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

BAKALÁŘSKÁ PRÁCE

Zranitelnost silniční sítě

Vedoucí bakalářské práce:
RNDr. Rostislav Vodák, Ph.D.
Rok odevzdání: 2013

Vypracovala:
Pavla Marešová
MATAP, III. ročník

Prohlášení

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

V Olomouci dne 16. dubna 2013

Poděkování

Na tomto místě bych chtěla poděkovat především svému vedoucímu bakalářské práce za dostatek trpělivosti, poskytnutí studijních materiálů a za pomoc a rady, které vedly k dokončení této práce. Dále bych chtěla poděkovat všem, kteří mi jakýmkoli způsobem pomohli při vzniku této práce.

Obsah

Úvod

Dnes jezdí autem skoro každý a silniční síť je stále vytíženější. V dnešní uspěchané době nemá také nikdo času nazbyt a neplánované zdržení na cestách může mít neblahé důsledky. Proto se zabýváme tím, jak síť silnic vylepšit, aby následky dopravní nehody, přírodní katastrofy či zneprůjezdnění silnice v důsledku prací na silnici byly co nejmenší. Cílem této práce je aplikovat nástroje z teorie grafů pro analýzu zranitelnosti silniční sítě. Toto téma jsem si vybrala také proto, že sama vlastním řidičský průkaz už několik let.

Cílem této práce nazvané *Zranitelnost silniční sítě* je uvést metodu pro výpočet důležitosti úseku na síti a provést následnou optimalizaci sítě. Abychom mohli s jakoukoli sítí vůbec pracovat, potřebujeme ji zjednodušit na nějaký model, a tím jsou grafy. Z tohoto důvodu je první kapitola věnována teorii grafů. Uvedeme v ní základní pojmy a vlastnosti potřebné pro práci se sítěmi. Pro realizaci přeprav na síti je jednou ze základních úloh nalezení nejkratší cesty mezi dvěma uzly v síti. Různé obdoby tohoto problému nalezneme v mnoha aplikacích. Využijeme ho i pro silniční síť, protože každý, kdo na silnici vyjede, se chce do cíle dostat co možná nejrychleji. Této problematice se proto věnuje druhá kapitola. Ve třetí kapitole, která je stěžejní kapitolou této práce, se dostaneme k otázce zranitelnosti a důležitosti. Uvedeme v ní různé indikátory důležitosti úseku, které v poslední části práce aplikujeme na část reálné silniční sítě České republiky a pomocí nich určíme nejvýhodnější zlepšení této části sítě.

1. Úvod do teorie grafů

Teorie grafů je částí matematiky, jejíž původ se datuje do 18. století. Tehdy řešil Leonhard Euler problém sedmi mostů města Královce. Městem protéká řeka, která vytváří ve městě ostrovy, a vede přes ní sedm mostů. Euler v roce 1736 přemýšlel, zda-li lze přes každý most přejít právě jednou a vrátit se do výchozího místa. Tedy jestli daná cesta jde nakreslit jedním tahem. Dokázal, že v Královci to možné není. Další známý historický problém související s teorií grafů je z roku 1852, kdy Francis Guthrie předložil „problém čtyř barev“. Poté, co obarvil politickou mapu Anglie pouze čtyřmi barvami, si kladl otázku, zda jde libovolná mapa obarvit s využitím maximálně čtyř barev tak, aby každé dvě sousedící země měli jinou barvu. Kladnou odpověď poskytli až o více než sto let později američtí matematici pomocí počítačového modelu.

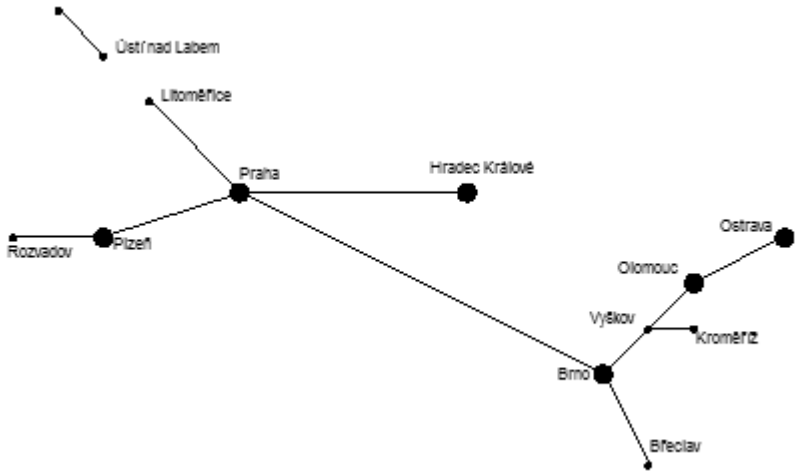
Jelikož grafem lze reprezentovat jakoukoli síť, je pole aplikací grafů hodně široké. Nevyužívají se pouze v informatice, ale najdeme je třeba i v chemii při studiu molekul. Pomocí grafů lze také popsat např. odkazy na webových stránkách, které odkazují z jednoho článku na druhý. Pod pojmem síť si čtenář možná představí něco jiného, a to síť dopravní. Tedy potrubí, železnice, letecké linky nebo silnice. A právě silniční sítí se budeme zabývat v této práci.

Silniční síť lze popsat pomocí grafu, tedy pomocí libovolně složitého systému uzlů a hran, a to následujícím způsobem. Pokud potřebujeme popsat pouze velmi zevrubnou silniční síť České republiky, třeba pouze síť dálnic, která je na obrázku 1, stačí za uzly považovat jen velká města, ke kterým vedou dálnice. Hrany grafu poté představují možné cesty mezi těmito městy. Jak dálniční síť zjednodušit na graf je ukázáno na obrázku 2. Tuto jednoduchou myšlenku lze rozšířit na všechny silniční komunikace. Abychom silniční síť vystihli co možná nejpřesněji a graf se dal prakticky použít, musíme do grafu zanést spoustu informací, jako například délku silnice, její vytíženost nebo třídu. Pokud pracujeme jen s grafem jednoho města, kde uzly představují křižovatky a hrany ulice, je třeba do grafu zahrnout i jednosměrné ulice. Proto v této kapitole zavedeme pojmy orientovaného a ne-

orientovaného grafu, ohodnocení grafu, cesty a vzdálenosti. Dále potom co je to komponenta souvislosti grafu a jak ji nalézt. V poslední části kapitoly je uvedeno, jak lze graf různě reprezentovat v programu.

Definice a věty v této kapitole jsou převzaty z [8].

Obrázek 1: Dálniční síť ČR

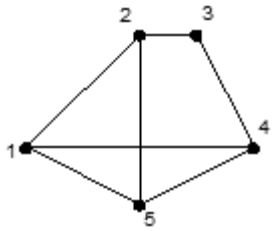


Obrázek 2: Dálniční síť ČR - zjednodušení na graf

1.1. Definice grafu

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 uzlů.*

Graf lze proto zadat obrázkem nebo výčtem vrcholů a hran. Prozatím hovoříme o neorientovaném grafu. Hrana {2,1} je tedy totožná s hranou {1,2}. Příklad zadání grafu je na obrázku 3.



Obrázek 3: V={1,2,3,4,5} E={{1,2},{2,3},{3,4},{4,5},{1,5},{2,5}}

Nyní se podívejme, jak definovat intuitivně chápané procházení grafem.

Definice 1.2. *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*.*

Mezi speciální případy grafů patří *kružnice délky n* (značíme C_n) – má $n \geq 3$ vrcholů, z nichž každé dva jsou různé s výjimkou vrcholů v_0 a v_n, tj. $v_0 = v_n$. Dále *úplný graf* (značíme (K_n)) – má n vrcholů, všechny navzájem pospojované, tedy $\binom{n}{2}$ hran. Dále *cestou délky n* nazýváme graf s n+1 vrcholy spojených za sebou n hranami. Ukázky těchto grafů jsou na obrázku 4.

Obrázek 4: Kružnice C_8, Cesta délky 4, Úplný graf K_5

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

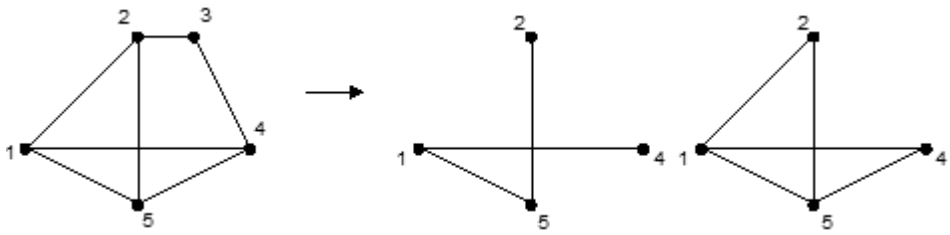
Například pro graf K_5 z obrázku 4, by byl $d_G(v) = 4$ pro kterýkoli vrchol. U kružnice je stupeň všech vrcholů $d_G(v) = 2$.

Věta 1.1. *Součet stupňů vrcholů v grafu je vždy sudý a je roven dvojnásobku počtu hran.*

Důkaz: [8], str. 3.

V aplikacích se můžeme setkat se situací, že máme k dispozici celý graf, ale my přitom chceme použít jenom některou jeho část. Například pokud známe silniční síť České republiky, ale jsme vyzváni k tomu, prošetřit silnice pouze Olomouckého kraje. Proto zavedeme definici podgrafu. Tato definice zaručuje, že nepřevezmeme pouze hrany bez koncových uzlů.

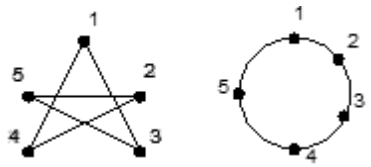
Definice 1.4. *Podgrafem grafu G rozumíme libovolný graf H na podmnožině vrcholů* $V(H)$, *kde* $V(H) \subseteq V(G)$, *který má za hrany libovolnou podmnožinu hran grafu G majících oba vrcholy ve V(H). Indukovaným podgrafem je podgraf, který obsahuje všechny hrany grafu G mezi dvojicemi vrcholů z V(H).*



Obrázek 5: Příklad grafu, jeho podgrafu a indukovaného podgrafu

Dalším důležitým pojmem z teorie grafů je pojem *izomorfismus*. Podle něj rozlišujeme dva navzájem opravdu různé grafy.

Definice 1.5. *Izomorfismus grafů G a H je bijektivní zobrazení $f : V(G) \to V(H)$, pro které platí, že každá dvojice vrcholů $u, v \in V(G)$ je spojená hranou v G právě tehdy, když je dvojice $f(u), f(v)$ spojená hranou v H. Grafy G a H jsou izomorfní, pokud mezi nimi existuje izomorfismus.*



Obrázek 6: Izomorfní grafy kružnice C_5

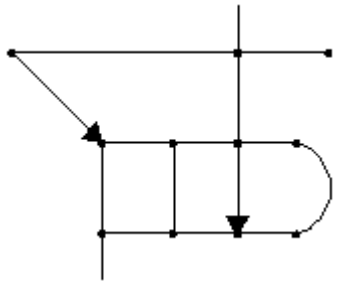
Grafy na obrázku 6 jsou izomorfní. Jedná se pouze o jiné nakreslení kružnice C_5. Zobrazení uzlů je následující $1 \to 1$, $3 \to 2$, $5 \to 3$, $2 \to 4$, $4 \to 5$. Izomorfní grafy musí mít stejný počet vrcholů i hran, jejich vrcholy musejí mít stejné stupně a izomorfismus f musí zobrazovat na sebe vrcholy stejných stupňů. Toto jsou nutná kritéria pro nalezení izomorfismu. Pokud nám existenci izomorfismu nevyloučí, neznamená to, že by existoval, a nezbývá nic jiného, než zkoušet všechny možnosti zobrazení.

