ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

Bc. Věra Jindrová

Řešení úlohy minimální kostry grafu s omezeními

Diplomová práce

2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
d ě k a n
Konviktská 20, 110 00 Praha 1

K617 ..... Ústav logistiky a managementu dopravy

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Věra Jindrová

Kód studijního programu a studijní obor studenta:

N 3710 – LO – Logistika, technologie a management dopravy

Název tématu (česky): **Řešení úlohy minimální kostry grafu s omezeními**

Název tématu (anglicky): Solution of the Constrained Minimal Spanning Tree Problem

Zásady pro vypracování

Při zpracování diplomové práce se řiďte osnovou uvedenou v následujících bodech:

- Úloha kostry grafu, minimální kostry grafu, minimální kostry grafu s omezeními
- Metody řešení - exaktní přístup, metaheuristiky
- Exaktní metoda - získání optimálního řešení
- Metoda Tabu Search obecně
- Rešeršní část - metaheuristika Tabu Search
- Aplikace Tabu Search na minimální kostru s omezeními
- Srovnání výsledků
- Úvaha o vlivu a rozměru aspiračního kritéria

Rozsah grafických prací: podle pokynů vedoucí diplomové práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury:

GLOVER, F., LAGUNA, M. Tabu search. Springer US, Vol. 1-3, ISBN 978-1-4613-7987-4, 1999.

SINGH, A. & BAGHEL, A. S. New Metaheuristic Approaches for the Leaf-Constrained Minimum Spanning Tree Problem. Asia - Pacific Journal of Operational Research, 25(4), 575-589, 2008.

Vedoucí diplomové práce: **Ing. Denisa Mocková, Ph.D.**

Datum zadání diplomové práce: **30. června 2014**
(datum prvního zadání této práce,které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **31. května 2015**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

L. S.

prof. Ing. Petr Moos, CSc.
vedoucí
Ústavu logistiky a managementu dopravy

prof. Dr. Ing. Miroslav Svítek
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Věra Jindrová
jméno a podpis studenta

V Praze dne .....30. června 2014

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 28. května 2015 ……

Podpis

Poděkování

Ráda bych touto cestou poděkovala všem, kteří mi byli nápomocni během zpracovávání této práce. Zejména děkuji doc. Ing. Denise Mockové, Ph.D. za odborné vedení a za rady, které mi poskytovala nejen během psaní této DP, ale po celou dobu mého studia.

Dále také velice děkuji mé rodině, přátelům a známým za trpělivost a podporu, které se mi dostávalo po celou dobu studia.

ABSTRAKT

Název práce: Řešení úlohy minimální kostry grafu s omezeními
Autor: Bc. Věra Jindrová
Ústav: Ústav logistiky a managementu dopravy (K617)
Vedoucí práce: doc. Ing. Denisa Mocková, Ph.D.

Předmětem této práce je představení metaheuristiky Tabu Search na příkladě minimální kostry grafu s omezeními malého rozsahu. Čtenář může následovat jednotlivé detailně popsané kroky a osvojit si tak postup řešení pomocí metaheuristiky Tabu Search. Na základě pochopení principu fungování metaheuristiky Tabu Search je možné tento přístup aplikovat na složitější úlohy. Rešeršní část poskytuje přehled problémů, které se pomocí dané metaheuristiky řešily a také které se v současné době začínají nově řešit. Tato proměna poskytuje nový náhled na zmíněnou metaheuristiku a nabízí se zavedení nových principů a nástrojů k efektivnímu řešení nově zvolených úloh.

KLÍČOVÁ SLOVA

Minimální kostra s omezeními, exaktní řešení, metaheuristiky, Tabu Search

ABSTRACT

Title: Solution of the Constrained Minimal Spanning Tree Problem
Author: Bc. Věra Jindrová
Department: Department of Logistics and Transport Management (K617)
Supervisor: doc. Ing. Denisa Mocková, Ph.D.

The object of this thesis is to show a metaheuristic approach called Tabu Search on an example of a minimum spanning tree with minor restrictions. The reader can follow the individual steps described in detail and learn the solving process using a metaheuristic approach Tabu Search. Based on the understanding of this approach of Tabu Search it can be applied to more complex tasks. The search section provides an overview of the problems which were solved by Tabu Search and which are currently starting a new deal. This transformation provides a new perspective on the referred metaheuristic and offers to introduce new principles and tools to effectively deal with the newly selected tasks.

KEY WORDS

Minimal Spanning Tree, exact solving method, metaheuristics, Tabu Search

Obsah

SEZNAM POUŽITÝCH ZKRATEK ... VII

1 ÚVOD ... 1

2 ÚLOHA KOSTRY GRAFU A JEJÍ MODIFIKACE ... 3

2.1 KOSTRA GRAFU ... 3
2.2 MINIMÁLNÍ KOSTRA GRAFU ... 4
2.2.1 Historický vývoj ... 5
2.2.2 Formulace úlohy ... 5
2.2.3 Matematický model úlohy ... 6
2.2.4 Přehled algoritmů pro minimální kostru ... 7
2.3 MINIMÁLNÍ KOSTRA GRAFU S OMEZENÍMI ... 9

3 METODY ŘEŠENÍ ... 11

3.1 EXAKTNÍ METODY ŘEŠENÍ ... 13
3.2 HEURISTIKY ... 13
3.3 METAHEURISTIKY ... 15
3.3.1 Definice ... 16
3.3.2 Specializace metaheuristik pro optimalizační problémy ... 18
3.3.3 Druhy metaheuristických algoritmů ... 18
3.3.4 Rozdíly mezi metaheuristickými koncepty ... 20

4 EXAKTNÍ ŘEŠENÍ PŘÍKLADU MINIMÁLNÍ KOSTRY GRAFU S OMEZENÍMI ... 21

5 TABU SEARCH ... 25

5.1 METODA TABU SEARCH OBECNĚ ... 25
5.2 REŠERŠE – SHRNUTÍ ZÁKLADNÍCH POZNATKŮ A HISTORIE TABU SEARCH ... 26
5.3 ALGORITMUS METODY TABU SEARCH ... 33
5.3.1 Algoritmus metody lokálního prohledávání ... 33
5.3.2 Algoritmus metody tabu prohledávání ... 34

6 APLIKACE TABU SEARCH NA KOSTRU GRAFU S OMEZENÍMI ... 36

6.1 POUŽITÍ KRÁTKODOBÉ PAMĚTI ... 36
6.2 POUŽITÍ STŘEDNĚ-DOBÉ PAMĚTI ... 52
6.3 SROVNÁNÍ OPTIMÁLNÍHO ŘEŠENÍ S ŘEŠENÍM POMOCÍ METAHEURISTIKY ... 61

7 ÚVAHA O VLIVU A ROZMĚRU ASPIRAČNÍHO KRITÉRIA ... 63

8 ZÁVĚR ... 64

LITERATURA ... 66

SEZNAM OBRÁZKŮ ... 71

SEZNAM TABULEK ... 72

Seznam použitých zkratek

ACO	Ant Colony Optimization, *Optimalizace mravenčí kolonie*
ATSP	Asymetric Traveling Salesman Problem, *Asymetrická úloha obchodního cestujícího*
BDCMST	Bounded Diameter Minimum Spanning Tree
DCMST	Degree Constrained Minimum Spanning Trees
EA	Evolution Algorithm, *Evoluční algoritmy*
LCMST	Leaf Constrained Minimal Spanning Tree, *Úloha minimální kostry s omezeným počtem listů*
MST	Minimal Spanning Tree, *Minimální kostra grafu*
NP	Not Polynomial, *nepolynomiální*
NP-C	Not Polynomial – Complete, *nepolynomiální kompletní*
NP-H	Not Polynomial – Hard, *nepolynomiální těžké*
OŘ	*Optimální řešení*
P	Polynomial, *polynomiální*
SMT	The Steiner Minimal Tree, *Šteinerův minimální strom*
STSP	Symetric Traveling Salesman Problem, *Symetrická úloha obchodního cestujícího*
TS	Tabu Search, *Zakázané porhledávání*
TSP	Traveling Salesman Problem, *Problém obchodního cestujícího*
ÚF	*Účelová funkce*
VNS	Variable Neighbourhood Search, *Variabilní prohledávání okolí*
VRP	Vehicle Routing Problem, *Rozvozní úloh*

1 Úvod

Náplň této práce spadá do oboru operačního výzkumu, který je možné charakterizovat jako vědecký nástroj na podporu rozhodování ve složitých reálných systémech. Jedním ze základních principů operačního výzkumu je hledání optimálního řešení zkoumaného problému s využitím aparátu matematického modelování. Mezi nejvýznamnější disciplíny operačního výzkumu se především řadí lineární, nelineární a dynamické programování, teorie her, teorie grafů, teorie rozvrhů, teorie front, teorie zásob, teorie obnovy, vícekriteriální rozhodování a další. Počátky zmíněných disciplín se datují mezi 30. a 40. léta 20. století. Mnoho známých úloh z oblasti operačního výzkumu bylo vyřešeno až během druhé světové války, poté v poválečném období a také později s rozvojem informačních technologií.

Optimalizační problémy se vyskytují v celé řadě odvětví lidské činnosti, např. v dopravě, ekonomii, obchodu, strojírenství, průmyslu či medicíně. Mnohé optimalizační úlohy, nejen dopravní, nejsou pro svoji častou vysokou složitost a komplexnost řešitelné exaktními metodami v požadovaném reálném čase. Z praktického hlediska je užitečné nalézt alespoň takové řešení, které je blízké optimu. To umožňují různé heuristické a metaheuristické přístupy. Praktická část práce je zaměřena právě na řešení úlohy minimální kostry grafu s omezeními pomocí metaheuristiky Tabu Search.

Úloha minimální kostry grafu s omezeními spadá do disciplíny teorie grafů, jejíž aparát je důležitým prostředkem pro zachycení a analýzu struktury zkoumaného systému.. Algoritmy pro hledání optimálních cest na grafech jsou využívány v rozsáhlé třídě matematických modelů operačního výzkumu a také jednou z nejčastěji vyhledávaných skupin algoritmů s širokým uplatněním zejména v oblasti silniční dopravy. Rovněž další oblasti teorie grafů (konstrukční úlohy na grafech, metody hledání kritické cesty, problematika okružních jízd apod.) slouží k řešení mnoha problémů z oblasti dopravy. Většina algoritmů je v reálných situacích řešena právě heuristickými či metaheuristickými přístupy, které nezaručují nalezení globálního optima, ale jsou schopny nalézt kvalitní řešení, které se hledanému optimu blíží.

Úlohu hledání minimální kostry grafu lze aplikovat např. při hledání nejlevnějšího propojení dané oblasti telefonním kabelem, dopravní či elektrickou sítí apod. Algoritmus hledání minimální kostry grafu může být využitý také v určitých krizových situacích – např. při níž dojde k částečnému nebo úplnému zneprůchodnění dopravních komunikací (zasypání, zatopení, sněhová kalamita apod.). Pro zajištění nejnutnějšího propojení strategických míst je zapotřebí alespoň částečně obnovit komunikační síť tak, aby každý ze strategických bodů byl propojen s ostatními. Ohodnocení hran znázorňuje vzdálenosti mezi krajními vrcholy hrany, tedy mezi jednotlivými strategickými body. Ohodnocení může být uvedeno v jednotkách určující

vzdálenost, čas či náklady na absolvování dané hrany. Nalezením minimální kostry grafu jsou určeny komunikace, které mají být bezprostředně zprůjezdněny, aby byly jednotlivé vrcholy dané sítě navzájem propojeny.

Text práce je v první části věnován základnímu představení úlohy kostry grafu a jejích modifikací, a to minimální kostry grafu a minimální kostry grafu s omezeními. Ve stručnosti jsou popsány tři základní algoritmy pro hledání minimální kostry grafu včetně jejich složitosti pro možnou pozdější návaznost v praktické části práce. V další části je dán prostor pro prezentaci a vzájemné porovnání možných metod řešení optimalizačních problémů, kterými jsou exaktní metody, heuristické a metaheuristické přístupy.

Praktická část práce se dělí na získání optimálního řešení pomocí exaktní metody a na řešení úlohy pomocí metaheuristiky Tabu Search. Následně jsou porovnány výsledky těchto dvou odlišných přístupů.

Práce si dává za úkol představení metaheuristiky Tabu Search na jednoduchém příkladě minimální kostry s omezeními. Čtenář může následovat jednotlivé detailně popsané kroky a osvojit si tak postup řešení pomocí metaheuristiky Tabu Search a poté jej aplikovat na složitější úlohy. Rešeršní část poskytuje přehled problémů, které se pomocí dané metaheuristiky řešily a také které se v současné době začínají nově řešit. Tato proměna poskytuje nový náhled na zmíněnou metaheuristiku a nabízí se zavedení nových principů a nástrojů k efektivnímu řešení nově zvolených úloh.

2 Úloha kostry grafu a její modifikace

Kostra grafu představuje vzájemné propojení všech míst na síti, které nesmí obsahovat kružnici. Jedná se o základní konstrukční úlohu na grafu z disciplíny zvané teorie grafů. Bez znalosti řešení této úlohy by se neobešlo hledání optimálního řešení celé řady složitějších úloh.

2.1 Kostra grafu

Kostra souvislého grafu G (Spanning Tree) představuje libovolný podgraf souvislého grafu G na množině všech jeho vrcholů, který hranami spojuje všechny vrcholy původního grafu a zároveň sám neobsahuje žádný cyklus. Kostra souvislého grafu G představuje strom [1].

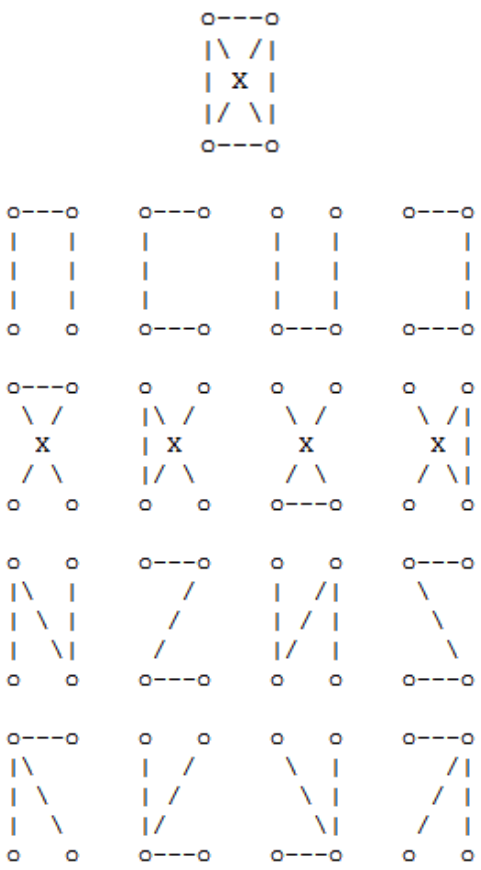
Definice kostry grafu

Nechť $G = (V, E)$ je souvislý graf. Libovolný strom (V, E'), kde $E' \subseteq E$, se nazývá kostrou grafu. V grafu s n vrcholy má každá kostra právě $n - 1$ hran. Graf může mít mnoho koster, například souvislý graf o čtyřech vrcholech má šestnáct koster grafu, viz Obrázek 2-1. Obecně pro každé $n \geq 3$ má úplný graf na daných n vrcholech právě n^{n-2} různých koster dle tzv. Cayleho vzorce [1]. Jedná se ovšem o různé kostry grafu, ne však neizomorfní. Problematikou zjišťování počtu neizomorfních koster grafu se zabývá práce [2].

Algoritmus pro hledání libovolné kostry

Následující základní algoritmus je schopen nalézt nějakou (blíže neurčenou) kostru grafu [3]:

1. Nechť $G = (V, E)$ je graf s n vrcholy a m hranami; hrany G jsou seřazeny libovolně do posloupnosti $(e_1, e_2 \ldots, e_m)$; je položeno $E_0 = 0$.
2. Byla-li již nalezena množina E_{i-1}, je spočtena množina E_i následovně:
   $E_i = E_{i-1} \cup \{e_i\}$, neobsahuje-li graf $(V, E_{i-1} \cup e_i)$ kružnici,
   $E_i = E_{i-1}$ jinak.
3. Algoritmus se zastaví, jestliže buď E_i již obsahuje $n - 1$ hran nebo $i = m$, tedy se prozkoumaly všechny hrany z G. Graf $T = (V, E_i)$ pak představuje kostru grafu G.



Obrázek 2-1: počet koster pro úplný graf *Zdroj: [4]*

2.2 Minimální kostra grafu

Při generování všech koster grafu jsou různé kostry daného grafu v zásadě rovnocenné - všechny obsahují stejný počet hran $h(G)$. Jiná situace ovšem nastane, pokud je každé hraně přiřazena nějaká nezáporná reálná délka. V takovém případě má smysl se zabývat hledáním takové kostry, která má nejmenší součet délek svých stran. Problém minimální kostry spočívá v nalezení kostry s minimální váhou – bez ohodnocení hran ztrácí pojem minimální kostra smysl [5].

Problém určení minimální kostry je motivován mnoha praktickými aplikacemi – od problému elektrifikace území, přes telekomunikační síť, až po možnou sjízdnost silnic či vybudování železniční sítě. Příkladem může být spojení obcí telefonním kabelem tak, aby bylo možné se dovolat z každé obce do každé a celková délka použitého kabelu byla minimální. Existuje-li silniční síť, z níž je třeba udržovat sjízdnou co nejkratší část, která vzájemně propojí všechny obce dané oblasti, je vhodné problém řešit nalezením minimální kostry na grafu silniční sítě. Stejně tak lze postupovat například v případě vybudování železniční sítě. Mezi městy určitého regionu je třeba navrhnout železniční síť tak, aby každá dvě města byla po železnici dosažitelná a přitom náklady na vybudování železniční sítě byly minimální. Problém minimální

kostry grafu patří mezi klíčové úlohy kombinatorické optimalizace. U jeho zrodu stáli dva významní čeští matematici.

2.2.1 Historický vývoj

Na přelomu let 1925 – 1926 byl český matematik Otakar Borůvka požádán pracovníkem Západomoravských elektráren Jindřichem Saxelem o vyřešení problému, *kudy a jak vést trasu, která měla spojovat několik desítek obcí v oblasti Moravy, aby byla co nejkratší, a tím co nejúspornější.* O hledání řešení pojednává článek *O jistém problému minimálním* na 16 stránkách. [3]

Borůvkův algoritmus nebyl na první pohled příliš přehledný, což si záhy uvědomil další vynikající český matematik Vojtěch Jarník. Na Borůvkův článek *O jistém problému minimálním* reagoval formou dopisu se shodným názvem, a to *O jistém problému minimálním* s podtitulem (Z dopisu panu O. Borůvkovi). [3]

Řešení problému minimální kostry grafu byla v obou podobách (Borůvky a Jarníka) nezávisle znovuobjevena ještě několikrát z důvodu skutečnosti, že většina prací byla psána v mateřském jazyku autora a často tak matematikům z jiných zemí nedostupná.

