Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Diplomová práce

Robustní Steinerův strom

Bc. Tereza Vudarčíková

© 2020 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Tereza Vudarčíková

Kvantitativní metody v ekonomice
Systémové inženýrství

Název práce

Robustní Steinerův strom

Název anglicky

Robust steiner tree

Cíle práce

Cílem práce je vytvoření robustního ekvivalentu úlohy Steinerova stromu s využitím lineárního celočíselného programování.

Metodika

Nastudování a popis teoretických východisek z oblastí operačního výzkumu, teorie grafů, komponent souvislosti, Steinerova stromu, lineárního celočíselného programování, robustí optimalizace, p-np problémů a doplňků programu MS Excel.

Následná formulace postupu algoritmu a přechodu na robustní verzi bude předvedena na jednoduchém příkladě. Demonstrace výpočtu bude prováděna v programu MS Excel, včetně doplňku programu Open Solver.

Poté bude zpracována samotná případová studie a zaznamenány výstupy.

Závěr obsahuje shrnutí postupu a výsledků práce.

Doporučený rozsah práce

60-70

Klíčová slova

Steinerův strom, robustní optimalizace, lineární celočíselné programování, p-np problémy, open solver

---

Doporučené zdroje informací

DEMEL, Jiří. Teorie grafů. 2. přeprac. vyd. Praha: České vysoké učení technické, 1991.

DIESTEL, Reinhard. Graph theory. Fifth edition. Berlin: Springer, [2017]. Graduate texts in mathematics. ISBN 978-3-662-53621-6.

JABLONSKÝ, J. *Operační výzkum : kvantitativní modely pro ekonomické rozhodování.* Praha: Professional Publishing, 2007. ISBN 978-80-86946-44-3.

Krichen, Saoussen, and Jouhaina Chaouachi. Graph-Related Optimization and Decision Support Systems, John Wiley & Sons, Incorporated, 2014. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=1784143.

ŠUBRT, T. *Ekonomicko-matematické metody.* Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, s.r.o., 2015. ISBN 978-80-7380-563-0.

Předběžný termín obhajoby

2020/21 ZS – PEF (únor 2021)

Vedoucí práce

Ing. Robert Hlavatý, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 29. 10. 2020

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 5. 11. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 27. 11. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Robustní Steinerův strom" jsem vypracovala samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autorka uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušila autorská práva třetích osob.

V Praze dne 20.11.2020 _____

Poděkování

Ráda bych touto cestou poděkovala Ing. Robertovi Hlavatému, Ph.D., který téma nadnesl a celou prací mě provedl. Díky jeho návrhům byl sestaven nový způsob řešení Steinerova stromu, který byl propojen s robustním programováním, a tak vznikl úplně nový přístup. Také děkuji za velkou pomoc s výpočty a praktickou částí, jako i s administrativou a odevzdáváním práce jako takové.

Robustní Steinerův strom

Abstrakt

Diplomová práce na téma „Robustní Steinerův strom“ obsahuje spojení dvou problematik, které dle autorčina vědomí spojeny do této doby nikdy nebyly, a to robustní optimalizace a Steinerův strom. Toto propojení je provedeno za účelem praktičtější aplikace algoritmu Steinerova stromu na reálné problémy. Jeho limity jsou především v tom, že do nich nelze zahrnout chyby v procesech nebo neurčitost. Toto řeší robustní optimalizace, která je svou podstatou odolná právě na neurčitost. Aby bylo možné úlohy řešit, byl pro práci využit program Microsoft Excel, a především jeho doplněk OpenSolver. Tím byl řešen nejdříve problém Steinerova stromu, který byl pro potřeby Excelu formulován jako úloha celočíselného lineárního programování na základě teoretických podkladů. Poté byl model rozšířen na robustní verzi. Úvod diplomové práce obsahuje samotné představení práce a nového přístupu k řešení Steinerova stromu. Také jsou zde uvedeny praktické případy užití nového algoritmu. Teoretická část obsahuje podklady k práci, tedy teorii, jež byla v praktické části použita. Ve vlastní práci se nejdříve řeší problém Steinerova stromu, jeho přeformulování na úlohu celočíselného lineárního programování a následně jeho převedení na robustní verzi. Vše je počítáno modelem v program Microsoft Excel. Kompletní algoritmus je ukázán na malé ilustrativní úloze s několika scénáři, jež by mohly nastat. Následně je řešena případová studie. Potom jsou výsledky a nová metodika diskutovány. V závěru je shrnutí celé diplomové práce.

Klíčová slova: Steinerův strom, robustní optimalizace, lineární celočíselné programování, p-np problémy, open solver

Robust Steiner tree

Abstract

Diploma thesis “Robust Steiner tree” contain a combination of two problematics, which have never been connected by this time, due to author´s knowledge. It is robust optimalization and Steiner tree. This combination is made in order to apply a more efficient application of the Steiner tree algorithm for the real issues. Limitations of this algorithm are mainly that it can´t involve process errors and uncertainty. This limitation is solved by robust optimalization, which is built to be uncertainty resistant. For the solutions of the issues is used Microsoft Excel and especially an add-on called OpenSolver. At first was solved the problem of the Steiner tree, formulated like a problem of an integer linear programming because of the needs of Excel inputs. This was solved based on theoretical background. Then the model was extended to a robust version. The introduction of the diploma thesis contains the introduction of the thesis itself and a new approach to solving the Steiner tree. There are also practical cases of an application of the new algorithm. The theoretical part contains texts for the work, the theory that was used in the practical part. In the main part of the thesis, at first is solved Steiner tree problem, its reformulation into the problem of integer linear programming and then its conversion to a robust version. Everything is calculated by a model in Microsoft Excel. For purpose of the demonstration complete set of steps of the algorithm there is a small illustrative task with several scenarios that could occur. Subsequently, a case study is solved. Then the results and the new methodology are discussed. In the end there is a summary of the whole diploma thesis.

Keywords: Steiner tree, robust optimalization, integer linear programming, p-np problems, open solver

Obsah

Seznam obrázků

Seznam tabulek

2 Úvod

V této práci jsem řešila úplně nový přístup k řešení problému Steinerova stromu. Je to algoritmus, který je užitečný při hledání minimální kostry grafu za přidání Steinerova uzlu. V reálném světě jej lze aplikovat na různé úlohy od úloh spedice, různé vlakové sítě s překladišti i projektové řízení. Nicméně pro reálné problémy často nemůžeme předpokládat, že bude dostačovat ideální řešení, které Steinerův strom poskytuje. Proto se k jeho algoritmu přidává robustní optimalizace, přesněji se algoritmus převede na robustní verzi. Kombinace těchto dvou přístupů proto umožní řešit stále stejné problémy jako obyčejný Steinerův strom, ale díky robustnosti bere v potaz odchylky od ideálního stavu a jistou míru neurčitosti. Právě proto je mnohem více užitečný ve veškerých problémech, které zahrnují lidský faktor, protože ten bývá největším zdrojem nejistot v procesech.

V práci kombinuji teoretické poznatky spolu s vlastní invencí, jenž je potřeba v převodu Steinerova stromu na robustní verzi. Tento převod byl poměrně komplikovaný a velmi mi s ním pomohl vedoucí práce.

Veškeré výpočty bylo nutné provádět pomocí počítače, ručně je to naprosto nereálné a nepraktické. Vyplnění modelu do tabulek programu je metodická práce, náchylná na chyby z nepozornosti, zvláště pak, pracujeme-li na velké úloze.

3 Cíl práce a metodika

3.1 Cíl práce

Cílem této práce je vytvořit robustní algoritmus Steinerova stromu, který je využitelný na širokou škálu problémů především z reálného prostředí. Potom bude algoritmus předveden na malé úloze a na případové studii. Budou interpretovány výsledky a zhodnocen nový algoritmus robustního Steinerova stromu.

3.2 Metodika

Nejdříve jsou v první části popsána teoretická východiska jako podklad pro praktickou část práce. Velmi okrajově je shrnuto téma operačního výzkumu. Poté je již větší pozornost věnovaná teorii grafů, zvláště pak Steinerovu problému a Steinerovu stromu, který je klíčovým prvkem vlastní práce, stejně tak jeho formulace jako úlohy celočíselného lineárního programování. Tato úprava je nevyhnutelná, jelikož potřebujeme model řešit v programu Microsoft Excel skrze doplněk OpenSolver určený k řešení takovýchto úloh. Poté jsou v teoretické části vysvětleny problematika lineárního a celočíselného lineárního programování. Také je stručně vysvětlena robustní optimalizace, bez které by se vlastní část práce neobešla. Popíší se i problémy P-NP a nakonec sekce je několik informací o OpenSolveru, pro vlastní práci nepostradatelném výpočetním nástroji.

Vlastní práce obsahuje nejdříve popis postupu propojení Steinerova stromu a robustního přístupu obecněji. Poté je přímo v průběhu řešení malé ilustrativní úlohy popisováno, co a jak se právě řeší. Vzhledem k tomu, že model OpenSolveru potřebuje, aby byl vkládaný problém celočíselné povahy, tak v první řadě na malé úloze ukazujeme, jak se formuluje Steinerův strom jako úloha celočíselného lineárního programování. Tato formulace vychází z teoretických poznatků. Malou úlohu tedy naformulujeme jako celočíselný model, což přímo převádíme do tabulek Excelu, aby mohl být spuštěn model OpenSolveru. Jakmile je model vyplněn, je OpenSolver spuštěn, čímž se vyřeší úloha Steinerova stromu a zjistí se hodnota účelové funkce. Potom se model převede na robustní verzi, doplní se odchylky a rozšíří model OpenSolver a zase se spustí, čímž je robustně vyřešena malá úloha.

Poté jsou shromážděna data pro případovou studii. Je nutné znát ceny hran, topologii grafu, význačné uzly a Steinerovy uzly. Také musíme znát odchylky cen hran. Data na studii jsou

nalezena vyhledáváním na internetu. Máme-li všechna potřebná vstupní data, aplikujeme stejný postup jako v malé úloze. U této úlohy již nejsou popisovány jednotlivé kroky algoritmu, ale jsou prezentovány průběžné a koncové výsledky. Navíc je komentován výstup OpenSolveru.

Následně je nový algoritmus Steinerova robustního stromu diskutován a jsou rozebrány výsledky případové studie. Je řešeno užití, výhody a nevýhody nového přístupu. Jsou také nadneseny možné reálné problémy, které by tímto přístupem mohly být řešeny.

V závěru je zhodnocení a shrnutí diplomové práce, výsledků úloh, postřehy autorky a návrhy na zlepšení postupu algoritmu.

4 Teoretická východiska

4.1 Operační výzkum

Tato vědní disciplína má svůj počátek kolem 30. až 40. let 20. století, kdy se na jejím startu podíleli například vědci G.B.Danzig a L.Kantorovič. Velký rozvoj a využití zažil operační výzkum především pro vojenské účely, analýzy strategických a taktických operací, během druhé světové války a následně během 50. let v reakci na ekonomický rozvoj. (Jablonský, 2007)

Operační výzkum je soubor jednotlivých disciplín, které zkoumají různé rozhodovací problémy a v rámci systému jej využijeme v situacích, kdy je zapotřebí koordinovat průběh operací, nebo systém samotný analyzovat. Primární metodou operačního výzkumu je matematické modelování. Tento nástroj umožňuje celou řadu simulací, kterou na reálném systému provádět nelze. Přesto je to pouze zjednodušený obraz systému. (Jablonský, 2007)

4.1.1 Disciplíny operačního výzkumu

Operační výzkum zahrnuje větší množství rozlišných modelů také v závislosti na ekonomické oblasti, ze které daný problém vychází. Základními disciplínami jsou matematické programování, které se zabývá optimalizačními úlohami a obsahuje úlohy lineárního a nelineárního programování. Další disciplínou je vícekriteriální rozhodování, jenž se zabývá rozhodovacími úlohami s více kritérii. Podstatným odvětvím je teorie grafů, podrobněji popsaná v kapitole 4.4 Teorie grafů. Odvětví, které se zabývá řízením zásobovacího procesu a skladových zásob, je teorie zásob. Mezi dalšími disciplínami jsou teorie hromadné obsluhy, modely obnovy, markovské rozhodovací procesy, teorie her a simulace. Především díky simulacím je možné analyzovat komplikované systémy. (Jablonský, 2007)

4.2 Lineární programování

Rozhodovací situace jsou řešeny několika skupinami modelů. Nejpočetnější oblastí takovýchto modelů je taková, kterou lze formulovat za pomoci lineárního programování. Základními vlastnostmi modelu lineárního programování je fakt, že lineární funkce optimalizovaných proměnných zahrnují funkce omezujících podmínek i účelovou funkci. (Gros, 2003)

V lineárním programování se pro modely používá následující terminologie. K nalezení řešení rozhodovacího problému je zapotřebí splnit optimální úroveň vstupních veličin, které se

nazývají optimalizované proměnné. Jako příklad lze uvést množství naloženého tovaru, sestavy přepravních cest atd. (Gros, 2003)

Dále do modelu patří takzvané technické koeficienty, které jsou často v průběhu řešení neměnné. Tyto koeficienty reprezentují například dané spotřeby dopravních prostředků, nebo strojů výroby, úroky, daně. (Linda a Volek, 2009)

Na pravé straně modelu jsou kapacitní omezení, které mohou být tří typů. Mohou být maximalizační, například maximum zdrojů, které lze využít, nebo maximální produkce z výrobní linky za určitý interval.

Opačným typem jsou minimalizační omezení. Mohou to být různé požadavky na minimální úroveň prvků jako je produkce, nebo zisk, minimální množství přepravovaného zboží. Omezení na pravé straně může také plnit přesnou podmínku, kterou si lze představit například jako přesně dosaženou jakost produktu. Dále prvky jako jsou variabilní náklady, ceny produktů, různé kvalitativní požadavky atd, jsou v modelu reprezentovány oceněním proměnných v účelové funkci. (Linda a Volek, 2009)

Množina omezujících podmínek může obsahovat tyto druhy omezení. Omezení „méně rovno“ se používají především pro zanesení kapacity různých typů zdrojů, materiálových, finančních, časových nebo lidských, do modelu. Dalším typem omezení je rovnice. Je používaná v případech složitější výrobní strukturou pro vyvažování hmotných vazeb, je-li nutné dodržet poměry v množství určitých výrobků nebo materiálů, které jsou na sobě závislé. (Gros, 2003)

4.2.1 Úlohy lineárního programování

Linda a Volek (2009) píší, že se počátky lineárního programování objevily kolem třicátých a čtyřicátých let 20. století. U vzniku stál sovětský matematik a ekonom Leonid Vitaliyjevič Kantorovič, který poprvé formuloval pár z optimalizačních problémů právě ve tvaru příkladů lineárního programování. Také ve své práci navrhl metodu, kterou by příklady bylo možné řešit. Pár let poté matematik a fyzik F. L. Hitschcock posunul lineární programování o optimalizační úlohy směřujícími k dopravním úlohám. Podle Lindy a Volka (2009) se o největší rozvoj lineárního programování zasloužili G. B. Danzig, R. Hurwitz a T. S. Koopmans, kteří koncem padesátých let minulého století dali formu všeobecné úloze lineárního programování a sestavili simplexový algoritmus. (Linda a Volek, 2009)

4.2.2 Hledání optimálního řešení

Jedním z nejjednodušších způsobů, jak nalézt řešení úlohy LP je metoda grafická. Využívá analytické geometrie, kdy touto metodou můžeme řešit úlohy o nejvýše třech proměnných. Tato podmínka vyplývá z možnosti zobrazení v maximálně trojrozměrném prostoru. Grafická metoda se proto používá pro jednodušší úlohy vzhledem k tomu, že úlohy LP v reálných problémech mají často více než tři proměnné. Je ovšem velmi názorná. (Linda a Volek, 2009)

Univerzální algoritmus zvaný simplexová metoda, nebo simplex byl vyvinut za účelem řešení rozhodovacích problémů. Je založen na principu postupného vylepšování prvotního řešení. (Gros, 2003)

Duální algoritmus je založen na předpokladu, že ke každé primární úloze LP můžeme vytvořit duální úlohu. Proměnné a jejich výsledné velikosti duálních úloh představují velmi užitečné informace pro reálné problémy. (Gros, 2003)

4.3 Celočíselné lineární programování

Celočíselné lineární programování se používá v případě, když jsou v modelu proměnné, které se nemohou spojitě měnit, ale mohou nabývat pouze předem určených hodnot, nebo mohou nabývat pouze celých čísel. Například objednávaná balení produktů, které nelze prodat po kusech atd. Případem lineárního celočíselného programování je takový, ve kterém proměnné mohou nabývat pouze hodnot nula nebo jedna, a říká se mu binární úloha. Taková proměnná může vyjadřovat například přítomnost nebo nepřítomnost nějakého prvku. Tyto binární úlohy mohou řešit plánování výroby, alokace zaměstnanců na pracovní pozice, nebo také hledání itinerářů pro okružní dopravní úlohy, ale také alokace prostředků pro různá odvětví podnikání. (Sierksma a Zwols, 2015)

Podle Kubišové (2014) se úlohy lineárního programování, které jsou doplněné o podmínky celočíselnosti v současnosti poměrně často objevují. Jejich řešení bývá poměrně náročné a vzhledem k tomu, že simplexová metoda vyhledává řešení na množině všech nezáporných reálných čísel.