Relace „být izomorfní“ rozděluje množinu grafů na třídy izomorfismu. Proto když mluvíme o grafu, mluvíme o celé třídě izomorfismu, tj. nezáleží nám na nakreslení grafu.

Nyní zavedeme pojem orientovaného grafu. Ne vždy lze totiž v síti procházet hranu v obou směrech, a potřebujeme tedy vyjádřit směr hrany. U silniční sítě toto zavedení potřebujeme pro vyjádření, že ulice či cesta je jednosměrná. U potrubní sítě by orientované hrany značily, kde kapalina vtéká a kde vytéká.

Definice 1.6. *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.*

Orientované grafy už nemusí být symetrické, tedy hrana (v, u) nemusí existovat. Mezi orientovanými a neorientovanými grafy existuje analogie. Na oboustrannou hranu lze totiž nahlížet jako na dvě jednostranné s opačnými směry.

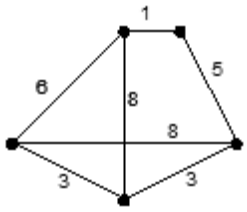


Obrázek 7: Příklad orientovaného grafu

Na obrázku 7 je uveden příklad orientovaného grafu, který by mohl znázorňovat třeba plán malé vesnice se dvěma jednosměrnými ulicemi.

V této práci budeme pracovat i s ohodnocením hran, protože není silnice jako silnice. To slouží k bližší specifikaci hrany a vyjadřuje nějakou její vlastnost. Co ohodnocení představuje musíme vždy specifikovat. Může reprezentovat vzdálenost mezi uzly, kapacitu hrany a mnoho dalšího. Příklad ohodnoceného grafu je na obrázku 8.

Definice 1.7. *Ohodnocený graf je graf G společně s ohodnocením hran w, kde $w : E(G) \to \mathbb{R}$. Pokud pro všechny hrany platí, že $w(e) > 0$, hovoříme o kladně ohodnoceném grafu.*



Obrázek 8: Příklad ohodnoceného grafu

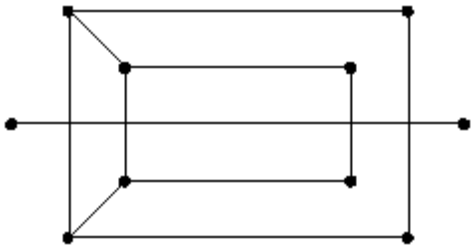
1.2. Souvislost grafu, komponenty

Jestliže graf modeluje nějakou síť, a to nejen silniční, ale třeba počítačovou nebo potrubní, je jednou ze základních otázek, zda máme možnost dostat se z každého uzlu do libovolného jiného uzlu. Právě uvedené sítě by tuto vlastnost mít měly, a proto graf, který je reprezentuje, nazýváme *souvislý*.

Definice 1.8. *Graf G je souvislý, pokud je tvořený nejvýše jednou komponentou souvislosti, tedy pokud jsou každé dva vrcholy spojené cestou.*

Věta 1.2. *Mějme relaci $\sim$ na množině vrcholů V(G) neorientovaného grafu G takovou, že pro dva vrcholy platí $u \sim v$, právě když existuje v G sled začínající v u a končící ve v. Pak $\sim$ je relace ekvivalence. Třídy této ekvivalence se nazývají komponenty souvislosti grafu G.*

Důkaz: [8], str. 14.



Obrázek 9: Komponenty souvislosti

Graf na obrázku 9 je tvořen dvěma komponentami souvislosti.

Protože nás v aplikacích bude zajímat, zda je graf souvislý i po nějakém lokálním výpadku, tj. po odstranění určitého počtu hran nebo vrcholů, uvedeme definice k-souvislých grafů.

Definice 1.9. *Graf G je hranově (vrcholově) k-souvislý, $k > 1$, pokud i po odebrání libovolných nejvýše k-1 hran (vrcholů) z G zůstane výsledný graf souvislý.*

Vysoká hranová souvislost tedy znamená, že síť je velmi odolná vůči výpadkům spojení, tj. existují alternativní cesty, jak se dostat do všech vrcholů. Vrcholová

souvislost je silnějším pojmem, protože síť zůstane dosažitelná i po odstranění určitého počtu vrcholů (samozřejmě mimo těch odstraněných).

1.3. Reprezentace grafu v programu

Mějme graf G s n vrcholy, které jsou značeny čísly 1,2,. . . ,n. Pro zadání grafu do programu si uveďme dva způsoby.

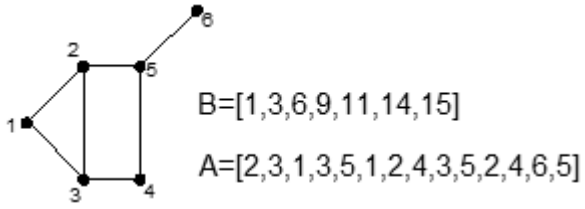
1. *Matice sousednosti M* – tento způsob slouží k zadání neohodnoceného grafu, neorientovaného i orientovaného. Matice je typu $n \times n$ obsahující pouze jedničky a nuly, kde $M(i, j) = 1$ znamená, že existuje hrana mezi vrcholy i a j, hodnota $M(i, j) = 0$, že hrana mezi vrcholy neexistuje. Pokud 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.

Reprezentace pomocí matice se hodí pro úplné grafy nebo grafy jim blízké. V opačném případě by matice obsahovaly velké množství nul a výpočet by byl paměťově náročný.

2. *Seznam sousedů* jednotlivých vrcholů je úspornější reprezentace grafu. V grafu jsou vrcholy číslovány od 1 a vzestupně. Princip spočívá v tom, že si ke každému vrcholu pamatujeme čísla vrcholů, do kterých z něj vede hrana. Mít uložené následovníky každého vrcholu v samostatném poli by bylo paměťově náročné, proto se používají pro uložení celého grafu pouze dvě pole: v poli A jsou uložena čísla všech vrcholů, do kterých vedou hrany a v poli B jsou pouze ukazatelé, které určují, kde v poli A začínají sousedé konkrétního vrcholu, který je reprezentován 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 vrcholu u jsou tedy vrcholy s čísly $A[B[u]], A[B[u]+1], \ldots, A[B[u+1]-1]$, viz příklad na obrázku 10. Velikost obou

polí se dá lehce určit předem. V poli A jsou potažmo všechny hrany grafu. Hrany obousměrné jsou započítány dvakrát. Na obrázku 10 je sedm obousměrných hran, proto je velikost pole A 14. Pole B by intuitivně mělo obsahovat pouze tolik prvků, kolik je vrcholů, v našem konkrétním příkladě 6. Jako poslední prvek pole B je ale navíc ukazatel na první volnou pozici pole A, proto je jeho hodnota o jedno větší než velikost pole A, tady konkrétně 15.



Obrázek 10: Zadání grafu pomocí seznamu sousedů

Jelikož je způsobů zadání více, dostáváme se k problému, který z těchto způsobů je výhodnější. Jak bylo řečeno, matice sousednosti je vhodná pro úplné grafy nebo grafy jim blízké. To ale jakákoli silniční síť není. Cesty (neboli hrany grafu) jsou po velmi krátké vzdálenosti přerušeny městem nebo vesnicí, což znamená další vrchol do grafu, ze kterého ale mnohdy nebude vycházet mnoho hran. Pokud si představíme nějakou odlehlejší vesnici v horách, vede do ní často jenom jedna nebo dvě silnice. Počítáme-li s ní ale do grafu například příslušného kraje, bude v něm přes 100 vrcholů, i když to bude ten v České republice nejmenší. Řádek v matici sousednosti příslušný této vesnici by obsahoval pouze jedno nebo dvě nenulová čísla a zbytek by byly samé nuly. Pokud bychom chtěli celorepublikovou síť znázornit maticí sousednosti, kde za uzly vezmeme jednotlivé obce, byla by to matice přibližně typu 6251×6251 [4], v níž by většina členů byla nulová. Kdybychom za uzly považovaly jednotlivé křižovatky, byla by matice podstatně větší. Proto je pro silniční síť výhodnější implementace pomocí seznamu sousedů, kde jsou pouze hrany, které opravdu existují.

1.4. Procházení grafu

Techniky procházení lze aplikovat nejen na silniční síť ale i v mnoha jiných aplikacích, jako je například průchod bludištěm. Dva základní postupy jsou průchod *do šířky* a *do hloubky*. Oba tyto způsoby dojdou ke stejnému cíli, ale liší se pořadím, v jakém jsou místa navštívena. Proto se každý z nich používá pro řešení jiných problémů. Procházení grafů je obdobné procházení stromů, ale protože u grafů neznáme pojem kořen, musíme zvolit uzel, ze kterého budeme graf procházet. Také musíme zaručit, že některý uzel nenavštívíme vícekrát. K tomu použijeme následující pojmenování stavu uzlů a hran. Uzel může mít stav iniciační, ten přidělíme všem na začátku. Poté, co najdeme nějakou hranu, která do něj vede, prohlásíme ho za nalezený, a jakmile probereme všechny hrany z něj vycházející, budeme mít stav zpracovaný. U hran je to podobné. Na začátku algoritmu je jim přidělen stav iniciační a poté, co už byla probrána od jednoho ze svých uzlů, změní se její stav na zpracovaný. Pro rozdíl mezi algoritmy je nejpodstatnější pomocná datová množina, označovaná jako úschovna U. Do ní se ukládají nalezené ale ještě nezpracované vrcholy. Podle toho jak je úschovna U implementována se rozlišují dva způsoby procházení grafu. Jestliže z U vybereme k dalšímu zpracovaní vrchol, který byl nalezený jako poslední, tj. U je implementována jako zásobník, jedná se o procházení „do hloubky“. Když dále prohledáváme od prvních nalezených vrcholů, tj. U je fronta, jedná se o procházení „do šířky“. Přepis algoritmu je uveden například v [8] na str. 16. Dále popsaný Dijkstrův algoritmus je určitou obdobou prohledávání do šířky, kdy z úschovny vybereme vrchol nejbližší k počátečnímu.

1.5. Metrika v grafu

V předcházející kapitole jsme zavedli pojem souvislost grafu, který zaručuje možnost dostat se z jednoho vrcholu grafu do druhého. Nyní nás bude zajímat jaká je délka nejkratší cesty. Stanovení této délky se liší pro neohodnocený a ohodnocený graf. V neohodnoceném je to minimální počet hran, které musíme po cestě z jednoho vrcholu do druhého navštívit. V ohodnoceném grafu bereme

v úvahu ohodnocení hran, tedy např. délky cest.