Metoda Otakara Borůvky byla zmíněna v práci Gustava Choqueta v roce 1938. Tatáž metoda byla nezávisle objevena v roce 1950 skupinou matematiků (K. Florek, J. Lukaszewicz, J. Perkal, H. Steinhaus a S. Zubrzycki) ve Wroclawi a v roce 1961 v Paříži Georgem Sollinem. Až pan Sollin tento algoritmus popsal anglicky, a proto většinou nese jeho jméno.

Řešení popsané Vojtěchem Jarníkem znovu nezávisle objevil J.B. Kruskal roku 1956. Loberman a Weinberger zveřejnili totéž řešení v roce 1957. Práce Roberta Prima z roku 1957 a nezávisle na ní práce Dijkstrova z roku 1959 popisuje opět metodu, kterou Jarník publikoval již v roce 1930. [3]

Třetí řešení, odlišné od předchozích dvou, podal roku 1956 Joseph B. Kruskal ve své práci *On the shortest spanning tree graph and travelling salesman problem* [3].

2.2.2 Formulace úlohy

Úloha optimálního spojení je formulována jako problém spojení *n* míst tak, aby všechna místa byla navzájem spojena a součet ohodnocení spojení všech míst byl minimální. Hledání minimální kostry má smysl u neorientovaných ohodnocených grafů. ,,*Nechť je dán neorientovaný graf* $G = (V, E)$*, který je hranově ohodnocený, tzn. každá hrana* $(i, j) \in E$ *je ohodnocena číslem* c_{ij} *(náklady na zřízení hrany). Uzly z množiny* $V = 1,2,\dots,n$ *představují místa, mezi kterými musí existovat spojení vytvořené z hran. Pokud chceme zajistit spojení*

v grafu G s nejmenšími celkovými náklady, pak z grafu G odstraníme některé hrany tak, aby zbývající hrany zajistily spojení mezi uzly a jejich ocenění bylo minimální [6].”

Hledá se tedy množina hran $E' \subset E$. Úloha spočívá v rozhodnutí, zda se hrana $i,j \in E$ vybere do množiny E', která tvoří tzv. minimální kostru grafu G. Každý vrchol $i \in V$ musí být možno propojit hranami z E' s libovolným vrcholem z V. Proto musí ležet alespoň na jedné hraně z E'. Pro účely matematické formulace zavedeme orientaci hran tak, že vznikne kostra s kořenem v uzlu 1 s orientací, která směřuje do tohoto uzlu 1 [6].

Z každého uzlu v kostře E' existuje jednoznačně cesta do uzlu 1. Proto jsou opatřeny touto orientací všechny hrany ležící na této cestě. Tím je jednoznačně určena orientace kostry E, a E' se stává orientovaným stromem s kořenem v uzlu 1. Protože vrchol 1 je kořen, nevychází z vrcholu 1 žádná hrana. Dále je třeba zajistit souvislost grafu. Všechny vrcholy kromě vrcholu 1 jsou zdroji produktu. Vrchol 1 je naopak koncový vrchol, kam všechny toky ze všech uzlů směřují [6].

V mnoha úlohách minimální kostry je třeba z modelu vyloučit některá spojení (tzv. zakázané hrany) a jiná spojení se naopak musí ve výsledném řešení nacházet (tzv. povinné hrany). Na základě požadavku je povinným hranám přiřazeno ohodnocení -1 a zakázané hrany jsou z grafu odebrány. Jsou-li v grafu hrany méně vhodné, avšak ne zakázané, je jim přiřazeno vyšší ohodnocení, než jaké mají ostatní hrany [3].

2.2.3 Matematický model úlohy

Matematický model minimální kostry grafu vypadá následovně [1]:

Minimalizovat

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}, \qquad (1)$$

za podmínek

$$x_{1j} = 0 \qquad \text{pro } j = 2,3,\dots,n, \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \text{pro } i = 2,3,\dots,n, \qquad (3)$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=2}^{n} y_{ji} = 1 \qquad \text{pro } i = 2,3,\dots,n, \qquad (4)$$

$$0 \le y_{ij} \le (n-1)x_{ij} \qquad \text{pro } i,j = 1,2,\dots,n, \qquad (5)$$

$$x_{ij} \in \{0;1\} \qquad \text{pro } i,j = 1,2,\dots,n, \qquad (6)$$

kde:

- c_{ij} jsou náklady na zřízení hrany (i,j);
- x_{ij} je bivalentní proměnná, $x_{ij} = 1$ pro hranu (i,j), která je prvkem množiny E';
  $x_{ij} = 0$ pro hranu (i,j), která není prvkem množiny E';
- y_{ij} je tok na každé hraně $(i,j) \in E$.

Rovnice (1) je účelová funkce, která minimalizuje celkové náklady na zřízení hrany. Ostatní rovnice jsou omezujícími podmínkami. Podmínka (2) se týká vrcholu 1, který je kořenem a nevychází z něj žádná hrana, proto musí být rovna nule. Podmínka (3) naopak zobrazuje ostatní uzly a říká, že musí existovat právě jedna cesta do kořene, resp. hrana vycházející z libovolného uzlu *i*. Každý vrchol $i \in 2,3,\dots,n$ je zdrojem velikosti 1, podmínka (4) ukazuje bilanci toku do tohoto uzlu a toku z tohoto uzlu. Omezení ve tvaru (5) zabraňuje toku hranou (i,j), která není prvkem E', platí pro ni $x_{ij} = 0$. Naopak pokud tok $y_{ij} > 0$, pak je nutné hranu (i,j) do množiny E' zařadit a proměnná $x_{ij} = 1$.

2.2.4 Přehled algoritmů pro minimální kostru

Hledání minimální kostry lze koncipovat několika způsoby. Jejich společným rysem je to, že pracují iteračně: postupně konstruují takové podgrafy daného grafu, které se přibližují hledané minimální kostře. Přibližování se může dosahovat buď vypouštěním hran výchozího grafu, nebo přidáváním hran k podgrafu, který je na počátku prázdný, a nebo konečně výměnou hran v nějaké výchozí kostře [7].

Kritérium efektivnosti splňují nejlépe algoritmy používající postupné přidávání hran. Minimální kostra se začíná vytvářet počínaje prázdnou množinou hran *T*, ke které se přidávají vhodné hrany tak dlouho, až se získá úplná minimální kostra. Základní invariantou tohoto postupu je podmínka, že množina hran *T* je podmnožinou hran nějaké minimální kostry – tato podmínka se nesmí přidáním nové hrany porušit [5]. Přehled různých algoritmů pro řešení úlohy minimální kostry včetně výpočetní složitosti jednotlivých přístupů znázorňuje Tabulka 2-1.

Tabulka 2-1: Přehled algoritmů *Zdroj:* ***Error! Reference source not found.***

Borůvkův-Sollinův	$O(m \log n)$
Jarníkův – Primův	$O(m \log n)$
Kruskalův hladový	$O(m \log n)$

Kruskalův hladový algoritmus (princip postupného výběru hran)

Poprvé byl publikován Josephem Kruskalem v *Proceedings of the American Mathematical Society v 2.* polovině 20. století. Algoritmus prochází hrany seřazené vzestupně podle jejich ohodnocení a přidává je k již vybraným hranám tak, aby nevznikl cyklus. Pokud by hrana s dalším nejnižším ohodnocením cyklus vytvořila, jednoduše je vynechána. S přidáváním hran se pokračuje tak dlouho, dokud počet vybraných hran nedosáhne počtu $n - 1$ [9]. Algoritmus se označuje jako hladový, neboť jednou dosažená rozhodnutí už nikdy nezmění, „hladově" postupuje přímo k řešení.

Při počítačovém zpracováni algoritmu je zřejmě největším problémem testování acykličnosti vznikajícího podgrafu. Ten totiž není souvislý a při testování acykličnosti je třeba brát v úvahu všechny jeho souvislé části.

Kruskalův hladový algoritmus vypadá následovně [3]:

1. Seřaď hrany podle velikosti $e_1 \leq e_2 \leq \cdots \leq e_m$.
2. $M := \emptyset$. Procházej hrany v pořadí podle velikosti. Pokud hrana e_i spojuje různé komponenty souvislosti M, tak ji přidej do M a sjednoť příslušné komponenty souvislosti.
3. Na konci je M minimální kostra.

Jarníkův-Primův algoritmus (princip množin sousedů)

Jarníkův (též zvaný Jarníkův – Primův nebo jen Primův)[1] algoritmus je k testování acykličnosti podgrafu přátelštější v tom smyslu, že vznikající podgraf je vždy souvislý. K němu se připojí nově přidávanou hranou vždy jen jeden dosud nezařazený uzel v, a to takový, který má k dosud vzniklému podgrafu nejblíže. Protože hodnota „vzdálenosti od kostry" se přidáním nových uzlů může změnit, je nutné po každém přidání hrany tuto vzdálenost přepočítat. Algoritmus končí ve chvíli, kdy kostra obsahuje všechny uzly grafu [9].

Jarníkův - Primův algoritmus vypadá následovně [9]:

1. $T := (V, M), V := \{v\}, M := \emptyset$.
2. V každém kroku přidáme do *M* nejlevnější hranu *e* řezu $\delta(V)$ a do V přidáme druhý konec hrany e neležící v V.

[1] Tento algoritmus poprvé popsal český matematik Vojtěch Jarník roku 1930, později byl znovuobjeven roku 1957 Robertem Primem a poté ještě jednou roku 1959 Edsgerem Dijkstrou. V zahraničí se téměř výlučně používá označení Primův algoritmus, vzácně pak Jarníkův algoritmus nebo DJP algoritmus.

3. Po $n-1$ krocích se zastavíme, máme minimální kostru *T*.

Borůvkův-Sollinův algoritmus

Borůvkův algoritmus funguje správně pouze na ohodnocených grafech, ve kterých je minimální kostra určena jednoznačně. Jedná se o složitější algoritmus, chová se jako Jarníkův algoritmus spuštěný zároveň ze všech vrcholů grafu najednou.

Borůvkův algoritmus si v průběhu udržuje les[2] T (množinu hran rozšířitelnou do minimální kostry). Na začátku je les tvořen izolovanými vrcholy (neobsahuje žádnou hranu). V každé iteraci je vybrán pro každý strom T_i (komponenta souvislosti) lesa T nejlevnější hranu řezu $\delta(T_i)$ a na závěr iterace je přidána do T. Je třeba si dát pozor, aby nějaká hrana nebyla přidána dvakrát. Mohlo by se stát, že jedna hrana bude spojovat dvě komponenty souvislosti a pro obě komponenty bude vybrána jako nejlevnější [9].

Algoritmus vyžaduje jednoznačnost minimální kostry, protože jinak by se mohlo stát, že pro komponentu T_u bude vybrána nejlevnější hrana řezu e_u a pro komponentu T_v jiná nejlevnější hrana e_v téhož řezu. Po přidání obou hran do kostry by vznikla kružnice [9].

Významnou výhodou Borůvkova algoritmu je snadná paralelizace spojování jednotlivých komponent řešení. Protože vyhledávání nejlevnějších hran vycházejících z daných komponent je zcela nezávislé na stavu ostatních komponent, tak lze tyto podproblémy řešit na samostatných výpočetních jednotkách.

2.3 Minimální kostra grafu s omezeními

Problém minimální kostry s omezeními spočívá v modifikaci úlohy minimální kostry, a to přidáním omezující podmínky. Takovou omezující podmínkou může být například limitovaný počet připojení k síti, kde d_i představuje počet hran, se kterými může být uzel *i* maximálně spojen. K matematickému modelu minimální kostry je přidáno následující omezení (7).

$$\sum_{\substack{j \\ (i,j)\in H}} x_{ij} + \sum_{\substack{k \\ (k,i)\in H}} x_{ki} \leq d_i \,, i \in V \tag{7}$$

Mezi nejznámější úlohy hledání minimální kostry s omezeními patří dle [10] následující úlohy:

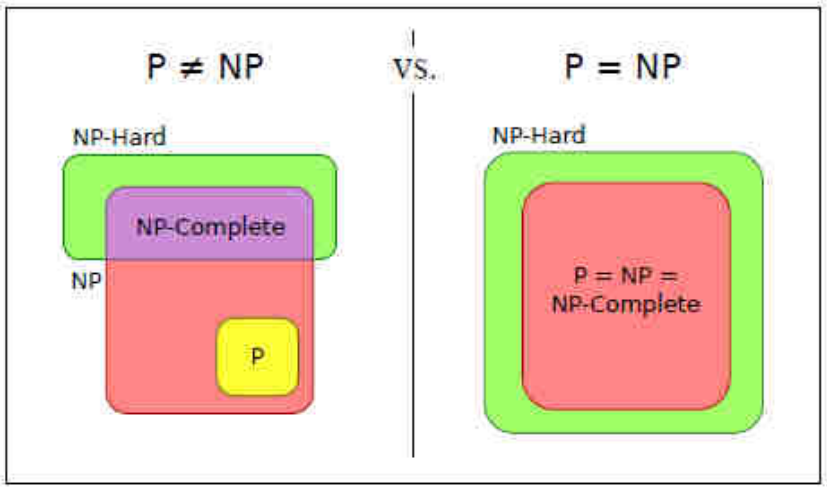
[2] Strom T je souvislý graf bez kružnic. Pokud jsou zakázány pouze kružnice, ale není vyžadována souvislost, je získán graf skládající se z několika stromů. Takovému grafu se říká les. Vrcholy stromu stupně 1 se nazývají listy.

- Problém *Bounded Diameter Minimum Spanning Trees* (BDCMST) je definován následovně. Nechť je dána počítačová síť o *n* uzlech, kde probíhá komunikace posíláním zpráv po stromu grafu počítačové sítě mezi počítači navzájem. Pouze jeden vrchol má privilegium ke vstupu do kritické sekce. Jestliže nějaký uzel chce mít toto privilegium, musí o něj požádat zasláním zprávy tomu uzlu, který dané privilegium právě má. Délka trvání této žádosti závisí na počtu hran mezi těmito dvěma uzly. V úloze BDCMST je tedy zapotřebí vytvořit komunikační strom s minimálními náklady a omezeným časem pro komunikaci. Tento problém je možný vyřešit v polynomiálním čase pouze v případě, je-li $D = 2$ či $D = 3$ a jsou-li všechny hrany grafu ohodnoceny stejnými náklady. V opačném případě se jedná o NP těžký problém. Nejčastěji výše popsaný problém řešen na základě metaheuristického přístupu hledání v proměnném okolí (Variable Neighbourhood Search = VNS), evolučních algoritmů (Evolutionary Algorithms = EA), mravenčí kolonie (Ant Colony Optimization = ACO) či jejich kombinací. Tento problém je hojně využíván při zpětné záchraně dat či na telekomunikačních sítích [11].
- Problém *Degree Constrained Minimum Spanning Trees* (DCMST) spočívá v hledání kostry, jejíž vrcholy by neměly překročit určitý maximální stupeň a jejíž celková délka hran je minimální. Například ústředny nebo přepínače mohou být fyzicky připojeny pouze na omezený počet spojujících drátů daného vedení. Při návrhu sítě s požadavkem na maximální spolehlivost, limituje zavedení omezeného stupně vrcholů poškození, které by mohlo být způsobeno právě ojedinělým výpadkem ústředny. Tato úloha je nejčastěji řešena pomocí heuristiky na bázi mravenčích kolonií [12].
- Problém *Leaf Constrained Minimal Spanning Tree* (LCMST) je využíván při problémech rozmístění zařízení, při navrhování sítí a velmi úzce souvisí s problémem p-mediánu (PMP)[3].
- *Problém hledání minimálního Steinerova stromu (The Steiner Minimal Tree, SMT)* spočívá v propojení všech koncových uzlů se zdrojovým uzlem (přes průběžné uzly (překladiště, pomocné stanice, ústředny, …) nebo přímo tak, aby celkové náklady byly minimální. V práci [14] je problém minimálního Steinerova stromu řešen vkládací heuristikou (*The Steiner Insertion Algorithm*) a heuristikou přírůstků (*The Incremental Optimization Problem*).

[3] Definice PMP a jeho praktické využití je k dohledání v [13].

3 Metody řešení

Před řešením jakékoliv reálné úlohy, je žádoucí mít základní představu o výpočetní složitosti daného problému dříve, než se přistoupí k nalezení algoritmu, který řeší problém dané úlohy. Existují dvě třídy úloh za předpokladu, že $P \neq NP$ – řešitelné v polynomiálním čase (třída P) a neřešitelné v polynomiálním čase (třída NP), která se dále dělí na problémy NP-těžké (NP-hard) a NP-úplné (NP-complete). Většina problémů, které je třeba v reálném světě řešit, je ze třídy NP-úplných. Skutečnost, že problém je NP-úplný znamená, že pokud třída $P \neq NP$, tak neexistuje polynomiální algoritmus, který by našel řešení pro každý případ problému. Pokud pro daný problém neexistuje efektivní algoritmus, výpočet optimálního řešení není praktický vzhledem k vysoké výpočetní náročnosti. Lze ovšem najít přibližné řešení, tzv. suboptimální řešení, pomocí heuristik či metaheuristik. Efektivní řešení polynomiálními algoritmy při řešení NP-úplných úloh je možné jen za podmínky, když se třída $P = NP$, což je velmi nepravděpodobné (dosud zcela nevyloučeno matematickými důkazy). Grafické znázornění daného problému zobrazuje Obrázek 3-1.



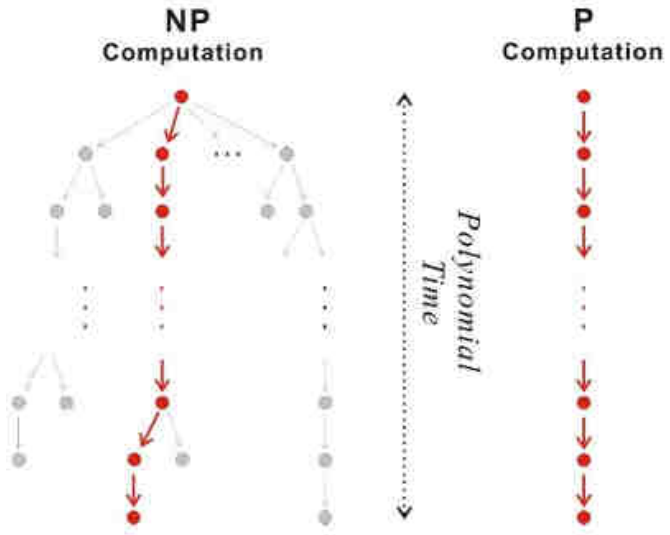
Obrázek 3-1: Grafická reprezentace problému NP *Zdroj: [15]*

Problémy ze třídy P jsou řešeny deterministickými algoritmy v polynomiálním čase, zatímco NP problémy jsou řešeny nedeterministickými algoritmy[4]. NP problém může být vyřešen několika způsoby, a tím pádem může být dosaženo různých řešení bez záruky optimality v polynomiálním čase, jak prezentuje Obrázek 3-2.