Modely celočíselného programování jsou tří typů. Prvním je skupina úloh, ve kterých je požadováno, aby veškeré proměnné modelu měly celočíselný tvar. Tato skupina úloh se podle

Kubišové (2014) nazývá ryze celočíselné úlohy LP. V další skupině úloh je požadavek na celočíselnost pouze u některých proměnných. Kubišová (2014) hovoří o této skupině jako o smíšeně celočíselných úlohách LP. Poslední skupinu tvoří problémy binárního programování. (Gros, 2003)

Přestože celočíselný charakter těchto úloh LP velmi omezuje množinu přípustných řešení, nalezení tohoto řešení ve skutečnosti jednodušší není a je i v současnosti předmětem výzkumů. Na druhou stranu již existuje velké množství nástrojů, které dokáže i tyto úlohy řešit. Například v programu Microsoft Excel je to nástroj Solver, nebo v české verzi Řešitel. (Gros, 2003)

Obecný model pro celočíselné LP a binární LP má tvar:

$$\frac{\max z = \sum_{j=1}^{n} c_j x_j}{\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i=1,2,\ldots,m} \tag{1}$$

- $x_j \geq 0 \quad \wedge \quad x_j$ jsou celočíselné pro $j = 1, 2, \ldots, n$, nebo (2)
- $x_j \geq 0 \quad \wedge \quad x_j$ jsou celočíselné pro vybraná j, nebo (3)
- $x_j = 0 \vee 1$ pro všechna $j = 1, 2, \ldots, n$. (4)

A v kanonické podobě se model celočíselného LP dá vyjádřit takto:

$$Z := \max \sum_{j \in J} c_j x_j \tag{5}$$

$$\sum_{j \in J} a_{ij} x_j \leq b_i \tag{6}$$

$$x_j \geq 0 \tag{7}$$

$$x_j \in Z_+ \tag{8}$$

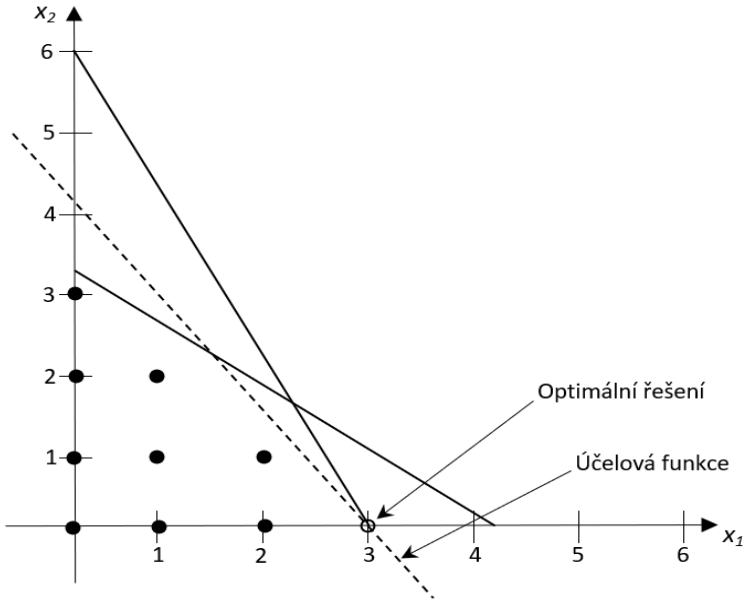
4.3.1 Metody hledání celočíselného řešení úlohy LP

Vzhledem k tomu, že počet přípustných řešení je obvykle poměrně velký, a i s výkonnou počítačovou technikou by prozkoumání této množiny byl nereálný v normálním čase. Proto bylo vyvinuto množství metod, které pomáhají najít řešení v rámci celočíselných čísel v reálním čase. Metody pro řešení úloh celočíselného LP jsou rozděleny do tří typů:

- Kombinatorické metody
- Heuristické metody
- Metody řezných nadrovin. (Linda a Volek, 2009)

Jednoduché úlohy lze také řešit graficky. Grafické řešení úloh celočíselného LP je podobné úlohám LP bez podmínky celočíselnosti, ovšem řešení se omezuje na množinu přípustných bodů, nikoliv na polygon přípustných řešení. V Gaussově rovině jsou zaneseny všechny podmínky matematického modelu úlohy LP, které odpovídají dvěma polorovinám. Hraniční přímky jsou zaneseny například v úsekovém tvaru, přičemž musí být dodržena podmínka nezápornosti. Množina přípustných řešení se nachází výhradně na bodech, které jsou průsečíky celočíselné sítě grafu. Dalším postupem je výběr přiměřené hodnoty účelové funkce, z čehož vyplyne daná izokvanta. Tuto izokvantu lze rovnoběžně přemístit směrem odpovídajícím zvětšování hodnoty účelové funkce, a to až do bodu, kde je nalezeno optimální řešení problému. (Kubišová, 2014)

Obrázek 1 Příklad grafického řešení celočíselné úlohy LP (Kubišová, 2014)



Zdroj: vlastní zpracování

Sierksma a Zwols (2015) popisují zaokrouhlovací postup, metodu, jejíž podstatou je jednoduchý přístup, a to ignorování podmínky celočíselnosti a řešení úlohy simplexovým algoritmem. Nalezené vrcholy v množině přípustných řešení modelu jsou obecnými body, nikoliv celočíselnými. Proto také simplexová metoda obvykle nenajde optimální řešení modelu celočíselného LP, ale může někdy zúžit oblast přípustných řešení natolik, že je výstup dostatečný. Paradoxně také řešení úlohy simplexovým algoritmem může vést takovým výsledkům, které jsou velmi vzdálené optimálnímu řešení problému celočíselného LP.

Kombinatorické metody

Podstatou kombinatorických metod je prozkoumávání přípustných řešení. Tyto metody pracují s postupy, které omezují počet možností, jenž je potřebné prozkoumávat. Jsou také poměrně náročné a zahrnují velké množství matematických operací, proti tomu poskytují přesné výsledky. Do kombinatorických metod patří metody větví a řezů, respektive hranic. (Linda a Volek, 2009)

Algoritmus větví a hranic je nejčastější metodou řešení modelů celočíselného a smíšeného celočíselného LP. Jeho podstatou je smršťování množiny přípustných oblastí na podstatně menší oblasti, což se nazývá větvení. Poté se počítají hranice na účelové funkci každé oblasti/submodelu, díky nimž se z uvažovaných oblastí mohou některé doposud přípustné oblasti vyloučit. Hranice získáme záměnou stávajících submodelů za jednodušší a tím, že vyřešíme jednodušší submodel, získáme hranici pro původní submodel, čímž jej ořežeme. Celý postup končí ve chvíli, kdy každý z jednodušších submodelů přinese nepřípustné řešení, nebo neobjeví lepší řešení, než bylo původní. Výstupem celého procesu je nalezení optimálního řešení, které je nejlepším řešením, objeveným v průběhu celého postupu algoritmu. (Sierksma a Zwols, 2015)

Heuristické metody

Tyto metody, někdy také zvané přibližné nebo aproximační, používají empiricko-iterativní algoritmus. Tyto metody jsou závislé také na typu matice, kterou je úloha specifikovaná. Ne všechny postupy jsou vhodné pro všechny úlohy, proto je nutné pečlivě vybírat postup pro každou řešenou úlohu. Heuristické metody pracují se zkušenostmi, díky nimž se vytváří postupy pro řešení rozdílných skupin problémů. Díky těmto metodám sice nenalezneme optimální řešení, ale alespoň řešení, která se optimálnímu blíží. Takovým řešením se říká suboptimální. Tato skupina metod hledání řešení úloh celočíselného LP je poměrně jednoduchá, ale jejich výhoda se ukáže až při řešení rozsáhlejších problémů. (Gros, 2003)

Sierksma a Zwols (2015) popisují přibližnou metodu, jejíž podstatou je jednoduchý přístup, a to ignorování podmínky celočíselnosti a řešení úlohy simplexovým algoritmem. Nalezené vrcholy v množině přípustných řešení modelu ale jsou obecnými body, nikoliv celočíselnými. Proto také simplexová metoda obvykle nenajde optimální řešení modelu celočíselného LP, ale někdy může zúžit oblast přípustných řešení natolik, že je výstup dostatečný. Toto řešení se

nazývá suboptimální. Paradoxně také řešení úlohy simplexovým algoritmem může vést takovým výsledkům, které jsou velmi vzdálené optimálnímu řešení problému celočíselného LP. Proto je nutné vhodně zvolit metodu, kterou konkrétní úlohy řešit.

Metody řezných nadrovin

Podstatou této metody je zmenšování množiny přípustných řešení až k nalezení optimálního řešení. Toto postupné ořezávání vede k tomu, že je optimální řešení úlohy celočíselného LP posunuto do hraničního bodu, nebo na hranu množiny přípustných řešení, přičemž z bodů na hraně lze optimální řešení vybrat některým ze známých algoritmů. Zkonstruovaná nadrovina, kterou ořezáváním dostaneme, separuje část řešení, která neobsahuje optimum. Různé algoritmy této skupiny metod se odlišují především způsobem, jakým konstruují řezné nadroviny. Pokud je potřebné sestavit více nadrovin k objevení celočíselného optimálního řešení, pak se může stát, že se naopak stávající problém zvětší a dochází k postupné, nepředvídatelné konvergenci. (Linda a Volek, 2009)

Nejznámější skupinou algoritmů sečných nadrovin jsou Gomoryho algoritmy. Nejprve se problém řeší obyčejným simplexovým algoritmem. Pokud tento postup nenalezne celočíselné řešení, je potřeba sestavit řeznou nadrovinu, která vychází z poslední simplexové tabulky, respektive jednoho z jejich funkčních omezení. Sestavená nadrovina se poté přidá k omezujícím podmínkám modelu. Původní simplexová tabulka se rozšíří o řádek a sloupec. Tím je porušena primární přípustnost řešení, a proto se problém dále musí řešit duálním simplexovým algoritmem. (Sierksma a Zwols, 2015)

4.4 Teorie grafů

Základním prvkem je graf, který slouží k popisu některých rozhodovacích problémů. Tento graf je definován množinou, která se skládá z podmnožin hran a vrcholů. Podmnožina vrcholů často udává prvky systému, například města v oblasti. Podmnožina hran pak popisuje vztahy mezi jednotlivými hranami. Jako celek může graf popisovat například reálnou dopravní síť, nebo strukturu podniku a procesy v něm probíhající. Zvláštním typem je síťový graf, který je určen k analýze projektů. (Fábry, 2011)

Z definice grafu je patrné, že nejsou nastavená pravidla pro to, kde jsou umístěny jednotlivé vrcholy, jaká je jejich poloha. Není definované ani to, jakého tvaru jsou hrany, jak jsou dlouhé, nebo zda leží podmnožina hran a podmnožina vrcholů v jedné rovině. Na druhou stranu existují

pro tyto podmnožiny pravidla, která říkají, že hrana nesmí projít sama sebou a také není možné, aby jedna hrana procházela více než dvěma vrcholy. (Bin a Zhongyi, 2010)

4.4.1 Základní pojmy teorie grafů (Fábry, 2011)

Vrchol, nebo také uzel je obvykle reprezentován kružnicí s vepsaným jedinečným identifikátorem v rámci konkrétního grafu. Mohou být popsány řadou čísel, nebo písmen. Jednotlivé uzly grafu propojují hrany, nebo cesty, které znázorňují vztahy a závislosti mezi uzly. Hrana má na obou koncích vrchol. Také můžeme hranu označit jako orientovanou, nebo neorientovanou. Pokud je hrana orientovaná, je na jednom z jejich konců šipka, která znázorňuje jediný možný směr přechodu po této hraně. Je-li hrana znázorněna pouze čarou bez šipky, je neorientovaná. Graf, který obsahuje pouze neorientované hrany, je neorientovaný. A naopak obsahuje-li pouze orientované hrany, celý graf je orientovaný. Pokud je potřeba popsat postup mezi dvěma vrcholy grafu, nazýváme tuto posloupnost cesta. (Fábry, 2011)

Sled

Sled je posloupností vrcholů a hran, který může být orientovaný, nebo neorientovaný, podle typu hran grafu. Sled je obvykle popisován počátečním a koncovým uzlem, také říkáme, že vede z vrcholu v_0 do vrcholu v_k. Tyto vrcholy spojuje, přičemž hrany a vrcholy na sebe vzájemně navazují. (Demel, 2015)

Typem sledu je i takový, který neobsahuje jedinou hranu a obsahuje pouze jeden uzel. Takovému sledu se říká triviální a lze jej označit za orientovaný i neorientovaný graf. (Demel, 2015)

Tah

Tah je typem sledu, ve kterém se neopakuje žádná hrana. Cestou nazýváme takový sled, který obsahuje každý uzel pouze jednou. Oba typy sledu mohou jak orientované, tak neorientované. (Demel, 2015)

Uzavřený sled

Začíná-li posloupnost ve vrcholu, ve kterém i končí, nazýváme ji kružnicí (také cyklem nebo uzavřenou cestou) uvádí Fábry (2011). Demel (2015) uvádí, že uzavřený sled (cesta) musí splňovat podmínky neopakování hran ani vrcholů, mimo počátečního, respektive koncového

vrcholu. Pokud je uzavřený sled neorientovaný, pak jej nazýváme kružnicí a pokud je orientovaný, je to cyklus. Cyklus můžeme také nazvat kružnicí, ale opačně to nelze.

Kvantifikace modelů teorie grafů

Pokud chceme docílit u modelů kvantitativní podobu, je nutné grafy ohodnotit. Rozlišujeme dva typy ohodnocení modelů, a to hranově a uzlově ohodnocený. Je-li graf hranově ohodnocený, každé hraně náleží určitá hodnota, cena. Uzlově ohodnocený graf je hodnocený naopak v uzlech. Graf může být hranově ohodnocený, uzlově ohodnocený, nebo hranově i uzlově zároveň. (Šubrt, 2015)

4.4.2 Typy grafů

Graf, který neobsahuje žádnou hranu, nazýváme diskrétním a může být jak orientovaný, tak neorientovaný. Prostý graf je takový, jehož hrany splňují podmínku násobnosti hran nejvýše rovno jedné. Často je pojmem graf myšlen právě prostý graf. Pokud je v grafu alespoň jedna hrana s násobností větší než jedna, nazýváme jej multigrafem. Multigraf může vzniknout i úpravou prostého orientovaného grafu, respektive jeho symetrizací. (Demel, 2015)

Souvislým grafem je nazýván takový graf, ve kterém existuje pro každou dvojici vrcholů minimálně jedna cesta, tah. (Fábry, 2011)

Strom

Máme-li graf, který neobsahuje žádný cyklus a je souvislý a neorientovaný, nazýváme jej strom. Základní podmínkou stromu je existence právě jedné cesty mezi každou dvojicí vrcholů grafu. (Fábry, 2011)

Demel (2011) uvádí pojem kořenový strom jako speciální typ stromu, který je navíc orientovaný. Nejčastěji udává aplikaci tohoto typu grafu v popisech hierarchických struktur. Jeho podstata spočívá v existenci kořenu, tedy význačného vrcholu, do kterého nevedou žádné hrany. Jiným názvem pro kořenový strom je větvení.

4.4.3 Komponenty souvislosti

Existuje-li spojení každé dvojice vrcholů grafu neorientovanou cestou, lze graf nazvat souvislým. Komponentami souvislosti nazýváme všechny podgrafy H grafu G. Graf G je souvislý a není součástí rozsáhlejšího souvislého podgrafu, je maximální. (Demel, 2015)

Postup pro hledání komponent souvislosti grafu je následující. Zvolíme si kterýkoliv uzel a libovolným algoritmem pro hledání neorientovaných cest (značkování vrcholů, prohledávání grafů do šířky, prohledávání grafů do hloubky) vyhledáme množinu, která obsahuje uzly, jež jsou neorientovaně dostupné ze zvoleného uzlu. Výsledný podgraf je komponentou souvislosti. Pokud takto nalezená množina neobsahuje veškeré uzly grafu, zvolíme další uzel mimo již nalezený podgraf a postup opakujeme do okamžiku, kdy v grafu neexistují žádné uzly nezařazené v některém z komponent souvislosti. (Demel, 2015)

Stromy jsou grafy, které neobsahují kružnice a jsou souvislé. Navíc jsou komponentami souvislosti lesa. (Demel, 2015)

Podgraf, který je tvořen všemi uzly původního grafu a je také grafem typu strom, se nazývá kostra grafu. (Fábry, 2011)

4.4.4 Minimální kostra

Kostru grafu lze nazvat minimální, pokud má nejnižší cenu. Součet cen hran této kostry je nejnižší v porovnání se všemi ostatními možnými kostrami grafu. (Demel, 2015)

Postup hledání minimální kostry v souvislém grafu, který má ohodnocené hrany, je následující. Existuje množství algoritmů pro výpočet minimální kostry, od postupu, kdy je kostra zřejmá okamžitě, po složité algoritmy, které jsou určené i pro hledání koster složitých grafů. (Demel, 2015)

Jedním z algoritmů hledajících minimální kostru je hladový algoritmus, známý také pod názvem Kruskalův (sestaven roku 1956). Pro postup tohoto algoritmu si musíme seřadit ohodnocené hrany grafu od nejlevnější po nejdražší. Poté z tohoto pořadí vytřídíme a vybereme hrany, které by v grafu nevytvořily kružnici. Vybrané hrany tvoří minimální kostru. (Demel, 2015)

Ještě starším je Jarníkův-Primův algoritmus z roku 1930. Na startu algoritmu je zvolen libovolný vrchol v. Potom vybíráme takovou hranu e, která splňuje podmínku: hrana e spojuje dvojici komponent A lesa L a minimálně pro jednu z nich lze vyvodit, že má původní hrana e nejmenší hodnotu v porovnání s hodnotami hran množiny dvojice komponent. Jako komponentu volíme pokaždé tu, která zahrnuje vrchol v. V průběhu výpočtu má les L jako komponenty souvislosti právě jeden strom a izolované vrcholy. Strom se zároveň rozrůstá přidáváním hran, jenž jej propojují za nejnižší cenu s dosud nezahrnutými izolovanými vrcholy. (Demel, 2015)

Nejstarším algoritmem pro výpočet minimální kostry je Borůvkův algoritmus, který pochází již z roku 1926. Je použitelný pouze za předpokladu, že jsou všechny ceny hran navzájem různé. Postup tohoto algoritmu je velmi rychlý, jelikož každý jeho krok redukuje počet komponent souvislosti grafu nejméně o polovinu. Právě tento algoritmus je podkladem pro nejrychlejší novodobé algoritmy výpočtů minimálních koster. (Demel, 2015)

4.4.5 Stupně souvislosti

V řadě případů lze rozlišovat méně nebo více souvislé grafy. Pokud bychom jako příklad uvažovali povodně, zajímá nás, na kolika místech bude trasa z jednoho bodu do druhého přerušena, nebo zdali budou přerušeny i ostatní alternativní trasy způsobem, že bude ochromeno spojení mezi těmito dvěma body.

Graf můžeme nazvat silně souvislým v případě, je-li orientovaný a každé jeho dva uzly lze propojit orientovanou cestou, a zároveň existuje mezi stejnou dvojicí uzlů i orientovaná cesta opačného směru. (Demel, 2015)

Hranový stupeň souvislosti grafu

Popisuje nejnižší počet hran, při jejichž odebrání dojde tomu, že se graf stane nesouvislým. Hranový stupeň souvislosti je takto popsán pro grafy, které mají dva a více vrcholů. V případě, že má graf pouze jeden vrchol, jeho hranový stupeň souvislosti je nula. Pro hranový stupeň souvislosti zároveň platí, že nesmí být vyšší, než je stupeň libovolného uzlu grafu.

Typ spojení dvou uzlů se může nazývat také mostem za předpokladu, že odstraněním této vazby se navýší počet komponent souvislosti grafu. (Demel, 2015)

Vrcholový stupeň souvislosti grafu

Stejně jako hranový stupeň řeší minimální počet prvků, v tomto případě vrcholů, které v případě odstranění z grafu způsobí jeho rozpad na graf nesouvislý. Takto lze definovat vrcholový stupeň souvislosti pro většinu grafů s výjimkou grafů úplných.

Artikulace je typ vrcholu, při jehož odebrání z grafu se zvýší počet komponent souvislosti grafu. (Demel, 2015)

4.5 Steinerův problém

Známe několik Steinerových problémů závislých na typu grafů, na kterých jsou popsané. Prvním je problém Euklidovského Steinerova stromu. Tento graf je souvislý a jeho hrany splňují podmínku, že vzdálenost mezi libovolnou dvojicí hran je euklidovská. Dalším ze Steinerových problémů je lineární Steinerův strom, který je také souvislý, ale rozdíl je v tom, že vzdálenost mezi libovolnou dvojicí jeho hran je lineární. Posledním problémem je Steinerův strom. (Krichen a Chaouachi, 2014) Cieslik (2001) říká, že řešení Steinerových problémů je primárně závislé na formě, jakou jsou popisovány vzdálenosti v prostoru.

Prömel a Steger (2002) píší, že historicky má položit základy tohoto problému francouzský matematik Pierre de Fermat, který se zabýval myšlenkou tří bodů v rovině, mezi které umístí další bod. Je-li tento přidaný bod umístěn vevnitř polygonu ohraničeného původními třemi body a kostra grafu bude tímto bodem procházet, bude suma vzdáleností mezi body kratší, než kdyby se uvažovala kostra po obvodu (tedy bez přidaného bodu). Na jeho myšlenku navázal italský fyzik Evangelista Torricelli, který navrhl geometrické řešení Fermatova problému. Zobecnil problém na *n* bodů, místo původních tří.