Definice 1.10. *Vzdálenost $d_G(u, v)$ dvou vrcholů u, v v neohodnoceném grafu G je dána délkou nejkratšího sledu mezi u a v. Pokud sled mezi nimi neexistuje, je $d_G(u, v) = \infty$.*
V ohodnoceném grafu (G,w) rozumíme délkou sledu $S = v_0, e_1, \ldots, v_n$ součet

$$d_G^w(S) = w(e_1) + \cdots + w(e_n).$$

Vzdáleností mezi dvěma vrcholy u, v pak

$$d_G^w(u, v) = min\{d_G^w(S) : S \text{ je sled s konci } u, v\}$$

Definice 1.11. *Metrikou grafu rozumíme soubor vzdáleností mezi všemi dvojicemi vrcholů grafu. Je to tedy matice, ve které prvek D(i,j) udává vzdálenost mezi vrcholy i a j.*

2. Hledání nejkratší cesty, Dijkstrův algoritmus

Pro nalezení nejkratší cesty je nejdůležitějším aparátem tzv. Dijkstrův algoritmus. Proto než přejdeme k samotnému principu algoritmu, zmiňme pár slov o samotném E.W. Dijkstrovi.

2.1. Edsger Wybe Dijkstra

E.W.Dijkstra byl přední nizozemský informatik. Narodil se v Rottterdamu 11. května 1930. Jeho matka byla matematička a otec chemik. Po studiu na gymnázium se rozhodl věnovat matematice a teoretické fyzice na Leidenské univerzitě, na které promoval roku 1956. O tři roky později získal také titul na Univerzitě v Amsterdamu. Už během studií pracoval v Mathematical Centrum v Amsterdamu, kde také potkal svou ženu. S ní se později odstěhoval do Eindhoveru, kde se stal profesorem na technické univerzitě. V roce 1972 obdržel Turingovu cenu (nejvýznamnější cena v oboru informatiky) za přínos rozvoji programovacích jazyků.[1]

[1] *http://cs.wikipedia.org/wiki/Turingova_cena*

Část svého života Dijkstra prožil také v americkém Austinu, kde pracoval až do svého odchodu do důchodu na podzim roku 1999. Nemocný se v únoru roku 2002 vrátil do svého domu na okraji Eindhoveru, kde půl roku poté, 6. srpna 2002, na rakovinu umírá.

I když Dijkstra vystudoval fyziku, celý život se zabýval programováním. Vývoj a podobu současné informatiky ovlivnil jako nikdo jiný. Své rukopisy psal po dobu asi 40 let a rozesílal je známým, kteří je šířili dále. Málokdy jsou delší než 15 stran a jsou postupně číslovány. Poslední z nich, číslo 1318, je z března 2002. Jsou známy pod označením „EWD“ a většinu z nich vydala univerzita v Austinu na CD v roce 2000 nebo jsou dostupné na webové stránce http://www.cs.utexas.edu/users/EWD/ . Mnohé tyto práce daly za vznik novým oblastem výzkumu. Stál například u vzniku programovacího jazyka ALGOL 60, zavedení řídících struktur a jako první zveřejnil mnohé programátorské problémy, které jsou dnes považovány za standartní a řešitelné, a část z nich také nese jeho jméno.

Obrázek 11: E.W.Dijkstra

Jedno z jeho nejznámějších děl je z roku 1959. Ve svém třístránkovém článku pod názvem „A note on two problems in connexion with graphs“ uvedl algoritmus pro nalezení nejkratší cesty v grafu, nyní nazývaný Dijkstrův algoritmus. Na jeho myšlenku a průběh se podíváme blíže.

2.2. Hledání nejkratší cesty

Problém nalezení nejkratší cesty můžeme v dnešní době nalézt v různých obměnách ve spoustě aplikací. Například internetový vyhledávač vlakových spojení prohledává po zadání výchozího a cílového města (neboli vrcholu) železniční síť a snaží se uživateli najít tu nejkratší cestu. Pokud přidáme požadavek, aby hledal pouze přímá spojení, neznamená to pro algoritmus nic jiného, než že musí pracovat pouze s grafem, který obsahuje přímé hrany z města (vrcholu) A do města B. Také všechny navigace pracují na podobném principu. Ty nejenže umí vypočítat nejkratší cestu mezi dvěma městy, ale protože za ohodnocení hran lze použít nejen délku cesty v kilometrech, ale také nejvyšší povolenou rychlost na této silnici, není výraznější problém spočítat i nejrychlejší cestu.

Pro praktické příklady nemusíme chodit až do světa technologií. Ten nejintuitivnější výpočet nejkratší cesty používáme skoro denně. Pokud se například student potřebuje dostat v Praze od Národního divadla na Hlavní nádraží, má hned několik možností, jak to provést. Dejme tomu, že je líný a chudý, a tedy nechce jít ani pěšky a ani nemá auto. Okamžitě se rozhodne pro metro, protože cestovat přes centrum Prahy autobusem je pouze pro otrlé. Nasedne tedy na Národní třídě do metra. Jenže tady se mu naskýtají další možnosti. Cesta totiž není jenom jedna (viz obrázek 12). Může totiž hned na Můstku přestoupit, jet jednu zastávku k Muzeu a od něj zase jednu zastávku a je v cíli. Vystupuje tedy už na třetí zastávce. Může ale také od divadla jet až na Florenc, tam přestoupit a opět jen jednou zastávkou k nádraží. To znamená sice jenom jeden přestup, ale zato bude vystupovat až na čtvrté zastávce. Pokud by nás tedy zajímal (třeba kvůli jízdence) jen počet zastávek, znamená to práci s neohodnoceným grafem a vybereme si první trasu. Jestliže ale víme, jak jsou stanice od sebe vzdálené a přibližný čas potřebný k přestupu, pracujeme s ohodnoceným grafem, a pak bude kratší druhá možnost. Tento problém má každý cestující vyřešený za pár vteřin, protože tyto základní informace zná. Princip řešení je jednoduchý. Pro nalezení té nejkratší cesty si předem spočítá všechny možnosti a potom z nich vybere tu nejlepší. To je ten nejprimitivnější algoritmus hledání nejkratší cesty. Není zcela

nepoužitelný, ale pro grafy s větším počtem hran a uzlů už optimální určitě není. Například kdybychom jeli do New Yorku, kde má metro pře 400 stanic a 26 linek, mohli bychom s takovýmto postupem zůstat na naší startovní stanici celou dovolenou.

Obrázek 12: Plán metra

2.3. Dijkstrův algoritmus

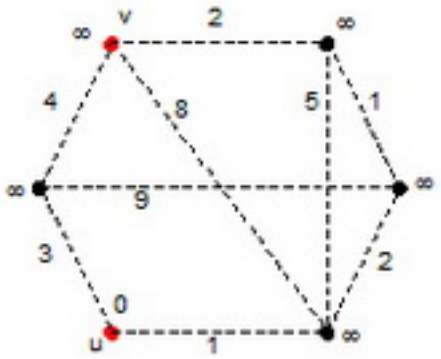
Dijkstrův algoritmus je určen pro nalezení nejkratší cesty v nezáporně ohodnoceném grafu. Pro stejný typ grafů se používá i Floyd-Warshallův algoritmus, Dijkstrův je ale rychlejší pro nalezení nejkratší cesty pouze mezi dvěma vrcholy. Dijkstrův algoritmus je variantou procházení grafu a svým průběhem připomíná prohledávání do šířky. Pro každý nalezený uzel si pamatuje i délku nejkratší cesty, po které jsme se do něj dostali, a k vyšetřování dalšího uzlu vybíráme ten, který má nejmenší vzdálenost od startovního uzlu. Do takového uzlu se už totiž kratší cestou nelze dostat. Po prošetření celého grafu udávají tyto vzdálenosti nejkratší cestu ze startovního uzlu do ostatních.

Popis algoritmu: Vstupní data algoritmu jsou graf (zadaný buď maticí nebo seznamem), délky hran a startovní a cílový uzel (na obr. 13 vrcholy u, v označeny červeně). „Před zahájením průchodu grafem bude mít startovní uzel přiřazenou nulovou vzdálenost (cesta do něj má nulovou délku) a všechny ostatní nekonečnou. Zatím jsme do nich totiž kratší než nekonečnou vzdálenost nenašli. Pro každý uzel také rozlišujeme zda je vzdálenost dočasná (tj. může se ještě zlepšit) nebo trvalá (výsledná nejkratší cesta do tohoto vrcholu).“ [15] Algoritmus začne

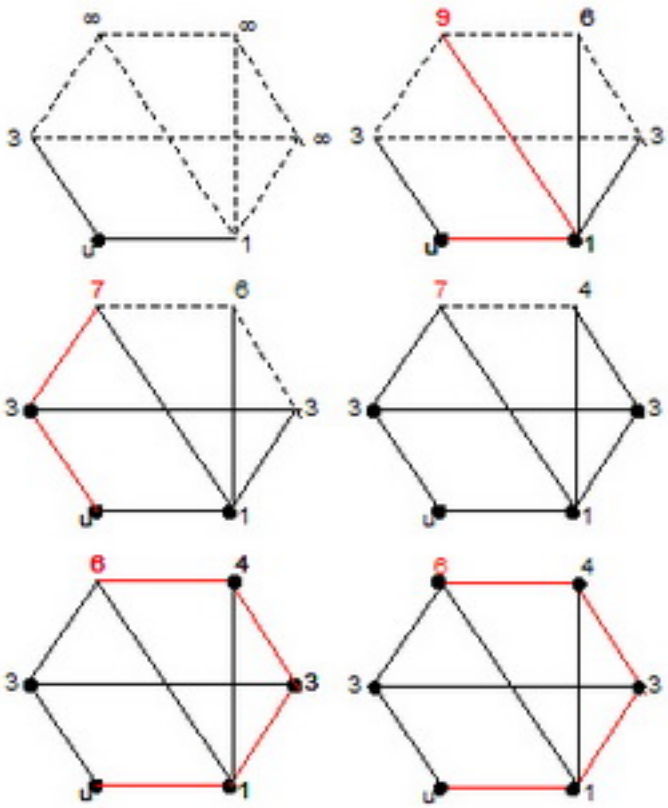
se startovním vrcholem a najde všechny hrany vedoucí k jeho sousedům. Poté začne porovnávat vzdálenosti. Protože jakákoli délka hrany je menší než nekonečno, přepíší se nekonečné vzdálenosti na konečné číslo a startovní vrchol prohlásíme za trvalý. Stejným postupem pokračujeme u dalších uzlů, kdy vybíráme vždy uzel s nejmenší vzdáleností od startovního uzlu. Jestliže je součet vzdálenosti prošetřovaného uzlu od startovního uzlu a délky hrany vedoucí do jeho souseda menší než poslední známá vzdálenost od startu tohoto souseda, vzdálenost se přepíše.

Výstupy Dijkstrova algoritmu mohou být dva. Buď nám stačí vypsat pouze nejkratší cestu ze startu do cíle, v tom případě algoritmus končí, jakmile je prošetřený cílový uzel. Ale pokud zaručíme prošetření všech uzlů grafu, bude ve výsledku u každého uzlu uvedena nejkratší možná vzdálenost od startovního uzlu.

Příklad Dijkstrova algoritmu je na obrázcích 13 a 14. Dosud neposuzované hrany jsou značeny čárkovanou, hrany, se kterými už algoritmus počítal, souvislou čarou. Prošetřené vrcholy jsou v řešení značeny tučně a hodnoty u nich uvedené udávají nejkratší známou vzdálenost ze startovního vrcholu. Na tomto příkladu se před prošetřením cíle prošetřily všechny ostatní uzly, proto po výpočtech známe nejkratší vzdálenosti ze startovního uzlu do všech ostatních.