Mnoho reálných i teoretických problémů vyžaduje nalezení řešení úloh ze třídy NP-H (NP-těžkých) či NP-C (NP-úplných). Optimalizační problémy představují rozsáhlou skupinu různých úloh, které se mohou velmi významně lišit, a to jak druhem možných řešení,

[4] Algoritmická řešitelnost souvisí s pojmem Turingův stroj, viz dále [16].

tak i například typem účelové funkce nebo významem optima. Vzhledem k potřebě nalezení řešení takto obtížných úloh se snížily nároky na výsledné řešení. Byly vyvinuty aproximační metody, které pro vybrané typy úloh dokážou zaručit, že nalezené řešení není od přesného optima vzdálenější, než je předem stanoveno. Jiné způsoby řešení nepožadují striktnost dodržování omezujících podmínek úlohy (relaxační metody) a některé vsadily na dostatečně vysoké pravděpodobnosti (např. Metoda Monte Carlo). Další skupinu metod tvoří právě metody založené na praxí ověřených postupech, které se nedají jednoznačně dokázat. Tyto metody jsou buď vytvořeny se záměrem optimalizovat pouze jeden konkrétní problém (heuristiky) a nebo naopak jsou navrženy tak obecně, že je možné jej aplikovat na více druhů úloh (metody umělé inteligence, metaheuristiky). Pro mnoho NP-těžkých (NP-úplných) úloh jsou tyto metody řešení jedinou přijatelnou možností, a to i přesto, že negarantují žádnou úroveň kvality nalezeného řešení a ani samotné nalezení přípustného řešení.



Obrázek 3-2: Výpočet NP a P problémů *Zdroj: [16]*

Deterministické algoritmy jsou v řešení složitých problémů velmi špatně použitelné, protože jejich časová náročnost roste, s lineárně rostoucím rozsahem problému, exponenciálně či logaritmicky, jak přehledně zobrazuje Tabulka 3-1. Naopak výhodou je oproti heuristickým algoritmům nalezení přesného, optimálního řešení.

Tabulka 3-1: Přehled výpočetního času algoritmů *Zdroj: [15]*

$g(n)$	$n = 10$	$n = 30$	$n = 50$	$n = 70$
n^2	0,000001 s	0,00003 s	0,00005 s	0,00007 s
n^3	0,001 s	0,027 s	0,125 s	0,343 s
n^5	0,1 s	24,3 s	5,2 min	28 min
2^n	0,001 s	17,9 min	35,7 roků	37 mil. roků
3^n	0,059 s	6,5 roku	$2{,}3 \times 10^{10}$ roků	10^{13} mil. roků

3.1 Exaktní metody řešení

Exaktní algoritmy představují takové metody řešení, jejichž výstupem je přesně požadovaný výsledek, a zároveň takové metody, které jsou schopny požadovaný výsledek při každém opakování znovu bezpečně nalézt v konečném (často neúměrně velkém) čase. Ne pro všechny úlohy je možné nalézt exaktní algoritmus nebo je jeho časová náročnost tak vysoká, že je prakticky nepoužitelný. Exaktní metody řešení vychází vždy z přesného pochopení tvorby stavového prostoru dané úlohy. Je zapotřebí, aby bylo vždy známo, jaké vlastnosti mají jednotlivá přípustná řešení a jaké jsou mezi nimi vazby.

Pokud lze účelovou funkci popsat pomocí analytické funkce s existující derivací, pak je možné snadné nalezení všech jejích stacionárních bodů[5], a tím pádem se dá poté zjistit, ve kterém z nich se nachází hledané optimum. Tento úkol nebývá jednoduchý, ale vzhledem ke znalosti funkčního předpisu pro celý definiční obor je možné odvodit vše potřebné. V některých případech se řešení pomocí derivace nehodí. Při úloze lineárního programování je účelová funkce sice popsána analyticky a derivace existuje na celém definičním oboru, ale nikde není nulová. Neobsahuje žádné stacionární body a optimum se nalézá v některém z rohů konvexního polyedru. Díky znalosti zákonitostí lineárního programování ale existují algoritmy, které přesné řešení najdou vždy. Jako příklad lze uvést známý simplexový algoritmus procházející postupně zmíněné rohové body, dokud nenarazí na hledané optimum[6] [17].

3.2 Heuristiky

Další metody řešení představují takové metody, které nejsou založeny na žádném exaktně ověřitelném postupu a nemohou tak zaručit žádné výsledné vlastnosti nalezených řešení. Tyto metody vznikají právě z nemožnosti použití exaktního algoritmu, a to buď z důvodu, že takový požadovaný algoritmus neexistuje, nebo by exaktní způsob řešení byl příliš časově náročný a daná úloha by tedy nebyla řešitelná v reálném čase.

Heuristika (z řečtiny *heuriskó*, *εύρίσκω* – *nalézt*, *objevit*) znamená zkusmé řešení problémů, pro něž není znám algoritmus nebo přesnější metoda. Jak uvádí J. Pearl [18], heuristické řešení je často jen přibližné, založené na dřívějším odhadu, zkušenosti nebo jednoduše na zdravém rozumu. První odhad se zpravidla postupně zlepšuje, heuristika ovšem nikdy nezaručuje nalezení globálně optimálního řešení. Tyto metody jsou tudíž označovány jako aproximační algoritmy. Heuristiky jsou schopny v relativné krátké době poskytnout řešení, které je uspokojivé vzhledem k zadanému souboru cílů a omezujících podmínek. Heuristické

[5] Body, ve kterých má funkce nulovou derivaci.

[6] Jakmile algoritmus nalezne lokální optimum, muže výpočet ukončit, protože v lineární optimalizaci je lokální optimum zároveň i globální.

metody představují speciální algoritmy vyvinuté pro speciální obtížné úlohy, jedná se o relativně jednoduché a rychlé metody, u nichž není možné zjistit, jak kvalitní řešení poskytují, tedy např. jak daleko je získané řešení od optima. Nevýhodou heuristických metod je omezený průzkum množiny přípustných řešení. Kvalitu získaného řešení může posoudit pouze jeho koncový uživatel. Jestliže je spokojen a řešení mu vyhovuje, použije pravděpodobně danou heuristiku i příště. Pokud řešení nesplňuje jeho očekávání, tak se pravděpodobně příště rozhodne pro jinou metodu.

Nejznámnějším příkladem heuristiky je hladový algoritmus (Greedy algorithm), který například problém obchodního cestujícího (TSP) řeší procházením měst v pořadí nejbližších sousedů. Začne tedy v jednom libovolném uzlu (městě), poté se přesune do dalšího uzlu, které je od předešlého nejblíže, a tak dále, dokud nenavštíví všechny uzly a nevrátí se zpět na začátek bez uzavření kružnice. Existují i úlohy, pro které hladový algoritmus zaručuje optimální řešení (např. určení minimální kostry), ale u složitějších úloh (ze třídy NP-H, NP-C tedy konkrétně např. určení minimální kostry s omezeními) nelze nijak předem určit, jak bude výsledek nakonec kvalitní.

Primární heuristiky

Primární heuristika spočívá v přechodu z přípustného řešení do takového přípustného řešení, jehož hodnota lokálního kritéria je lepší než hodnota kritéria předchozího řešení. Výchozí řešení je tedy postupně zlepšováno při nenarušení podmínek přípustnosti.

Duální heuristiky

Duální heuristika, na rozdíl od primární, vychází z nepřípustného řešení a přechází k řešení s menší mírou nepřípustnosti tak, aby se lokální kritérium zhoršilo co nejméně.

Oba zmíněné postupy končí právě tehdy, když ze současného řešení již není možné povoleným postupem přejít k následujícímu řešení s lepší hodnotou lokálního kritéria nebo s menší mírou nepřípustnosti [19].

Oba přístupy se mohou při konstrukci heuristických metod vzájemně kombinovat tak, že duální heuristika poskytne výchozí řešení pro primární heuristiku, která toto řešení dále zlepšuje. Heuristik je obecně známo několik typů, nejznámější z nich jsou dle Tuzara (2007) [19] tyto:

- vkládací heuristiky,
- heuristiky s výhodnostními koeficienty,
- výměnné heuristiky,
- dekompoziční heuristiky,
- heuristiky využívající metody matematického programování.

Obecně lze konstatovat, že heuristické metody se váží ke konkrétnímu problému, zatímco metaheuristiky představují všeobecné postupy (vycházející např. z fyziky, biologie, …), které jsou aplikovatelné na řešený problém. Poslední dobou, s nárůstem výpočetního výkonu počítačů, se začaly masivně používat metaheuristické metody. Jejich výhodou je schopnost za určitých okolností opustit nalezený lokální extrém účelové funkce a přejít do jiných částí množiny přípustných řešení, kde je větší pravděpodobnost nalezení řešení s lepší hodnotou účelové funkce, což klasické heuristiky neumožňují. Ani heuristické metody ani metaheuristiky nezaručují ovšem nalezení optimálního řešení.

3.3 Metaheuristiky

Metaheuristiky jsou skupinou aproximativních výpočetních metod používaných pro optimalizační problémy, u nichž není vyžadováno nalezení optimálního řešení. Pomocí metaheuristických algoritmů lze prohledávat stavový prostor úlohy, avšak bez záruky nalezení dostatečně kvalitního řešení v rozumném čase. Slovo metaheuristika vzniklo spojením dvou původem řeckých slov meta a heuristika. Předpona *meta* se používá k označení přesahu kořenového pojmu a znamená *něco za* nebo přeneseně i *nad*. Termín heuristika (z řeckého *heuriskó* znamenající objevovat nebo vynalézat) označuje metody, které vznikly pozorováním a jsou ověřeny pouze zkušeností bez exaktní prokazatelnosti. Metaheuristiky představují obecné algoritmy pro řešení obtížných úloh, tzv. heuristiky nové generace, které řeší komplexní problémy, které nemohou být vyřešeny optimalizačními metodami v reálném čase **Error! Reference source not found.**.

Fred Glover prvně použil výraz metaheuristika v článku [20] z roku 1986, kde jím označuje nadvrstvu nad běžnou heuristikou. Jako alternativní název pro stejné metody se občas používal i pojem moderní heuristiky, viz [21], od konce 90. let 20. stol. se ale již téměř výhradně používá velmi rozšířený pojem *metaheuristika*. Většina metaheuristických přístupů má svůj původ v 80. letech 20. století, i když v některých případech sahají kořeny až do poloviny 60. a 70. let, a od počátku 80. let stále narůstají na popularitě. Řada specializovaných časopisů a konferencí se věnuje této problematice. EU/ME[7] představuje komunitu zabývající se metaheuristikami, která je sponzorována neziskovou organizací EURO[8]. Zmíněná komunita má přibližně 1400 členů a jedná se tím pádem o největší základnu pro možnou komunikaci mezi výzkumníky metaheuristik po celém světě.

[7] EU/ME představuje největší pracovní skupinu zabývající se metaheuristikami na světě; dostupná na: http://uahost.uantwerpen.be/eume/index.php

[8] Asociace evropských společností zabývajících se operačním výzkumem zaštiťována IFORS (Mezinárodní federací společností operačního výzkumu), dostupné na: http://www.euro-online.org/web/pages/1/home

3.3.1 Definice

Šandera (2014) **Error! Reference source not found.**, odborná komunita se zatím jednoznačně neshodla na přesném vymezení pojmu metaheuristika. V literatuře lze nalézt několik existujících definic (nebo alespoň zobecněných popisů), které si v mnoha ohledech odpovídají a pokrývají podobné třídy algoritmů, v dílčích detailech se ale liší. Příkladem mohou být následující definice.

- [22]: *„Metaheuristika je množina algoritmických konceptů, které mohou být využity k definování heuristických metod aplikovatelných na široké spektrum různých problémů. Jinak řečeno, metaheuristiky mohou být chápány jako zobecněné heuristické metody navržené k určování postupu problémově závislých heuristik (lokální prohledávání, konstrukční heuristiky) směrem k oblastem ve stavovém prostoru obsahující vysoce kvalitní řešení. Metaheuristika je tedy obecný algoritmický rámec aplikovatelný na mnoho různých problémů s relativně malým počtem úprav nutných pro přizpůsobení se konkrétnímu problému.“*
- [23]: *„Metaheuristika je proces iterativního generování, který řídí podřízenou heuristiku kombinováním rozdílných konceptů pro prohledávání a využívání stavového prostoru pomocí učících se strategií k strukturování informací pro efektivní nalezení téměř optimálního řešení.“*
- [24]: *„Metaheuristika je iterativní proces, který řídí a mění operace podřízených heuristik, aby efektivně vytvářely kvalitní řešení. Může manipulovat s úplným (nebo neúplným) jednotlivým řešením nebo s celou množinou řešení během každé iterace. Podřízené heuristiky mohou být vysoko (nebo nízko) úrovňové procedury, jednoduchá lokální vyhledávání a nebo jen konstrukční metody.“*
- [25]: *„Metaheuristika je inteligentní iterativní proces, který vykonává prohledávání a může být aplikovaný na optimalizační problémy, jako např. problém obchodního cestujícího.“*
- [26]: *„Metaheuristický algoritmus může být definován jako vysokoúrovňová obecná metodologie (šablona), která je používána jako návod k odvození heuristik pro řešení specifických optimalizačních problémů.“*
- Yang (2008) ve své knize [27] poznamenal, že žádná obecně uznávaná definice zatím neexistuje a trendem poslední doby je pojmenovávat metaheuristikou jakýkoliv stochastický algoritmus s randomizací a lokálním prohledáváním, čehož se drží ve svých publikacích i on.

Hlavními problémy uvedených popisů je buď přílišná obecnost vymezení metaheuristických metod, nebo naopak vyžadování příliš konkrétní vlastnosti daných metod. Jak uvádí Šandera

(2014), metaheuristika ve skutečnosti nemusí být inteligentní[9], samoučící se[10] a nemusí být ani nutně stochastická[11]. Tyto silné předpoklady mohou být z definice vypuštěny, protože zbytečně zužují třídu využitelných metod.

Základním principem jakéhokoliv metaheuristického modelu je práce s dvojicemi typu (x, y), kde x je navštívený bod a $y = f(x)$ je zjištěná hodnota optimalizačního kritéria (účelové funkce), Šandera (2014) [17]. Z takto uspořádaných informací musí být metaheuristika schopna generovat nové body x, které mají jistou šanci[12] na lepší hodnotu y. Reálná výpočetní zařízení jsou navíc omezena limitovanou pamětí, a proto musí metaheuristika neustále rozhodovat, které páry (x, y) z paměti odstraní a které si ponechá pro další použití. Základními schopnostmi metaheuristických algoritmů tedy musí být dle Šandery (2014) [17] následující:

- schopnost vyhodnotit kvalitu konkrétního bodu x (tzv. vyhodnotit hodnotu účelové funkce);
- schopnost uchovat nebo zpracovat získané informace (dvojice (x, y));
- schopnost rozhodnout o odstranění získaných informací z paměti;
- schopnost vygenerovat nová řešení na základě uchovaných informací.

Množina metaheuristických metod je velmi široká a v posledních letech se prudce rozšiřuje. Nejjednodušší metodou, ale také nejméně efektivní, je metoda lokálního prohledávání (Local Search). Tento algoritmus trpí základní nectností gradientních metod, tzn., že nejspíše skončí v lokálním optimu a nedosáhne globálního optima. Z nalezeného lokálního extrému se algoritmus bohužel nedokáže posunout, což značně snižuje jeho samostatnou použitelnost. Běžně je však využíván jako doplněk jiné metody řešení. Jinou zvolenou metodou se dosáhne konečného řešení a nakonec je spuštěno lokální prohledávání, kterým se dojde k nejbližšímu lokálnímu extrému. Druhá možnost nasazení lokálního prohledávání využívá vlastnost, že metoda velmi rychle konverguje. Lze ji tedy ve zvoleném čase pouštět opakovaně s tím, že pokaždé se vyjde z odlišného výchozího řešení. Tím pádem bude pravděpodobně pokaždé dosaženo odlišného lokálního extrému. Pokud je nakonec vybrán nejlepší z dosažených extrémů, je získáno poměrně kvalitní řešení, které s trochou štěstí může být i globálně optimální.

Rozmanitost jednotlivých aplikací metaheuristik je zmíněna v následující literatuře: Osman a Laporte (1995), Laporte a Osman (1995b), Stewart, Liaw a White (1994), Nissen (1993), Collins, Eglese a Golden (1988); v následujících sbornících: Laporte a Osman (1995a), Pesch

[9] Inteligence je stejně problematický nedefinovatelný pojem.
[10] Samoučící je pojem vzbuzující přílišná očekávání a kladoucí vysoké nároky na práci s informacemi.
[11] Z principu není nutné, aby se algoritmus choval při každém běhu různě.
[12] Právě tento pojem je hlavním rysem metaheuristik - zlepšení hodnoty účelové funkce je negarantované a stojí pouze na heuristickém ověření.

a Voss (1995), Glover *a kol.* (1993); v následujících knihách: Aarts a Lenstra (1995), Osman a Kelly (1995), Rayward-Smith (1995), Michalewicz (1994), Holland (1994), Reeves (1993), Davis (1991), Goldberg, Aarts a Korst (1989), van Laarhoven a Aarts (1987), v následujících zápisech: Osman a Kelly (1995), Albrecht, Reeves a Steel (1993), Forrest (1993), Belew a Booker (1991), Shaffer (1989) a Grefenstette (1985).

3.3.2 Specializace metaheuristik pro optimalizační problémy

Šandera (2014) [17], metaheuristika je často chápána v kontextu optimalizačních metod jako návod, jak zkonstruovat algoritmus generující body stavového prostoru pouze na základě předchozích získaných znalostí. Při obecném pohledu na optimalizovanou funkci $f: X \to Y$ to znamená, že pomocí metaheuristiky lze vytvářet algoritmy (heuristiky), které pro danou úlohu dokáží generovat posloupnost bodů $(x, f(x))$ tak, aby se hodnoty $f(x)$ s časem zlepšovaly (ve smyslu optimalizačního kritéria). Jako základní předpoklad pro úspěšný běh metaheuristiky tedy musí být zajištěno, že nalezené dvojice hodnot $(x, f(x))$ spolu navzájem nějak korelují. Taková korelace nemusí existovat globálně na celém stavovém prostoru, ale stačí, když určitý vztah mají mezi sebou body pouze na nějakém lokálním okolí. V případě, že by neexistovala žádná závislost mezi polohou bodu x ve stavovém prostoru a hodnotou jeho účelové funkce $f(x)$, neměla by se metaheuristika podle čeho rozhodovat a generování nových bodů by tedy probíhalo zcela náhodně.

Metaheuristiky tedy obecně nepotřebují žádné apriorní informace o vlastnostech účelové funkce, protože jejich základní schopností je pohyb v neznámém prostoru a vyhledávání tamních nejlepších hodnot. Zjednodušeně lze chování metaheuristik přirovnat k hledání nejvyšší hory v neznámé krajině za naprosté tmy. Algoritmus musí o dalších krocích rozhodovat pouze ze znalosti nadmořských výšek v předchozích navštívených místech. Je zřejmé, že pokud je ráz krajiny předem znám, je větší šance na nalezení efektivnějšího způsobu prohledávání. Některé metaheuristiky tedy mohou pracovat pro určitý typ úloh výrazně lépe než pro jiné, ale dopředu nemusí být nutně úplně jasné, které typy jsou pro danou metaheuristiku vhodné a které nikoliv. A jelikož je vysoce nad schopnosti každého průzkumníka pamatovat si všechna navštívená místa, musí být získané informace nějakým způsobem navíc postupně zpracovávány a nepotřebné postupně odstraňovány [17].