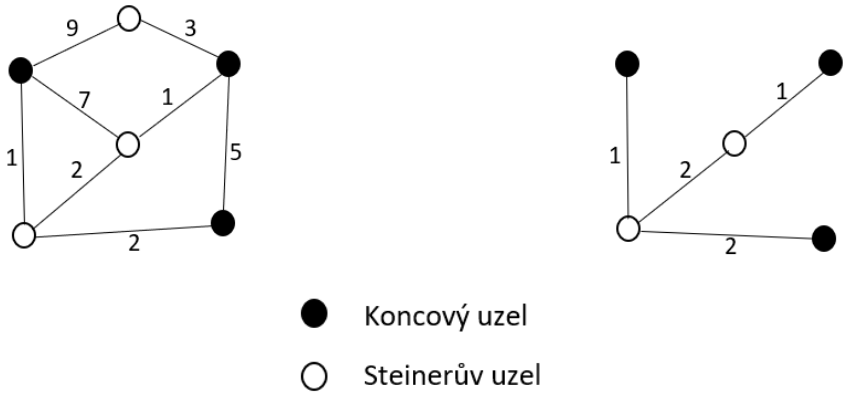
Na původní Fermatovu myšlenku a Torricelliho geometrický rozvoj se zaměřovalo větší množství vědců. Jedním z nich byl Jacob Steiner. (Prömel a Steger, 2002) Švýcarský matematik Jacob Steiner (narozen 1796), který je jedním ze zakladatelů syntetické a projektivní geometrie, je tedy autorem Steinerova problému. Základním problémem je hledání nejkratšího nebo nejlevnějšího možného propojení/cesty mezi vrcholy grafu. (Du, Smith a Rubinstein, 2000)

4.5.1 Steinerův strom

Je to problém, který stejně jako jiné algoritmy hledá minimální kostru grafu. S tím rozdílem, že k tomu využívá přidaných Steinerových uzlů. Výsledná podoba Steinerova stromu také závisí na vybraném optimalizačním algoritmu. (Prömel a Steger, 2002)

Tento problém popisujeme na ohodnoceném grafu, kde by všechny koncové (nebo také podstatné) uzly měly být propojeny. Problém Steinerova stromu spočívá v hledání takového stromu, který by obsahoval všechny koncové (podstatné) uzly za případného použití Steinerových uzlů, a to vše s dosažením co nejmenší celkové ceny. Tedy hledá podgraf o nejkratší možné délce za splnění podmínek minimální kostry. V Euklidovském prostoru lze říct, že problém Steinerova stromu je NP-složitým problémem. (Krichen a Chaouachi, 2014)

Obrázek 2 Problém Steinerova stromu



Zdroj: vlastní zpracování

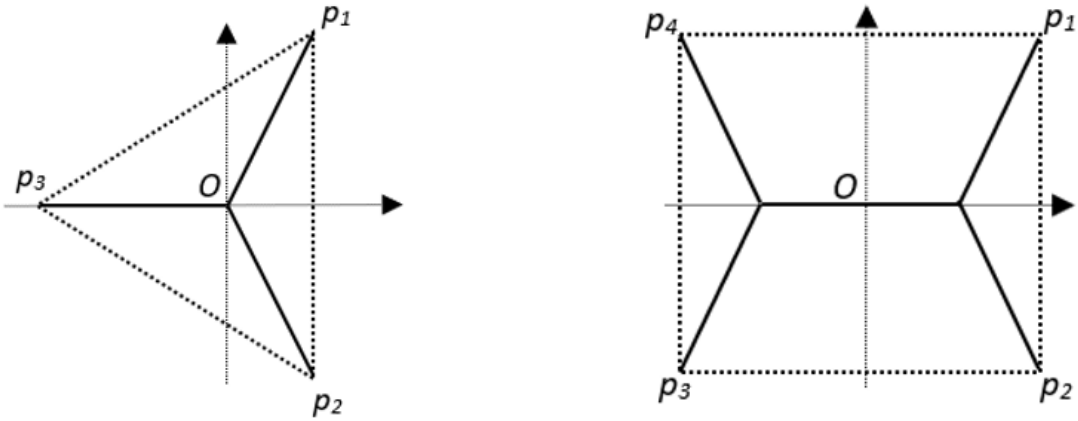
Steinerův uzel je takový uzel, který může být v rámci potřeby do grafu přidán. Není určeno, se kterým dalším uzlem (i Steinerovým) musí tvořit hranu. Můžeme si jej představit jako křižovatku mezi sklady, nebo rozvodnu obslužných sítí v zástavbě. Takový uzel má jediný účel, a to zmenšit minimální kostru grafu. (Krichen a Chaouachi, 2014)

Topologie tohoto grafu zobrazuje cestu koncovými uzly, případně Steinerovými uzly. Umístění jednotlivých uzlů v prostoru nemusí být přesně známé, podstatné je samotné uspořádání a ohodnocení. Koncové uzly mohou být stupně jedna a výše, Steinerovy uzly oproti tomu musí být stupně nejméně dva. Pokud uvažujeme, že Steinerův uzel má snižovat cenu minimální

kostry grafu a zároveň jím kostra nutně procházet nemusí, je tato podmínka logická. (Laarhoven, 2010)

Marchese a Massaccesi (2014) popisují pár velmi známých příkladů *Obrázek 2*, které znázorňují problém Steinerova stromu, které znázorňují hledání nejkratší cesty.

Obrázek 3 Řešení pro vrcholy trojúhelníku a čtverce



Zdroj: vlastní zpracování

Cieslik (2001) vysvětluje výraz Steinerův poměr jako vztah mezi délkou MST (minimum spanning tree/minimální kostra grafu) a SMT (Steiner minimum tree/Steinerův minimální strom) v tom samém grafu. Matematicky se tento vztah vyjádří jako délka SMT ku délce MST. Účelem řešení minimálního Steinerova stromu v grafu je zmenšení minimální kostry grafu. Z toho vyplývá, že minimální kostra grafu bude vždy delší než minimální Steinerův strom.

4.5.2 Formulace Steinerova stromu jako model celočíselného programování v grafech (Diané a Plesník, 1993)

Je dán graf G s těmito vlastnostmi: je spojitý, neorientovaný, neobsahuje smyčky nebo cykly ani multihrany a jeho hrany mají kladnou cenu. Pro tento graf existuje množina význačných uzlů označená jako $Z \subseteq V(G)$. Podstatou Steinerova problému je hledání minimální ceny kostry Z v G v grafech. Poté máme ještě množinu hran $E(G)$, která spolu s množinou $V(G)$ tvoří graf nebo podgraf G. Jejich kardinalita je následující. Nechť $n := |V(G)|$, $m := |E(G)|$ a $p := |Z|$. Cena podgrafu je určena součtem cen všech jeho hran, jež jsou značeny jako c_{ij}.

Jakýkoliv problém celočíselného lineárního programování může být relaxován od celočíselnosti proměnných a poté řešen jako problém lineárního programování, například simplexovým algoritmem. Pokud některou z metod řešení LP získáme hodnotu účelové funkce, představuje tato hodnota dolní odhad minimální ceny Steinerova stromu. V tomto modelu značíme počet proměnných jako α a počet podmínek jako β.

Problém celočíselného LP z hlediska rozměrů úlohy je definován takto:

$\alpha = 2m + n - 1$ proměnné ($2m$ binární a $n - 1$ celočíselných proměnných) a

$\beta = 4m + 3n + p - 4$ podmínky.

Postup této formulace je následující. Nejdříve si vybereme libovolný uzel $v_0 \in Z$. Poté nahradíme každou hranu ij z G dvěma opačně orientovanými hranami ji a ij každá o ceně c_{ij} a zároveň vyloučíme hrany jež vstupují do námi vybraného uzlu v_0. Nově vzniklý podgraf se značí $\vec{G}$. Máme kostru T ze Z v grafu G a všechny hrany z T orientujeme tak, aby uzel v_0 byl kořenem nového orientovaného stromu $\vec{T}$, který je podgrafem $\vec{G}$.

Předpokládáme, že $V(G) = \{1, 2, \dots, n\}$, $Z = \{1, 2, \dots, p\}$ a $v_0 = 1$. Dále předpokládáme $V^-(j) := \{i \in V(\vec{G}) | ij \in E(\vec{G})\}$. . Každé hraně ij přiřadíme proměnné $xij \in \{0, 1\}$. Každou hranu z podgrafu $\vec{G}$ nazýváme 1-hranou, nebo 0-hranou, zapsané jako $x_{ij} = 1$ a $x_{ij} = 0$. Hrana zapsaná jako $x_{ij} = 0$ patří výhradně do množiny $\vec{T}$ a potom do každého $j \in V(\vec{T}) - \{1\}$ vstupuje právě jedna 1-hrana. Všechny uzly $j \in V(\vec{G})$ jsou určené proměnnou u_j, která reprezentuje vzdálenost mezi j a kořenem 1 náležícím $\vec{T}$ za předpokladu že $j \in V(\vec{T})$. Proměnná u_j udává kolik je hran v jedinečné orientované cestě $1 - j$. Položme $u_j := -1$ v případě, že j nepřísluší množině $\vec{T}$ a poté platí, že jsou všechny $u_j \leq n - 1$. Model má následující podobu:

minimalizace $\sum_{ij \in E(\vec{G})} c_{ij} x_{ij}$ (9)

vyhovující

$\sum_{i \in V^-(j)} x_{ij} \leq 1$ $\forall j \in V(\vec{G}) - \{1\}$ (10)

$n \sum_{i \in V^-(j)} x_{ij} \geq u_j + 1$ $\forall j \in V(\vec{G}) - \{1\}$ (11)

$(n + 1) \sum_{i \in V^-(j)} x_{ij} \leq n(u_j + 1)$ $\forall j \in V(\vec{G}) - \{1\}$ (12)

$1 - n(1 - x_{ij}) \leq u_j - u_t \leq 1 + n(1 - x_{ij})$ $\forall ij \in E(\vec{G})$ (13)

$$x_{ij} \in \{0,1\} \qquad \forall ij \in E(\vec{G}) \qquad (14)$$

$$u_1 = 0, \qquad u_i \geq 0 \qquad \forall i \in Z - \{1\} \qquad (15)$$

Označme $\vec{H}$ jako podgraf grafu $\vec{G}$, jenž je tvořen 1-hranami a prozkoumejme jeho vlastnosti. Dále víme, že žádná 1-hrana z $\vec{H}$ nevstupuje do vrcholu 1. Podle podmínek výše platí, že nejvíce jedna 1-hrana vstupuje do jakéhokoliv uzlu. Do každého Z-uzlu, pro který platí $j \neq 1$, vchází 1-hrana ij. Poté lze říct, že $u_j \geq 1$ a $u_i = u_j - 1$, což lze aplikovat i pro i místo j, až na to, že platí $u_i = 0$. Potom lze prokázat, že každému Z-uzlu j náleží uzel i a platí $u_i = 0$ a orientovaná cesta $i - j$ sestávající z 1-hran. Další z podmínek říká, že žádná 1-hrana nemůže vstoupit do uzlu i, ale zároveň pokud se $i \neq 1$, pak do uzlu i 1-hrana vstoupit musí a z toho vyplývá že $i = 1$. Lze tedy říct, že všechny Z-uzly jsou dosažitelné z uzlu 1 po cestě sestávající z 1-hran.

Z podmínek výše také vyplývá, že jsou vyloučeny všechny koncové uzly, jež nejsou ze Z-uzlů a $\vec{H}$ musí být minimální kostrou Z.

Hodnoty proměnných u_j jsou omezeny na reálné s podmínkou $-1 \leq u_j \leq n-1$ nebo může být požadována jejich celočíselnost a z modelu v některých případech doopravdy vystupují pouze celá čísla. Pokud $u_j > -1$ je minimální neceločíselná hodnota, pak bychom dospěli ke sporu, kde by bylo další $u_i = u_j - 1$.

Tento model lze také kompletně převést na binární. Za podmínky $-1 \leq u_j \leq n-1$ lze položit $\overline{u_j} := u_j + 1$ ($j \in V(G) - Z$) a následně vyjádřit $\overline{u_j}$ binární proměnnou $y_{j0}, \ldots, y_{jt} \in \{0,1\}$: $\overline{u_j} = 2^0 y_{j0} + \cdots + 2^t y_{jt}$.
kde $t = [\log_2 n]$. Následně získáme model s nanejvýš $2m + (n-1)([\log_2 n] + 1)\, 0 - 1$ proměnných a nanejvýš $4m + 3n - 3$ omezení, přičemž m udává počet hran a n počet uzlů původního grafu G.

4.6 Robustní optimalizace

Počátky problému robustní optimalizace se datují do sedmdesátých let minulého století. Její rozvoj byl spjat s rozvojem algoritmů pro řešení optimalizačních úloh. Hlavními výzkumníky této oblasti byli například Charnes, Cooper a spol., dále Soyster, Ben-Tal a Nemirovski atp. (Bertsimas a Brown, 2009)

Základem robustní optimalizace je zjistit takové optimální řešení, které zachová proveditelnost pro většinu realizací i za předpokladu, že do modelu vstoupí nejistá data. To je primární odlišnost od obyčejného matematického modelu, který zná všechny vstupní parametry. Robustní přístup proti klasickému bere do úvahy nejednoznačná data, která mohou ovlivnit přípustnost výsledku modelu. (Bertsimas a Brown, 2009)

Robustní optimalizace využívá formulace cílového programování, což tvoří sady řešení, jež jsou významně méně náchylné na provedení modelových dat. Ve všeobecnosti má robustní optimalizace mnoho plusů proti stochastické lineární optimalizaci, které je alternativní metodologií. Přes všechny výhody, není bezvýhradně lepší, ale je určitě více všeobecně využitelná. (Kouvelis a Yu, 1997)

Je to metodologie, která se často zabývá „worst-case“ případy. Používá se pro případy, jejichž parametry pochází z odhadů, tudíž by tyto parametry mohly být poznamenány chybami odhadů. Také je vhodné použít robustní optimalizaci, pokud jsou v modelu nějaká podstatná omezení, jejichž splnění je podmínkou přípustnosti řešení. Nebo také v případě, že je účelová funkce, respektive optimální řešení, poměrně citlivá na chaos. (Yaman, Karasan a Pinar, 2001)

Obecně jsou v optimalizačních modelech známa všechna vstupní data, ovšem hodnoty některých koeficientů mohou nabývat neurčitých hodnot. Tato neurčitost pochází z toho, že známe minulý průběh nějakého problému a z něj vyvozujeme, jak by se podobná situace mohla vyvíjet v budoucnu, přičemž je to právě pouze neurčitý odhad. Neurčitá data ovšem nelze opomíjet v rámci zachování správnosti řešení problému. (Büsing and D'Andreagiovanni, 2013)

Může se stát, že se model stane neproveditelným, nebo přestane být optimální. Za předpokladu, že by se řešila úloha z reálného prostředí, jsou tyto dvě situace velmi nežádoucí. Proto se v minulosti objevilo několik metod, které neurčitost naopak zohledňují a umí s ní počítat. V základně předpokládají, že se neurčité hodnoty v modelu celočíselného programování objeví pouze v souvislosti s některými součiniteli a_{ij} a i kdyby se neurčitost objevila u zbylých součinitelů, existuje metoda, ve které je neurčitostí zasažena jen matice součinitelů. (Büsing and D'Andreagiovanni, 2013)

Jeden z nejčastěji používaných modelů určených pro robustní optimalizaci sestavili vědci Dmitris Bertsimas a Melvyn Sim (oba se zabývají operačním výzkumem). Jejich model stojí na několika předpokladech.

Prvním je, že si ke každému součiniteli, který je neurčitý, přiřadíme nějakou očekávanou hodnotu, kolik by mohla být jeho cena. Hodnotu můžeme odhadnout na základě sousedních součinitelů nebo například historických dat. Také se musí u tohoto součinitele určit, o kolik nejvíce se může zhoršit, tedy jeho maximální odchylka. Dále předpokládáme, že se hodnota neurčitého součinitele objeví v intervalu, tvořeném přičtením a odečtením maximální odchylky k odhadu součinitele.

Poté platí, že neurčití součinitelé jsou stochasticky náhodné proměnné určené vlastním rozpětím odchylky podle neurčeného symetrického rozdělení.

Poslední podmínkou je, že si pro každé omezení řešitel určí hodnotu, které odpovídá počtu součinitelů, které se v ten samý okamžik, respektive průchod výpočtem, zhorší o vybranou odchylku. (D'Andreagiovanni and Raymond, 2014)

Pokud existuje množina odchylek, která všechny tyto předpoklady splní, nazývá se množina nejistot, nebo neurčitá množina. Také je v modelu zaveden parametr Γ, který zaručuje jeho stálost. Čím je tento parametr Γ větší, tím více je model chráněn vůči chybě s čímž zároveň vzniká cena robustnosti. Na druhou stranu to způsobí pokles optimální hodnoty, což je následkem vyřazení nerobustních variant řešení z přípustné množiny. (D'Andreagiovanni and Raymond, 2014)

4.6.1 Formulace modelu robustní optimalizace (Bertsimas and Sim, 2004)

Máme nějaké *i*-té omezení úlohy $a_i'x \leq b_i$ a množinu J_i, která obsahuje koeficienty $a_{ij}, j \in J_i$ u nichž je předpokládán výskyt neurčitosti. Uvažujeme symetricky rozvržené hodnoty se střední hodnotou, která odpovídá nominální hodnotě a_{ij} na intervalu $[a_{ij} - \hat{a}_{ij}, a_{ij} = \hat{a}_{ij}]$. Poté máme parametr Γ_i pro každé i z intervalu$[0, |J_i|]$, který upravuje robustnost vůči postoji k riziku (0 – žádné riziko, $|J_i|$ – maximální riziko). Dále značíme p vektor neurčitých koeficientů a δi značí nejistotu vah.

Uvažujeme následující model:

maximalizace $c'x$ (16)

za podmínek

$\sum_j a_{ij}x_j$ (17)

$+ \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij} y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\, \hat{a}_{ij} y_t\} \leq b_i \qquad \forall i$ (18)

$-y_j \leq x_j \leq y_j \qquad \forall j$ (19)

$1 \leq x \leq u$ (20)

$y \geq 0.$ (21)

Za předpokladu, že je Γ_i celočíselné, můžeme říct, že i-tá podmínka je splněna $\beta_i(x, \Gamma_i) = \max_{\{S_i | S_i \subseteq J_i, |S_i| = \Gamma_i\}} \{\sum_{j \in S_i} \hat{a}_{ij} |x_j|\}$. Pokud by se Γ_i rovnalo nule a $\beta_i(x, \Gamma_i)$ také, pak podmínka vyjadřuje totéž jako podmínka v nominální verzi problému (bez aspektů neurčitosti).

Aby bylo možné přeformulovat model výše na model lineární optimalizace, je nutné splnit následující podmínky:

Máme vektor $\boldsymbol{x}^*$ a zajišťující funkci pro i −tou vazbu

$$\beta_i(x^*, \Gamma_i) = \max_{\{S_i \cup \{t_i\} | S_i \subseteq J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \{\sum_{j \in S_i} \hat{a}_{ij} |x_j^*| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\, \hat{a}_{it_i} |x_j^*|\}, \qquad (22)$$

ekvivalentní účelové funkci následujícího problému lineární optimalizace:

$$\beta_i(x^*, \Gamma_i) = \max \sum_{j \epsilon J_i} \hat{a}_{ij} |x^*{}_j| z_{ij} \qquad (23)$$

Vyhovující $\sum_{j \epsilon J_i} z_{ij} \leq \Gamma_i$ (24)

$0 \leq z_{ij} \leq 1 \qquad \forall j \epsilon J_i.$ (25)

Nyní je možné model přeformulovat na ekvivalentní robustní model lineární optimalizace (tzv. *robust counterpart*):

maximalizace $c'x$ (26)

za podmínek

$$\sum_j a_{ij}x_j + z_i\Gamma_i + \sum_{j\epsilon J_i} p_{ij} \leqslant b_i \qquad \forall i \qquad (27)$$

$$z_i + p_{ij} \geq \hat{a}_{ij}y_j \qquad \forall i, j \in J_i \qquad (28)$$

$$-y_j < x_j \leq y_j \qquad \forall j \qquad (29)$$

$$l_j \leq x_j \leq u_j \qquad \forall j \qquad (30)$$

$$p_{ij} \geq 0 \qquad \forall i, j \in J_i \qquad (31)$$

$$y_j \geq 0 \qquad \forall j \qquad (32)$$

$$z_i \geq 0 \qquad \forall i. \qquad (33)$$

Uvažujeme duální problém k problému lineární optimalizace (23)-(25):

minimalizace $\sum_{j\in J_i} p_{ij} + \Gamma_i z_i$ (34)

za podmínek

$$z_i + p_{ij} \geq \hat{a}_{ij}|x_j^*| \qquad \forall i, j \in J_i \qquad (35)$$

$$p_{ij} \geq 0 \qquad \forall i, j \in J_i \qquad (36)$$

$$z_i \geq 0 \qquad \forall i. \qquad (37)$$

Vzhledem k tomu, že problém lineární optimalizace (23)-(25) je přípustný (díky silné dualitě) a omezený pro všechny $\Gamma_i \in [0.|J_i|]$, pak můžeme říct, že duální problém (34)-(37) je též ohraničený a přípustný a hodnoty jejich účelových funkcí se shodují. Pokud bychom použili podmínku (22), lze uvést, že $\beta_i(x^*, \Gamma_i)$ je ekvivalentní k účelové funkci problému (34)-(37). Pokud bychom substituovali problém (16)-(21), dostaneme se k tomu, že je tento problém ekvivalentem lineárního optimalizačního problému (26)-(33).