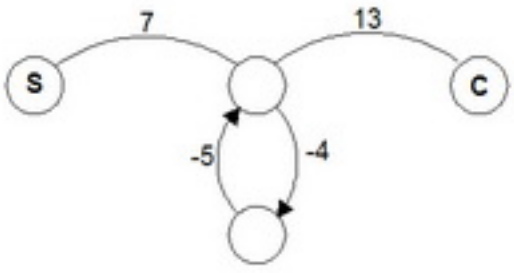


Obrázek 13: Příklad



Obrázek 14: Řešení příkladu

Dijkstrův algoritmus nefunguje pro záporné ohodnocení hran z toho důvodu, že bychom mohli jít nejdříve do vzdálenějšího vrcholu a poté projitím hrany se záporným ohodnocením celkovou délku cesty zase zkrátit.



Obrázek 15: Graf se záporným cyklem

2.4. Další algoritmy pro nalezení nejkratší cesty

K algoritmům nalezení nejkratší cesty patří také Floyd-Warshallův algoritmus, který se užívá, pokud potřebujeme znát cesty mezi každou dvojicí vrcholů. Jeho výstupem je totiž matice typu n×n pro n vrcholů v grafu, kde na pozici (i,j) je vzdálenost mezi dvojicí vrcholů i a j. Proto není efektivní ho používat, pokud nás zajímá jen cesta z jednoho vrcholu místo do všech dvojic.

Jak bylo v předchozí kapitole řečeno, Dijkstrův algoritmus zaručuje správnost výsledku pouze pro nezáporné ohodnocení hran. Patří k nejpoužívanějším, protože ať už ohodnocení hran představuje cokoli, zpravidla nezáporné bývá. Nemusí to být pouze vzdálenost, ale také náklady na cestu, spotřeba a jiné. Pokud je ohodnocení hran obecné, lze i přesto spočítat nejkratší cestu. Ohodnocení může reprezentovat třeba cenu, kterou nás bude průjezd hranou stát. Záporné číslo by tedy znamenalo, že někdo zaplatí za cestu nám. Pokud by takový graf obsahoval záporné cykly, nejkratší cesta by podle Dijkstrova algoritmu byla -∞. Došli bychom totiž do vrcholu, kde cyklus začíná a s každým průchodem tohoto cyklu by se cesta zkracovala, a tedy bychom stále vydělávali.

Pro případy s obecným ohodnocením hran proto existují jiné algoritmy. Jako třeba Bellman-Fordův algoritmus.

3. Zranitelnost silniční sítě

V této kapitole nejprve zavedeme pojmy potřebné k výpočtu zranitelnosti silniční sítě, a poté se podíváme na metodu E. Jeneliuse pro výpočet. Tu aplikujeme dvěma způsoby na reálné silniční sítě. Nejprve na určení důležitosti jednotlivých hran v síti, a potom na stanovení nejefektivnějšího vylepšení sítě.

3.1. Definice pojmů zranitelnost, spolehlivost a důležitost silniční sítě

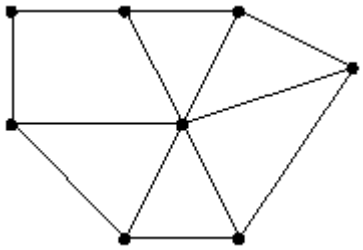
Pojem zranitelnost doposud nemá všeobecně uznávanou definici. Přesný význam tohoto pojmu vysvětlují různi autoři jinak a jejich odlišnosti jsou dány

různým kontextem, ve kterém je toto slovo používáno. Proto na úvod této kapitoly uvedeme různé definice pojmu *zranitelnost*.

Například A. Holmgren v [9] používá tuto definici: „Zranitelnost je citlivost na hrozby a nebezpečí“. Jiní autoři (jako například Laurentius v [13]) kladou důraz na nízký výskyt těchto událostí a jako zranitelnost označují citlivost sítě na vážná rizika nebo na náhlé nepředvídatelné události. Abrahamsson v [1] konstatuje, že celkově zkoumání zranitelnosti shrnuje myšlenka „Malé rány kácí velké stromy“. Tedy relativně malá nehoda, pokud se stane na nešťastném místě a v nevhodný čas, způsobí závažné škody nebo dokonce zkolabování celé sítě. Podobný přístup použila i Berdica v [2], kde definuje zranitelnost jako citlivost na selhání, která mohou vyústit ve značné snížení provozuschopnosti dané sítě. S touto definicí se ztotožníme v této práci.

Silniční sítě už se ve svém principu staví odolné. Pokud selže nějaká hrana, ve většině případů existuje cesta, po které se v síti dá přepravovat. Je to jedna z jejich základních myšlenek. Jejich zranitelnost se projeví, když je síť vytížená a její kapacita dosahuje maxima. Pak stačí nepatrný zásah a celá síť se řetězovou reakcí zhroutí.

Použití pojmu zranitelnost se v kontextu silničních sítí občas zaměňuje s pojmem *spolehlivost* sítě. Ten ale představuje něco jiného. L. H. Immers v [10] považuje ukazatel spolehlivosti spíše za uživatelsky užitečný. Definuje ji totiž jako míru jistoty, se kterou může uživatel odhadnout svůj cestovní čas. Ta závisí na dostupných informacích, které má, na případně alternativních cestách a také na stálosti a výkyvu časů, které můžeme znát ze zkušeností. Spolehlivost je tedy nízká, pokud jsou naše odhady pravidelně chybné. Najdeme ale i jiné definice spolehlivosti. Jako například *spolehlivost spojení*, která se zaměřuje na pravděpodobnost, že dva uzly zůstanou propojené, tj. existuje mezi nimi cesta i přesto, že došlo k selhání jedné nebo více hran. Na obrázku 16 můžete vidět příklad v tomto ohledu spolehlivé sítě, protože mezi každou dvojicí uzlu existuje dostatečný počet alternativních cest.



Obrázek 16: Příklad spolehlivé sítě

Uveďme nyní příklad, na kterém bude patrný rozdíl mezi konceptem zranitelnosti a spolehlivosti. D'Este a Taylor uvedli názorný příklad na australské síti. Došlo k selhání největší dálnice Eyre Highway mezi Perthem a Adelaide, třeba z důvodů záplav. Celá síť zůstane nadále propojená a navíc pravděpodobnost záplav je velice nízká, proto by ukazatel spolehlivosti předem neukazoval žádný větší problém. Následky tohoto výpadku jsou ale podstatné. Další možná cesta totiž zahrnuje 5000 km dlouhou objížďku.

Ještě zavedeme pojem *důležitost* a stanovíme, kterou složku sítě považujeme za *kritickou*. Nyní nebudeme rozlišovat, o které části grafu mluvíme, proto pod pojmem *složka* budeme rozumět uzel, hranu, soubor hran nebo soubor uzlů. Jestliže je pravděpodobnost selhání složky vysoká, nazýváme ji *slabou* a pokud jsou následky tohoto selhání velké, označíme ji za *důležitou*. Jakmile je složka slabá a důležitá současně, považujeme ji za *kritickou*. Čím je složka kritičtější, tím vážnější škody nastanou, pokud selže.

3.2. Výpočet důležitosti úseku v dopravní síti

V této podkapitole si předvedeme metodu hodnocení zranitelnosti silniční sítě, kterou uvedl E. Jenelius v [11].

Nechť V je množina všech uzlů na uvažované dopravní síti a E množina všech hran. Předpokládáme, že problémová situace je taková, když hrana (nebo soubor hran), obecně nazýváme *úsek*, je přerušena nebo uzavřena, což nutí všechny

cestující, kteří přes ní chtěli projet, použít jinou, alternativní cestu.[2] Dále předpokládáme, že cestující se při volbě cesty rozhodují tak, aby při cestě z výchozího bodu do cíle minimalizovali svoje náklady. Čili předpokládáme, že cestovní náklady mezi každou dvojicí start-cíl (značíme OD) [3] jsou známé. Výchozí uzel budeme značit i, cílový j a přerušený úsek k. Nechť c_{ij}^0 představuje cestovní náklady na přepravu z uzlu i do uzlu j v neporušené síti a c_{ij}^k náklady při přerušení úseku k. Základ pro výpočet důležitosti tedy tvoří nárůst nákladů

$$\Delta c_{ij}^k = c_{ij}^k - c_{ij}^0.$$

Za cestovní náklady budeme považovat vzdálenost mezi uzly. Kromě možného nárůstu nákladů musíme počítat i s horším následkem selhání úseku k. Výpadek může zapříčinit rozdělení sítě na několik vzájemně nedostupných částí, tedy komponent, a cestovní náklady mezi uzly v odlišných částech by narostly na nekonečno. Proto zavádíme koncept neuspokojeného přepravního proudu u_{ij}^k

$$u_{ij}^e = \begin{cases} x_{ij}, & \text{jestliže } c_{ij}^k = \infty, \\ 0, & \text{jestliže } c_{ij}^k < \infty, \end{cases} \tag{1}$$

kde x_{ij} je velikost přepravního proudu mezi uzly i a j, tj. počet jízd z i do j, které jsou neuskutečnitelné po selhání úseku k. Přerušení úseku může tedy mít dva následky. Buď dojde pouze k navýšení nákladů (popř. zůstanou stejné, pokud existuje stejně nákladná alternativní cesta) a každá dvojice uzlů zůstane dostupná. Takový úsek nazýváme nc-úsek[4]. Množinu všech takovýchto úseků značíme E^{nc}. Nebo výpadek způsobí rozpad na komponenty. Počet takových úseků je dán vztahem $E^c = E \setminus E^{nc}$.

Důležitost úseku můžeme počítat vzhledem k jednomu vrcholu, ke skupině vrcholů nebo pro celou síť. Uvažujme nc-úsek označený k. V případě výpočtu pro celou síť zjistíme Δc_{ij}^k pro všechny OD dvojice uzlů. Každé dvojici je také přidělena hodnota *váhové funkce* w (zn. w_{ij}). Tou může být např. spotřeba paliva,

[2] Selhání uzlu je ekvivalentní se selháním všech hran do něj vedoucích.
[3] v angličtině OD pair (origin-destination)
[4] z anglického non cut

hustota dopravy na této hraně a jiné.

Důležitost úseku k s ohledem na celou síť je

$$I_{net}(k) = \frac{\sum_i \sum_{j \neq i} w_{ij}(c_{ij}^k - c_{ij}^0)}{\sum_i \sum_{j \neq i} w_{ij}}, k \in E^{nc}, \tag{2}$$

pro $k \in E^c$ je $I_{net}(k) = \infty$.

Ukazatel $I_{net}(k)$ je roven nekonečnu pro hrany, které zapříčiní rozpad na nesouvislý graf. Proto jsme v úvodní kapitole definovali pojem m-souvislého grafu. Pokud můžeme odebrat $m-1$ hran a síť zůstává souvislá, ale při odebrání m-té hrany bude už mít nějaká cesta mezi dvěma uzly délku nekonečno, je síť m-souvislá.