3.3.3 Druhy metaheuristických algoritmů

Algoritmus metaheuristik je založen na inspiraci známými procesy - zdaleka nejčastějším zdrojem inspirace při vývoji nových metaheuristik je živá příroda, matematika, statistika nebo umělá inteligence. Autoři těchto algoritmů se odvolávají na evoluční principy, na genetické

zákonitosti, na chování mravenců, ptáků, včel i netopýrů. Metaheuristiky lze dle Šandery (2014) [17] kvalitativně rozlišovat z hlediska výkonnosti a chování algoritmu, tedy jestli se jedná o populační nebo jednoduché prohledávání. Při jednoduchém (single based) je v paměti udrženo pouze jedno hlavní řešení, kdežto u populačních algoritmů je jich udržována celá skupina a nová řešení jsou generována na základě její kombinace. Velmi významnými jsou i velikosti oblastí, které jsou prohledávány. Při lokálních typech algoritmů jsou nová řešení generována v blízkosti původních, naopak při globálních metodách je možné nové řešení generovat od původní skupiny velmi daleko.

Sörensen a Glover (2013) [28] dělí metaheuristiky do třech základních tříd na základě způsobu, jakým jsou získávána řešení daného problému. Metaheuristiky lokálního prohledávání opakovaně provádějí drobné změny v dosavadním řešení. Jedná se o relativně malé pohyby v okolí stávajícího řešení, tím pádem prozkoumávaná řešení si jsou relativně blízká. Do této třídy patří například metoda Tabu Search. Konstruktivní metaheuristiky vytvářejí řešení z jejich strukturálních částí. Jedná se v řadě případů o adaptace hladových algoritmů, kdy k dosavadnímu řešení je v každé iteraci přidáno nejlepší možné řešení. Konstruktivní metaheuristiky mnohdy využívají fázi lokálního hledání právě po konstruktivní fázi pro zvýšení kvality výsledného řešení. Do této třídy patří například metoda ACO. Populační metaheuristiky kombinují stávající řešení ze souboru[13] a vytváří z něj nové řešení. Mezi hlavní zástupce v této třídě se řadí evoluční algoritmy, například genetické algoritmy, evoluční programování aj. Zmíněné třídy se nicméně navzájem nevylučují a řada metaheuristických algoritmů kombinuje nápady z různých zmíněných tříd - v takovém případě se jedná o hybridní metaheuristiky.

Typičtí zástupci dnešních metaheuristik jsou založeni buď na jednoduchých intuitivních pravidlech pohybu v neznámém prostoru (Hill Climbing, Tabu Search,…), nebo na inspiraci přírodními fenomény, jako je evoluce, genetika (genetické algoritmy) nebo pohyb zvířat (Ant Colony Optimization, Bee Colony, …). Přístup mravenčích kolonií má mezi novými metaheuristikami výsadní postavení získané především atraktivností vlastního pojmenování.

Šandera (2014) [17], i přes značné rozdíly mezi jednotlivými metaheuristikami je jejich nejzákladnější princip stále stejný, a to vyzkoušet velké množství různých řešení z množiny přípustných řešení a vybrat z nich právě to nejlepší. Metaheuristiky implicitně předpokládají jistou souvislost mezi reprezentací řešení a hodnotou jeho účelové funkce, což představuje hlavní rozdíl oproti zcela náhodnému prohledávání. Z toho plyne, že většina odlišností pramení z rozdílných předpokladů o struktuře účelové funkce a o významu jednotlivých získaných informací v průběhu prohledávání.

[13] Obvykle se soubor řešení nazývá populace.

3.3.4 Rozdíly mezi metaheuristickými koncepty

Z implementačního pohledu jsou většinou metaheuristiky chápány jako modifikující se množina řešení. Mezi klíčové charakteristiky jednotlivých přístupů, které definují rozdíly mezi jednotlivými koncepty, patří Šandera (2014) [17]

- velikosti množiny řešení,
- doba, po kterou může jedno řešení přímo odvozovat nová řešení,
- počet prvků (rodičů) přímo podílejících se na produkci nových řešení,
- způsob uchovávání získaných informací,
- způsob předávání uchovaných informací,
- druh stochasticity,
- počet generovaných řešení

Každá metaheuristika trpí určitými nedostatky. Některé algoritmy vznikají pouze jako vylepšení dosavadních metod k překonání jejích problémů. Nicméně z *No Free Lunch teorému*[14] plyne, že dané vylepšení nikdy nemůže překonat všechny své nedostatky. Řešitelé úloh s pomocí metaheuristik si proto musí být vědomi jejich omezení a neočekávat nemožné. Jako nejčastější problémy jsou Šandera (2014) [17] uváděny předčasná konvergence[15], nedostatečná šíře prohledávané oblasti, možnost generování stále stejných řešení nebo přílišná variabilita a neschopnost se ustálit. Ke zvýšení výkonu metaheuristik se lze dostat několika odlišnými cestami. Od vývoje nových základních konceptů přes jejich různé kombinace a hybridizace až po využívání nejmodernějších hardwarových nástrojů.

[14] Tato exaktně prokázaná věta říká, že žádný algoritmus nemůže efektivně vyřešit všechny problémy. Nelze tedy zkonstruovat heuristiku, která by dokázala optimalizovat všechny účelové funkce lépe než kterákoliv jiná [17].

[15] Příliš rychlá konvergence do lokálního extrému bez možnosti dále prohledávat stavový prostor.

4 Exaktní řešení příkladu minimální kostry grafu s omezeními

Příklad minimální kostry s omezeními, jehož zadání je inspirováno příkladem v [29], je nejdříve řešen exaktní metodou. Daný algoritmus byl naprogramován v jazyce JAVA 1.7 pro získání všech neizomorfních koster grafu jak v případě kostry grafu bez omezení, tak kostry grafu s omezeními (penalizacemi). Následně je tento příklad řešen pomocí metaheuristiky Tabu Search v kapitole 6, kde je podrobně představen postup krok za krokem při ručním řešení příkladu minimální kostry grafu s omezeními.

Nechť je dána minimální kostra s omezeními, které zabraňují společnému výskytu určitých hran ve výsledné kostře grafu, nebo které dovolují některým hranám zahrnutí do výsledné kostry grafu jen za předpokladu, že je v dané kostře grafu zahrnuta i jiná hrana. Bez těchto omezení by bylo možné daný problém řešit přímo hladovým algoritmem, např. iterativním přidáváním nejlevnější hrany, která netvoří cyklus s předchozími zvolenými hranami, dokud není dosaženo výsledné minimální kostry grafu.

Zvolený problém je znázorněn na grafu, který se skládá z pěti uzlů, jak je patrné na následující sérii iterací. Minimální kostra daného grafu je tvořena sedmi hranami. Celkem je zapotřebí provést minimálně dvě iterace (ve větvi B), nebo pět iterací (v podvětvi b) větvě A), či sedm iterací (v podvětvi a) větvě A) pomocí metaheuristiky Tabu Search k získání výsledné minimální kostry grafu s omezeními s nejnižší hodnotou celkových nákladů. U každé hrany jsou uvedeny náklady dané hrany. Názvy jednotlivých hran (AB/AC/AD/BE/CE/CD/DE) představují binární proměnné, kde každá hrana je definováno jako

$$hrana\ j = \begin{cases} 1, jestliže\ hrana\ j\ je\ zahrnuta\ do\ kostry\ grafu \\ 0, jestliže\ hrana\ j\ není\ zahrnuta\ do\ kostry\ grafu \end{cases}$$

Tím pádem mohou být omezující podmínky pro danou kostru grafu zapsány následovně:

$AD = DE$, pak penalizace = 0, jinak penalizace = 100

$AB + AD + CD \leq 1$, pak penalizace = 0

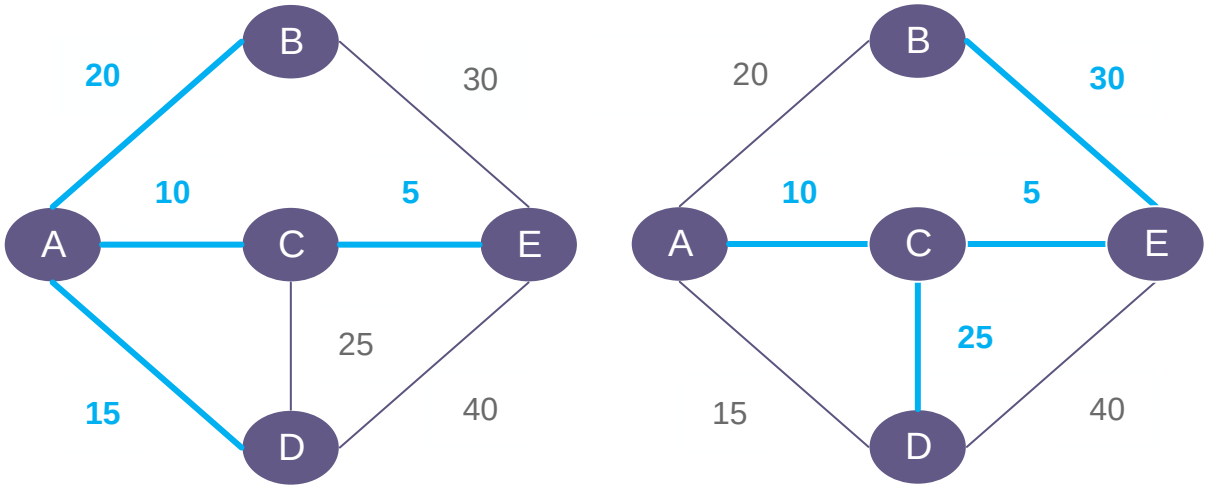
$AB + AD + CD = 2$, pak penalizace = 100

$AB + AD + CD = 3$, pak penalizace = 200

Aby bylo možné ohodnotit i ty kostry grafu, kde jsou dané podmínky porušeny, jsou zavedeny tzv. penalizace ve výši 100 jednotek nákladů pro každou hranu, která porušuje zadaná omezení. První omezující podmínka říká, že v kostře grafu smí být zahrnuta hrana AD

a zároveň hrana DE, nebo ani jedna z těchto hran. Je-li tato podmínka porušena, dochází k penalizaci ve výši 100 jednotek nákladů. Druhá podmínka znamená, že nejvýše jedna z uvedených tří hran AB, AD, CD může být zahrnuta do kostry grafu, aby výsledná kostra nebyla zatížena penalizací. Třetí podmínka určuje, že nejvýše dvě z uvedených tří hran AB, AD, CD mohou být zahrnuty do kostry grafu, aby penalizace byla ve výši 100 jednotek. Jsou-li v kostře grafu zahrnuty všechny tři hrany AB, AD, CD, je výsledná kostra grafu penalizovaná 200 jednotkami nákladů, jak definuje čtvrtá podmínka Výchozím přípustným řešením pro následné řešení metaheuristikou Tabu Search je minimální kostra získaná bez ohledu na zadaná omezení, která se projevují penalizací nákladů na dané hrany.

Jelikož zvolený příklad sestává z pěti vrcholů, je zřejmé dle Cayleho vzorce uvedeného v podkapitole 2.1, že je nejdříve třeba získat n^{n-2} různých koster, tedy celkem 125. Neizomorfních koster grafu je v tomto případě pouze 26. Aby se zamezilo možné duplicitě některých koster grafu a případnému vynechání některých variant, bylo generování potřebného počtu různých koster naprogramováno v jazyce JAVA 1.7. Nejdříve byly zjištěny všechny neizomorfní kostry grafu včetně jejich ohodnocení, viz Tabulka 4-1. Minimální kostru zvoleného grafu prezentuje Obrázek 4-1 vlevo. V dalším kroku byly jednotlivé kostry grafu ohodnoceny patřičnými penalizacemi, v případě, že zvolené hrany v dané kostře grafu nesplňovala zadaná omezení, viz Tabulka 4-2. V posledním kroku řešení byla vybrána minimální kostra grafu s ohledem na zadané penalizace, viz Obrázek 4-1 vpravo. Řešení minimální kostry grafu s ohledem na zadané penalizace má hodnotu účelové funkce o 20 jednotek nákladů vyšší než v případě bez ohledu na zadané penalizace.



Obrázek 4-1: Minimální kostra grafu bez ohledu na zadané penalizace (vlevo), minimální kostra grafu s ohledem na zadané penalizace (vpravo)

Tabulka 4-1: Výčet neizomorfních koster grafu bez penalizací

Vybrané hrany kostry grafu				Celkové ohodnocení
AB	BE	AC	AD	**75**
AB	BE	AC	CD	**85**
AB	BE	AC	DE	**100**
AB	BE	CE	AD	**70**
AB	BE	CE	CD	**80**
AB	BE	CE	DE	**95**
AB	BE	AD	CD	**90**
AB	BE	CD	DE	**115**
AB	AC	CE	AD	**50**
AB	AC	CE	CD	**60**
AB	AC	CE	DE	**75**
AB	AC	AD	DE	**85**
AB	AC	CD	DE	**95**
AB	CE	AD	CD	**65**
AB	CE	AD	DE	**80**
AB	CE	CD	DE	**90**
AB	AD	CD	DE	**100**
BE	AC	CE	AD	**60**
BE	AC	CE	CD	**70**
BE	AC	CE	DE	**85**
BE	AC	AD	CD	**80**
BE	AC	AD	DE	**95**
BE	AC	CD	DE	**105**
BE	CE	AD	CD	**75**
BE	CE	AD	DE	**90**
BE	AD	CD	DE	**110**

Tabulka 4-2: Výčet neizomorfních koster grafu s penalizacemi

Vybrané hrany kostry grafu				Celkové ohodnocení
AB	BE	AC	AD	**275**
AB	BE	AC	CD	**185**
AB	BE	AC	DE	**200**
AB	BE	CE	AD	**270**
AB	BE	CE	CD	**180**
AB	BE	CE	DE	**195**
AB	BE	AD	CD	**390**
AB	BE	CD	DE	**315**
AB	AC	CE	AD	**250**
AB	AC	CE	CD	**160**
AB	AC	CE	DE	**175**

AB	AC	AD	DE	**185**
AB	AC	CD	DE	**295**
AB	CE	AD	CD	**365**
AB	CE	AD	DE	**180**
AB	CE	CD	DE	**290**
AB	AD	CD	DE	**300**
BE	AC	CE	AD	**160**
BE	AC	CE	CD	**70**
BE	AC	CE	DE	**185**
BE	AC	AD	CD	**280**
BE	AC	AD	DE	**95**
BE	AC	CD	DE	**205**
BE	CE	AD	CD	**275**
BE	CE	AD	DE	**90**
BE	AD	CD	DE	**210**

5 Tabu Search

Následující část práce je zaměřena na přiblížení metaheuristiky zvané Tabu Search. Následně je tento přístup aplikován na vybranou úlohu minimální kostry s omezením a poté porovnán s exaktním řešením. Metaheuristika Tabu Search umožňuje přejít k řešení s horší hodnotou účelové funkce, jestliže není výhodnější žádný jiný pohyb. Metaheuristika Tabu Search zvyšuje výkon metaheuristiky Local Search, ze které vychází, díky relaxaci základních pravidel. Zavádějí se tzv. zákazy (tabu) k zabránění zkoumání již dříve zkoumaných řešení. Jestliže bylo potenciální řešení prověřováno před krátkým časovým úsekem, je označen jako „tabu“ a algoritmus jej již dále opakovaně nebere v úvahu po dobu setrvání zakázaných pohybů v tzv. tabu listu.

5.1 Metoda Tabu Search obecně

Algoritmus metaheuristiky Tabu Search byl poprvé zveřejněný koncem 80. let minulého století autorem Glover [30], [31]. Později byl přístup Tabu Search detailněji popsán v práci autorů Glover, Taillard a Werra (1993). Metoda tabu prohledávání je založena na metodě lokálního prohledávání. Během hledání nejlepšího řešení se vytváří tzv. seznam zakázaných změn (Tabu list), který obsahuje taková řešení, ke kterým se již nemá vrácet, protože již byla zpracována. Tabu list představuje tedy seznam pevné délky *n*, ve kterém je uvedeno posledních *n* změn ohodnocení, konkrétně dvojice (proměnná, hodnota). Při každém novém ohodnocení proměnné nějakou hodnotou je tato dvojice zapsána na začátek tabu listu a nejstarší (poslední) dvojice je z něj odebrána. Vždy, když se vybírá hodnota pro nějakou proměnnou, je zjišťováno, zda již není tato dvojice obsažena v tabu seznamu. Je-li obsažena, je zakázáno tuto hodnotu proměnné nastavit. Zmíněná technika tedy zamezuje opětovnému nastavení stejné hodnoty stejné proměnné v nějakém krátkém časovém intervalu, který je určen délkou tabu listu.

Při prohledávání sousedních řešení, je vždy hledáno řešení s nejnižší hodnotou účelové funcke s ohledem na seznam zakázaných změn. Metoda Tabu Search se přesouvá pouze do takového řešení v okolí posledního řešení, které není v zakázaném seznamu. Tabu list je obnovován v průběhu celého algoritmu. Tato metaheuristika je ovšem oproti lokálnímu prohledávání schopna se při optimalizaci vyhnout lokálnímu extrému a neuvíznout v něm stejně jako jiné pokročilé heuristické metody. Nalezená nejnižší hodnota účelové funkce může být vyšší než hodnota současného řešení (je-li současné řešení v lokálním minimu). Pokud je nejlepší sousední řešení obsaženo v tabu listu, nemůže být vybráno (je zakázané – tabu) a hledá se druhé nejlepší řešení.

Glover [30], slovo *tabu* (nebo *taboo*) pochází z jazyku tongan užívaného v Polynésii, kde daný výraz znamená posvátné věci, kterých se nesmí dotýkat. Nyní to také znamená zákaz, který je stanovený společenským zvykem. V metodě Tabu Search se tabu status zakázaných změn mění v závislosti na čase a okolnostech a na základě vyvíjející se paměti. Tabu status může být zrušen kvůli výhodnější alternativě. Jinými slovy v algoritmech implementujících tabu list existuje funkce, která povoluje výběr řešení, i když se toto řešení nachází v tabu listu. Daná funkce se nazývá aspirační kritérium (*aspiration criterion*). Vybrání zakázané výměny může nastat např. tehdy, když nastavením této hodnoty dané proměnné je získáno dosud nejlepší nalezené řešení. Délka tabu seznamu je vstupním parametrem. Díky větší hodnotě je zamezeno delším cyklům, na druhou stranu porovnání řešení s tabu seznamen zabere více času. Použití tabu seznamu dovoluje při optimalizaci vyskočit z lokálního extrému, čímž se stává hledání flexibilnějším.