Robustní model lineární optimalizace (26)-(33) má $n + k + 1$ proměnných a $m + k + n$ vazeb, přičemž má $k = \sum_i |J_i|$ neurčitých hodnot. Ve skutečných situacích je matice A řídká.

4.6.2 Absolutní robustní stromy

Máme definovaný absolutní „worst-case“ scénář grafu, který je kostrou grafu a cena této kostry je maximum. Z tohoto předpokladu je zřejmé, že v této kostře jsou všechny ceny hran vázané ke svým omezením shora, přičemž zbylé hrany se cenově pohybují kdekoliv na svých zadaných intervalech. (Yaman, Karasan a Pinar, 2001)

Pokud máme kostru grafu, jejíž absolutní „worst-case“ scénář má minimální cenu, pak tuto kostru nazýváme absolutní robustní kostrou. Pokud uvažujeme takovýto scénář, pak absolutní robustní kostra odpovídá minimální kostře grafu. Oproti předchozímu případu všechny hrany kostry stojí maximum na svém intervalu. (Yaman, Karasan a Pinar, 2001)

Problematikou absolutní robustní kostry s konečnou množinou scénářů se zabývali Kouvelis a Yu (1997), kteří vysvětlují dva případy koster v závislosti na otevřenosti, respektive uzavřenosti množiny scénářů. Prvním případem absolutní robustní kostry je NP-úplný problém, ve kterém je množina scénářů pevně ohraničená. Druhým typem je silný NP-složitý problém, který je specifický neohraničenou množinou scénářů.

Bereme-li do úvahy teorie výše, pak lze říct, že absolutní robustní kostra grafu, s hranami určenými intervaly cen, je řešitelná vypočítáním minimální kostry grafu za předpokladu, že všechny její hrany stojí méně než jejich vrchní intervalová hodnota. Minimální kostru lze nalézt v polynomiálně omezeném čase. (Yaman, Karasan a Pinar, 2001)

4.6.3 Relativní robustní stromy

K této problematice je zaveden nový pojem robustní odchylka kostry. Je to rozdíl (který je maximem) mezi cenami kostry relativního „worst-case“ scénáře a minimální kostry grafu. Pokud je robustní odchylka kostry naopak minimální, pak takovéto kostře říkáme relativní robustní kostra. (Yaman, Karasan a Pinar, 2001)

Relativní „worst-case“ scénář kostry je takový scénář, jehož hrany na kostře mají ceny, které jsou na svém horním intervalu a zbylé hrany mají naopak ceny na dolním odhadu svého intervalu. (Yaman, Karasan a Pinar, 2001)

Tuto problematiku také rozvinuly Kouvelis a Yu (1997), kteří svou prací prokázali, že relativní robustní kostra grafu je NP-úplný problém, pokud je množina scénářů omezená a naopak je-li otevřená, jedná se o silně NP-složitý problém.

4.7 P a NP problémy

Fortnow (2009) říká, že se už skoro padesát let podstatně mění a vyvíjí počítačová věda, respektive informatika, a to od momentu, kdy kanadsko-americký počítačový vědec a matematik Stephen Arthur Cook publikoval svou zásadní práci v oblasti NP úplných problémů

pod názvem *The Complexity of Theorem-Proving Procedures* (v překladu Komplexnost Postupů Dokazování Vět) v roce 1971. Na vzestupu jsou také výpočetní kapacity, oproti tomu jejich cena šla zásadně dolů. Počítačem řízené výpočty jsou v dnešní době využívané skoro ve všech oblastech vědeckých výzkumů, jako jsou například biologie, fyzika, ekonomie atd. Obecně všechna vědní pole, které využívají širokopásmové výpočetní modelování, simulace a rozhodovací problémy.

Na rozdíl od Fortnowa (2009) Uday (2018) píše, že počátky P a NP problémů byly položeny ještě dříve. Podle něj je problematika spjata s prací rakouského matematika Kurta Gödela, který již roku 1956 s jinými vědci probíral téma komplexnosti NP úplných úloh.

P a NP jsou třídami složitosti. Zkratka P znamená polynomiální problém, NP je zkratka pro nedeterministický polynomiální problém. P i NP problémy jsou především spjaty s rozhodovacími úlohami, přičemž každá optimalizační úloha má svou verzi rozhodovací úlohy. (Kolář, 2009)

4.7.1 Třída složitosti P

P problémy jsou třídou rozhodovacích problémů, jenž jsou řešitelné v polynomiálním čase nějakým algoritmem. Tento algoritmus má omezený počet kroků fixní polynomiální funkcí, a to na velikosti vstupů. P problémy se spojují s přívlastkem lehké, nebo jednoduché. (Fortnow, 2009)

Třídu P problémů lze popsat na úloze hledání minimální cesty mezi libovolnými dvěma vrcholy prostého neorientovaného grafu. Úloha je popsaná prostým grafem a dvěma jeho vrcholy, mezi kterými chceme najít cestu. Výsledkem úlohy minimální cesty jsou všechny existující posloupnosti vrcholů, které tvoří nejkratší průchod vrcholy. (Kolář, 2009)

Fortnow (2009) uvádí příklad, kdy je ve skupině velké množství žáků. Tyto žáky potřebuje učitel rozdělit do dvojic na projekty tak, aby byly žáci ve dvojicích navzájem kompatibilní, přičemž informaci o tom, kteří žáci jsou kompatibilní se kterými, známe. Existuje teoretická možnost prozkoumat úplně všechny dvojice žáků, nicméně i třída 40 žáků by vyžadovala prozkoumat velmi mnoho párů. Proto roku 1965 kanadský informatik a matematik Jack Robert Edmonds sestavil efektivní algoritmus, který problém tohoto typu dokáže vyřešit

v polynomiálním čase. Také představil svůj návrh věty o efektivním výpočtu. Jeho práce pak dala základ třídě problémů P.

4.7.2 Třída složitosti NP

Pokud známe pro rozhodovací problém nějaký nedeterministický algoritmus, pak je tato úloha ve třídě NP. Také platí, že zmíněný algoritmus probíhá v polynomiálním čase. Tyto algoritmy probíhají ve dvou fázích. První fáze je nedeterministická, což znamená, že se odhadne a uloží řetěz znaků, které mohou být řešením. Také je možné, že se daný řetěz znaků bude měnit pro rozlišné fáze výpočtů. Následuje fáze deterministická, kdy se vypočítá deterministický algoritmus. Ten má na vstupech k dispozici mimo jiné právě i řetěz znaků získaný předchozí fází. V této fázi je výstupem výsledné řešení (nebo také nekonečný cyklus), respektive validace řešení, které vzešlo z nedeterministické fáze. (Kolář, 2009)

Původně byl pojem NP spjat s větami o nedeterministickými stroji, což jsou takové stroje, které disponují dvěma a více kroky z aktuální pozice. Nyní se NP úlohy spojují spíše s pojmem nedeterministického polynomiálního času. (Cook, 2000)

Jako příklad NP problému lze uvést například situace, kdy máme velkou skupinu lidí a chceme je posadit k oválnému stolu. Podmínkou ale je, aby lidé, kteří se třeba nesnesou, nebo neznají, neseděli vedle sebe, přičemž tyto informace o všech osobách jednotlivě máme. Tento problém se nazývá termínem Hamiltonovská kružnice. V této chvíli můžeme navrhnout nějaký potenciálně správný zasedací pořádek. Přesně to ovšem nevíme, ale můžeme efektivně zjistit, zda je zasedací pořádek validní. (Fortnow, 2009)

4.7.3 NP úplnost

Zásadní množinou úloh třídy NP jsou takové, které můžeme popsat jako nejsložitější úlohy z této třídy. Říká se jim NP-úplné problémy. Podmínkou zařazení úlohy do této skupiny je možnost redukovat na NP-úplné problémy všechny ostatní problémy třídy NP v polynomiálním čase. (Kolář, 2009)

V principu si chceme práci na NP úplných problémech ulehčit. A to tak, že se snažíme model nějak redukovat. Máme-li například řešit úlohu, je výhodné zjistit, jestli tato úloha nepatří do další větší skupiny úloh, které umíme už nějakým algoritmem vyřešit. Zjistíme-li, že takovým

algoritmem úloha lze vyřešit, zredukujeme vstupy naší původní úlohy tak, aby lépe odpovídaly požadavkům na vstupy do toho algoritmu. (Kolář, 2009)

4.7.4 P versus NP

Podstatu tohoto problému lze shrnout následovně. V reálném světě je někdy obtížnější vůbec vyřešit nějakou úlohu než poté zjišťovat, zda jsou výstupy toho řešení správné. V tomto příkladě je třída P problémů zástupcem optimalizačním úloh, které je možné efektivně vyřešit a proti tomu jsou NP problémy zástupci optimalizačních úloh, jejichž správnost řešení lze efektivně ověřit. (Goldreich, 2010)

Další otázkou problému P versus NP je, zda je obtížnější prověřit správnost důkazů vět nebo dokazovat věty obecně. Dá se tedy říct, že v tomto případě P třída reprezentuje rozhodovací úlohy, které jsou dobře řešitelné a NP třída zastupuje množiny, které mají efektivně verifikovatelné věty o příslušnosti k množině. (Goldreich, 2010)

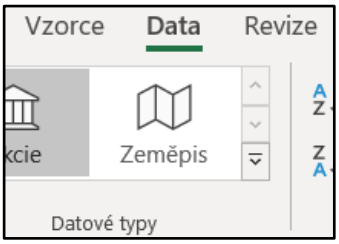
P versus NP problém se zabývá otázkou, zda NP problémy ve skutečnosti nejsou P problémy. Respektive zda jsou všechny jazyky řešitelné nějakým nedeterministickým algoritmem v polynomiálním čase stejně tak jako deterministický algoritmus v polynomiálním čase. P a NP problémy jsou často svázány s teoretickou informatikou a modelem počítače, popsaném například Alanem Turingem roku 1936, dobře známým pod názvem Turingův stroj. (Cook, 2000)

4.8 OpenSolver

OpenSolver je doplněk programu Microsoft Excel. Tento software je založen na principu open source, což znamená, že je přístupný veřejnosti a má otevřený zdrojový kód. Slouží jako nástroj pro lineární, nelineární a celočíselnou optimalizaci. V programu Microsoft Excel se tento doplněk nachází na kartě „Data“ v pravé části v oddíle „OpenSolver“. (www.opensolver.org, 2020)

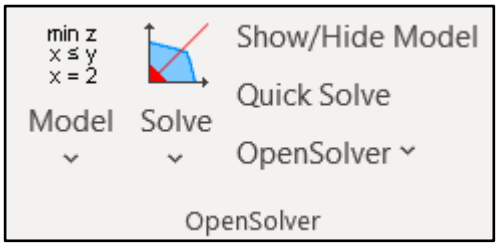
Jako doplněk Excelu, který není obsažen v základním balíčku programu Micrsoft Excel je nutné si tento doplněk přidat skrze samotný Excel. Tlačítko Soubor → Možnosti → Doplňky → (ve spodní části v sekci Spravovat:) Přejít… → poté ze souboru vybrat nainstalovaný balíček OpenSolver → vybrat jej v sekci Dostupné doplňky → OK.

Obrázek 4 Microsoft Excel, záložka Data



Zdroj: Microsoft Excel

Obrázek 5 Microsoft Excel, záložka Data, oddíl OpenSolver



Zdroj: Microsoft Excel

Program Microsoft Excel sám o sobě obsahuje standardní nástroj pro optimalizaci méně složitých problémů, který se jmenuje Solver, nebo také Řešitel, přístupný z oddílu „Analýza“. Tento nástroj je pro Microsoft Excel dodáván společností Frontline Systems. Solver je doplněk, který je snadno ovladatelný a vhodný například pro účely výuky, kdy si studenti mohou vyzkoušet techniky modelování nebo optimalizace. Ale pokud je tento nástroj použit na reálné situace, je možné, že už nebude dostačovat, je totiž omezen na 200 proměnných. Existuje i rozšíření na nástroj Premium Solver, také od společnosti Frontline Systems, který je již placený. (Dunning and Mason, 2010)

OpenSolver vznikl jako doplněk pro Microsoft Excel spojením software skupiny COIN-OR a Excelu. COIN-OR totiž disponovali software, který byl schopen optimalizovat lineární i celočíselné programování a dokázal operovat ve větším rozsahu než originální nástroj Excelu Solver. Tento software se nazývá CBC. OpenSolver tedy dovoluje uživateli pracovat skrze Excel s optimalizačním nástrojem COIN-OR CBC. (Dunning and Mason, 2010)

„Computational Infrastructure for OR“ ve zkratce COIN-OR je skupina výzkumníků. Tato nezisková společnost má oficiální název COIN-OR Foundation, Inc. Společnost se zabývá vývojem open source nástrojů podporujících řešení problémů operačního výzkumu. Také

spravuje úložiště software nástrojů, sloužících k psaní kódů optimalizace, nebo jako v případě doplňku OpenSolver samostatných balíčků. (Salzman, Ladányi and Ralphs, 2004)

V roce 2020 je aktuální verzí doplňku OpenSolver 2.9.0, který využívá SolveEngine společnosti Satalia. Nabízí rychlé řešení problémů lineárního a celočíselného programování a je také kompatibilní s modely obyčejného Solveru. Jeho limitace spočívají už pouze v omezené operační paměti počítače, nicméně je možné, že rozsahem velké modely mohou trvat při spuštění OpenSolver delší časový úsek.

Hlavními tvůrci doplňku jsou Andrew Mason, nebo v poslední době vývoj od Jacka Dunna z MIT. K původnímu vývoji také přispěli studenti katedry Inženýrských věd Univerzity Auckland. (www.opensolver.org, 2020)

5 Vlastní práce

Ve vlastní práci bude řešen problém pomocí Steinerova stromu, který bude následně převeden na svoji robustní verzi, aby byl celý algoritmus použitelný obecněji, na široké spektrum problémů z různých oblastí od projektování staveb, přepravních problémů po projektového řízení atd. Pro algoritmus Steinerova stromu je nutné znát tyto prvky. Musíme znát vrcholy, nebo uzly, kterými mohou být například různá nádraží, překladiště, úkoly projektového řízení a obecně různé mezníky. Poté potřebujeme vědět, které z těchto uzlů, jsou pro nás podstatné natolik, že by řešení nebylo přípustné, pokud by kostra grafu těmito uzly neprocházela. Také potřebujeme další uzel, tentokrát nepodstatný, tedy Steinerův uzel, nebo více uzlů, které mohou být nějakým mezikrokem v úloze. Ještě je podstatné znát ceny hran mezi uzly a možnou chybu každé hrany.

Celý algoritmus bude řešen v programu Microsoft Excel za pomoci jeho doplňku OpenSolver. Nejprve v programu vyřešíme obyčejnou úlohu Steinerova stromu, která ale musí být převedena na celočíselnou úlohu lineárního programování, aby bylo možné ji pomocí OpenSolveru vyřešit. Tento převod bude proveden na základě teoretických poznatků z kapitoly 4.5.2 od autorů Diané a Plesníka (1993). Poté bude úloha rozvinuta na robustní verzi.

Vzhledem k rozsahu následných výpočtů v Excelu bude celý postup nejprve prezentován na malé ilustrativní úloze, kde budou znázorněny a popsány všechny kroky algoritmu. Bude také rozebráno několik variant, jež by mohly nastat. Algoritmus totiž umožňuje uživateli zadat počet hran, které se zhorší. Po zadání a spuštění modelu zadaného do OpenSolveru tento doplněk vypočítá, kolik bude nejhorší hodnota účelové funkce tak, že zhorší nejdražší hrany podle jejich zadané chyby (zhorší jich tolik, kolik zadáme, že se jich zhorší). Pořadí hran, které by se řadily do zhoršených, je určeno od nejdražší po nejlevnější. Model OpenSolveru také při každém svém spuštění porovnává všechny varianty kostry grafu, tedy hran, které jsou do kostry vybrány, a to i v závislosti na tom, kolik, jaké a o kolik zhoršené hrany jsou uživatelem aktuálně zadány do modelu, že se zhorší.

Po malé úloze bude již následovat případová studie. Ta už nebude rozebrána v takovém detailu jako malá úloha, ale budou prezentovány především výstupy robustního Steinerova stromu z programu Microsoft Excel.

5.1 Malá ilustrativní úloha

Pro malou ilustrativní úlohu byla vybrána čtyři místa v Česku, která jsou pro tuto úlohu podstatnými uzly a jsou to města Lovosice (1), Trutnov (2), České Budějovice (3) a Cheb (4). Mezi těmito městy existuje po obvodu vlaková trať, přičemž známe vzdálenosti mezi jednotlivými místy. Dále jsou do úlohy zahrnuta další dvě města, kterými jsou Praha – Uhříněves (5) (překladiště) a Plzeň (6). Obě tato místa jsou Steinerovými uzly.

5.1.1 Úvod do úlohy

V úloze se hledá minimální kostra grafu Steinerova stromu, kde jsou cenami hran vzdálenosti mezi jednotlivými místy. V následující *Tabulce 1* jsou potřebná data k řešení úlohy. Data byla získána z mapových podkladů ze serveru www.google.cz/maps (nicméně vzdálenosti jsou přibližné). Vzhledem k tomu, že se řeší neorientovaný graf, postačí uvést vzdálenosti mezi jednotlivými místy pouze v jednom směru, ve kterém není podstatné.

Tabulka 1 Vzdálenosti mezi místy

zápis hran	Trasa mezi místy		Vzdálenost v km
x_{12}, x_{21}	Lovosice	Trutnov	132
x_{15}, x_{51}	Lovosice	Praha – Uhříněves	67
x_{14}, x_{41}	Lovosice	Cheb	128
x_{16}, x_{61}	Lovosice	Plzeň	100
x_{32}, x_{23}	České Budějovice	Trutnov	203
x_{35}, x_{53}	České Budějovice	Praha – Uhříněves	114
x_{36}, x_{63}	České Budějovice	Plzeň	116
x_{34}, x_{43}	České Budějovice	Cheb	196
x_{46}, x_{64}	Cheb	Plzeň	83
x_{65}, x_{56}	Plzeň	Praha – Uhříněves	91
x_{25}, x_{52}	Trutnov	Praha – Uhříněves	110

Zdroj: vlastní zpracování

Jsou určeny i maximální odchylky jednotlivých tras *Tabulka 2*, které jsou v tomto případě určeny na základně spočítání vzdálenosti mezi místy, kdyby byla na trase výluka.