Pokud bychom za váhovou funkci použili velikost přepravního proudu x_{ij}, znamená to, že růst cestovních nákladů mezi dvěma uzly závisí na hustotě dopravy mezi nimi.

$$I_{net}^{dem}(k) = \frac{\sum_i \sum_{j \neq i} x_{ij}(c_{ij}^k - c_{ij}^0)}{\sum_i \sum_{j \neq i} x_{ij}}, k \in E^{nc}. \tag{3}$$

Jestliže jsou všechny hodnoty hodnoty w_{ij} shodné (např. $w_{ij} = x_{ij} = 1$, tj. z každého uzlu i do každého uzlu j se uskuteční právě jedna přeprava), lze globální důležitost úseku u vypočítat takto:

$$I_{net}^{glob}(k) = \frac{1}{N^d(N^d-1)} \sum_i \sum_{j \neq i} (c_{ij}^k - c_{ij}^0), k \in E^{nc}, \tag{4}$$

kde N^d je počet uzlů.

Důležitost úseku k vzhledem k velikosti neuspokojeného přepravního proudu lze vypočítat takto:

$$I_{net}^{uns}(k) = \frac{\sum_i \sum_{j \neq i} u_{ij}^k}{\sum_i \sum_{j \neq i} x_{ij}}, k \in E, \tag{5}$$

pro $k \in E^{nc}$ je $I_{net}^{uns}(k) = 0$.

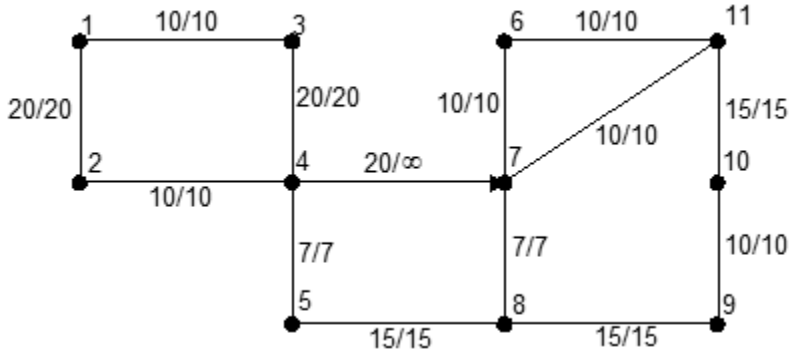
Jak si můžete všimnout, je ve tvaru zlomku, ve kterém dělíme celkovým množstvím přeprav. Výsledné číslo je tedy procentuální podíl neuspokojených přeprav

z celkového počtu přeprav.

Ukažme si aplikaci výše uvedených vzorců na modelovém příkladě.

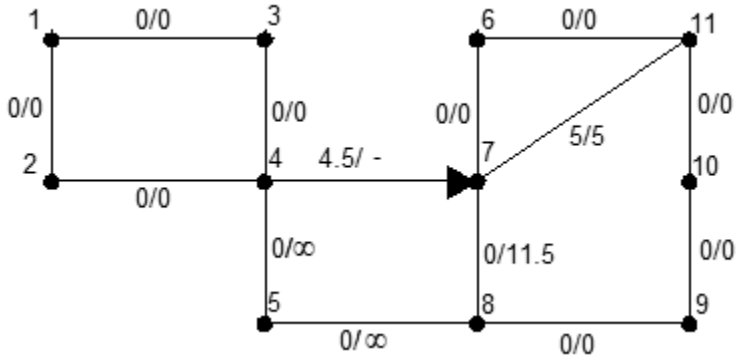
Modelový příklad

Modelová silniční síť na obrázku 17 se skládá z jedenácti uzlů. Skoro všechny úseky této sítě jsou neorientované a lze je tedy procházet v obou směrech. Pouze úsek 4-7 je orientovaný. Ohodnocení hran značí náklady na průchod hranou, a to ve tvaru tam/zpět, kdy směr tam je cesta z uzlu s nižším číslem k vyššímu a zpět od vyššího k nižšímu. U orientované hrany jsou přirozeně nastaveny náklady na průchod v jednom směru na nekonečno.



Obrázek 17: Modelová silniční síť

Uvažujme za dvojici start-cíl uzly s čísly 1 a 11. Do vztahů pro výpočty budeme uvažovat *pouze tuto dvojici*. Mezi těmito uzly existují dvě stejně nákladné cesty s hodnotou 60, a to 1-3-4-7-11 a 1-2-4-7-11. Za váhovou funkci x_{ij} budeme považovat počet jízd, který zvolíme $x_{1,11} = x_{11,1} = 10$ a ostatní $x_{ij} = 0$. Tedy všichni cestující jezdí pouze mezi uzly 1 a 11, a v každém směru je jich právě 10. To může nastat například v situaci, kdy všechny ostatní uzly jsou křižovatky.

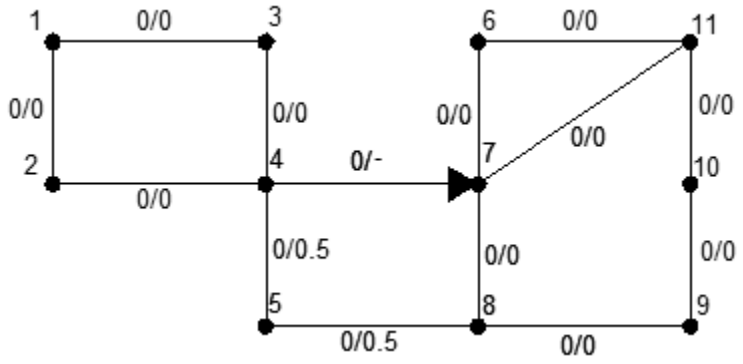


Obrázek 18: Hodnoty $I_{net}^{dem}(k)$

Na obrázku 18 jsou u každé hrany uvedeny hodnoty dvou dílčích členů ze vztahu (3), ve kterých není nulová váhová funkce. U každé hrany vyjadřují hodnotu tohoto ukazatele při selhání této hrany k. Pro přehlednost jsou uvedeny v následujícím tvaru: Před lomítkem je hodnota pro přepravy z 1 do 11, za lomítkem pro přepravy z 11 do 1. Hodnota $I_{net}^{dem}(k)$ by pak byla jejich součtem. U jednosměrné je hodnota jenom jedna, protože opačný směr není možný. Podívejme se například, co se stane v případě, že hrana 4-7 nebude průjezdná. Tehdy totiž musíme graf obejít spodní částí přes uzly 5 a 8, což je o 9 jednotek nákladnější cesta. Proto

$$I_{net}^{dem}(4-7) = \frac{\sum_i \sum_{j \neq i} x_{ij}(c_{ij}^k - c_{ij}^0)}{\sum_i \sum_{j \neq i} x_{ij}} = \frac{10 * 9}{10 + 10} = \frac{9}{2}. \tag{6}$$

Podobná situace nastává i při výpadku hrany 7-8. Zde se navýšení nákladů projeví pouze u přeprav z uzlu 11 do 1, v opačném směru nemá výpadek této hrany na nejkratší cestu vliv. Složitější je to s cestou z uzlu 11 „zpět“ do 1. Už ze zadání sítě je zřejmé, že na této cestě mají nejpodstatnější roli hrany 8-5 a 5-4. Ty nelze nijak obejít, protože hrana 7-4 neexistuje, a tedy při jejich výpadku je cílový uzel nedosažitelný. Proto je u těchto hran na obrázku 18 znak ∞. Nekonečna se u hran vyskytly už při přerušování jednotlivých hran, proto je tato síť pouze souvislá (1-souvislá).



Obrázek 19: Hodnoty $I_{net}^{uns}(k)$

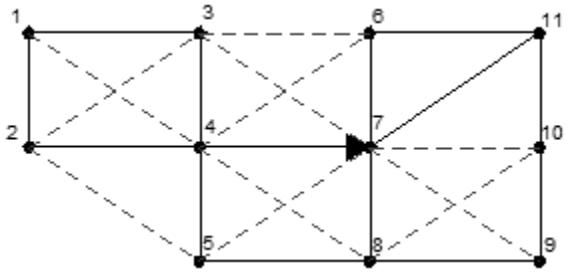
Hodnoty, které jsou znázorněny na obrázku 19, udávají procento neuspokojených cestujících, tj. podíl těch, kteří se při výpadku konkrétní hrany nedostanou do cíle. Opět jsou hodnoty v obrázku rozdělené na dvě části a celková hodnota $I_{net}^{uns}(k)$ je jejich součtem. Například hrana 10-11 má ohodnocení $0/0$, protože při jejím selhání ještě existuje možnost, jak se dostat z uzlu 1 do 11 i zpět, což je jediná dvojice uzlů, s kterou v příkladu počítáme. Jedinou nenulovou hodnotu mají hrany 5-4 a 8-5. Při selhání jedné hrany (popř. obou současně) už není možnost jak se dostat z uzlu 11 do 1. Protože počet přeprav z 11 do 1 je deset a počet všech přeprav dvacet je ukazatel $I_{net}^{uns}(5-4) = I_{net}^{uns}(8-5) = \frac{10}{20} = 0.5$. Do cíle se v takovém případě tedy nemůže dostat 50 % cestujících.

3.3. Výpočet důležitosti úseku po přidání hrany

V předchozí kapitole jsme se zabývali tím, co se stane se sítí při výpadku kterékoli hrany. Zkoumání zranitelnosti sítě se v takovém případě zaměřuje na následky selhání určité hrany, a to bez ohledu na pravděpodobnost tohoto selhání. Pravděpodobnost selhání určitého úseku nebereme v úvahu z toho důvodu, že existují případy selhání, které lze jen těžko předpovídat. Tím myslím teroristický útok, živelnou katastrofu či válečný konflikt. Jak jsme uvedli, selhání určité hrany (nebo souboru hran) může mít za následek až úplné odříznutí jednoho či více uzlů od zbytku sítě.

V této části se zaměříme na zlepšení sítě po přidání nějaké dosud neexistující hrany. Ve stávajícím grafu silniční sítě budeme spojovat uzly, které zatím mezi sebou silnici nemají, zjistíme, jaký vliv by nová silnice měla na celou síť, a tím budeme moci stanovit, která města by se měla propojit novou silnicí. Vrátíme-li se k příkladu dálniční sítě ČR, vidíme, že mezi Prahou a celou Moravou je pouze dálnice D1. Proto také, když se na ní stane větší nehoda, tvoří se na ní dlouhé zácpy. Bylo by třeba výhodné spojit přímo Prahu s Ostravou, aby se dálnici na Brno odlehčilo. Takovou stavbu silničářům ale nedoporučíme, protože jsme do grafu nezahrnuly ostatní velké silnice, po kterých se dá dálnici vyhnout.

Z teoretického hlediska lze tvořit hrany mezi každou dvojicí uzlů. Vzali bychom konkrétní město a spojovali ho přímo se všemi ostatními, a to provedli pro všechna města v síti. V takovém případě bychom se ale nevyhnuli křížení silnic. Což sice není neřešitelný problém, cesty bychom mohli přemostit, pokud ale zvážíme reálné použití, není to příliš praktické, především z pohledu financí. Nikdo nám totiž z města, ze kterého už na jednu ze světových stran vede několik silnic, nezačne stavět další. Uvedeme si to na modelovém příkladu, ke kterému vezmeme silniční síť z obrázku 16 minulé kapitoly. Nemá smysl spojovat novou silnicí třeba uzly 5 a 9, když mezi nimi existuje cesta přes vrchol 8. Mohlo by být výhodné spojit přímo uzly 9-1, aby mezi „pravou“ a „levou“ částí grafu neexistovala pouze jedna obousměrná cesta. Při stavbě takové silnice bychom museli přemostit hned tři cesty. Proto nové hrany, které se nekříží s původními úseky a lze je reálně postavit, jsou na obrázku 20 znázorněny přerušovanou čarou.