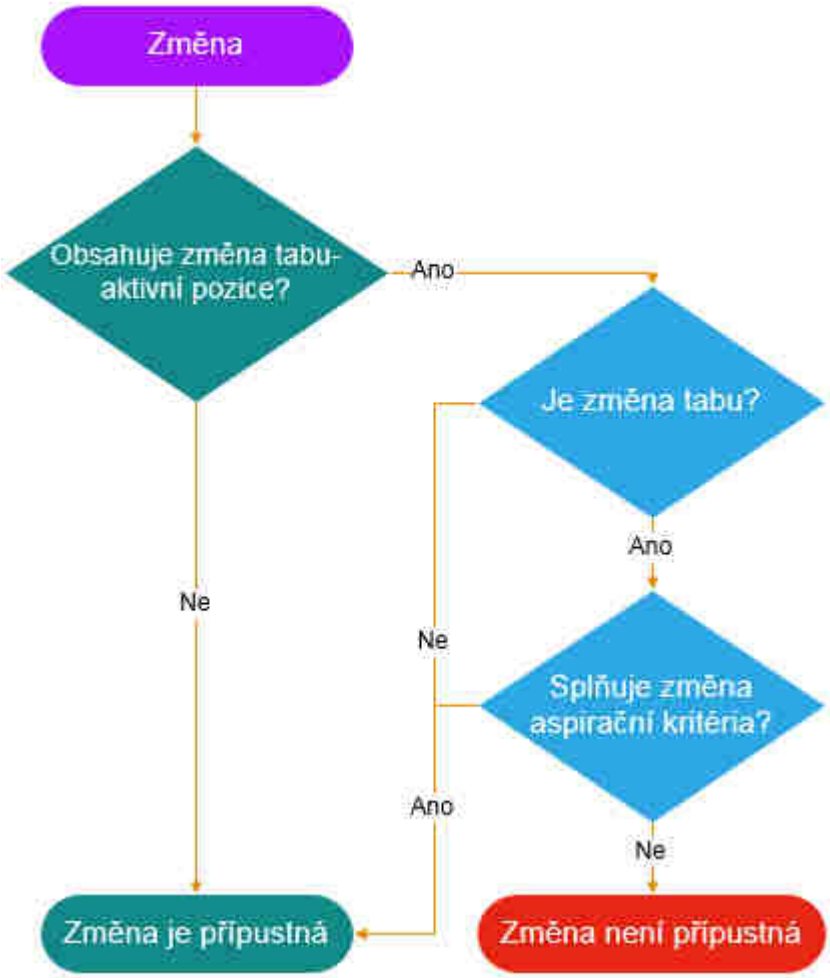
5.2 Rešerše – shrnutí základních poznatků a historie Tabu Search

Principy fungování metody Tabu Search mohou být přehledně zobrazeny a shrnuty pomocí rozhodovacího stromu, viz Obrázek 5-1, Obrázek 5-2 a Obrázek 5-3. Přesunutí se do dalšího řešení je přípustné, jestliže se nejedná o zakázaný pohyb uvedený v tabu listu, pokud je při pohybu splněno aspirační kritérium či pokud změna neobsahuje tabu-aktivní pozice.

Obrázek 5-2 znázorňuje rozhodovací strom metaheuristiky Tabu Search s krátkodobou pamětí. Glover (2001) [29], jádro metaheuristiky Tabu Search spočívá v procesu s krátkodobou pamětí. Mnoho strategických úvah, které jsou podkladem pro tento proces, se znovu objeví v zesílené míře, ale ne výrazně věcně změněné v paměti dlouhodobého procesu. Krátkodobá paměť metaheuristiky Tabu Search představuje formu agresivního průzkumu, který se snaží o realizaci nejlepšího možného pohybu (s nejvyšším ohodnocením) s ohledem na splnění zadaných omezujících podmínek. Tato omezení představují tzv. Tabu restrikce, které jsou navrženy tak, aby se předešlo obrácení směru či někdy opakování některých tahů. Požadovaného se dosáhne označením vybraných vlastností daných pohybů za zakázané (tabu). Hlavním cílem tabu restrikcí je povolit metaheuristice Tabu Search jít za rámec bodů lokálního optima a zároveň ještě vytvářet velmi kvalitní pohyby v každém kroku.

Bez těchto restrikcí by mohla daná metaheuristika vybrat „nejlepší“ pohyb z lokálního optima a poté teoreticky v dalším kroku spadnout zpět do lokálního optima díky výběru nejlepšího pohybu v daném místě. Obecně, tabu restrikce jsou určeny pro to, aby se zabránilo právě takovému cyklickému chování. V širším kontextu jsou tabu restrikce určeny k tomu, aby přiměly proces hledání sledovat novou trajektorii, jestliže dojde k zacyklení v užším slova smyslu

(což znamená přehodnocení některých dřívějších řešení). Tato omezení nefungují izolovaně, ale jsou vyváženy použitím aspiračních kritérií.



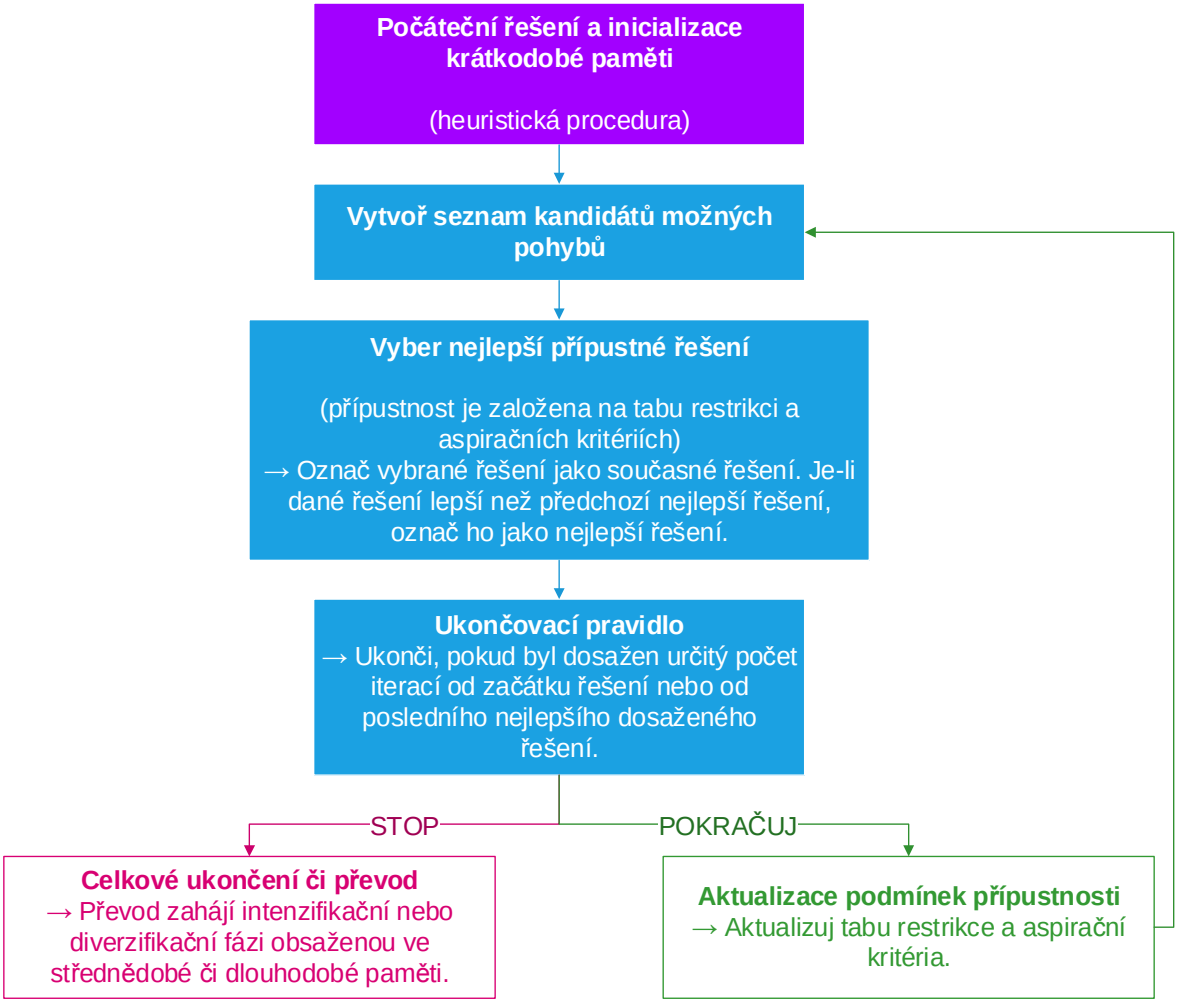
Obrázek 5-1: Rozhodovací strom přípustnosti dané změny *Zdroj: [29]*

Nejběžnější a nejjednodušší aspirační kritérium spočívá v povolení určitého pohybu, přestože je uveden v tabu listu, a to jen v případě, jestliže daný pohyb zajistí lepší hodnotu účelové funkce, než byla dosud získána. Jsou používána jiná a daleko komplikovanější aspirační kritéria, ale pouze zřídka, jak je například praktikováno v [32], [33].

Obrázek 5-3 znázorňuje rozhodovací strom výběru nejlepšího přípustného řešení. Glover (2001) [29], kritický krok, který ztělesňuje agresivní orientaci krátkodobé paměti, je výběr nejlepšího přípustného pohybu ze seznamu kandidátů. Nejprve se každý z pohybů ze seznamu kandidátů[16] ohodnotí podle pořadí. V mnoha nastaveních může být zpočátku vyhodnocení pohybu založeno na změně hodnoty účelové funkce (to znamená rozdíl mezi hodnotami účelové funkce před a po aplikaci daného pohybu). V ostatních případech,

[16] Problematika tvorby a aktualizace kandidátních listin, které jsou zvláště důležité pro problémy vrstev (layer problems), je popsána v Glover (1989a) [30].

kdy důsledky daného pohybu nelze tak jednoduše určit nebo nejsou-li všem proměnným v současné době přiřazeny hodnoty, může být ohodnocení pohybů založeno na generování uvolněných či přibližných řešeních nebo může jednoduše využívat lokální míry atraktivity (stejně jako v lokálních rozhodovacích pravidlech pro problémy job-shop scheduling).



Obrázek 5-2: Rozhodovací strom metaheuristiky Tabu Search s krátkodobou pamětí *Zdroj: [29]*

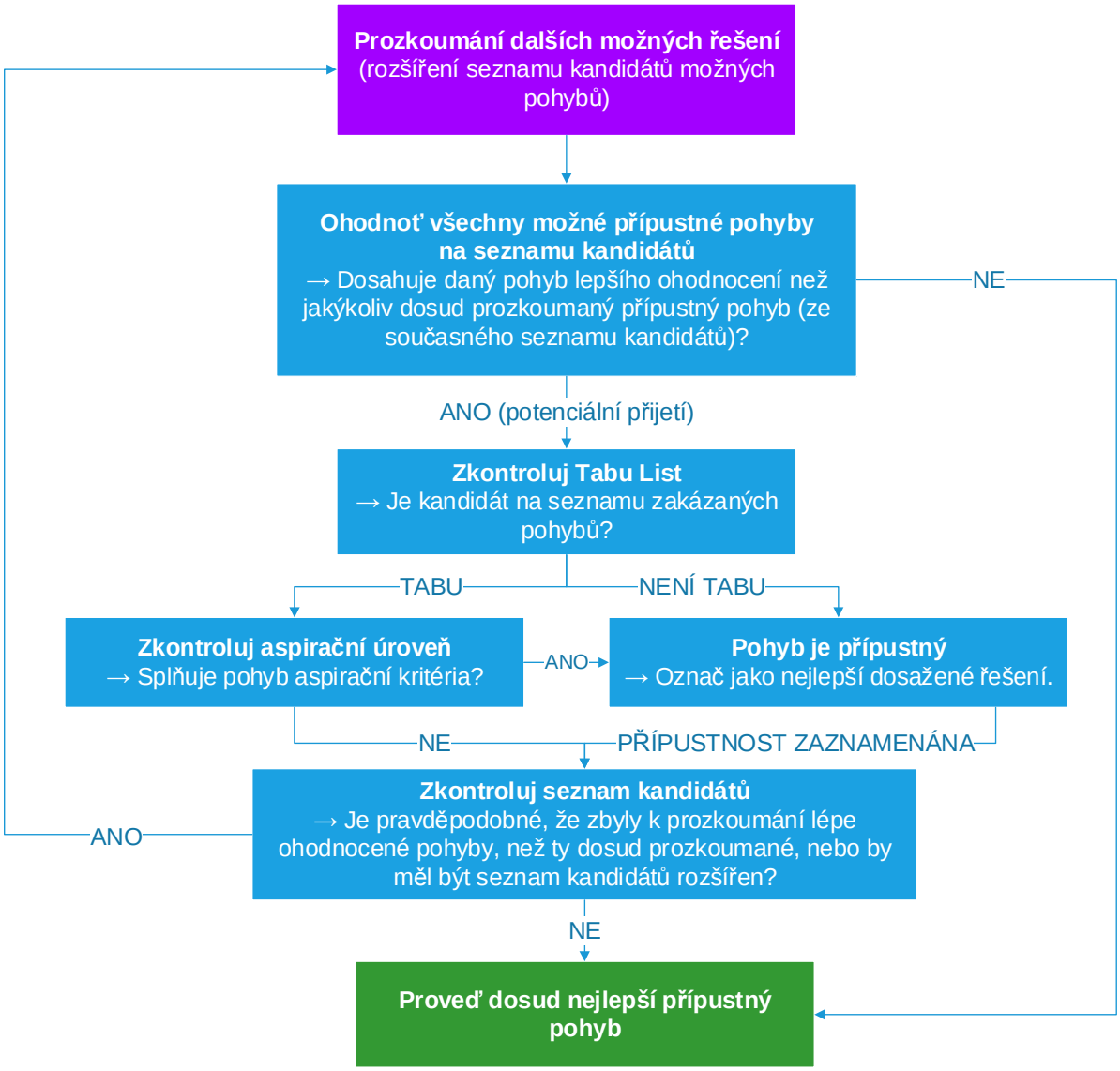
Glover (2001) [29], počet tahů, které jsou označeny jako zakázané (tabu), je obecně malý vzhledem k počtu tahů, které jsou k dispozici. Za předpokladu, že náklady na vyhodnocení tahů nejsou velké, je obvykle výhodné nejdříve ověřit, zda daný pohyb má vyšší ohodnocení než jeho přípustný předchůdce před tím, než se bude ověřovat jeho tabu status. Kontrola tabu statusu je prvním krokem při zkoumání přípustnosti daného pohybu. V případě, že daný pohyb není zakázaný (tabu), je okamžitě přijat mezi přípustné pohyby. V opačném případě mají aspirační kritéria možnost přepsat tabu status, což poskytuje danému pohybu druhou šanci na kvantifikování jako přípustný pohyb.

Zkoumání dalšího pohybu může být zahrnuto ve strategii seznamu kandidátů. Jsou-li v některých případech tabu restrikce a aspirační kritéria dostatečně omezující, žádný

z dostupných pohybů nebude považován za přípustný. „Nejméně nepřípustný“ pohyb je uložen, aby se vypořádal s takovou možností, a je vybrán, pokud se neobjeví žádné přípustné alternativy.

Je třeba zdůraznit, že různé, i metaheuristické, přístupy se mohou navzájem kombinovat. Metaheuristika Tabu Search je šasto doprovázena přístupy genetických algoritmů [34], [35], Lagrangeových relaxací [35], programování s omezeními (Constraint Programming) [37], [38], [39] a celočíselného programování (Integer Programming) [40].

Co se týče nových trendů ve využívání metaheuristiky Tabu Search, již se opouští od klasických aplikací (problémy z teorie grafů, rozvrhovací problémy, rozvozní problémy) a přechází se k novým, a to ke kontinuální optimalizaci (Continuous Optimization) [41], [42], [43], [44], [45], vícekriteriální optimalizace (Multi-criteria Optimization) [44], [46], stochastickému programování (Stochastic Programming) [47], [48], smíšenému celočíselnému programování (Mixed Integer Programming) [49], [50], problémům rozhodování se v reálném čase (Real-time Decision Problems) [44], [51] atd. Tyto nové aplikace mohou vést k rozšíření technik zahrnutých v aparátu metaheuristiky Tabu Search.



Obrázek 5-3: Rozhodovací strom výběru nejlepšího přípustného řešení *Zdroj: [29]*

5.2.1 Typické úlohy metody tabu prohledávání

Tabulka 5-1 ilustrativně znázorňuje široký přehled aplikací metaheuristiky Tabu Search. Jednotlivé úlohy jsou uváděny v originálním ustáleném anglickém znění. Následně jsou představeny základní úlohy operačního výzkumu, a to problém minimální kostry, problém obchodnícho cestujícího a rozvozní úlohy, které byly řešeny i metaheuristikou Tabu Search.

Dále jsou představeny tři významné úlohy, které jsou mnohdy řešeny metaheuristikou Tabu Search, a to úloha minimální kostry, úloha obchodního cestujícího a rozvozní problémy.

Tabulka 5-1: Ilustrativní přehled aplikací metaheuristiky Tabu Search *Zdroj: [52]*

Scheduling Flow-Time Cell Manufacturing Heterogeneous Processor Scheduling Workforce Planning Classroom Scheduling Machine Scheduling Flow Shop Scheduling Job Shop Scheduling Sequencing and Batching	**Telecommunications** Call Routing Bandwidth Packing Hub Facility Location Path Assignment Network Design for Services Customer Discount Planning Failure Immune Architecture Synchronous Optical Networks
Design Computer-Aided Design Fault Tolerant Networks Transport Network Design Architectural Space Planning Diagram Coherency Fixed Charge Network Design Irregular Cutting Problems	**Production, Inventory and Investment** Flexible Manufacturing Just-in-Time Production Capacitated MRP Part Selection Multi-item Inventory Planning Volume Discount Acquisition Fixed Mix Investment
Location and Allocation Multicommodity Location/Allocation Quadratic Assignment Quadratic Semi-Assignment Multilevel Generalized Assignment Lay-Out Planning Off-Shore Oil Exploration	**Routing** Vehicle Routing Capacitated Routing Time Window Routing Multi-Mode Routing Mixed Fleet Routing Traveling Salesman Traveling Purchaser
Logic and Artificial Intelligence Maximum Satisfiability Probabilistic Logic Clustering Pattern Recognition/Classification Data Integrity Neural Network |Training and Design	**Graph Optimization** Graph Partitioning Graph Coloring Clique Partitioning Maximum Clique Problems Maximum Planner Graphs P-Median Problems
Technology Seismic Inversion Electrical Power Distribution Engineering Structural Design Minimum Volume Ellipsoids Space Station Construction Circuit Cell Placement	**General Combinational Optimization** Zero-One Programming Fixed Charge Optimization Nonconvex Nonlinear Programming All-or-None Networks Bilevel Programming General Mixed Integer Optimization

i. Minimální kostra

Úloha minimální kostry, resp. úloha minimální kostry s omezeními není v mnoha případech řešena metodou Tabu Search. Je-li tomu tak, většinou se tak autoři různých publikací rozhodnou pro porovnání výsledků více metaheuristických přístupů navzájem nebo se jedná o speciálně obtížnou úlohu.

Autoři Singh a Baghel (2008) řešili ve své práci [52] úlohu minimální kostry s omezeným počtem listů (LCMST), která má za cíl nalezení minimální kostry s alespoň *n* listy, pomocí metaheuristiky ACO (ACO-LCMST) a Tabu Search (TS-LCMST). Došli k závěru, že výkon obou zmíněných metaheuristik je velmi podobný vzhledem ke kvalitě řešení, ale TS-LCMST je daleko rychlejší než ACO-LCMST.

