-101.4361	30 32/ 1 37/ 5 9	-101.7565
20	17 33/ 28 33/ 35 36	-101.4072	3 23/ 3 36/ 18 28	-101.7284

Tabulka 8: 20 nejlepších hran a jim odpovídající hodnota účelové funkce

Příloha 4: Znázornění sítí 1 a 2



Obrázek 31: Grafické znázornění sítě 1 společně s čísly vrcholů



Obrázek 32: Grafické znázornění sítě 2 společně s čísly vrcholů

Literatura

[1] Barrat, A., Barthélemy, M., Vespignani, A., Dynamical Processes on Complex Networks, Cambridge: Cambridge University Press, 2012

[2] Daqing Li, Bowen Fu, Yunpeng Wang, Guangquan Lu, Yehiel Berezin, H. Eugene Stanley, and Shlomo Havlin, Percolation transition in dynamical traffic network with evolving critical bottlenecks, PNAS 2015 112 (3) 669-672

[3] Doleželová, J., Analýza zranitelnosti silniční sítě. Olomouc, 2015. Diplomová práce. Univerzita Palackého v Olomouci, Fakulta přírodovědecká, Katedra matematické analýzy a aplikací matematiky

[4] Durrett, R., Random Graph Dynamics, 1. vydání, Cambridge: Cambridge University Press, 2010

[5] Grimmett, G., Probability on Graphs, 1. vydání, Cambridge: Cambridge University Press, 2010

[6] Hliněný, P., Základy teorie grafů [online], dostupné z: http://is.muni.cz/do/1499/el/estud/fi/js10/grafy/Grafy-text10.pdf,

[7] Marešová, P., Zranitelnost silniční sítě. Olomouc, 2013. Bakalářská práce. Univerzita Palackého v Olomouci, Fakulta přírodovědecká, Katedra matematické analýzy a aplikací matematiky

[8] Newman, M., Networks: An Introduction, New York: Oxford University Press, 2010

[9] Töpfer, P., Algoritmy a programovací techniky, 1. vydání, Praha: PROMETHEUS, 1995