Obrázek 20: Modelový příklad s možnými přidanými hranami

Při výpočtu postupujeme následovně. Vybereme uzel a k němu budeme postupně přiřazovat další uzly. Zjistíme, jestli mezi nimi existuje hrana, pokud ano, přistoupíme k dalšímu uzlu, pokud ne, tak tyto dva uzly hranou pomyslně spojíme a vypočítáme, jestli se někde v grafu protíná s jinou hranou. Jestliže zjistíme, že existuje průsečík, víme, že takovou hranu postavit nejde. Tímto způsobem zjistíme přípustné hrany. Jakmile na takovou hranu narazíme, přidáme ji do grafu. Pro tento upravený graf postupujeme stejně jako v předchozí kapitole a napočítáme důležitost jednotlivých hran. Protože přidaná hrana síť zlepší, jsou tyto hodnoty v porovnání s původním grafem lepší.

Po prošetření všech nových hran zjistíme, která naši síť vylepšila nejvíce, a tu můžete doporučit k budoucí výstavbě. Nejlepší hranu určíme tak, že spočítáme průměr z jednotlivých hodnot $I_{net}(k)$ pro nově vzniklé sítě, a čím menší tento průměr bude, tím je hrana lepší.

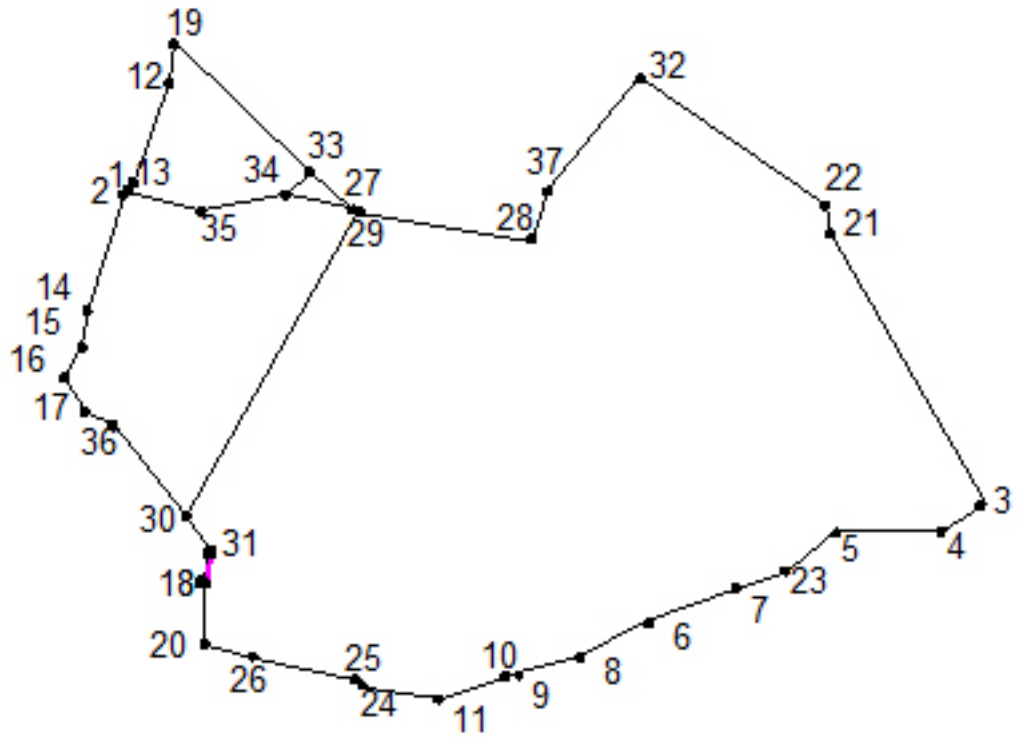
Analogicky můžeme přidávat všechny kombinace dvojicí (trojicí,...) hran, které na sebe můžou nebo nemusí navazovat.

4. Aplikace na reálné síti

V této závěrečné kapitole se podíváme na aplikaci námi zavedených ukazatelů na reálnou silniční síť. Máme k dispozici části silniční sítě Zlínského kraje, jejíchž znázornění je na obrázku 21 a 23. Upozorňuji, že v těchto nákresech délky hran nevystihují skutečnou vzdálenost mezi uzly. Data jsou načtena z textových souborů, jejichž popis je uveden v příloze 4.

1. První část sítě

Z načtených dat máme k dispozici počet uzlů společně s hranami a vzdálenostmi mezi nimi. Pro výpočet důležitosti jednotlivých úseků potřebujeme ještě znát váhovou funkci. My za ni zvolíme velikost přepravního proudu, jak je uvedeno ve vztahu (3) na straně 25. Informace o hustotě provozu na jednotlivých hranách je ale těžko dostupná. Mnohem dostupnější je informace o počtu obyvatel v jednotlivých uzlech. Proto skutečnou hodnotu nahradíme odhadem.



Obrázek 21: První část silniční sítě Zlínského kraje - zjednodušení na graf

Je rozumné domnívat se, že mezi velkými městy cestuje daleko více lidí než mezi malými vesničkami, a také že mezi bližšími městy jezdí více lidí než mezi vzdálenějšími. Hodnotu váhové funkce mezi uzly a a b tedy zjistíme z následující rovnice:

$$x_{ab} = P\frac{(P_a + P_b)\frac{1}{d_{ab}}}{2\sum_i \sum_{j>i}(P_i + P_j)\frac{1}{d_{ij}}}, \tag{7}$$

kde P je počet všech obyvatel v síti, tj. $P = \sum_{i=1}^{n} P_i$. Hodnoty P_a, P_b udávají počet obyvatel v uzlech a, b a hodnota d_{ab} délku nejkratší cesty mezi uzly a, b. Ve jmenovateli zlomku se tyto hodnoty sečtou přes všechny uzly v síti, kde využíváme toho, že jsou všechny hrany neorientované.

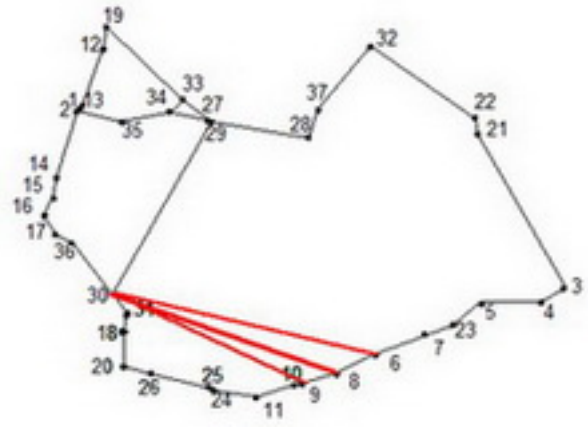
Než přistoupíme k vylepšování sítě, podívejme se na její původní podobu. Spuštěním programu z přílohy 2 zjistíme, že nejdůležitější je v ní hrana spojující uzly 18 a 31. Tato hrana je zvýrazněna na obrázku 21. Protože jsme výpočet

důležitosti založili na odhadu velikosti přepravního proudu, je nejdůležitější právě tato hrana, která je poměrně krátká a spojuje dva uzly s velkým počtem obyvatel.

Po spuštění programu uvedeného v příloze 3 je výsledkem matice N typu 267×43, protože počet všech hran, které je v námi zvolené síti možné přidat, je právě 267. Pro vyhodnocení výsledků jsme ji tedy převedli na matici H, která má pouze tři sloupce. V prvních dvou sloupcích jsou čísla uzlů, mezi které se přidala hrana, a ve třetím je průměrná důležitost hrany, kterou spočítáme jednoduše jako průměr sloupců 3 až 43 (viz předchozí kapitola). Ze všech možností vybereme pouze prvních deset nejlepších. Viz tabulka.

Uzel 1	Uzel 2	Průměrná důležitost	Uzel 1	Uzel 2	Průměrná důležitost
8	30	1448.5	10	30	1612.3
9	30	1465.7	8	31	1638.6
6	30	1471.3	6	31	1641.4
7	30	1556.1	11	30	1665.0
23	30	1556.3	5	30	1672.3

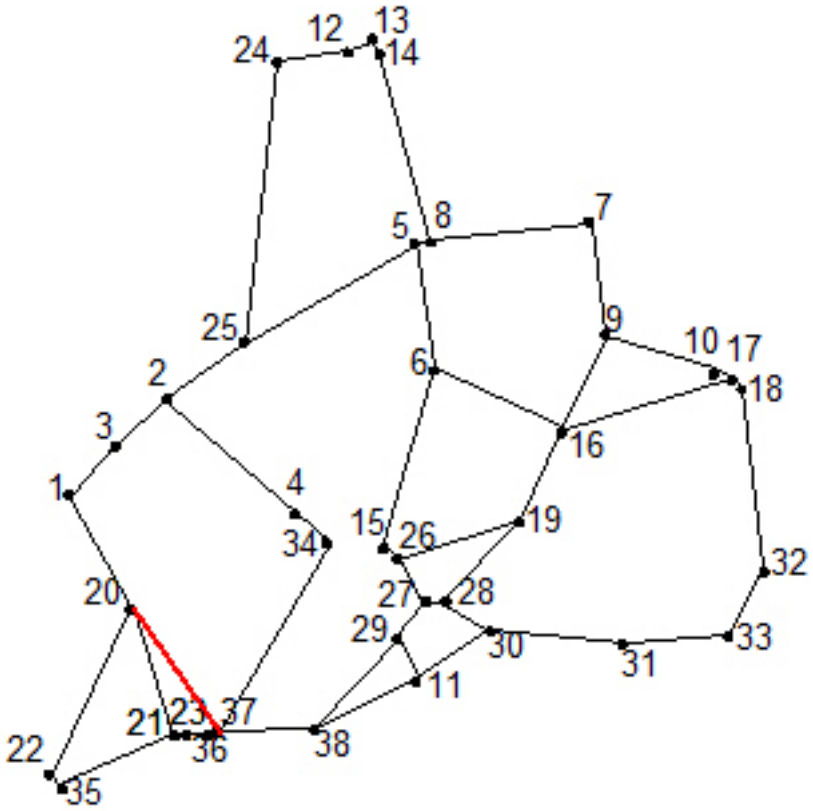
Za nejlepší vylepšení sítě považujeme tu hranu, která nejvíce sníží důležitost jednotlivých hran, potažmo dává nejnižší průměr těchto důležitostí. Proto pro největší vylepšení sítě můžeme ke stavbě navrhnout hrany 8-30 nebo 9-30. Uzel 30 má totiž bezkonkurenčně nejvíce obyvatel v celé síti.