[54] Kasperski, Makuchowski a Zieliski (2012) řešili úlohu minimax ztráty úlohy minimální kostry s intervalovými daty (MinMax Regret Minimum Spanning Tree Problem with Interval Data) pomocí metaheuristiky Tabu Search, která se osvědčila i při problémech většího rozsahu. V řešené úloze nejsou přesně známé náklady na hrany, pouze možné intervaly výše jednotlivých nákladů na hranu. Předpokládá se, že hodnoty jednotlivých nákladů na hranu nezávisí na nákladech zbývajících dosud nezvolených hran.

ii. TSP

Metoda Tabu Search je jednou z nejpreferovanějších metaheuristik pro řešení problému obchodního cestujícího. Pro porovnání výsledků metody Tabu Search a jiných metaheuristik je možné např. nahlédnout do následujících prací [55], [56], [57], [58]. Obsáhlejší přehled výpočetních metod problému TSP je k nahlédnutí v práci autorů Applegate, Bixby, Chvátal, a Cook (2006) [59] a autorů Gutin a Punnen (2002) [60].

Ačkoli symetrický problém obchodního cestujícího (STSP) je vhodně řešen pomocí metaheuristik, problém se často vyskytuje u asymetrické varianty problému obchodního cestujícího (ATSP). Možné důvody jsou rozebrány v práci [61]. Basu a Ghosh (2008) [62] poznamenali, že většina publikované literatury zaměřené na problém TSP řešeného pomocí metaheuristiky Tabu Search, se zabývá pouze symetrickou verzí TSP s méně než 500 uzly[17]. Toto zjištění je nečekané, jelikož osvědčená metaheuristika jako je Tabu Search, je určena k řešení rozsáhlých praktických problémů obchodního cestujícího. Řešením asymetrické varianty TSP pomocí třech modifikací přístupu Tabu Search se zabývá Basu (2012) [63] a autoři Basu, Gajulpalli a Ghosch (2013) [64].

Problémem obchodního cestujícího se ziskem (TSP with profit) se zabývají autoři Feillet, Dejax, a Gendrau (2005) [65]. Jedná se vlastně o dvoukriteriální problém obchodního cestujícího s opačnými cíli – jedním je dosažení zisku, což nutí obchodníka cestovat, zatímco druhým cílem je minimalizovat cestovní náklady (s možností vynechat některé vrcholy).

iii. VRP

Rozvozní problémy jsou velmi studovanou kapitolou mezi optimalizačními problémy. Mnoho variant rozvozních úloh bylo v literatuře dosud postupně prezentováno, jako např. rozvozní úlohy s více výchozími místy (multi-depot), s časovými okny (time-windows), se smíšeným vozovým parkem (mixed fleet), a další.

Metoda Tabu Search byla pro problém VRP prvně představena autorem Willard (1989), který prvně transformoval VRP problém na TSP problém. V práci Puerza a Franca (1991)

[17] Pouze 11 prací se do r. 2010 zabývalo úlohami TSP s více než 400 uzly [63].

se přechází z jednoho řešení do dalšího pomocí výměny vrcholů mezi dvěma cestami. Osman (1991, 1993) využívá kombinace 2-optimálních pohybů – přeřazení vrcholů na jiné cesty a výměna vrcholu mezi cestami. Taillard (1992) ve své práci rozděluje při řešení úlohy VRP pomocí přístupu Tabu Search sadu vrcholů do jednotlivých klastrů prostřednictvím přesunu vrcholů z jedné trasy do druhé. Další algoritmus byl vyvinut autory Semet a Taillard (1993) pro problém reálného rozvozního problému s určitými vlastnostmi. Později se autoři Gendreau, Hertz a Laporte (1994) věnují ve své práci [58] přístupu zvaném Taburoute. Jedná se o novou verzi metaheuristiky Tabu Search. Pomocí přístupu Taburote je řešena rozvozní úloha s kapacitami a s omezeními vzdálenosti rozvozu.

Práce [66] se věnuje řešení VRP metaheuristikou Tabu Search, a to konkrétně třem typům náhodně vygenerovaných problémů VRP (s nedostatkem zdrojů, kritický a s příliš mnoho zdroji). Výsledky jsou následně porovnávány s dalšími přístupy. Prvním je Hill Climbing (HC), při němž jsou řešení opakovaně vylepšována. Nevýhodou tohoto přístupu je možnost zabřednutí v lokálním minimu. V převážné většině případů poskytuje Tabu Search lepší výsledky než Hill Climbing.

Autoři Ceschia, Gaspero a Schaerf (2010) [67] ve své práci řeší rozvozní úlohu s heterogenními vozidly, s časovými okny a s náklady v závislosti na zvoleném dopravci. V mnoha publikacích jsou řešeny zjednodušené verze rozvozních problémů. Reálné příklady jsou mnohem komplexnější a náročnější. Proto se autoři rozhodli vyřešit reálný problém rozvozní úlohy s různorodým vozovým parkem (heterogeneous fleet), s vícedenním horizontem plánování (multi-day planning horizon), s komplexní nákladovou funkcí pro jednotlivá vozidla závislou na dopravci (a complex carrier-dependent cost function for vehicles) a s možností ponechání nenaplánovaných objednávek (the possibility of leaving orders unscheduled). Různé modifikace metaheuristiky Tabu Search jsou tedy aplikovány na rozvozní problém s vlastním vozovým parkem (Vehicle Routing Problem with Private Fleet) a s objednaným dopravcem (Vehicle Routing Problem with Common carrier) a následně jsou porovnány výsledky jednotlivých verzí.

5.3 Algoritmus metody Tabu Search

Algoritmus metaheuristiky tabu prohledávání (Tabu Search) vychází z metody lokálního hledání, proto jsou v následující části práce popsány oba algoritmy a vzájemné rozdíly.

5.3.1 Algoritmus metody lokálního prohledávání

Algoritmus základní metaheuristiky lokálního prohledávání je následující:

1. Je zvoleno výchozí přípustné řešení $x \epsilon X$ a je položeno $x^* = x$;
2. Je definováno $U(x)$ jako okolí řešení x;
3. Nechť $x' \epsilon U(x)$ je nejlepší řešení z okolí bodu x;
4. Pokud $f(x') < f(x^*)$, pak $x^* = x'$, kde x^* je dosud nejlepší nalezené řešení;
5. Je-li splněno ukončovací pravidlo, algoritmus končí, jinak se přejde na krok 2.

Metoda lokálního prohledávání tedy iterativně prohledává okolí aktuálního řešení, a pokud najde nějaké lepší řešení v tomto okolí, aktualizuje dosavadní řešení. Následně se celý proces opakuje s novým řešením, dokud není splněna podmínka zastavení, kterou může být fakt, že v okolí se žádné lepší řešení nenachází, případně po dosažení předem stanoveného počtu iterací.

5.3.2 Algoritmus metody tabu prohledávání

Algoritmus metaheuristiky tabu prohledávání vychází z metody lokálního hledání, jak je patrné z následujícího algoritmu:

1. Je zvoleno výchozí přípustné řešení $\mathrm{x} \epsilon \mathrm{X}$, je položeno $\mathrm{x}^* = \mathrm{x}$ a toto přípustné řešení je vloženo do Tabu seznamu. Při inicializaci algoritmu je Tabu seznam prázdný, po každé iteraci se do zakázaného seznamu přidá transformace inverzní k transformaci, která poskytla nejlepší řešení v rámci sousedství v předcházejícím kroku. (Např. do zakázaného seznamu se zapíší pořadová čísla vyměněných úloh). Pokud došlo k překročení definované povolené délky zakázaného seznamu, jsou z tohoto seznamu odstraněny nejstarší prvky (metoda FIFO – First In First Out);
2. Je definováno $U(x)$ jako okolí řešení x;
3. Nechť $x' \epsilon U(x)$ je nejlepší řešení z okolí bodu x při respektování Tabu seznamu (tj. pokud nejsou všechna řešení z okolí v Tabu seznamu), pak je toto řešení přidáno do TABU seznamu a porovnáno s dosud nejlepším nalezeným řešením:
   Pokud $f(x') < f(x^*)$, pak $x^* = x'$, kde x^* je dosud nejlepší nalezené řešení;
4. Je-li splněno ukončovací pravidlo, algoritmus končí, jinak se přejde na krok 2. Ukončovacím kritériem může být např. určitý počet kroků nebo nejčastěji absence změny x^* po určitém počtu kroků.

Je zřejmé, že uchovávání takového seznamu je poměrně náročné na paměť, zejména při výpočtech s mnoha iteracemi. Z tohoto důvodu bývá do algoritmu zakomponované tzv. *expirační kritérium*, které definuje počet iterací, po kterých je daný přechod ze seznamu zakázaných přechodů odstraněný. Implementace této vlastnosti je většinou poměrně jednoduchá, postačí nastavení velikosti záznamu zakázaných přechodů na pevnou velikost a zabezpečení funkcionality fronty, čímž po naplnění seznamu vždy při přidání nového

přechodu z něho automaticky vypadne nejstarší přechod. Významnou úsporu paměti je možné dosáhnout použitím vhodných údajových struktur při uchovávání údajů o zakázaných přechodech. Nejhorší variantou by bylo ukládání obou řešení, tedy původního a nového, což by vzhledem ke spotřebě paměti bylo neúnosné. Ideálním řešením tohoto problému je ukládání jen minimálního množství údajů (atributů) o daném přechodu tak, aby se spolehlivě vyloučilo opakování tohoto přechodu až do doby, než vypadne ze zakázaného seznamu, jak uvádí [52].

Tento způsob uchovávání tabu údajů přináší možnost problému spočívajícího v (i když díky expiračnímu kritériu jen dočasného) zákazu dobrých řešení, jak uvádí [52]. Proto metoda tabu prohledávání zavádí tzv. *aspirační kritérium*. Jedná se o pravidlo, které umožňuje za určitých okolností vykonat i přechod, který je aktuálním tabu seznamem zakázaný. Nejběžnějším aspiračním kritériem bývá povolení přechodu k řešení s lepší hodnotou účelové funkce, než je hodnota účelové funkce nejlepšího dosud nalezeného řešení (které se tedy z tohoto důvodu musí uchovávat) a povolení zakázaného přechodu v případě, že neexistuje žádny povolený přechod (povolené okolí aktuálního řešení je kvůli tabu záznamu prázdné).

Algoritmus tabu prohledávání pracuje deterministicky, tzn., vybírá si z okolí aktuálního řešení vždy ten nejlepší možný přechod, avšak díky tabu seznamu a aspiračním kritériím je schopný se vyhnout uvíznutí v lokálním optimu. Díky tomu může poskytovat poměrně kvalitní řešení.

V mnoha případech je při řešení problému metaheuristikou Tabu Search zapotřebí zahrnout další prvky pro zajištění plně efektivního hledání co nejlepšího řešení. Pro zajištění zintenzivnění prohledávacího procesu se využívá středně-dlouhodobá paměť. Pro diverzifikaci prohledávání je nutné změnit množinu prohledávaných sousedních pohybů. Je totiž možné, že se prohledávání nedostane do nejzajímavějších částí množiny přípustných řešení, a tudíž získané řešení se dosti vzdálí od globálního optima. Prohledávání je tedy nuceno se zaměřit na dosud neprozkoumanou oblast přípustných řešení, což je většinou založeno na dlouhodobé paměti.

6 Aplikace Tabu Search na kostru grafu s omezeními

Na následujícím jednoduchém příkladu minimální kostry s omezeními, jehož zadání je inspirováno příkladem v [29] a bylo představeno v kapitole 4, je přehledně vysvětlen přístup řešení metaheuristiky Tabu Search. Výchozím přípustným řešením pro následné řešení metaheuristikou Tabu Search je minimální kostra získaná bez ohledu na zadaná omezení.

Pro užití metaheuristiky Tabu Search je zapotřebí zavést pojem záměna hran (*edge swap*), což představuje přidání hrany a zároveň odebrání jedné z původních hran za účelem transformace současné kostry do nové kostry s nižšími náklady. Odebraná hrana je vždy taková hrana, která tvoří cyklus po přidání nové hrany. Je uvažováno obvyklé pravidlo volby nového přípustného řešení na základě krátkodobé paměti Tabu Search. V daném příkladě byla aplikována krátkodobá paměť po dobu dvou iterací. Jinými slovy přidaná hrana do kostry grafu nesmí být odebrána po dobu dvou iterací od jejího přidání, jelikož je uvedena v zakázaném seznamu (*tabu list*). Vybraný pohyb, resp. záměna dvou hran, reprezentuje přípustný pohyb s nejvyšším ohodnocením, jímž je takový přípustný pohyb, který vytváří novou kostru s nejnižšími náklady (včetně možných penalizací za porušení stanovených omezení). Pro definování zakázaných (*tabu*) omezení je stanovena přidaná hrana jako vlastnost daného pohybu, kterému je přiřazen tabu status (ve chvíli, kdy je hrana přidána ke stávající kostře grafu). Tabu klasifikace je tím pádem přenesena na pohyby, které obsahují hranu s omezením zakazující budoucí pohyb, který by vedl k odstranění hrany, po dobu existence dané hrany v tabu seznamu.

6.1 Použití krátkodobé paměti

V daném příkladě je umožněno pouze dvěma hranám, aby byly obsaženy v tabu seznamu v jakémkoli čase. Tato skutečnost znamená, že každá přidaná hrana zůstane tabu během dvou iterací a poté je odstraněna z tabu listu (jehož délka = 2), čímž je osvobozena od svého tabu statusu. Aspirační kritérium, které bylo vybráno k přepsání tabu statusu, je jednoduchým kritériem. Zmíněné kritérium umožňuje současnému pohybu zahrnout zakázanou hranu, jestliže výsledná kostra je lepší, než dosud nejlépe ohodnocená nalezená kostra. Pro sledování fungování metody Tabu Search jsou podrobně graficky rozebrány jednotlivé iterace v obrízkové sérii iterací. Výchozím přípustným řešením je optimální řešení minimální kostry grafu bez ohledu na možné penalizace, viz Obrázek 4-1 (vlevo). Ukončovací pravidlo

bylo stanoveno na konec výpočtu po druhém dosažení stejného řešení s nejmenší hodnotou účelové funkce v každé větvi řešení.

Jednotlivé zakázané hrany jsou podbarveny v tabulce červeným pozadím, v obrázku jsou uvedeny s červenou poznámkou „tabu“. Optimální řešení je označeno žlutě v popisu příslušné iterace. Je-li daná hrana zvolena do aktuální kostry grafu, je podbarvena modře. Označení „drop“ znamená, že daná hrana opouští v dalším kroku kostru grafu, označení „add → tabu“ určuje, která hrana bude v dalším kroku zařazena do kostry grafu a do tabu listu.

Iterace 1: Mezi současnými možnostmi přidání nebo odebrání hran na kostře grafu výchozího přípustného řešení pro vytvoření nové kostry grafu existují dvě nevýhodnější změny se stejnou hodnotou celkových nákladů. První variantou (A) je přidat hranu BE a odstranit hranu AB. Druhou variantou (B) je přidat hranu CD a odstranit hranu AD. Uvedený pohyb eliminuje porušení zadaných podmínek na přípustných hranách a zároveň rovněž redukuje výši penalizace ze 200 na 100, zatímco zvyšuje zbývající složku nákladů také z 50 na 60 (přidáním 25-15, což uvádí rozdíl nákladů mezi přidanou a odebranou hranou).

A) Iterace 1: Došlo k přidání hrany BE a odstranění hrany AB. Tento pohyb eliminuje porušení zadaných podmínek na přípustných hranách a zároveň redukuje výši penalizace ze 200 na 100, zatímco zvyšuje zbývající složku nákladů z 50 na 60 (přidáním 30-20, což představuje rozdíl nákladů mezi přidanou a odebranou hranou). Zmíněný pohyb získává tabu status, čímž je propůjčen tabu stav na pohyby, které odstraní tuto hranu.

A) Iterace 2: Podle zvoleného pravidla pro identifikaci pohybové vlastnosti, která je označena jako tabu, byla v první iteraci přidána hrana BE. Nejlepší změnu v celkových nákladech tvoří mezi zbývajícími možnými pohyby přidání hrany DE a odstranění hrany AC. Tato kostra grafu nebude nijak penalizována, nicméně vybraný pohyb zhoršuje náklady samotné kostry grafu z 60 na 90 jednotek nákladů.

A) Iterace 3: Hrana DE, přidaná do kostry grafu pomocí výměny v iteraci 2, se stává spolu s hranou BE zakázanou hranou. Nejlepší variantou vzhledem ke změně celkových nákladů je přidání hrany AC a odebrání hrany CE. Jako v předešlé iteraci nejsou porušeny zadané podmínky, nicméně hodnota celkových nákladů vzrostla o 5 jednotek. Tímto pohybem opět nastává situace, kdy se nabízejí dvě možnosti záměn hran jako nejlepší přípustná řešení vzhledem k zakázanému seznamu hran. Jelikož je hrana AC v tabu listu a daný pohyb nesplňuje aspirační kritérium vytvořením kostry grafu s lepším celkovým ohodnocením než dosud získaným, není daný pohyb realizován.

Současné přípustné pohyby, u kterých se zdá, že by mohly poskytnout lepší řešení, porušují zadaná omezení. Při volbě zakázané výměny hran CE a AC by bylo dosaženo identické kostry grafu jako v iteraci 2 větvě A, resp. v předešlé iteraci. Kvůli penalizacím poskytují zmíněné

pohyby ve výsledku vyšší náklady, tedy i horší řešení. Zmíněné současné nejvýhodnější pohyby jsou atraktivnější než současné zakázané pohyby, a to z důvodu možnosti prozkoumání takových koster grafu, které dosud ještě nebyly v předešlých iteracích zkoumány. Výše popsaná skutečnost ukazuje, že tabu omezení nemá vždy vliv na preferovanou volbu.

A) Iterace 3 a): Hrana BE je propuštěna ze svého tabu postavení, protože tabu seznam se skládá pouze ze dvou prvků, jimiž aktuálně je hrana AC a DE. První variantou nejlepšího přípustného řešení vzhledem k současnému tabu seznamu je přidání hrany AB a odebrání hrany BE. Zatímco celkové náklady kostry se snižují o 10 jednotek, dochází k porušení druhé omezující podmínky, a tedy k penalizaci ve výši 100 jednotek.

A) Iterace 4 a): V aktuálním tabu seznamu se nachází hrana AB a AC. Nejlepším dostupným pohybem je přidání hrany CE a odebrání hrany AD, čímž celkové náklady kostry grafu klesnou o dalších 10 jednotek. V tomto případě dochází k porušení první omezující podmínky, tedy dané řešení je opět penalizováno ve výši 100 jednotek nákladů.

A) Iterace 5 a): Hrana AB spolu s hranou CE tvoří dvě naposledy přidané hrany, tudíž jsou označeny jako zakázané hrany. Nejlepším současným pohybem je přidání hrany CD a odebrání hrany DE, což opět sníží celkové náklady kostry grafu o 15 jednotek. Při této podobě kostry grafu dochází k porušení druhé omezující podmínky, tedy k penalizaci ve výši 100 jednotek.