Tabulka 2 Maximální odchylka

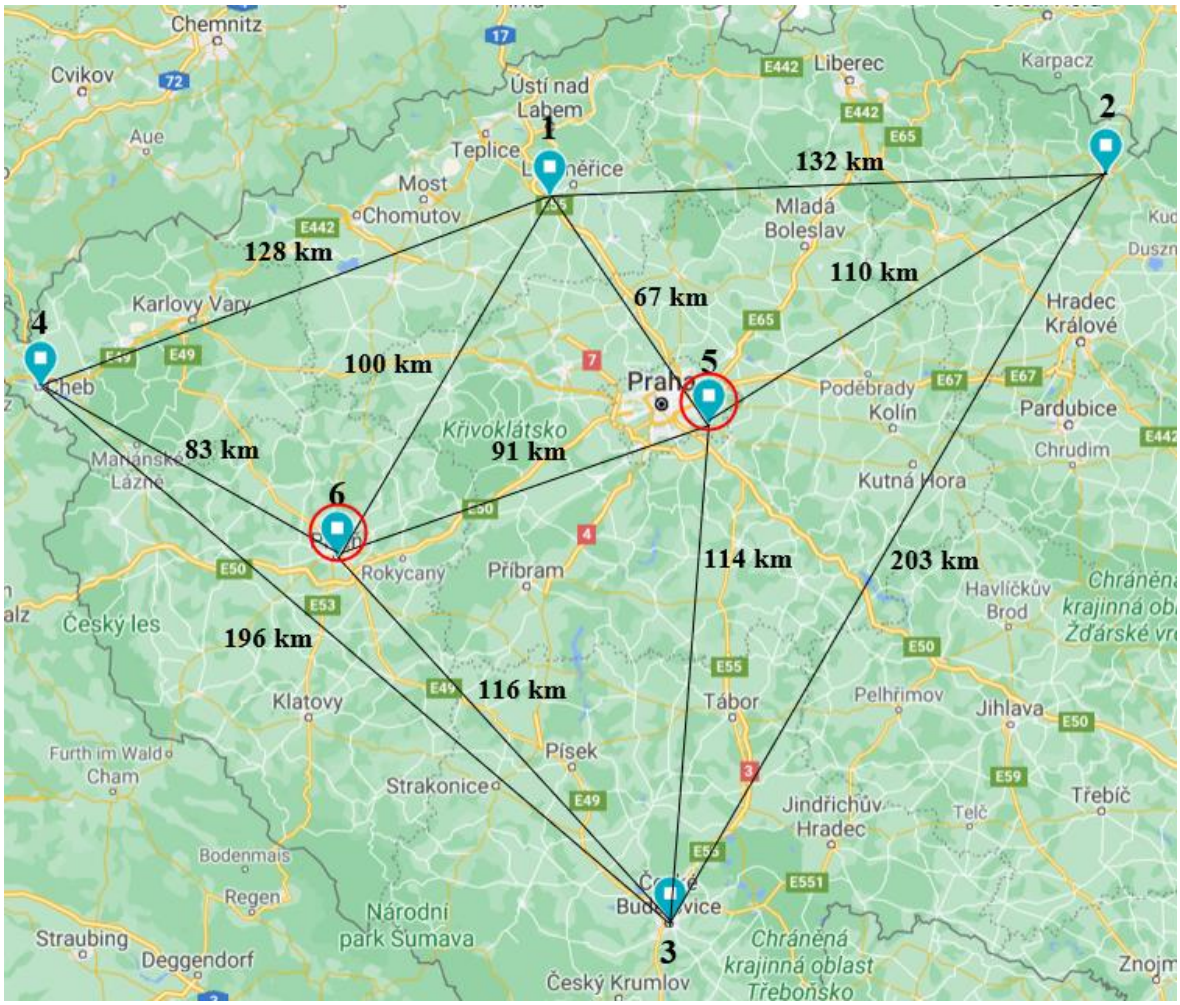
Trasa mezi místy		Maximální odchylka δ
Lovosice	Trutnov	40
Lovosice	Praha – Uhříněves	5
Lovosice	Cheb	30
Lovosice	Plzeň	10
České Budějovice	Trutnov	60
České Budějovice	Praha – Uhříněves	13
České Budějovice	Plzeň	15
České Budějovice	Cheb	50
Cheb	Plzeň	11
Plzeň	Praha – Uhříněves	12
Trutnov	Praha – Uhříněves	17

Zdroj: vlastní zpracování

Na následujícím obrázku č. 5 je schéma míst a tras mezi nimi, včetně vzdáleností. Červeně zakroužkované body jsou Steinerovými uzly. Zbylé uzly (1, 2, 3 a 4) jsou význačné, tedy potřebujeme, aby kostra grafu jimi procházela a zároveň potřebujeme najít minimální cenu účelové funkce. Pokud je graf takto jednoduchý, lze kostru grafu odhadnout i krátkým zkoumáním, nicméně v reálných problémech to již možné není.

Z obrázku je patrné, že počet uzlů $n = 6$, počet hran $m = 11$ a význačných uzlů je $p = 4$. Také bychom základním zkoumáním schématu odhadnout, jakými hranami by mohla být tvořena minimální kostra, a tedy i odhadnout hodnotu účelové funkce. Díky výrazným rozdílům ve vzdálenostech mezi jednotlivými uzly (a také díky malému počtu uzlů) odhadneme, že minimální kostra bude mezi uzly 5-3, 5-2, 5-1 a 1-4 s cenou 419. Ale je to pouze odhad, pro pozdější postup potřebujeme vyřešit úlohu OpenSolverem.

Obrázek 5 Malá úloha - mapa



Zdroj: www.google.cz/maps + vlastní zpracování

V tuto chvíli máme kompletní zadání pro sestavení modelu Steinerova stromu formulovaného jako model celočíselného programování.

5.1.2 Postup převodu úlohy Steinerova stromu na formulaci celočíselného LP

Prvním krokem převodu je výběr uzlu v_0. Není podstatné, který uzel to je, nicméně musí být z množiny význačných uzlů, v tomto případě z uzlů 1, 2, 3 nebo 4. V této úloze byl vybrán uzel 1, Lovosice. To znamená, že vyloučíme všechny hrany, jenž do něj vstupují.

Takto můžeme začít vyplňovat model v programu Microsoft Excel, kam na začátku vepíšeme všechny hrany mimo těch, které vstupují do uzlu 1. K těmto hranám se přidá pro každý uzel j proměnná u_j. V tomto případě bude celkem šest proměnných u_j.

Náplň řádku je následující:

proměnné $x_{12}, x_{14}, x_{15}, x_{16}, x_{23}, x_{25}, x_{32}, x_{34}, x_{35}, x_{36}, x_{43}, x_{46}, x_{52}, x_{53}, x_{56}, x_{63}, x_{64}, x_{65}$, dále proměnné $u_1, u_2, u_3, u_4, u_5, u_6$. Do řádku za tyto proměnné se přidají pravé strany rovnice značené jako b a $breal$. Nad tímto řádkem je proměnný řádek, jenž bude vyplněn do OpenSolveru do sekce „Variable Cells:“. Tento řádek nám po kompletním vyplnění modelu a spuštění OpenSolveru řekne, které hrany jsou v minimální kostře. Zároveň jsou to binární hodnoty podle podmínky (14) z teorie.

V tuto chvíli začneme vyplňovat oddíly tabulky modelu, který každý odpovídá jedné podmínce podle Dianého a Plesníka (1993). Rovnice i nerovnice si upravíme tak, aby žádné proměnné nebyly na pravé straně.

První oddíl odpovídá podmínce (10). Sloupec $breal$ je získán skalárním součinem proměnných modelu (modře zvýrazněny) a samotných řádků modelu.
Nerovnice sekce:

$x_{ij} \leq 1$

Tabulka 3 Podmínka (10)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
1	1	1																						1	0
			1	1	1	1																		1	0
							1	1	1															1	0
										1	1	1	1											1	0
														1	1	1	1							1	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce podmínky (11) a upravená nerovnice (hodnota *n* udává počet uzlů grafu):

$(n * x_{ij}) - u_j \geq 1$

Tabulka 4 Podmínka (11)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
6	6	6																	-1					1	0
			6	6	6	6														-1				1	0
							6	6	6												-1			1	0
										6	6	6	6									-1		1	0
														6	6	6	6						-1	1	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce podmínky (12) a převedená nerovnice:

$$[(n+1) * x_{ij}] - (n * u_j) \leq n * 1$$

Tabulka 5 Podmínka (12)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
7	7	7																	-6					6	0
			7	7	7	7														-6				6	0
							7	7	7												-6			6	0
										7	7	7	7									-6		6	0
														7	7	7	7						-6	6	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce podmínky (13.1) a převedená první část nerovnice, která se skládá ze dvou nerovnic:

$$(n * x_{ij}) - u_j + u_i \leq (n-1)$$

Tabulka 6 Podmínka (13.1)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
6																		1	-1					5	0
	6																		-1	1				5	0
		6																	-1			1		5	0
			6																1	-1				5	0
				6																-1	1			5	0
					6															-1		1		5	0
						6														-1			1	5	0
							6											1			-1			5	0
								6												1	-1			5	0
									6												-1		1	5	0
										6								1				-1		5	0
											6								1			-1		5	0
												6									1	-1		5	0
													6									-1	1	5	0
														6				1					-1	5	0
															6					1			-1	5	0
																6					1		-1	5	0
																	6					1	-1	5	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce podmínky (13.2) a druhá část nerovnice:

$(-n * x_{ij} - u_j + u_j) \geq (-n - 1)$

Tabulka 7 Podmínka (13.2)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
-6																		1	-1					-7	0
	-6																		-1	1				-7	0
		-6																	-1			1		-7	0
			-6																1	-1				-7	0
				-6																-1	1			-7	0
					-6															-1		1		-7	0
						-6														-1			1	-7	0
							-6											1			-1			-7	0
								-6												1	-1			-7	0
									-6												-1		1	-7	0
										-6								1				-1		-7	0
											-6								1			-1		-7	0
												-6									1	-1		-7	0
													-6									-1	1	-7	0
														-6				1					-1	-7	0
															-6					1			-1	-7	0
																-6					1		-1	-7	0
																	-6					1	-1	-7	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce pro proměnnou u_1, část podmínky (15), tedy ta proměnná, která byla přiřazena k uzlu v_0. Rovnice:

$u_1 = 0$

Tabulka 8 Podmínka (15)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
																		1						0	0

Zdroj: vlastní zpracování + Microsoft Excel

Sekce pro druhou část podmínky (15), její nerovnice:

$u_i \geq 0$

Tabulka 9 Podmínka (15)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
																			1					0	0
																				1				0	0
																					1			0	0

Zdroj: vlastní zpracování + Microsoft Excel

V této části jsou ceny hran, které do úlohy vstupují a pole účelové funkce, které je v tuto chvíli ještě nulové, protože ještě není naplněný model pro OpenSolver, který by účelovou funkci a kostru spočítal.

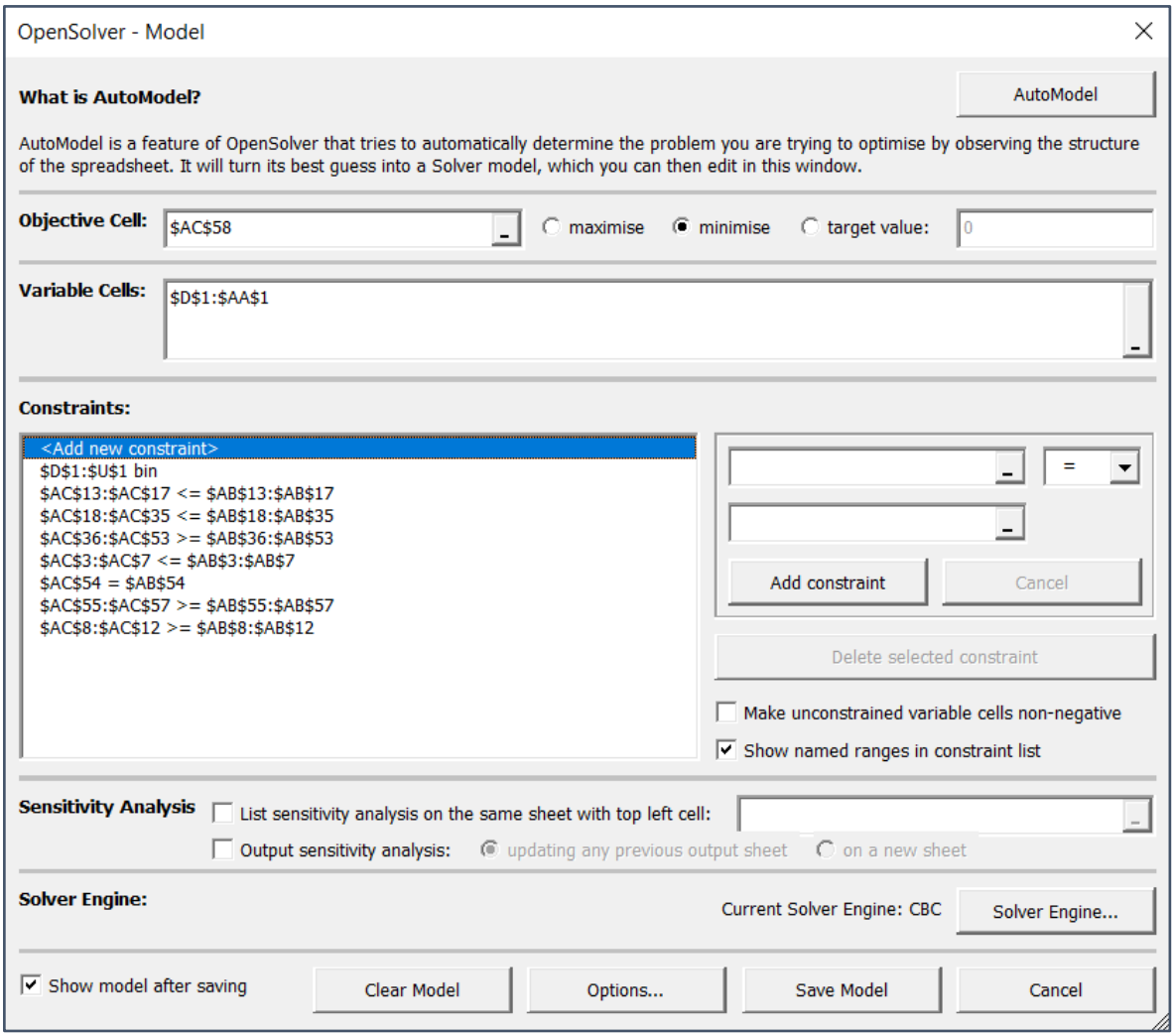
Tabulka 10 Ceny hran a Účelová funkce (růžově zvýrazněna)

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91								0

Zdroj: vlastní zpracování + Microsoft Excel

Po vyplnění všech potřebných sekcí, doplnění cen a vložení vzorce pro skalární součin lze otevřít model OpenSolveru a vyplnit jej, tedy zadat do něj všechny podmínky.

Obrázek 6 Screen vyplněného modelu pro OpenSolver

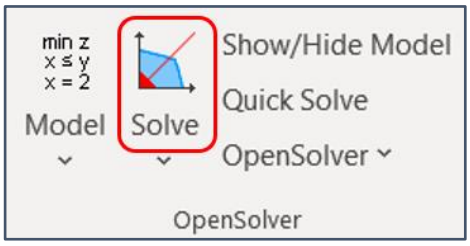


Zdroj: vlastní zpracování + OpenSolver

Pro účelovou funkci (Objective Cell) nastavíme minimalizaci. Je to růžově označená buňka výše. Proměnné (Variable Cells) jsou výše v řádku označeném modře. Zároveň je tento řádek

určen podmínkou binárnosti, což je vidět v podmínkách (Constraints). Řádek proměnných je také zatím nulový, hodnoty 1 nebo 0 se zde objeví až po spuštění modelu OpenSolver. V podmínkách jsou také vložené všechny podmínky modelu. Ovšem OpenSolver u porovnává pouze sloupec $breal$ ku b, přičemž $breal$ už je vypočítaný skalární součin.
Takto vyplněný model můžeme uložit a následně nechat OpenSolver model vyřešit.

Obrázek 7 Spuštění modelu



Zdroj: Microsoft Excel

OpenSolver provede výpočty v řádu jednotek sekund, napíše novou hodnotu účelové funkce a určí, které hrany jsou v minimální kostře grafu. V *Tabulce 11* je patrné, že do minimální kostry byly zařazeny hrany $x_{52}, x_{53}, x_{14}, x_{15}$. Výstup odpovídá odhadu kostry výše. Také je splněna podmínka zahrnutí všech význačných uzlů do minimální kostry grafu. Dále je vidět, že byl využit i Steinerův uzel 5, ale uzel 6 použit nebyl. OpenSolver musel zjistit, že pro model není výhodné jím procházet a jeho zahrnutí do minimální kostry by zvýšilo hodnotu účelové funkce, kterou chceme minimalizovat.

Tabulka 11 Minimální kostra Steinerova stromu

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	2	2	1	1	-1	
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

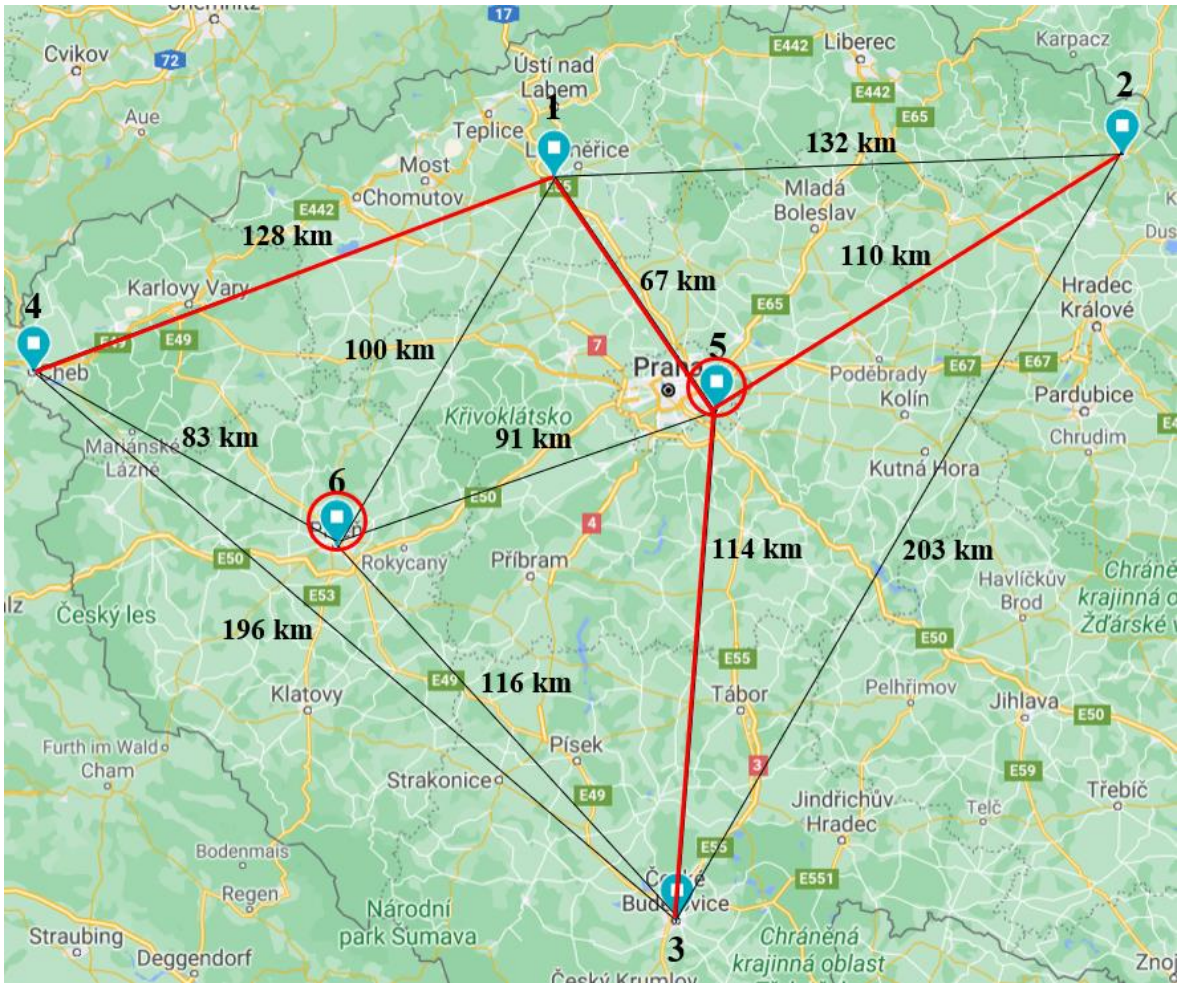
V *Tabulce 12* se změnilo pole pro účelovou funkci vypočítané OpenSolverem na 419, což je také správně odhadnuto výše.