Obrázek 22: Nejpřínosnější vylepšení první části sítě

2. Druhá část sítě

Nyní se podíváme na další část silniční sítě. V jejím původním stavu je nejdůležitější hrana spojující uzly 36 a 37, která je opět krátká a spojuje dva uzly s více než tisícovkou obyvatel. Protože je tato hrana velmi krátká, nezakreslujeme ji kvůli přehlednosti do obrázku. Jak je vidět na obrázku 23, je síť složitější a obsahuje více hran než v části 1, takže při zjišťování, které hrany můžeme přidat, nám možnosti ubývají rychleji. Ve výsledku je možno přidat 136 různých hran, což je o více než 100 méně než v části 1. Všech 136 možností uvádět nebudeme. Zmíním pouze prvních 10, a nejlepší zakreslíme do sítě červeně. Z možných vylepšení je nejlepší hrana 20-37, protože propojí dva velké uzly.

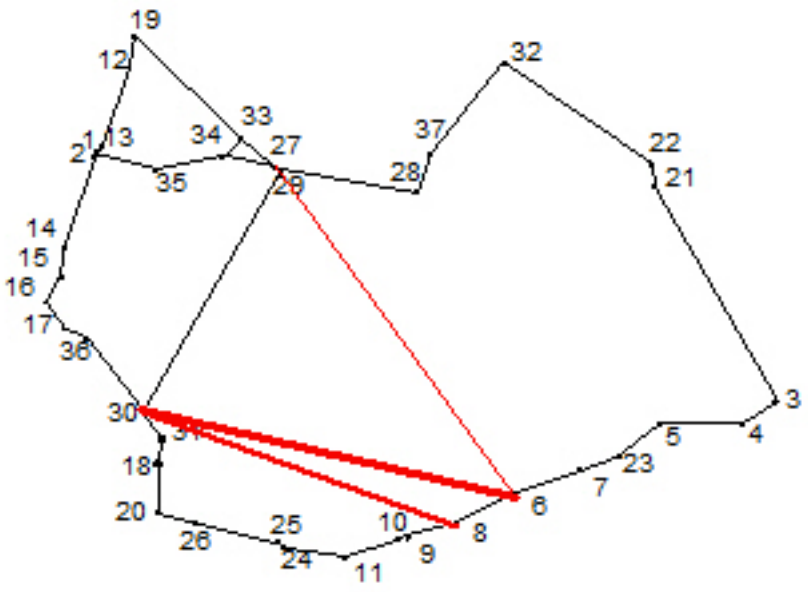


Obrázek 23: Druhá část silniční sítě Zlínského kraje a její vylepšení

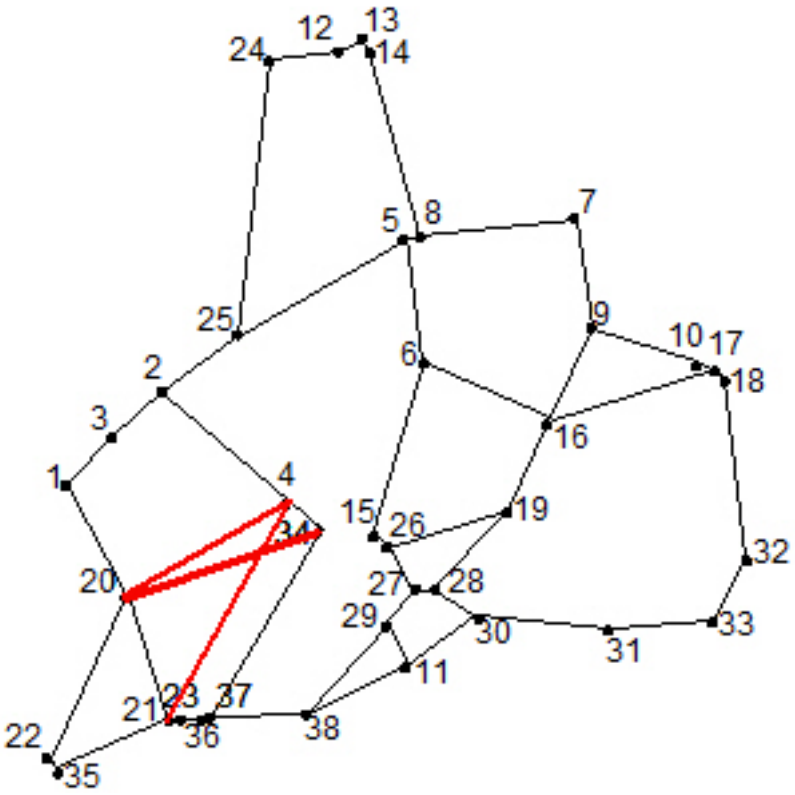
V následující tabulce je uvedeno 10 nejlepších úprav sítě.

Uzel 1	Uzel 2	Průměrná důležitost	Uzel 1	Uzel 2	Průměrná důležitost
20	37	430.6832	21	36	454.7311
23	37	433.0419	4	21	454.8235
20	34	442.0427	35	38	455.4361
21	34	447.5594	23	34	469.0946
4	20	448.5099	20	36	471.5060

Doposud jsme volili za váhovou funkci odhad hustoty průjezdu. Jak jsme zmínili v teoretické části práce, můžeme váhovou funkci volit i konstantní. Nebrali bychom v potaz rozdíl mezi silnicemi a $w_{ij} = 1, \forall i, \forall j, j \neq i$. V takovém případě jsou výsledky jiné a jsou znázorněny na obrázcích 24 a 25. Nejpřínosnější vylepšení je značeno nejtučnější čarou.



Obrázek 24: Vylepšení pro první část sítě s konstantní váhovou funkcí



Obrázek 25: Vylepšení pro druhou část sítě s konstantní váhovou funkcí

Závěr

V této práci jsme uvedli konkrétní aplikaci teorie grafů v silniční dopravě. K volbě tématu nás vedla skutečnost, že silniční síť České republiky tvoří jeden ze základních pilířů přepravy lidí a zboží, a s jejich vzrůstajícím počtem je stále vytíženější. Metoda uvedená v naší práci může pomoci ke zlepšení jakékoliv sítě, a to určením nejefektivnějšího úseku pro případnou budoucí výstavbu. Stavění silnic není zrovna levná záležitost, a proto by nově postavená cesta mezi některými městy nemusela mít velký význam a přínos.

Protože napsání této práce bylo inspirováno běžnou praxí, chtěli jsme, aby příklad v kapitole 4 byl opravdu reálný. Rozhodli jsme se ho tedy postavit na základě skutečné silniční sítě Zlínského kraje.

Výsledky bakalářské práce jsou aplikovatelné a využitelné nejen v optimalizaci silniční sítě, lze je využít k efektivnímu vylepšení obdobných sítí. Jejím zpracováním jsem získala mnoho zkušeností a zároveň si výrazně prohloubila znalosti práce v programu Matlab, které jistě využiji v dalším studiu a popřípadě v budoucí praxi.

Programy pro výpočet, které jsme použili, nejsou úplné a z pohledu programátora je lze určitě optimalizovat tak, aby výpočet pro velké sítě nebyl zbytečně časově náročný. Optimalizace využitých programů ale nebyla cílem naší práce.

Příloha 1

Program výpočtu nejkratší cesty v programu Matlab.

```
function [d,hranynaceste,uzly]=cesta2(n,ukaz,nasled,hrany,delky,...
                                      ...,start,cil)

%Program vrací délku nejkratší cesty mezi uzly start a cil a případně po
%odpoznamkování od řádku c.53 i nejkratší cestu ve formě uzlu a hran.
%n je počet vrcholů v grafu
%pole ukaz a nasled slouží k zadání grafu pomocí seznamu sousedů
%pole hrany obsahuje čísla hran
%pole delky obsahuje délky hran
%start a cíl jsou čísla startovního a cílového uzlu

%stav[i]...zpracovanost  vrcholu
%vzdal[i]...zatím nalezená nejkratší cesta

vzdal=ones(1,n)*inf;  %všem vrcholům přiřadíme nekonečnou vzdálenost...
stav=zeros(1,n);      %...a neprošetřený stav
vzdal(start)=0;      %vzdálenost výchozího vrcholu změníme na nulu
j=start;
nejpred=zeros(1,n); %pole nejpred bude na i-té pozici obsahovat číslo uzlu,
                    %který navštívíme na nejkratší cestě těsně před uzlem i
                    %stejnou roli bude mít i pole "nejpredhrana", které
                    %pouze místo čísel uzlů bude obsahovat čísla hran
nejpredhrana=zeros(1,n);
%budeme hledat nejkratší cestu, dokud neprošetříme cil
while stav(cil)==0 %any(stav==0) dokud neprošetříme všechny vrcholy
    stav(j)=1; %nalezení nejbližšího nezpracovaného vrcholu

    k=ukaz(j); %odečtením čísel k a l získáme stupeň prošetřovaného vrcholu,
    l=ukaz(j+1); %neboli počet jeho následovníků
    %pro každého neprošetřeného následovníka právě prošetřovaného vrcholu,
    %zjistíme nejkratší cestu do něj vedoucí
    for r=1:(l-k) %r=stupen vrcholu
        if stav(nasled(ukaz(j)+r-1))==0;
            if (vzdal(j)+ delky((ukaz(j)+r-1)))<vzdal(nasled(ukaz(j)+r-1));
                %porovnávání délek nových cest už s nalezenými vzdálenostmi
                vzdal(nasled(ukaz(j)+r-1))=vzdal(j)+delky((ukaz(j)+r-1));
                nejpred(nasled(ukaz(j)+r-1))=j;
                nejpredhrana(nasled(ukaz(j)+r-1))=hrany(ukaz(j)+r-1);
            end;
```

```
        end;
    end;
    min=0;
    %nalezení dalšího nejbližšího nezpracovaného vrcholu
    for i=1:n
        if stav(i)==0 && vzdal(i)>0;
            if (min==0)
                min=i;
            elseif (vzdal(i)<vzdal(min))
                min=i;
            end
        end
    end
   j=min;
end;
%pokud chceme znát i konkrétní cestu do cíle, odpoznámkujeme následující
%cest=zeros(1,n);
%pom=n;
%s=cil;
%cest(pom)=s;     %cest=(0 0 0 ... cil)
%while nejpred(s)~=0;
   % s=nejpred(s);
   % pom=pom-1;
   % cest(pom)=s;
%end;
d=vzdal(cil);
%vzdalenosti=vzdal;
%uzly=nonzeros(cest);
%for t=2:length(uzly)
%hranynaceste(t-1)=nejpredhrana(uzly(t));
%end;
```