A) Iterace 6 a): Současnými zakázanými hranami je hrana CE a hrana CD, nicméně tabu list nezabraňuje výběru nejlepšího možného pohybu, a tudíž jako nejvýhodnější záměna hran se jeví jednoznačně přidání hrany BE a odebrání hrany AB.

A) Iterace 7 a): Zmíněným pohybem bylo dosaženo dosud nejlepší změny v celkových nákladech, které klesají na hodnotu 70 jednotek bez jakékoliv penalizace. Všechny omezující podmínky jsou splněny. Současná kostra grafu je tedy novým lokálním optimem a dosud nejlepším řešením. Vybraný následující nejvýhodnější pohyb zhoršuje náklady kostry grafu. Jen z řešení pomocí metaheuristiky Tabu Search by nebylo možné zjistit, zda se jedná o optimální řešení či nikoliv. Jelikož daný problém byl díky jeho malému rozměru řešen i exaktně, je možné tvrdit, že se jedná o optimální řešení, viz Tabulka 4-1. Žádný pohyb nevede k lepšímu řešení vzhledem k zadaným omezením. V některých realizacích nejsou tabu seznamy aktivovány, dokud není dosaženo prvního optimálního řešení.

Jak je patrné z následujících iterací, 7. iterace větve A) a podvětve a) je spolu s 11. iterací větve A) a podvětve a) optimálním řešením vzhledem k zadaným omezujícím podmínkám minimální kostry grafu. Dále je zřejmé, že posloupnost jednotlivých nejlepších pohybů vzhledem k aktuálnímu seznamu zakázaných hran tvoří od 3. iterace větve A) a podvětve a) cyklus až do 14. iterace větve A) a podvětve a), kde je získána identická minimální kostra

se stejnými aktuálními zakázanými hranami a se stejným ohodnocením celkových nákladů jako minimální kostra v 3. iteraci větve A). Z tohoto důvodu bylo i přes stanovené ukončovací pravidlo pokračováno ve výpočtu až do 14. iterace. Jelikož by 15. iterace větve A byla identická s iterací 4 a) nebo 4 b) větve A, není jíž do série iterací zahrnuta. V dané podvětvi a) byla získána dvě identická optimální řešení minimální kostry grafu se zadanými omezujícími podmínkami, a to v 7. a 11. iteraci, nicméně s rozdílnými zakázanými hranami. Nejedná se tedy o naprosto shodná řešení. Následující posloupností iterací bylo dokázáno, že v této větvi se daná řešení cyklicky opakují a že i přes zpočátku se měnící zakázané hrany se dojde opět k již dosaženému nejlepšímu řešení s identickými hodnotami celkových nákladů a tabu seznamu.

A) Iterace 3 b): Nyní je zapotřebí prozkoumat druhou variantou nejlepšího přípustného řešení vzhledem k současnému tabu seznamu ze 3. iterace větvě A). Tato druhá varianta představuje přidání hrany CE a odebrání hrany AD. Zatímco celkové náklady kostry se snižují jako v iteraci 3 a) o 10 jednotek, v této variantě dochází k porušení první omezující podmínky, a tedy rovněž k penalizaci ve výši 100 jednotek.

A) Iterace 4 b): Současný seznam zakázaných hran je tvořen hranou AC a DE. Pohyb s nejvyšším ohodnocením představuje nyní přidání hrany CD a odebrání hrany DE.

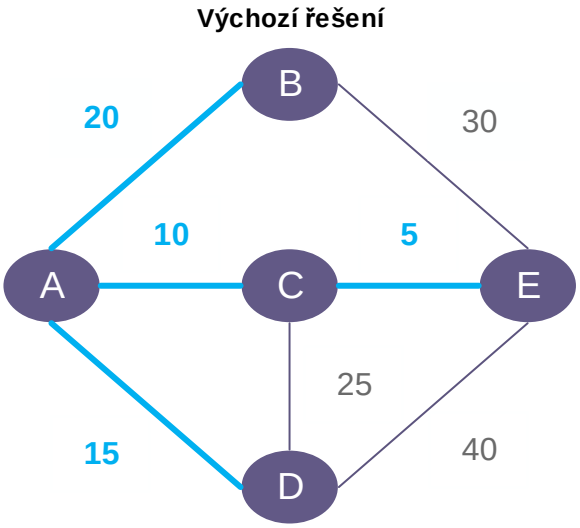
A) Iterace 5 b): Právě tímto zmíněným pohybem je dosaženo identické minimální kostry grafu jako v iteraci 7 a 11 větvě A) a podvětvě a). Jediný rozdíl tkví v aktuálním seznamu zakázaných hran, jež nyní tvoří hrany CD a CE. Obdobným způsobem se pokračuje ve výpočtu až do iterace 7 b), která odpovídá minimální kostře dosažené v iteraci 8 a), jediný rozdíl je v pořadí hran uvedených v aktuálnímu tabu seznamu, což hraje velkou roli při dalších iteracích. Opět se tedy nejedná o identická řešení, jelikož následující získané kostry grafu by již nebyly ani zdálnivě podobné. V iterace 9 b) je dosaženo druhého optima v této větvi a výpočet končí. Kdyby výpočet dále pokračoval, došlo by v následující iteraci opět k dalšímu větvení řešení.

Nyní je třeba se vrátit na počáteční zkoumání nejvýše ohodnocené záměny hran v 1. iteraci. Mezi přípustnými variantami řešení přidání nebo odebrání hran na kostře grafu výchozího přípustného řešení pro vytvoření nové kostry grafu existují dvě nevýhodnější změny se stejnou hodnotou celkových nákladů. První varianta (A) byla v předešlých odstavcích již podrobně rozebrána. Druhá varianta (B) spočívá v přidání hrany CD a odstranění hrany AD. Uvedený pohyb eliminuje porušení zadaných podmínek na přípustných hranách a zároveň rovněž redukuje výši penalizace ze 200 na 100, zatímco zvyšuje zbývající složku nákladů také z 50 na 60 (přidáním 25-15, což uvádí rozdíl nákladů mezi přidanou a odebranou hranou).

B) Iterace 1: Pohyb s nejvyšším ohodnocením představuje nyní přidání hrany BE a odstranění hrany AB. Seznam zakázaných hran, jež aktuálně tvoří pouze hrana CD, nezabraňuje realizaci lépe ohodnoceného pohybu.

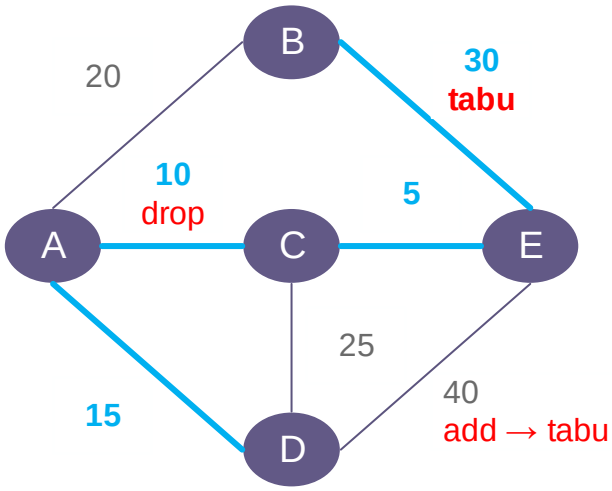
B) Iterace 2: Výše zmíněnou záměnou hran bylo dosaženo naprosto identické kostry grafu včetně identického seznamu zakázaných hran jako v iteraci 7 větvě A) a podvětve a). Jedná se tedy o dosud nejlepší nalezené řešení s celkovou hodnotou nákladů ve výši 70 jednotek. Následná série iterací dokazuje, že následující vývoj záměny hran je shodný s postupem od iterace 7 větve A) a podvětve a) dále.

Požadovaným mezním řešením minimální kostry se zadanými omezujícími podmínkami je kostra grafu znázorněna v následujících iteracích: *A) Iterace 7 a)*, *A) Iterace 11 a)*, *A) Iterace 5 b), A) Iterace 9 b), B) Iterace 2* a *B) Iterace 6*. Ve zmíněném příkladě představují zmíněné kostry grafu optimální řešení vzhledem k zadaným omezujícím podmínkám. Identického optimálního řešení včetně seznamu zakázaných hran bylo dosaženo v následujících iteracích: *A) Iterace 7 a) = B) Iterace 2, dále A) Iterace 9 b)* a *A) Iterace 11 a) = B) Iterace 6*.



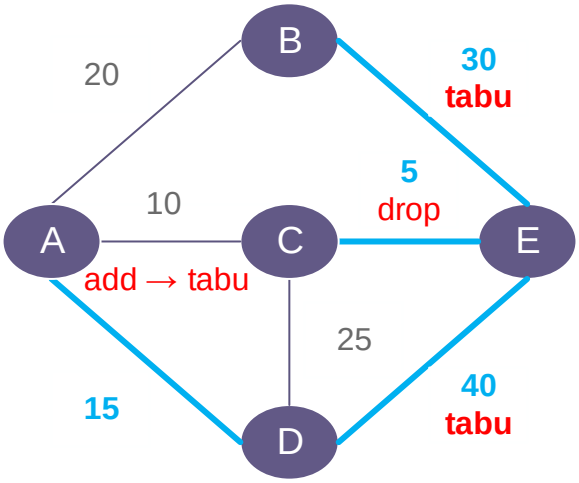
+	-	f
BE	CE	75 + 200 = **275**
BE	AC	70 + 200 = **270**
BE	AB	60 + 100 = **160**
CD	AD	60 + 100 = **160**
CD	AC	65 + 300 = **365**
DE	CE	85 + 100 = **185**
DE	AC	80 + 100 = **180**
DE	AD	75 + 100 = **175**

A) Iterace 1



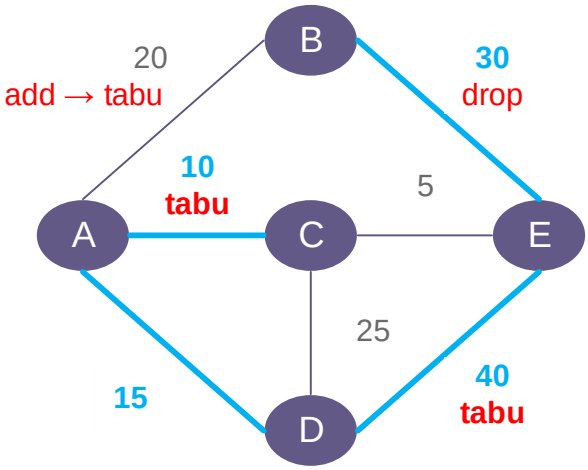
+	-	f
AB	BE	50 + 200 = **250**
AB	AC	70 + 200 = **270**
AB	CE	75 + 200 = **275**
CD	AD	70 + 100 = **170**
CD	AC	75 + 200 = **275**
DE	CE	95 + 0 = **95**
DE	AD	85 + 100 = **185**
DE	AC	90 + 0 = **90**

A) Iterace 2



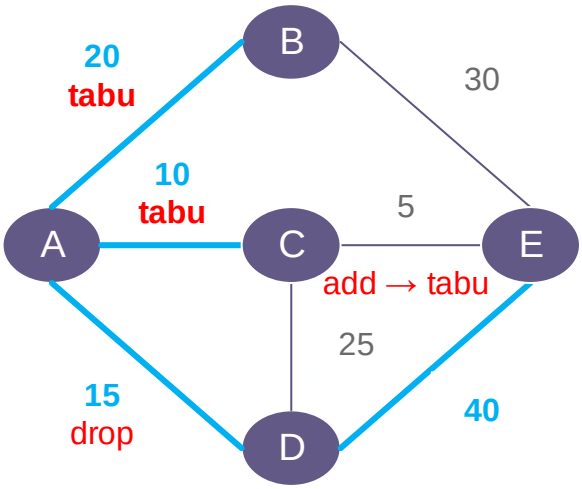
+	-	f
AB	BE	80 + 100 = **180**
AB	DE	70 + 200 = **270**
AB	AD	95 + 100 = **195**
CD	CE	110 + 100 = **210**
CD	DE	75 + 200 = **275**
AC	AD	85 + 100 = **185**
AC	CE	95 + 0 = **95**
AC	DE	60 + 100 = **160**

A) Iterace 3 a)



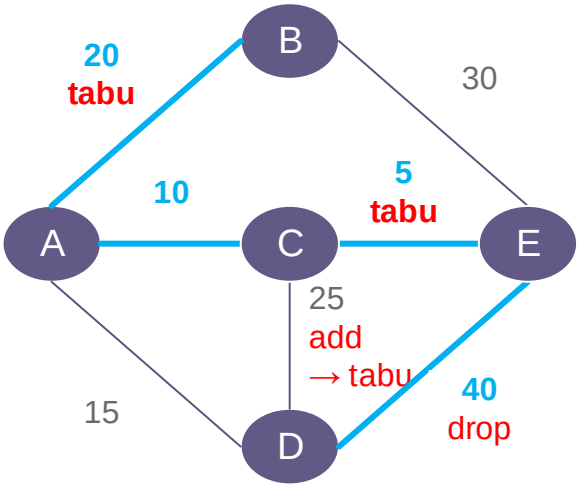
+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

A) Iterace 4 a)



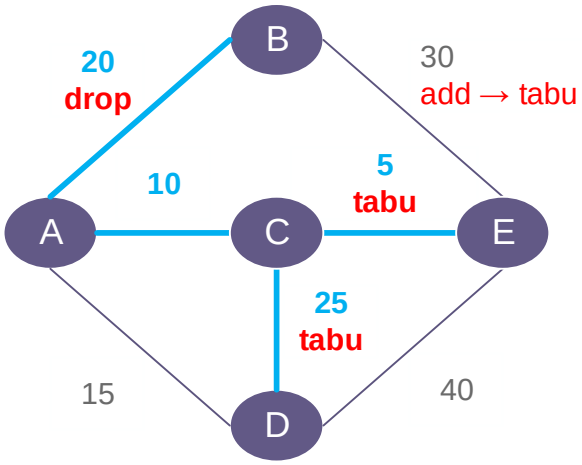
+	-	f
BE	AB	95 + 0 = **95**
BE	DE	75 + 200 = **275**
BE	AD	100 + 100 = **200**
CD	AC	100 + 200 = **300**
CD	AD	95 + 200 = **295**
CE	AC	80 + 100 = **180**
CE	AD	75 + 100 = **175**
CE	DE	60 + 200 = **260**

A) Iterace 5 a)



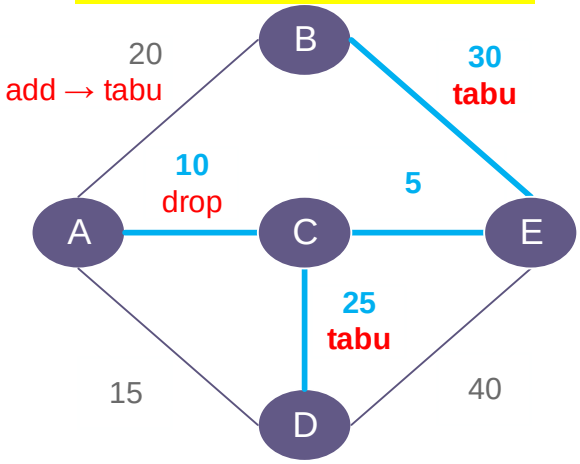
+	-	f
AD	AC	80 + 100 = **180**
AD	CE	85 + 100 = **185**
AD	DE	50 + 200 = **250**
CD	CE	95 + 200 = **295**
CD	DE	60 + 100 = **160**
BE	AB	85 + 100 = **185**
BE	AC	95 + 100 = **195**
BE	CE	100 + 100 = **200**

A) Iterace 6 a)



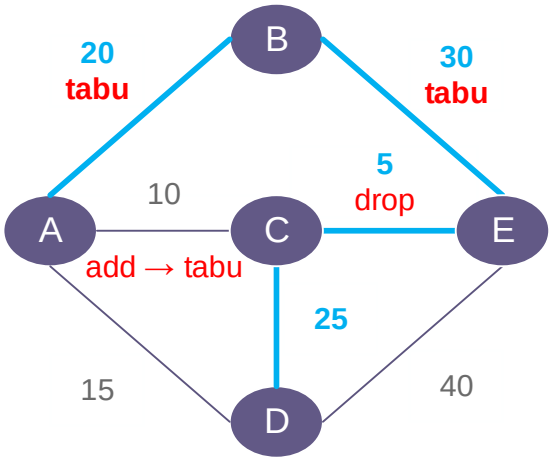
+	-	f
AD	CD	70 + 200 = **270**
AD	AC	65 + 100 = **165**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CD	95 + 100 = **195**
DE	CE	95 + 200 = **295**

A) Iterace 7 a)



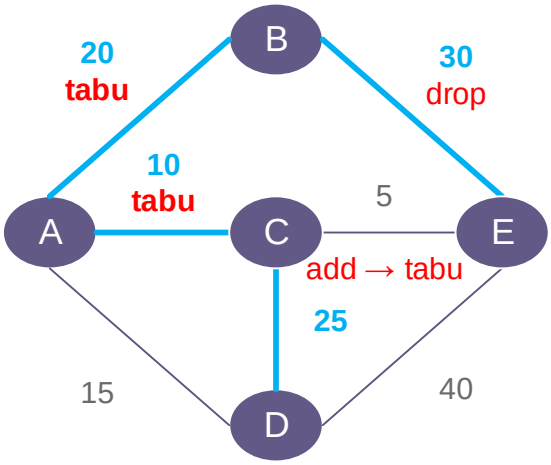
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CD	85 + 100 = **185**
DE	CE	105 + 100 = **205**

A) Iterace 8 a)



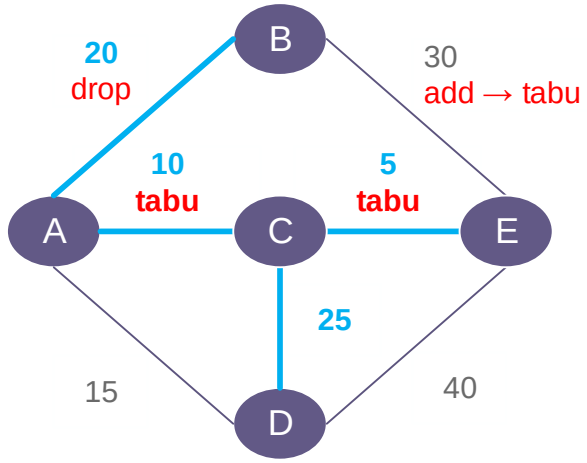
+	-	f
AC	AB	70 + 0 = **70**
AC	CE	85 + 100 = **185**
AD	CD	70 + 200 = **270**
AD	AB	75 + 200 = **270**
AD	CE	90 + 300 = **390**
AD	BE	65 + 300 = **365**
DE	CD	95 + 100 = **195**
DE	CE	105 + 200 = **305**

A) Iterace 9 a)