Tabulka 12 Výsledná účelová funkce

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	u1	u2	u3	u4	u5	u6	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91								419

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Obrázek 8 Výsledná minimální kostra Steinerova stromu



Zdroj: vlastní zpracování

V tento moment je obyčejná úloha Steinerova stromu formulovaná jako úloha celočíselného lineárního programování vyřešena. Známe hodnotu účelové funkce i minimální kostru grafu. Pokud bychom ve výstupech modelu OpenSolver chtěli vidět nějakou změnu, museli by být pozměněny například vzdálenosti mezi městy, nebo by muselo být do množiny význačných uzlů zahrnuto další město, které tam ještě není, například Plzeň.

5.2 Robustní verze

5.2.1 Model robustního Steinerova stromu

Spojením formulace robustního modelu lineární optimalizace a úlohy Steinerova stromu formulovaného jako model celočíselného lineárního programování vznikl následující model:

minimalizace E

za podmínek

$$\sum_j c_{ij}x_{ij} + \sum_{(i,j)\epsilon J} p_{ij} + \Gamma z - E \leq 0$$

$$\sum_{i \in V^-(j)} x_{ij} \leq 1 \qquad \forall j \in V(\vec{G}) - \{1\}$$

$$n \sum_{i \in V^-(j)} x_{ij} \geq u_j + 1 \qquad \forall j \in V(\vec{G}) - \{1\}$$

$$(n+1) \sum_{i \in V^-(j)} x_{ij} \leq n(u_j + 1) \qquad \forall j \in V(\vec{G}) - \{1\}$$

$$1 - n(1 - x_{ij}) \leq u_j - u_t \leq 1 + n(1 - x_{ij}) \qquad \forall ij \in E(\vec{G})$$

$$-\delta_{ij}x_{ij} + p_{ij} + z \geq 0 \qquad \forall ij \in E(\vec{G})$$

$$p_{ij} \geq 0 \qquad \forall ij \in E(\vec{G})$$

$$z \geq 0$$

$$x_{ij} \in \{0,1\} \qquad \forall ij \in E(\vec{G})$$

$$u_1 = 0, \qquad u_i \geq 0 \qquad \forall i \in Z - \{1\}$$

5.2.2 Popis převedení úlohy na robustní verzi

Ve chvíli, kdy máme vyřešenou úlohu obyčejného Steinerova stromu, nastává fáze, kdy model můžeme upravit tak, aby byl robustní. Vycházíme z teoretických poznatků Bertsimase a Sima (2004) a jejich formulace modelu robustní optimalizace (kapitola 4.6.1). Použijeme nelineární model (16)-(21) s jeho převedením na ekvivalentní robustní model lineární optimalizace (26)-(33).

Nejdříve rozšíříme model o množinu proměnných p_{ij}, tak, že je ke každé proměnné x_{ij}, která už je zařazená v modelu. Cena každé proměnné p_{ij} je rovna jedné.

Tabulka 13 Přidaná množina proměnných p_{ij}

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p12	p32	p52	p23	p43	p53	p63	p14	p34	p64	p15	p25	p35	p65	p16	p36	p46	p56
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Zdroj: vlastní zpracování + Microsoft Excel

Úpravu na robustní verzi musíme provést převedením účelové funkce na levou stranu. Účelovou funkci bude reprezentovat proměnná E (žlutě zvýrazněno), kterou když přidáme na levou stranu, rozšíříme tím model o jeden sloupec.

Tabulka 14 Rozšíření levé strany modelu o účelovou funkci

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p12	p32	p52	p23	p43	p53	p63	p14	p34	p64	p15	p25	p35	p65	p16	p36	p46	p56	z	E

Zdroj: vlastní zpracování + Microsoft Excel

V robustní formulaci optimalizačních problémů minimalizujeme $c'x$ vyhovující $Ax \leq b$. Pomocí substituce vyjádříme $c'x \leq E$ a následně převedeme E na levou stranu $c'x - E \leq 0$. E má tím pádem cenu v řádku cen hodnotu -1.
Zařadíme také proměnnou z (výše zvýrazněná žlutě), která udává počet zhoršení v simulaci modelu. Doplníme jej takto: $c'x + (p_{ij} + z) - E \leq 0$.

Ještě potřebujeme do modelu doplnit všechny odchylky. Ty jsou značené δ a určené nerovnicí $p_{ij} + z - \delta_{ij} x_{ij} \leq 0$. Máme zadání, kde jsou odchylky určené absolutní hodnotou, tady reprezentují prodloužení trasy o možnou objízdnou trasu, která by mohla nastat. Jinak by mohly být odchylky zapsané také například podílem. Lze pomocí funkcí Excelu do buněk pro odchylky napsat například poměr -0,5 * cena dané hrany. Tato formulace by říkala, že se cena té hrany zhorší maximálně o 50 %. Vždy je ovšem pole odchylky (pokud vůbec je vyplněné, což nemusí) záporné. V nerovnici se $\delta_{ij} x_{ij}$ převádělo z pravé strany, kde měl výraz kladnou povahu, proto je vlevo záporný.

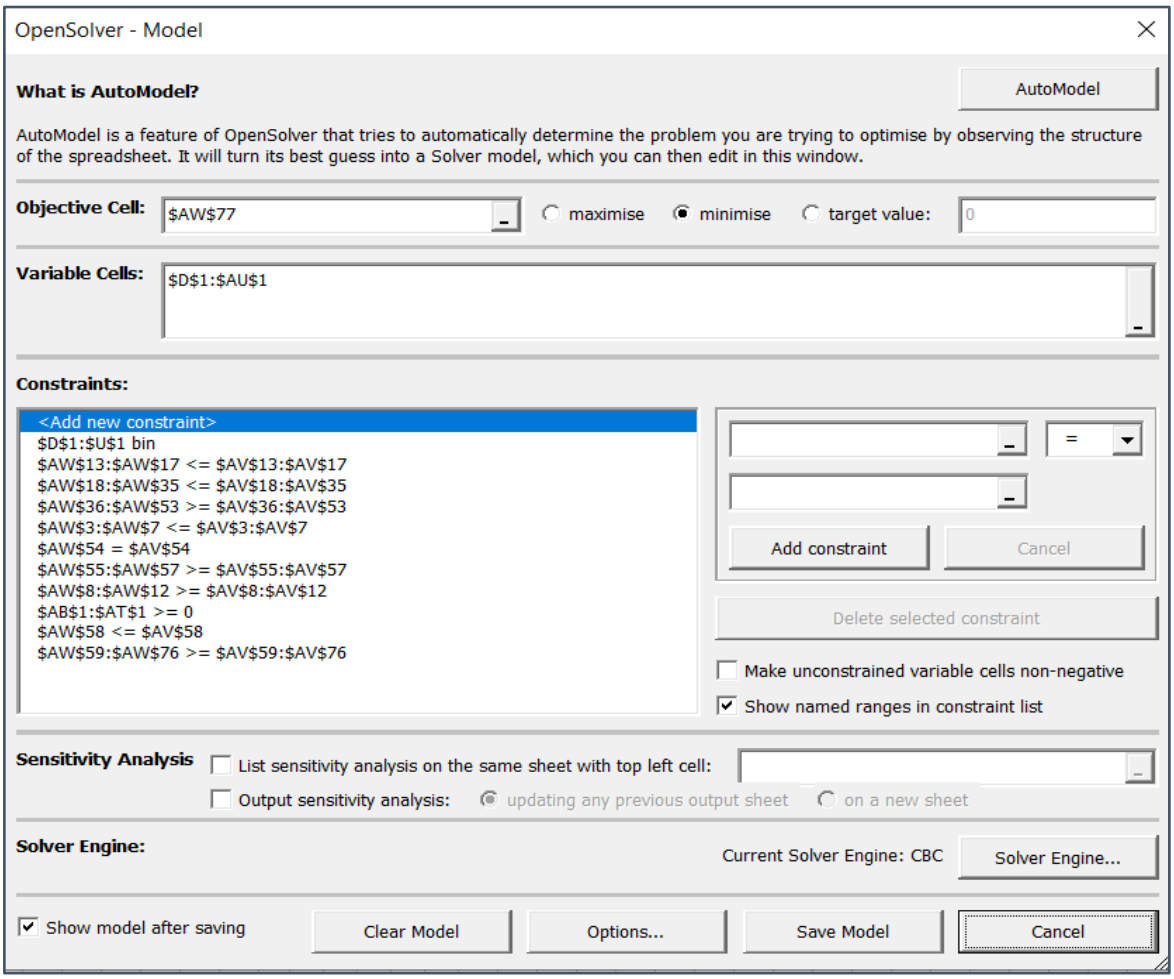
Tabulka 15 Model doplněný o odchylky, z a E

x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	p12	p32	p52	p23	p43	p53	p63	p14	p34	p64	p15	p25	p35	p65	p16	p36	p46	p56	z	E
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	-1
-40																		1																			1	
	-60																		1																		1	
		-17																		1																	1	
			-60																		1																1	
				-50																		1															1	
					-13																		1														1	
						-15																		1													1	
							-30																		1												1	
								-50																		1											1	
									-11																		1										1	
										-5																		1									1	
											-17																		1								1	
												-13																		1							1	
													-12																		1						1	
														-10																		1					1	
															-15																		1				1	
																-11																		1			1	
																	-12																		1		1	
																																					1

Zdroj: vlastní zpracování + Microsoft Excel

Je-li do tabulky doplněno vše, musí se podle podmínek ještě rozšířit model v OpenSolveru viz *Obrázek 8*. Je nutné rozšířit proměnné buňky modelu, s čímž se pojí úprava původního skalárního součinu, který musí být také rozšířen podle nových proměnných (p_{ij}, z, E). Dále přidáme podmínku nezápornosti a nenulovosti pro proměnné p_{ij}. Nová účelová funkce se přesunula podstatně níže, proto přeznačíme pozici její buňky, aby odpovídala nové tabulce.

Obrázek 9 Model OpenSolveru upravený na robustní verzi



Zdroj: vlastní zpracování + OpenSolver

Nyní lze začít do řádku cen vyplňovat hodnotu z. Pokud nastavíme v *Tabulce 15* hodnotu $z = 0$, znamená to, že byl vymodelován ideální scénář a žádná z hran se nezhoršila. Za takových předpokladů vyjde hodnota účelové funkce i minimální kostra stejně jako v předchozím modelu.

5.2.3 Happy day scénář

Tabulka 16 Happy day scénář

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	30	419		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	0	-1	0	0
-40																		1		0	30
	-60																	1		0	30
		-17																1		0	13
			-60															1		0	30
				-50														1		0	30
					-13													1		0	17
						-15												1		0	30
							-30											1		0	0
								-50										1		0	30
									-11									1		0	30
										-5								1		0	25
											-17							1		0	30
												-13						1		0	30
													-12					1		0	30
														-10				1		0	30
															-15			1		0	30
																-11		1		0	30
																	-12	1		0	30
																			1		419

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Poznámka: pro lepší názornost a přehlednost byly sloupce proměnných u_{ij} a p_{ij} v Excelu zakryty, protože se v nich pro nás nic nemění.

Do modelu bylo nastaveno $z = 0$. To znamená, že se nezhorší ani jedna proměnná. A proto, spustíme-li OpenSolver, výstup z modelu je totožný s obyčejným Steinerovým stromem výše. Účelová funkce má hodnotu 419 a do minimální kostry jsou vybrané hrany $x_{52}, x_{53}, x_{14}, x_{15}$. Jejich pozice v tabulce jsou vyznačeny oranžově. Tento scénář není v reálných situacích příliš obvyklý, naopak obvykle téměř nemožný. Čím více je proces závislý na lidech, zvedá se jeho neurčitost, a tedy doopravdy nelze s tímto scénářem, zvláště pak například v problémech projektového řízení, počítat. Na druhou stranu jej lze využít pro představu a porovnání, například v úvodní správě, pokud bychom reportovali nějaký projekt.

5.2.4 Scénář s 1, 2, 3 zhoršenými proměnnými

Tabulka 17 Scénář s 1 zhoršenou proměnnou

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	30	449		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	1	-1	0	0
-40																		1		0	30
	-60																	1		0	30
		-17																1		0	13
			-60															1		0	30
				-50														1		0	30
					-13													1		0	17
						-15												1		0	30
							-30											1		0	0
								-50										1		0	30
									-11									1		0	30
										-5								1		0	25
											-17							1		0	30
												-13						1		0	30
													-12					1		0	30
														-10				1		0	30
															-15			1		0	30
																-11		1		0	30
																	-12	1		0	30
																			1		**449**

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Tabulka 18 Scénář se 2 zhoršenými proměnnými

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	17	466		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	2	-1	0	0
-40																		1		0	17
	-60																	1		0	17
		-17																1		0	0
			-60															1		0	17
				-50														1		0	17
					-13													1		0	4
						-15												1		0	17
							-30											1		0	0
								-50										1		0	17
									-11									1		0	17
										-5								1		0	12
											-17							1		0	17
												-13						1		0	17
													-12					1		0	17
														-10				1		0	17
															-15			1		0	17
																-11		1		0	17
																	-12	1		0	17
																			1		**466**

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Tabulka 19 Scénář se 3 zhoršenými proměnnými

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	13	479		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	3	-1	0	0
-40																		1		0	13
	-60																	1		0	13
		-17																1		0	0
			-60															1		0	13
				-50														1		0	13
					-13													1		0	0
						-15												1		0	13
							-30											1		0	0
								-50										1		0	13
									-11									1		0	13
										-5								1		0	8
											-17							1		0	13
												-13						1		0	13
													-12					1		0	13
														-10				1		0	13
															-15			1		0	13
																-11		1		0	13
																	-12	1		0	13
																			1		479

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Tabulka 20 Scénář se 4 zhoršenými proměnnými

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	484		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	4	-1	0	0
-40																		1		0	0
	-60																	1		0	0
		-17																1		0	0
			-60															1		0	0
				-50														1		0	0
					-13													1		0	0
						-15												1		0	0
							-30											1		0	0
								-50										1		0	0
									-11									1		0	0
										-5								1		0	0
											-17							1		0	0
												-13						1		0	0
													-12					1		0	0
														-10				1		0	0
															-15			1		0	0
																-11		1		0	0
																	-12	1		0	0
																			1		484

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

V žádném z těchto případů se neměnila minimální kostra. Je to tím, že i se zhoršením se modelu vyplatí vést kostru skrze originální hrany, vybrané i původním modelem. Postupně se k hodnotě účelové funkce také připočítává zhoršení jednotlivých hran. V případě $z = 1$ jsme vlastně

zadali, že se zhorší hrana x_{14} , která je z kostry nejdražší. Její odchylka má hodnotu 30, proto se účelová funkce zhorší na 449.

Dále jsou ceny odchylek zhoršovaných hran přidávány k účelové funkci v pořadí x_{53}, x_{52}a x_{15}. Vzhledem k tomu, že ani při $z = 4$ se nemění výběr hran do minimální kostry, lze říct, že ani při situaci, kdy jsou na všech trasách z kostry objížďky, nevyplatí se jezdit po obvodu.

5.2.5 Scénář s 5 zhoršenými proměnnými

Tabulka 21 Scénář s 5 zhoršenými proměnnými

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	484		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	5	-1	0	0
-40																		1		0	0
	-60																	1		0	0
		-17																1		0	0
			-60															1		0	0
				-50														1		0	0
					-13													1		0	0
						-15												1		0	0
							-30											1		0	0
								-50										1		0	0
									-11									1		0	0
										-5								1		0	0
											-17							1		0	0
												-13						1		0	0
													-12					1		0	0
														-10				1		0	0
															-15			1		0	0
																-11		1		0	0
																	-12	1		0	0
																			1		484

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Porovnáme-li hodnotu účelové funkce při $z = 4$ a $z = 5$, vidíme, že jsou totožné. Tento scénář prokazuje, že pro tento algoritmus není podstatné, zhorší-li se hrany, které aktuálně nejsou zahrnuty do minimální kostry. Mohly by se zhoršit i úplně všechny hrany, a přesto by hodnota účelové funkce zůstala stejná. V realitě to lze vysvětlit takto. Pokud bychom jeli trasou A z Prahy do Brna o délce 208 km a předběžně bychom počítali s objížďkou o délce 15 km a alternativou trasy je trasa B o délce 240 km, pak by se nám ani v nejhorším případě (nutnost jet po objížďce) nevyplatilo o trase B uvažovat.

5.2.6 Scénář se změnou minimální kostry

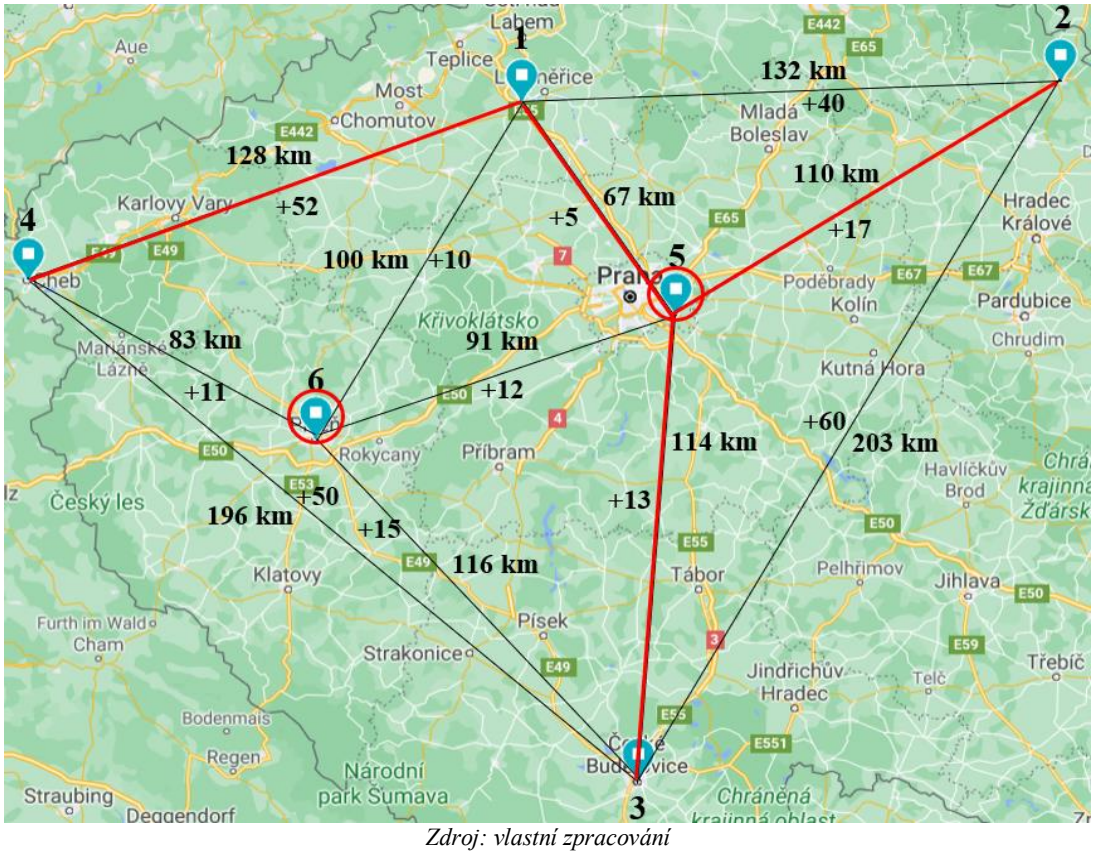
Pro účely lepšení názornosti tohoto scénáře byla změněna hodnota odchylky pro hranu x_{14}.