Příloha 2

Program výpočtu důležitosti úseku v programu Matlab.

```
function[I]=dleprujezdu_opr(n,ukaz,nasled,hrany,delky,pocet_obyv)
% počítání zranitelnosti jednotlivých hran podle rovnice (2) str.25, kdy za
% váhovou funkci volíme odhad průjezdu uvedený ve vztahu (6) na str.32
delkya=delky; %načtené délky si uložíme, neboť u nich bude docházet ke změně
I=zeros(1,max(hrany)); %vektor I bude na j-té pozici obsahovat hodnotu
                       %důležitosti  j-té hrany
s=0;
%v následujících dvou cyklech napočítáme hodnoty váhové funkce všech
%dvojicí vrcholů, dle průjezdu
for i=1:n
        for j=i+1:n
            [r]=cesta2_opr(n,ukaz,nasled,hrany,delky,i,j);
            s=s+((pocet_obyv(i)+pocet_obyv(j))/r);
        end
end;
for a=1:n
    for b=a+1:n;
    [r]=cesta2_opr(n,ukaz,nasled,hrany,delky,a,b);
    w(a,b)=(sum(pocet_obyv))*(((pocet_obyv(a)+pocet_obyv(b))/r)/(2*s));
    w(b,a)=w(a,b);
    end;
end;
%Pokud bereme všechny váhy rovny 1, odpoznámkujeme následující a
%zapoznámkujeme předchozí cykly
%w=ones(n,n);

%ověření, že součet matice se rovna součtu všech obyvatel
%sum(pocet_obyv);
%P=sum(sum(w));

%výpočet důležitosti všech hran
for j=1:length(hrany)
    i=hrany(j); %očíslování hrany
    V=0;
    F=0;
    for start=1:n
      for cil=1:n
        if start~=cil;
           [d]=cesta2_opr(n,ukaz,nasled,hrany,delky,start,cil);
```

```
            delky(j)=inf;
            [c]=cesta2_opr(n,ukaz,nasled,hrany,delky,start,cil);
            V=V+w(start,cil);
            F=F+(w(start,cil)*(c-d));
            delky(j)=delkya(j);
          end;
        end;
      end;
      I(i)=I(i)+F/V;
end;
```

Příloha 3

Program, který do grafu přidává možné hrany a počítá důležitosti úseků v novém (vylepšeném) grafu.

```
function [H]=finale_opr(ukaz,nasled,hrany,souradnice,vzdal,pocet_obyv)
%Program finále je závěrečným programem pro výpočet důležitosti úseků ve
%vylepšené síti. Ze zadané sítě zjistí, které hrany už existují a bude
%přidávat dosud neexistující hrany. Z těch postupně přidá pouze ty, které se
%neprotínají s nějakou stávající, a spočítá důležitosti hran.
%Výstupem programu je matice, která v prvních dvou sloupcích udává čísla
%uzlů, mezi kterými se přidala hrana, a v dalších sloupcích hodnoty
%důležitostí jednotlivých hran.
ukaza=ukaz; %načtená data si uložíme, neboť přidáváním hrany u nich bude
             docházet ke změně
nasleda=nasled;
hranya=hrany;
vzdala=vzdal;
n=length(ukaz)-1; %počet uzlů
M=zeros(max(hrany),2);%matice, která na j-tém řádku bude udávat,
                      které dva vrcholy hrana j spojuje
%v následujícím cyklu napočítáme matici M
for i=1:length(ukaz)-1
   for j=0:(ukaz(i+1)-1)-(ukaz(i));
    k=nasled(ukaz(i)+j);
    if i<k
    M(hrany(ukaz(i)+j),:)=[i,k];
    end;
   end;
end;

pom=1;
%budeme procházet všechny dvojice vrcholů a přidávat hrany
for a=1:n;
  for b=a+1:n;
        a
        b
        %zjistíme následovníky uzlu a
     for j=1:(ukaz(a+1))-(ukaz(a));
         k(j)=nasled(ukaz(a)+j-1); % seznam následovníků uzlu a
     end;
   %Zjistíme, zda mezi uzly a a b existuje hrana. Pokud ne, tak
   %testujeme, zda nová hrana mezi těmito uzly neprotne už existující hranu.
```

```
if (any(k==b)==0);
     P=0; %počet průsečíku
     for h=1:max(hrany); %počítání průsečíku se stávajícími hranami
     e=M(h,1); %krajní body hrany
     f=M(h,2);
     A=[souradnice(a,1)-souradnice(b,1),souradnice(f,1)-souradnice(e,1);
      souradnice(a,2)-souradnice(b,2),souradnice(f,2)-souradnice(e,2)];
     g=[souradnice(f,1)-souradnice(b,1);souradnice(f,2)-souradnice(b,2)];
     %ověříme podmínky pro existenci průsečíku
         if rank (A)~=1;
           if rank(A)==rank([A g]) %existuje průsečík
              l=A\g; %hodnoty parametru lambda, gama
                if (any(l<=1e-14) || any(l>=1-(1e-14)))==1;
                   P=P+0; %průsečík leží mimo graf
                else P=P+1;
                end;
           end;
         end;
     end;
     if P==0 %neexistuje průsečík
        %přidání hrany mezi a b, tedy přepsání původních polí dat
        for i=1:length(ukaz)-a
            ukaza(a+i)=ukaz(a+i)+1;
        end;
        for l=1:length(ukaz)-b
            ukaza(b+l)=ukaza(b+l)+1;
        end;
        for j=0:(length(nasled)-(ukaz(a+1)-1))
          if j==0
           nasleda(ukaz(a+1))=b;
           hranya(ukaz(a+1))=max(hrany)+1;
           vzdala(ukaz(a+1))=sqrt(((souradnice(a,1)-souradnice(b,1))^2)+
               +((souradnice(a,2)-souradnice(b,2))^2));
          else nasleda(ukaz(a+1)+j)=nasled(ukaz(a+1)+j-1);
           hranya(ukaz(a+1)+j)=hrany(ukaz(a+1)+j-1);
           vzdala(ukaz(a+1)+j)=vzdal(ukaz(a+1)+j-1);
          end;
        end;
        for m=0:(length(nasled)-(ukaz(b+1)-1))
         if m==0
          nasleda(ukaza(b+1)-1)=a;
          hranya(ukaza(b+1)-1)=max(hrany)+1;
          vzdala(ukaza(b+1)-1)=sqrt(((souradnice(a,1)-souradnice(b,1))^2)+
```

```
                    +((souradnice(a,2)-souradnice(b,2))^2));
                 else nasleda(ukaza(b+1)-1+m)=nasled(ukaz(b+1)+m-1);
                    hranya(ukaza(b+1)-1+m)=hrany(ukaz(b+1)+m-1);
                    vzdala(ukaza(b+1)-1+m)=vzdal(ukaz(b+1)+m-1);
                 end;
               end;%je přidaná obousměrná hrana a b
               %v novém grafu spočítáme důležitosti jednotlivých hran pomocí
               %funkce dleprujezdu_opr
               [I]=dleprujezdu_opr(n,ukaza,nasleda,hranya,vzdala,pocet_obyv);
               %výsledná matice N bude na řádcích obsahovat nejprve čísla
               %uzlů, mezi kterými se vytvořila hrana, a poté hodnoty
               %vektoru I
                N(pom,:)=[a,b,I];
                pom=pom+1;
            end;
           %přepsání na původní graf
            ukaza=ukaz; nasleda=nasled; hranya=hrany; vzdala=vzdal;
    end;
  end;
end;
%pro porovnání jednotlivých vylepšení hodnoty I zprůměrujeme
for i=1:size(N,1)
    H(i,:)=[mean(N(i,3:size(N,2)-1)) N(i,1) N(i,2)];
end
%pro přehlednost matici setřídíme
H=sortrows(H);
```

Příloha 4

Než přistoupíme ke spuštění programu, uveďme jak vypadají jeho vstupy. Pro malé, testovací sítě můžeme potřebná pole ukazatelů, následovníků, hran, souřadnic, vzdáleností a počtu obyvatel zadat ručně. Při reálných aplikacích se takto ovšem nepostupuje. Data do programu Matlab načítáme z jednoduchých textových souborů, ve kterých jsou zakódované všechny informace. Ukázka jednoho řádku takového souboru je následující.

```
2514A037;327  2514A038;102;2514A037  2514A038;118;9
 2514A036;217;2514A036  2514A037;140;8
 2514A092;561;2514A037  2514A092;2230;162
```

Řádek začíná kódem uzlu a počtem obyvatel v tomto uzlu. Následují všichni jeho následovníci s počty obyvatel a vzdáleností uvedenou nejdříve v metrech a poté v sekundách. Souřadnice uzlů máme v dalším textovém souboru, který na řádku obsahuje kód uzlu 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 a vyhodnocovat a ukládat informaci, kterou zjistil, pokud narazil na mezeru.
Nejprve tedy načteme data následujícím způsobem:

```
sit=importdata('místo, kde máme soubor uložen\zlin_cast1_trb.txt','\n');
[deko_uzly deko_hrany uk nasl vzdal hrany pocet_obyv]=NACTIDATA4(sit);
sit=importdata('místo, kde máme soubor uložen\cast1_uzly_trb.txt','\n');
souradnice=NACTISOURADNICE(sit,deko_uzly);
vzdalenosti_km=vzdal(1,:);
```

Program z přílohy 3 spustíme tímto příkazem:

```
[H]=final_opr(uk,nasl,hrany,souradnice,vzdalenosti_km,pocet_obyv)
```

Předchozí příkazy pro spuštění jsou obsaženy v přiloženém m-souboru pod názvem prikazy_pro_spusteni.m.

Součástí práce jsou i programy NACTISOURADNICE, NACTIDATA4 a TESTPOLE, které slouží k dekódování textových souborů, a které mi poskytl RNDr. Rostislav Vodák, Ph.D.

Literatura

[1] Abrahamsson, T., Characterization of vulnerability in the road system, Department of Infrastructure and Planning, KTH, Stockholm, 1997

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

[3] Černý, J., Základní grafové algoritmy [online], dostupné z: http://kam.mff.cuni.cz/~kuba/ka/min_cesta.pdf, [citováno 20.2.2013]

[4] Demografická ročenka krajů [online], dostupné z: http://www.czso.cz/csu/2012edicniplan.nsf/t/29003BA607/$File/40271200.pdf, [citováno 20.3.2013]

[5] D'Este, G.M., Taylor, M.A.P., Modelling network vulnerability at the level of the national strategic transport network, Journal of the Eastern Asia Society for Transportation Studies 4, 1-14, 2001

[6] D'Este, G.M., Taylor, M.A.P., Network vulnerability: An approach to reliability analysis at the level of national strategic transport networks, Proceedings of the 1st International Symposium on Transportation Network Reliability, Pergamon, Oxford, 2003

[7] Graph theory [online], dostupné z: http://en.wikipedia.org/wiki/Graph_theory, [citováno 15. 3. 2013].

[8] Hliněný, P., Základy teorie grafů [online], dostupné z: http://is.muni.cz/do/1499/el/estud/fi/js10/grafy/Grafy-text10.pdf, [citováno 20.2.2013]

[9] Holmgren, A., Vulnerability analysis of electrical power delivery networks, Licentiate thesis TRIDA-LWR LIC 2020, KTH, Stockholm, 2004

[10] Immers, L.H. a kol., Robustness and resilience of transportation networks, In: Proceesinds of the 9th international scientific conference Mobilita, Bratislava, Slovensko, 2004

[11] Jenelius, E., Petersen, T., Mattsson, L-G.: Importance and exposure in road network vulnerability analysis, Transport Research Part A 40, 2006, 537-560.

[12] Krzysztof R., Edsger Wybe Dijkstra (1930–2002): A Portrait of a Genius [online], dostupné z: http://homepages.cwi.nl/ãpt/ps/dijkstra.pdf, [citováno 15.2.2013]

[13] Laurentius, E., The vulnerability of the city, 1994

[14] Slivoně, M., Několik přístupů k identifikaci kriticky důležitých úseků na dopravní síti [online], dostupné z:
http://pernerscontacts.upce.cz/12_2008/slivone.pdf, [citováno 20.3.2013]

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

[16] http://www.rsd.cz/sdb_intranet/sdb/img/mapy/cr_ssud.jpg [online]