Tabulka 22 Scénář beze změny kostry

0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	506		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	4	-1	0	0
-40																		1		0	0
	-60																	1		0	0
		-17																1		0	0
			-60															1		0	0
				-50														1		0	0
					-13													1		0	0
						-15												1		0	0
							-52											1		0	0
								-50										1		0	0
									-11									1		0	0
										-5								1		0	0
											-17							1		0	0
												-13						1		0	0
													-12					1		0	0
														-10				1		0	0
															-15			1		0	0
																-11		1		0	0
																	-12	1		0	0
																			1		506

Zdroj: vlastní zpracování + výstup doplňku OpenSolver

Obrázek 10 Minimální kostra s odchylkami při ÚF = 506



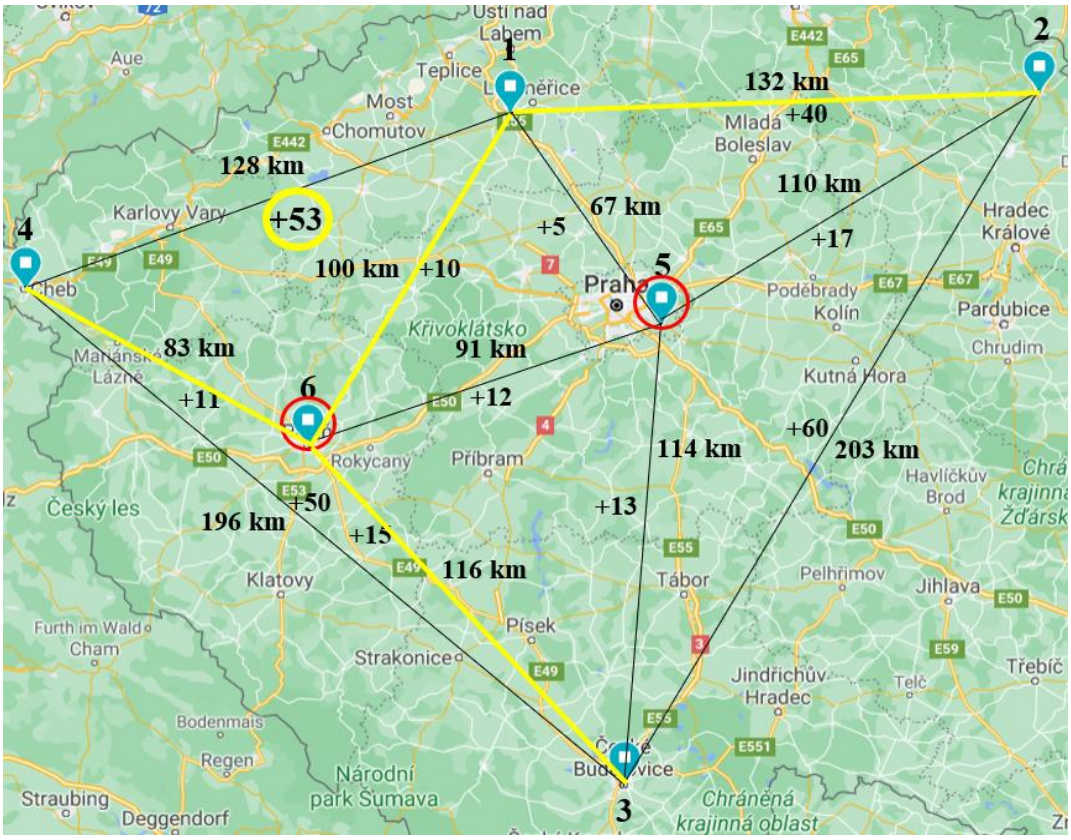
Zdroj: vlastní zpracování

Tabulka 23 Scénář se změnou kostry

1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	10	507		
x12	x32	x52	x23	x43	x53	x63	x14	x34	x64	x15	x25	x35	x65	x16	x36	x46	x56	z	E	b	breal
132	203	110	203	196	114	116	128	196	83	67	110	114	91	100	116	83	91	4	-1	0	0
-40																		1		0	0
	-60																	1		0	10
		-17																1		0	10
			-60															1		0	10
				-50														1		0	10
					-13													1		0	10
						-15												1		0	0
							-53											1		0	10
								-50										1		0	10
									-11									1		0	0
										-5								1		0	10
											-17							1		0	10
												-13						1		0	10
													-12					1		0	10
														-10				1		0	0
															-15			1		0	10
																-11		1		0	10
																	-12	1		0	10
																			1		507

Zdroj: vlastní zpracování + Microsoft Excel

Obrázek 11 Minimální kostra s odchylkami při ÚF = 507



Zdroj: vlastní zpracování

V těchto dvou scénářích je patrné, jak jedna minimální změna v odchylce může způsobit výměnu hran minimální kostry. Máme-li v obou scénářích $z = 4$ a měníme odchylku hrany x_{14} z 52 na 53, zhorší se hodnota účelové funkce zdánlivě logicky pouze o 1. Při lepším prozkoumání ale vidíme, že jsou najednou v minimální kostře jiné hrany a to x_{12}, x_{16}, x_{64} a x_{16}. Stalo se tak proto, že zvýšením chyby x_{14} na 53 se cena této hrany se zhoršením dostala na takovou úroveň, že je levnější ji obejít skrze hrany x_{16} a x_{64}. Směrem dolů, je potřebné dojít do uzlu 3, ovšem hrana x_{34} je poměrně drahá a jít z uzlu 6 skrze uzel 5 není výhodné, protože je to další Steinerův uzel, tedy do něj chodit nemusíme a cestu by v tomto případě pouze prodlužoval. Další variantou průchodu do uzlu 3 je z uzlu dva, nicméně tady je stejný problém, jako u hrany x_{34}, hrana x_{23} je drahá. Zbývá tedy hrana x_{63}. Nyní máme obsazené uzly 1, 3 a 4 a zůstává zapojit uzel 2. Nabízejí se varianty součtu hran z uzlů 1 a 3 skrze Steinerův uzel 5, ovšem ani jeden ze součtů cen není menší, než kterákoliv přímá hrana z uzlů 1 a 3. Co se týče ceny je výrazně levnější hrana x_{12}, která je také vybrána jako poslední hrana minimální kostry grafu.

Kdybychom porovnávali například pouze dvě cesty, které by spojovaly totožné uzly (byly by si navzájem alternativou), přičemž by byla první cesta tvořena hranou S o ceně 6 a odchylce 5 a druhá cesta s hranami T a U o cenách 8 a 2. Hrana S by byla zařazena do minimální kostry, přičemž hrana T a U nikoliv, pak by se mohlo stát toto. Ve chvíli, kdyby hodnota z byla taková, že by nezhoršovala hranu S, tak by tato hrana nadále zůstávala součástí minimální kostry. Ve chvíli, kdy uživatel modelu zhorší z právě natolik, že se ke zhoršeným hranám přidá právě jen hrana S, pak by se účelová funkce grafu zhoršila o 11. Kdyby ale neexistovala alternativa v podobě druhé cesty skrze hrany T a U, které spolu stojí pouze 10 jednotek.

Ovšem i v tomto velmi malém grafu nelze uvažovat takto jednoduše. Jak dokázaly scénáře i malá změna v odchylce nemusí nutně znamenat pouze výměnu jedné jediné hrany, ale může úplně vyměnit hrany minimální kostry grafu.

5.2.7 Výstup z malé ilustrativní úlohy

Do modelu lze vkládat různá zhoršení, odchylky i jednoduše měnit například ceny hran, při zjištění chyby v přepisu vstupů problému do tabulky Excelu. Proto můžeme jednoduše modelovat různé situace, které mohou nastat. Ideální scénář pro představu před zahájením projektu, nebo běžnou situaci, kdy předpokládáme zhoršení 50 % hran kostry. OpenSolver při

každém spuštění modelu přepočítá celý model, aby zjistil, zda hrana, kterou jsme právě přidali do zhoršení, nezvýší účelovou funkci natolik, že existuje levnější alternativa kostry. Pokud ano, vymění sám hrany kostry a vypočítá novou cenu účelové funkce.

5.3 Reálný případ užití robustního Steinerova stromu

Pro případovou studii byl vybrán projekt Smíchov City. V programu Microsoft Excel se bude řešit plán rozvodných sítí v nové zástavbě. Smíchov City je projekt proměny okolí smíchovského nádraží developerské společnosti Sekyra Group v obytně kancelářský areál na ploše 20 hektarů. Tento projekt se letos po 15 letech přípravy začal realizovat. Během plánů a veřejné soutěže se projekt setkal i s velkým nepochopením ze strany veřejnosti v podobě petic proti výstavbě a studií s negativním postoji některých politických stran (http://praha5.zeleni.cz/smichov-city/). Přesto se tento rok začalo s prvními pracemi na projektu. Stavět se začne na severní straně u dopravního uzlu Na Knížecí.

V této úloze se bude řešit, jak budou v areálu rozvrženy rozvodné sítě, kterými uzly povede (tedy minimální kostru grafu) a samotnou cenu účelové funkce (tedy maximální cenu, kterou rozvodné sítě tohoto projektu budou stát).

Původně byl prostor v okolí Smíchovského nádraží dlouhé roky neudržovaný, využíval se například jako skládka materiálů, nebo prostor pro různé squaty. Dělo se tak i přesto, že je areál defacto v širším centru města, dobře dostupný veřejnou dopravou a v blízkosti městského okruhu. K tomu je v sousedství jednoho z nejvýznamnějších vlakových nádraží v Praze, které vypravuje vlaky západním i jižním směrem od města. Lokalita není přímo v historické části města, nicméně občanská vybavenost této oblasti je úplná. Jsou zde v blízkosti všechny typy dopravní obsluhy, školy, školky, lékařská péče, obchody, kina a divadla i sportoviště.

Na *Obrázku 12* je současný vzhled areálu, kde developer naplánoval výstavbu. Vpravo dole je vidět autobusové nádraží Na Knížecí, odkud jezdí mimo i dálkové spoje. Pohled je ze směru od centra, respektive od oblasti Anděla, směrem na jih, kde je v pozadí vidět údolí Vltavy, vpravo Dívčí hrady a Barrandov. Vlevo je částečně vidět Smíchovské nádraží, do kterého se napojuje železniční trať ze směru od Hlavního nádraží přes most nad ulicí Nádražní.

Obrázek 12 Aktuální vzhled areálu

Zdroj: https://www.zalepsismichov.cz/index.php/clanky/vsechny-clanky/98-smichov-city-setkani-s-developerem

A takto vypadá developerský projekt společnosti Sekyra Group (*Obrázek 13*). Jeho podoba je výsledkem dvou nadnárodních soutěží a kombinací práce řady ateliérů. V areálu mají být mimo obytných prostor také kanceláře, obchody, restaurace, nebo sportoviště. Celkově by kompletně dokončený projekt měl poskytnout bydlení nebo kanceláře až pro 15 tisíc osob, na celkové zastavěné ploše 400 tisíc m^2. Podle propočtů developera by tato investice měla stát kolem 20 miliard.

Obrázek 13 Projekt Smíchov City

Zdroj: https://www.zalepsismichov.cz/index.php/clanky/vsechny-clanky/98-smichov-city-setkani-s-developerem

Obrázek 14 Jižní část projektu Smíchov City

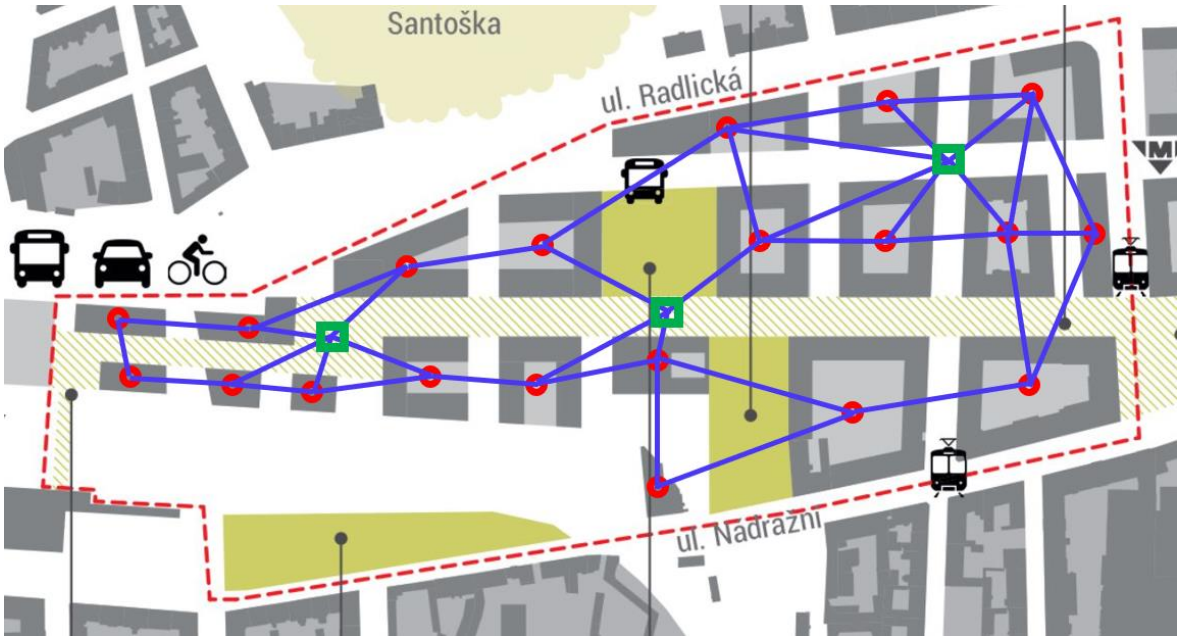
Zdroj: https://www.zalepsismichov.cz/index.php/clanky/vsechny-clanky/98-smichov-city-setkani-s-developerem

5.3.1 Zadání případové studie

Poznámka: developerský projekt společnosti Sekyra Group s názvem Smíchov City je reálný, nicméně v této diplomové práci je řešen pouze teoretický problém rozvodu sítí v této zástavbě. Developer nezadal autorce diplomové práce žádnou reálnou úlohu. Všechna data k projektu pochází z veřejně dostupných zdrojů.

V následující *Obrázku 15* je nakreslen graf rozvodné sítě. Červeně jsou význačné uzly a zeleně Steinerovy uzly. Steinerovy uzly jsou v této úloze reprezentací rozvodných budek mimo budovy a hrany k nim jsou levnější také z důvodu, že ve velké části nevedou pod povrchem v takové hloubce. Přesná poloha uzlů a vedení hran není pro řešení úlohy podstatná, potřebujeme znát především ceny hran, topologii a to, které uzly jsou význačné a které Steinerovi.

Obrázek 15 Graf rozvodné sítě



Zdroj: https://www.iprpraha.cz/uploads/assets/dokumenty/kps/brozury/brozura%20smichov_151208.pdf

Dále byla zjištěna přesná délka spojení jednotlivých uzlů pomocí mapových podkladů https://mapy.cz/, cena výkopových prací ze serveru https://www.bagrovani-praha.cz/cenik/, případně pomocí stránky https://www.adbasro.cz/kalkulacka a průměrná cena materiálů, z čehož byla vypočítána cena jednotlivých hran. K této ceně je započítaná i lidská práce.

Cena jednoho metru výkopu hlubokého tři metry je přibližně 2400 Kč, s tím, že počítáme i s odvozem části zeminy. Dále je pro zavedení sítí potřeba dalších strojů, jejichž hodina práce může stát v součtu až 1200 Kč/hodina. Díky stránce https://www.eprehledy.cz/prumerne_platy_podle_profese.php byla zjištěna cena práce dělníka, která je v průměru přibližně 150 Kč/hodinu a tím pádem nás četa může stát přibližně 1200 Kč/hodinu. Průměrně může práce (kompletně s výkopy, pokládkou materiálů, zapojení a zakopání výkopu) na jednom metru délky rozvodné sítě trvat 2 hodiny. Materiál na jeden metr rozvodů byl odhadnut na cenu 1350 Kč.

Tabulka 24 Ceny a odchylky pro model případové studie

						stroje			**CENY HRAN**	
hrany		délka v m	cena za materiál v Kč	čas práce v hod	cena za práci v Kč	cena výkopu v Kč	další stroje cena v Kč	cena hrany v Kč	koeficient pro obtížnější práce v Kč	**ODCHYLKY** v Kč
4	**23**	40	54000	80	12000	96000	48000	210000	**216300**	34608
23	**22**	48	64800	96	14400	115200	57600	252000	**259560**	41530
22	**21**	37	49950	74	11100	88800	44400	194250	**200078**	32012
21	**20**	50	67500	100	15000	120000	60000	262500	**270375**	43260
20	**19**	39	52650	78	11700	93600	46800	204750	**210893**	33743
19	**18**	40	54000	80	12000	96000	48000	210000	**216300**	34608
18	**17**	62	83700	124	18600	148800	74400	325500	**335265**	53642
17	**16**	93	125550	186	27900	223200	111600	488250	**502898**	80464
16	**12**	90	121500	180	27000	216000	108000	472500	**486675**	77868
12	**11**	90	121500	180	27000	216000	108000	472500	**486675**	77868
11	**10**	90	121500	180	27000	216000	108000	472500	**486675**	77868
10	**9**	75	101250	150	22500	180000	90000	393750	**405563**	64890
9	**8**	79	106650	158	23700	189600	94800	414750	**427193**	68351
8	**7**	120	162000	240	36000	288000	144000	630000	**648900**	103824
7	**6**	74	99900	148	22200	177600	88800	388500	**400155**	64025
6	**5**	82	110700	164	24600	196800	98400	430500	**443415**	70946
5	**4**	55	74250	110	16500	132000	66000	288750	**297413**	47586
1	**5**	28	37800	56	8400	67200	33600	147000	**147000**	14700
1	**6**	35	47250	70	10500	84000	42000	183750	**183750**	18375
1	**20**	36	48600	72	10800	86400	43200	189000	**189000**	18900
1	**21**	24	32400	48	7200	57600	28800	126000	**126000**	12600
1	**22**	36	48600	72	10800	86400	43200	189000	**189000**	18900
2	**7**	61	82350	122	18300	146400	73200	320250	**320250**	32025
2	**15**	63	85050	126	18900	151200	75600	330750	**330750**	33075
2	**18**	20	27000	40	6000	48000	24000	105000	**105000**	10500
2	**19**	61	82350	122	18300	146400	73200	320250	**320250**	32025
3	**8**	85	114750	170	25500	204000	102000	446250	**446250**	44625
3	**9**	52	70200	104	15600	124800	62400	273000	**273000**	27300
3	**10**	59	79650	118	17700	141600	70800	309750	**309750**	30975
3	**13**	50	67500	100	15000	120000	60000	262500	**262500**	26250
3	**14**	50	67500	100	15000	120000	60000	262500	**262500**	26250
3	**15**	82	110700	164	24600	196800	98400	430500	**430500**	43050
16	**18**	90	121500	180	27000	216000	108000	472500	**486675**	77868
13	**12**	81	109350	162	24300	194400	97200	425250	**438008**	70081
13	**11**	54	72900	108	16200	129600	64800	283500	**292005**	46721
13	**10**	87	117450	174	26100	208800	104400	456750	**470453**	75272
13	**14**	65	87750	130	19500	156000	78000	341250	**351488**	56238

14	**15**	65	87750	130	19500	156000	78000	341250	**351488**	56238
15	**8**	64	86400	128	19200	153600	76800	336000	**346080**	55373
								12663000		

Zdroj: vlastní zpracování

Byl připočítán koeficient pro náročnější práce, které vedou ve velké části pod nebo skrze zastavěné plochy, které jsou náročnější jak cenově, tak materiálově, kdy je potřeba výkop zajistit lépe než výkop, který nad sebou nenese větší tíhy. Odchylky jsou určeny možným zpožděním prací, vyplýtvaným materiálem nebo chybami při pracích a to tak, že k hranám, které vedou mimo zastavěné plochy bylo připočteno maximální zhoršení 10 % a pro hrany pod zastavěnými plochami je maximální zhoršení hran (tedy odchylka) určena na 16 %.

5.3.2 Případová studie - algoritmus Steinerova stromu

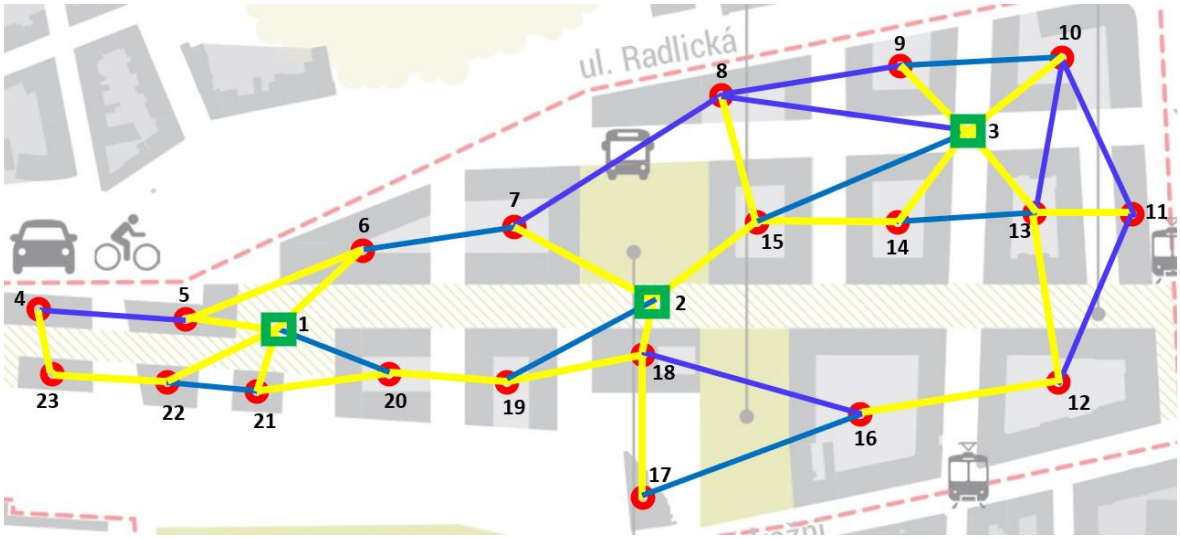
Pro Steinerův strom jsme vyplnili Excel, tedy to samé, co pro malou úlohu, nicméně v podstatně větším měřítku a na problémů s reálnými (přestože lehce zaokrouhlenými) vstupy. V této úloze je počet uzlů $n = 23$, počet hran $m = 39$ a počet význačných hran je $p = 36$. Z toho vyplývá, že jsou v projektu tři Steinerovi uzly, které reprezentují rozvodné boudy úplně mimo plochu budov, jak je vidět na *Obrázku 15* (kde jsou označeny zelenými čtverci). Pro tuto fázi v modelu ještě nejsou zahrnuty odchylky ani žádná neurčitost, tedy model ještě není robustní. Jde nám zatím čistě o to, najít minimální kostru rozvodné sítě a určit čistou hodnotu účelové funkce.

V této části práce nebudu vkládat žádné výstupy, respektive tabulky program Excel, jelikož je jejich rozměr na vkládání nepraktický (tabulka o rozměrech 242 na 100 polí).

Po správném zapsání hodnot do tabulky, vyplnění pravé strany, doplnění o vzorec pro skalární součin $breal$ ku b, můžeme vyplnit i model OpenSolveru. Poté lze model spustit a nechat OpenSolver úlohu vyřešit. V této chvíli je patrné, že OpenSolver pracuje s podstatně větším počtem vstupů a variant, protože jeho výpočet trvá až pět krát déle. Na následujícím *Obrázku 16* je vypočítaná minimální kostra, znázorněná žlutě. Účelová funkce má hodnotu 5 932 449 Kč.

Opět je zde na první pohled patrné, že byly zahrnuty všechny význačné uzly, což byl předpoklad přípustnosti modelu. Vidíme, že do kostry byly i zahrnuty všechny Steinerovy uzly a jejich využití zlevňuje cenu účelové funkce.

Obrázek 16 Případová studie - Steinerův strom



Zdroj: vlastní zpracování + OpenSolver

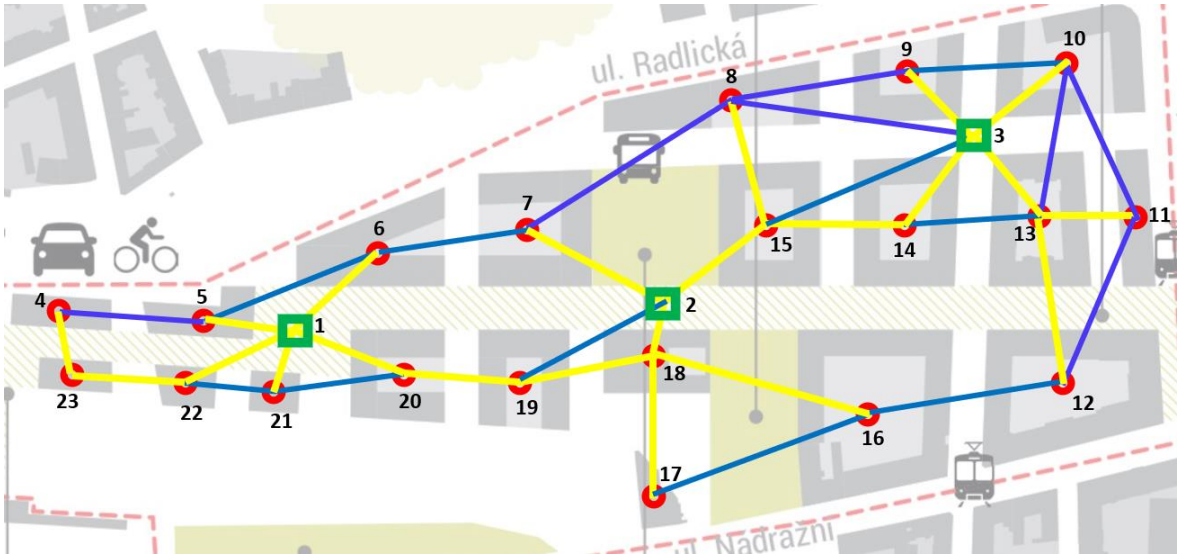
5.3.3 Případová studie – převedení na robustní model

Ve chvíli, kdy máme hotový Steinerův strom a jeho úlohu vyřešenou i OpenSolverem, můžeme model převést na robustní verzi. Jako v malém příkladu přidáme proměnné p_{ij}, zhoršení z a odchylky δ. Odchylky máme zadané v *Tabulce 24*.

Na *Obrázku 17* je vyřešený problém případové studie robustního Steinerova stromu. Určili jsme si, že se zhorší zhruba 30 % prací, což z 23 hran odpovídá přibližně 7 hranám ($z = 7$). Je vidět změna oproti *Obrázku 16*, který není robustní a nejsou v něm připuštěny žádné chyby. Do minimální kostry grafu se dostaly jiné hrany, ale stále byl zachován požadavek zahrnutí všech význačných hran do kostry. A opět jsou v kostře i Steinerovi uzly, což prokazuje, že se developerovi vyplatí postavit vymodelované rozvodné boudy.

Účelová funkce nového řešení se zhoršením je 6 252 527 Kč. Z toho vyplývá, že zhorší-li se 30 % hran, což může projektový tým předpokládat například z předchozích zkušeností, pak to developerskou společnost může stát proti původnímu plánu o 320 078 Kč navíc. V rozpočtu si tedy manažer může udělat adekvátní nárazník právě v této výši.

Obrázek 17 Případová studie - Robustní Steinerův strom



Zdroj: vlastní zpracování + OpenSolver

6 Výsledky a diskuse

Výsledkem případové studie je změna minimální kostry grafu při předpokládaném zhoršení průběhu prací o 30 %. Toto je výsledkem zapojení robustního přístupu, který donutí model změnit původní množinu hran v minimální kostře. Poté co byl OpenSolver spuštěn se vstupní hodnotou *z*, která odpovídá 30 % zhoršení, byl celý model přepočítán a byla jím zjištěna levnější minimální kostra. Došlo k tomu právě v důsledku zhoršení 30 % hran. A to tak, že cena některé z hran po přičtení své odchylky zvedne hodnotu účelové funkce natolik, že OpenSolver najde levnější kostru, nebo jen objížďku ke zhoršené hraně.

Výstupem vlastní práce je také obecně použitelný algoritmus robustního Steinerova stromu, jenž je okamžitě aplikovatelný na další reálné úlohy. Mimo plánování rozvodných sítí by šlo metodu využít například pro plánování cest v letecké, lodní, železniční i silniční dopravě, nebo na druhou stranu v projektovém řízení. Pokud by byl algoritmus použit z projektovém řízení, hrany by mohly reprezentovat činnosti mezi různými milníky (uzly), nebo dodanými komponentami systému či aplikace. Jejich cena by byla vyjádřena například využitím zdrojů, jak lidských, tak finančních a dalších. V tom případě si lze jednoduše představit odchylky jako zpoždění s pracemi, nebo chyba ve vývoji. Projektoví manažeři, mají-li již nějaké zkušenosti ve firmě, ve které pracují, by měli být schopni odhadnout, o kolik se kteří členové týmu mohou se svou prací opozdit, nebo zda jejich práce vykazuje větší počet chyb. Také mohou za svou praxi odpozorovat o kolik se může zpozdit dodavatel, o kolik bude stát víc vývoj konkrétního typu komponenty atd.

6.1 Výhody a nevýhody robustního Steinerova stromu

Velkou výhodou je, že je nyní algoritmus Steinerova stromu odolný vůči neurčitým vstupům. Také je příhodné, že lze pomocí OpenSolveru poměrně rychle a efektivně zjistit, co se stane, předpokládáme-li zhoršení o 10 %, 30%, 50 %, nebo 90 % atd. Vidíme hned, zda minimální kostra zůstane stejná, jen se zhorší cena účelové funkce, nebo model vymění kompletně hrany minimální kostry a tím nám řekne, že počítáme-li s těmito odchylkami a může se stát až tolik změn, nemá smysl jít původní kostrou. Je tedy vhodné rovnou vybrat kostru, kterou si nasimulujeme jako nejhorší pravděpodobnou variantu. Může to mít přínos v tom, že budeme mít připravenou rezervu, kterou v nejhorším případě spotřebujeme na skutečné zhoršení.

Kdybychom zůstali na původní minimální kostře a zhoršilo by se tolik hran, kolik jsme předpokládali, pak by byl výsledek dražší, než kdybychom brali v potaz modelem navrhnutou kostru. Zamezilo by se tím nečekaným výdajům.

Nevýhodou zatím je zadávání větších úloh do Excelu. To je podle mě velký prostor pro inovaci. Problém je v tom, že zadáváme-li do Excelu větší počet proměnných, stává se výrazně nepřehledným. Kupříkladu případová studie z této práce řešila problém a 39 hranách (to znamená minimálně 39 sloupců), což by bylo samo o sobě už trochu na hraně s přehledností, ale nejspíš by to ještě bylo v pořádku. Ovšem model ani zdaleka nekončí na čistém počtu hran. Co se týče sloupců, už v prvním oddíle je jich pro tuto případovou studii 2*39 (mínus hrany vstupující do kořenového uzlu). K nim se přidá počet uzlů, tady je to 23 a také proměnná p, přičemž platí, že každé x má své p. Ještě jsou v sloupcích proměnné z a E a pravá strana modelu se dvěma sloupci b a $breal$. Vzhledem k množství podmínek, které musí být splněny, je počet řádků modelu ještě větší. V řešené případové studii jsem se dostala na plochu o velikosti 317*177 buněk tabulky. Je tedy poměrně jednoduché udělat někde překlep, nebo přehlédnout špatně vložená data. Navrhovala bych vývoj podobného doplňku jako je OpenSolver, do kterého by se pouze zadaly hrany, jejich ceny a odchylky, jejich význačnost a Steinerovy uzly a poté by si uživatel pouze měnil hodnotu z a spouštěl jednotlivé varianty bez obav, že je někde v rozsáhlé tabulce Excelu někde chyba. Excel sám o sobě by se také v ještě větším počtu hran a vstupních dat mohl začít zasekávat, což na přehlednosti tabulky také nepřidá.

7 Závěr

Výsledkem práce je úspěšné spojení Steinerova stromu a robustního přístupu. Úspěch tohoto spojení je prokázán ve vlastní části práce nejprve malým ilustrativním příkladem, který je podrobně popsán a je na něm i vidět mechanismus změny kostry. Úspěšné spojení přístupů je patrné i v řešení případové studie, kdy model zhodnotil, že je pro developera výhodné zahrnout do projektu Steinerovy uzly, které reprezentují rozvodné boudy.

Výstupem práce je tedy nový algoritmus robustního Steinerova stromu. Je odolný vůči neurčitosti vstupů, odchylkám a je schopen uvažovat pro kostru i uzly, které přímo nemusí být součástí minimální kostry.

Došlo se k závěru, že je algoritmus využitelný pro reálné problémy a mohl by být užitečný pro mnoho typů úloh. Stálo by tedy za námahu robustní Steinerův strom dále zkoumat a rozvíjet, nebo jak bylo navrženo, napsat na něj stroj, do kterého by se vložily vstupní hodnoty a uživatel už by poté pouze simuloval různé scénáře dle svého uvážení.

8 Seznam použitých zdrojů

Ben-Tal, A., El Ghaoui, L. and Nemirovski A. (2009), *Robust Optimization*, Princeton University Press, Princeton. [online], Available: ProQuest Ebook Central. [21 November 2020].

Bertsimas, D. and Brown, D.B. (2009), "Constructing Uncertainty Sets for Robust Linear Optimization", *Operations research*, vol. 57, no. 6, pp. 1483-1495,1532.

Bertsimas, D. and Sim, M. (2003). "Robust discrete optimization and network flows". *Mathematical Programming*, 98(1–3), pp.49–71.

Bertsimas, D. and Sim, M. (2004). "The Price of Robustness". *Operations Research*, 52(1), pp.35–53.

Brožová, H. and Houška, M. (2002). *Základní metody operační analýzy*, Praha: Česká zemědělská univerzita.

Büsing, C. and D'Andreagiovanni, F. (2013). Robust Optimization under Multi-band Uncertainty - Part I: Theory. *arXiv:1301.2734 [cs, math]*. [online], Available: https://arxiv.org/abs/1301.2734.

Büsing, C. and D'Andreagiovanni, F. (2013). "A New Theoretical Framework for Robust Optimization Under Multi-Band Uncertainty". *Operations Research Proceedings*, pp.115–121.

Cieslik, D. (2001), The Steiner Ratio, Springer, New York, NY. [online], Available: ProQuest Ebook Central. [21 November 2020].

Cook, S. (2000). *The P versus NP problem*. [online], CiteSeer. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.98.358 [21 Nov. 2020].

D'Andreagiovanni, F., Krolikowski, J. and Pulaj, J. (2015). "A fast hybrid primal heuristic for multiband robust capacitated network design with multiple time periods". *Applied Soft Computing*, 26, pp.497–507.

D’Andreagiovanni, F. and Raymond, A. (2014). "Multiband Robust Optimization and its Adoption in Harvest Scheduling". *FORMATH*, 13(0), pp.97–122.

Demel, J. (2015). *Grafy a jejich aplikace,* Vyd. 2., (Vlastním nákladem 1.)., Libčice nad Vltavou: J. Demel.

Diané, M. and Plesník, J. (1993). An integer programming formulation of the Steiner problem in graphs. *ZOR Zeitschrift for Operations Research Methods and Models of Operations Research,* 37(1), pp.107–111.

Du, D.-Z., Smith, J.M. and Rubinstein, J.H. eds. (2000). *Advances in Steiner Trees*. *Combinatorial Optimization*. Boston, MA: Springer US. [online], Available: https://link.springer.com/book/10.1007%2F978-1-4757-3171-2 [21 Nov. 2020].

Dunning, I. and Mason, A.J. (2010). *(PDF) OpenSolver: Open Source Optimisation for Excel*. [online], ResearchGate. Available: https://www.researchgate.net/publication/266484155_OpenSolver_Open_Source_Optimisation_for_Excel.

Fábry, J. (2011). *Matematické modelování*, Praha: Professional Pub.

Fortnow, L. (2009). "The status of the P versus NP problem". *Communications of the ACM*, 52(9), pp.78–86.

Goldreich, O. (2010) *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511761355.

Gros, I. (2003). *Kvantitativní metody v manažerském rozhodování*, Praha: Grada.

Chartrand, G., Lesniak, L. and Zhang, P. (2016). *Graphs & digraphs* Sixth edition., Boca Raton: CRC Press.

Jablonský, J. (2007). *Operační výzkum : kvantitativní modely pro ekonomické rozhodování*. Praha: Professional Publishing.

Kolář, J. (2009). *Teoretická informatika*, V Praze: České vysoké učení technické.

Kouvelis, P. and Yu, G. (1997). *Robust Discrete Optimization and Its Applications*. *Nonconvex Optimization and Its Applications*. Boston, MA: Springer US.

Krichen, S, and Chaouachi, J. (2014), Graph-Related Optimization and Decision Support Systems, John Wiley & Sons, Incorporated, Somerset. [online], Available: ProQuest Ebook Central. [21 November 2020].

Kubišová, A. (2014). *Operační výzkum*, Jihlava: Vysoká škola polytechnická Jihlava.

Linda, B. and Volek, J. (2014). *Lineární programování*, Vyd. 5., Pardubice: Univerzita Pardubice.

Linda, B. and Volek, J. (2012). *Teorie grafů - aplikace v dopravě a veřejné správě*, Pardubice: Univerzita Pardubice.

Marchese, A. and Massaccesi, A. (2014). The Steiner tree problem revisited through rectifiable G-currents. *Advances in Calculus of Variations*, [online], Available: https://arxiv.org/pdf/1408.2696.pdf [21 Nov. 2020].

Mulvey, J.M., Vanderbei, R.J. and Zenios, S.A. (1995). "Robust Optimization of Large-Scale Systems". *Operations Research*, 43(2), pp.264–281.

OpenSolver for Excel. (2020). *OpenSolver for Excel*. [online], Available: https://opensolver.org/ [21 Nov. 2020].

P versus NP problem. (2020). *Britannica Academic*. Retrieved 21 November 2020, [online], Available: https://academic-eb-com.ezproxy.techlib.cz/levels/collegiate/article/P-versus-NP-problem/475865.

Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems. (1999). *Journal of Computer and System Sciences*, [online], 58(1), pp.193–210. Available: https://www.sciencedirect.com/science/article/pii/S0022000098916051 [21 Nov. 2020].

Prömel, H.J. and Steger, A. (2002). *The Steiner Tree Problem: A Tour through Graphs, Algorithms, and Complexity*. [online], *www.springer.com*. Vieweg+Teubner Verlag. Available: https://www.springer.com/gp/book/9783528067625 [21 Nov. 2020].

Saltzman, M., Ladányi, L. and Ralphs, T. (2004). *The COIN-OR Open Solver Interface: Technology Overview*. [online], Available: https://www.researchgate.net/publication/267817357_The_COIN-OR_Open_Solver_Interface_Technology_Overview.

Sierksma, G. and Zwols, Y. (2015). *Linear and integer optimization: theory and practice* 3rd edition., Boca Raton: CRC Press, Taylor & Francis Group.

Šubrt, T. (2015). *Ekonomicko-matematické metody*, Upravené vyd. 2., Plzeň: Vydavatelství a nakladatelství Aleš Čeněk.

Tsui, T. (2018). Two theorems about the P versus NP problem. *arXiv:1805.01755 [cs, math]*. [online], Available: https://arxiv.org/abs/1805.01755 [21 Nov. 2020].

Uday, A. (2018). A dual identity based symbolic understanding of the Godel's incompleteness theorems, P-NP problem, Zeno's paradox and Continuum Hypothesis. *arXiv:1807.09600 [math]*. [online], Available: https://arxiv.org/abs/1807.09600.

Van Laarhoven, J.W. (2010). Exact and heuristic algorithms for the Euclidean Steiner tree problem. *Theses and Dissertations*. [online], Available: https://ir.uiowa.edu/etd/755/ [21 Nov. 2020].

Xiong, B. and Zheng, Z. (2010). *Graph theory*, Shanghai: East China Normal University Press.

Yaman, H., Karaşan, O.E. and Pınar, M.Ç. (2001). "The robust spanning tree problem with interval data". *Operations Research Letters,* 29(1), pp.31–40.