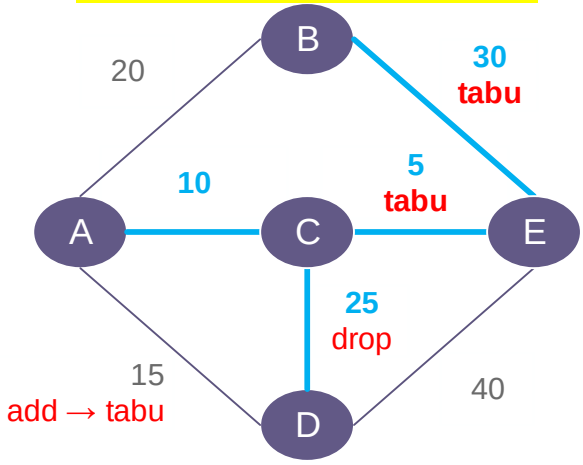
+	-	f
AD	AC	90 + 300 = **390**
AD	CD	75 + 200 = **275**
CE	AC	80 + 100 = **180**
CE	BE	60 + 100 = **160**
CE	AB	70 + 0 = **70**
DE	CD	100 + 100 = **200**
DE	BE	95 + 200 = **295**
DE	AC	115 + 200 = **315**
DE	AB	105 + 100 = **205**

A) Iterace 10 a)



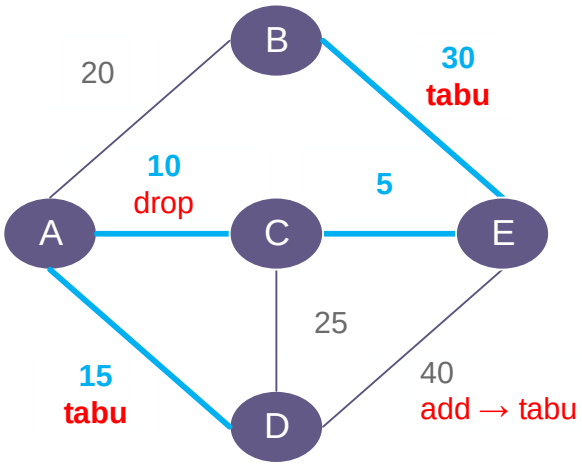
+	-	f
AD	CD	70 + 200 = **270**
AD	AC	65 + 100 = **165**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CD	95 + 100 = **195**
DE	CE	95 + 200 = **295**

A) Iterace 11 a)



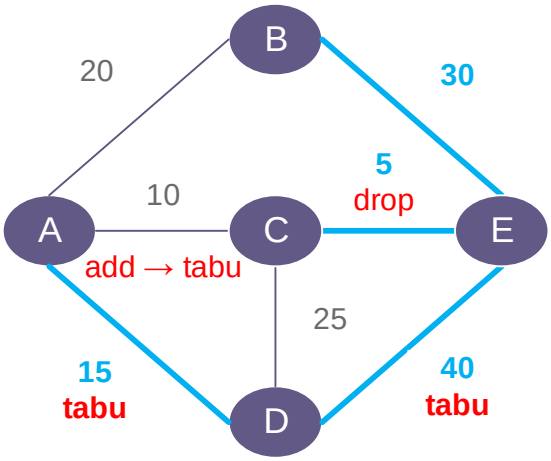
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CD	85 + 100 = **185**
DE	CE	105 + 100 = **205**

A) Iterace 12 a)



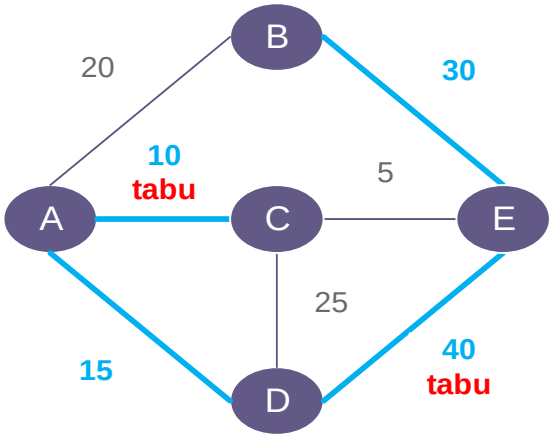
+	-	f
AB	BE	50 + 200 = **250**
AB	AC	70 + 200 = **270**
AB	CE	75 + 200 = **275**
CD	AD	70 + 100 = **170**
CD	AC	75 + 200 = **275**
DE	CE	95 + 0 = **95**
DE	AD	85 + 100 = **185**
DE	AC	90 + 0 = **90**

A) Iterace 13 a)



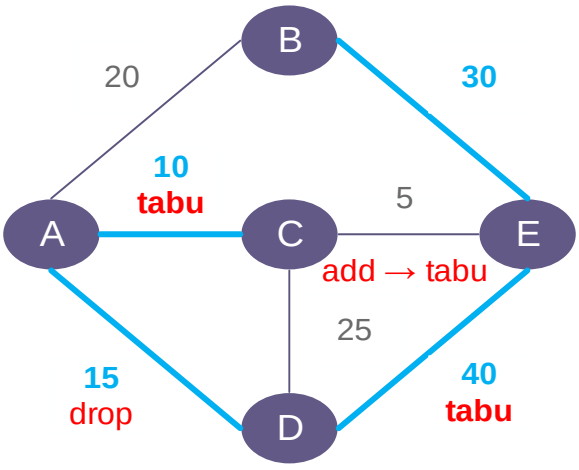
+	-	f
AB	BE	80 + 100 = **180**
AB	DE	70 + 200 = **270**
AB	AD	95 + 100 = **195**
CD	CE	110 + 100 = **210**
CD	DE	75 + 200 = **275**
AC	AD	85 + 100 = **185**
AC	CE	95 + 0 = **95**
AC	DE	60 + 100 = **160**

A) Iterace 14 a)



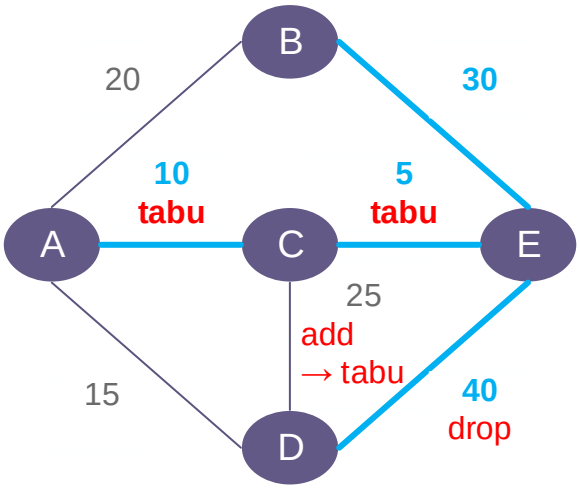
+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

A) Iterace 3 b)



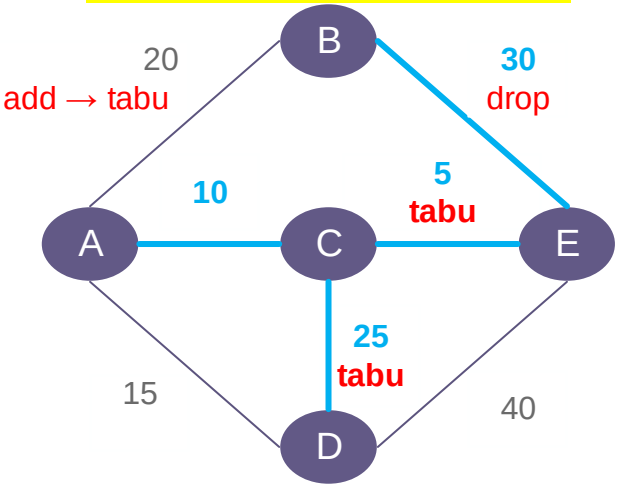
+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

A) Iterace 4 b)



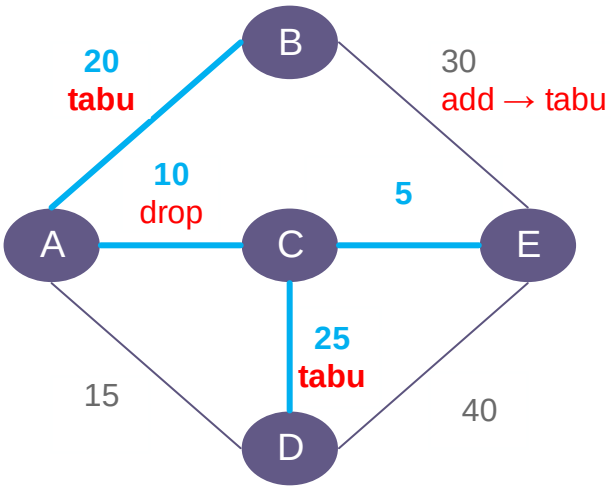
+	-	f
AB	AC	95 + 100 = **195**
AB	BE	75 + 100 = **175**
AB	CE	100 + 100 = **200**
CD	DE	70 + 0 = **70**
CD	CE	105 + 100 = **205**
AD	AC	90 + 0 = **90**
AD	DE	60 + 100 = **160**
AD	CE	95 + 0 = **95**

A) Iterace 5 b)



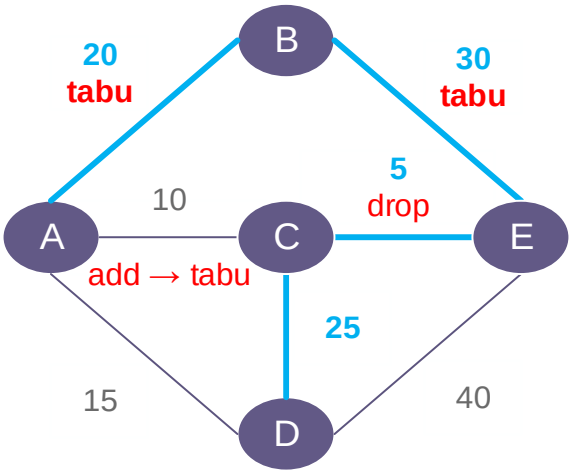
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CD	85 + 100 = **185**
DE	CE	105 + 100 = **205**

A) Iterace 6 b)



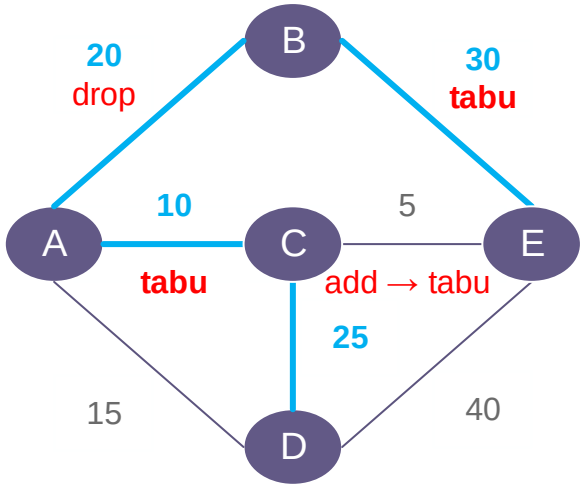
+	-	f
AD	CD	50 + 200 = **250**
AD	AC	65 + 300 = **365**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CD	95 + 100 = **195**
DE	CE	95 + 200 = **295**

A) Iterace 7 b)



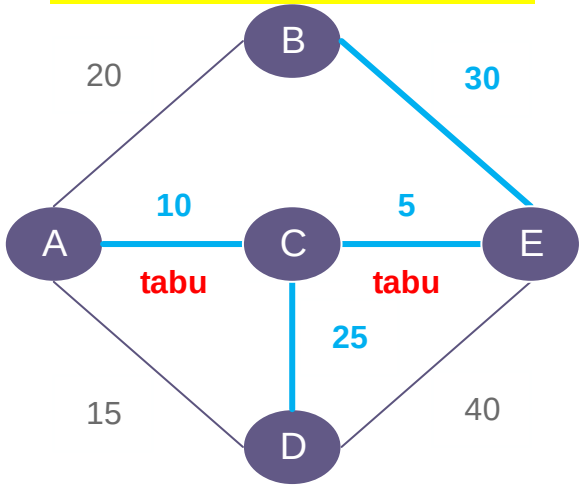
+	-	f
AC	AB	70 + 0 = **70**
AC	CE	85 + 100 = **185**
AD	CD	70 + 200 = **270**
AD	AB	75 + 200 = **270**
AD	CE	90 + 300 = **390**
AD	BE	65 + 300 = **365**
DE	CD	95 + 100 = **195**
DE	CE	105 + 200 = **305**

A) Iterace 8 b)



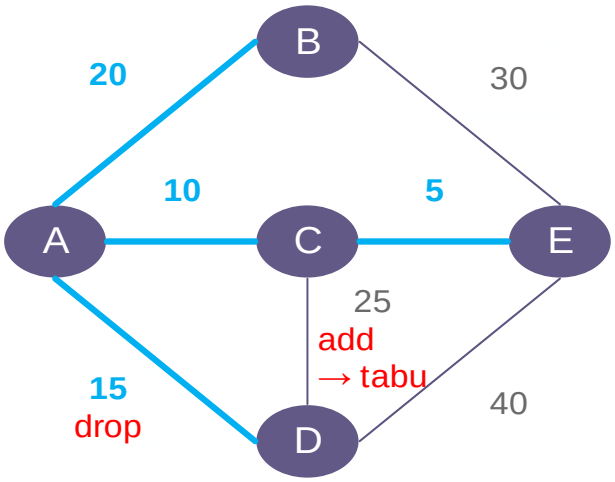
+	-	f
AD	AC	90 + 300 = **390**
AD	CD	75 + 200 = **275**
CE	AC	80 + 100 = **180**
CE	BE	60 + 100 = **160**
CE	AB	70 + 0 = **70**
DE	CD	100 + 100 = **200**
DE	BE	95 + 200 = **295**
DE	AC	115 + 200 = **315**
DE	AB	105 + 100 = **205**

A) Iterace 9 b)



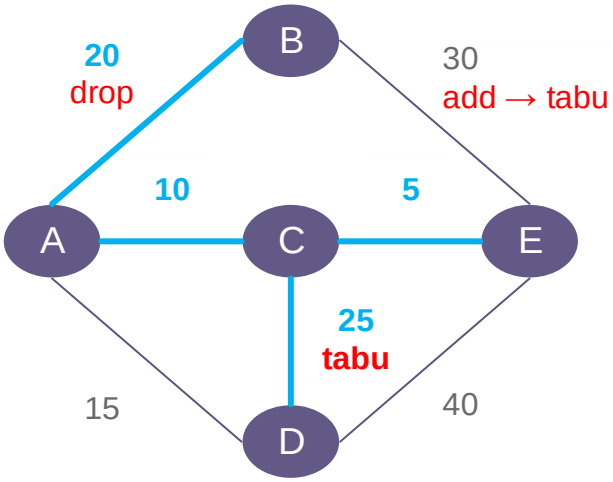
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CE	105 + 100 = **205**
DE	CD	85 + 100 = **185**

Výchozí řešení



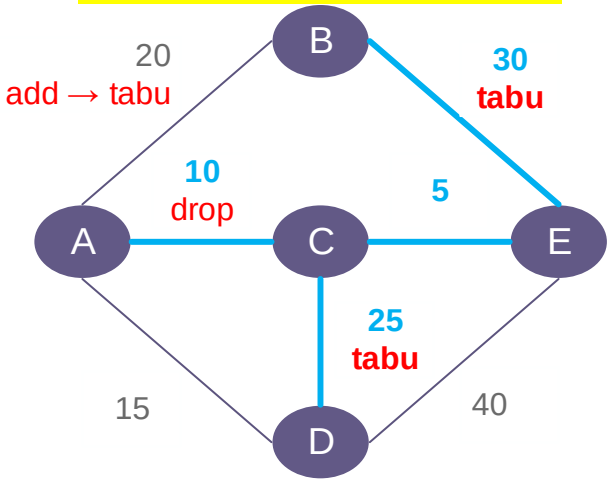
+	-	f
BE	CE	75 + 200 = **275**
BE	AC	70 + 200 = **270**
BE	AB	60 + 100 = **160**
CD	AD	60 + 100 = **160**
CD	AC	65 + 300 = **365**
DE	CE	85 + 100 = **185**
DE	AC	80 + 100 = **180**
DE	AD	75 + 100 = **175**

B) Iterace 1



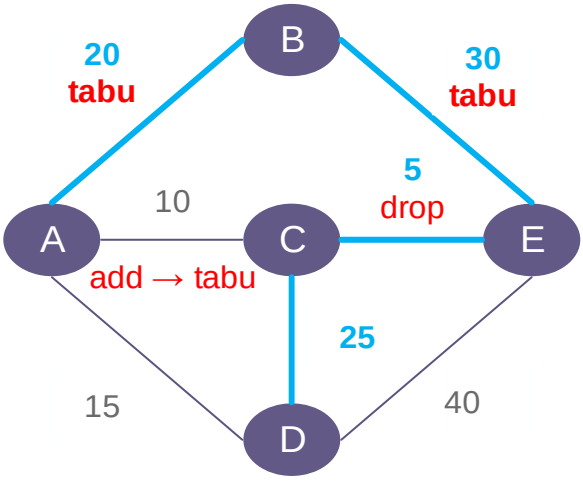
+	-	f
AD	AC	65 + 300 = **365**
AD	CD	50 + 200 = **250**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CE	95 + 200 = **295**
DE	CD	95 + 100 = **195**

B) Iterace 2



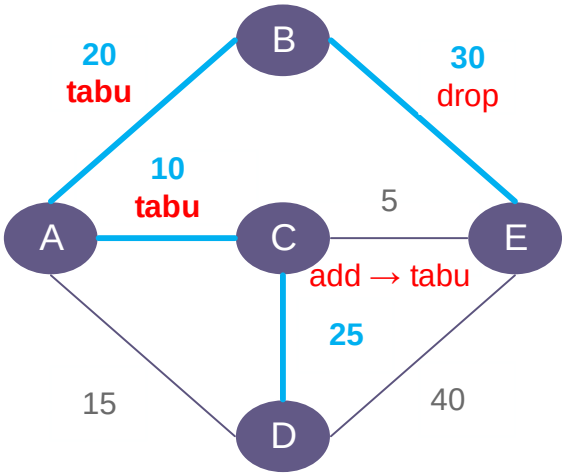
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CE	105 + 100 = **205**
DE	CD	85 + 100 = **185**

B) Iterace 3



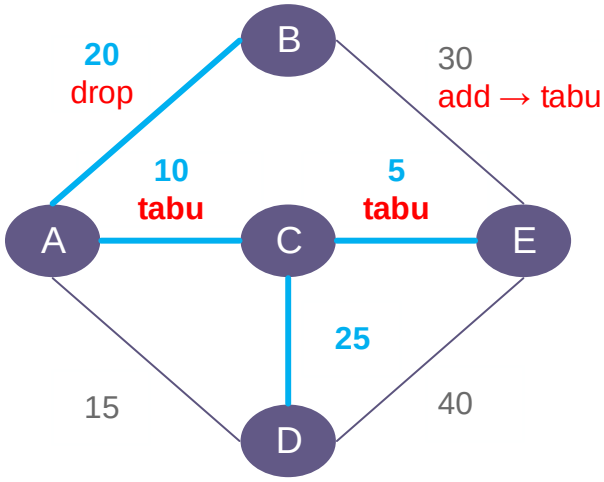
+	-	f
AC	AB	70 + 0 = **70**
AC	CE	85 + 100 = **185**
AC	BE	60 + 100 = **160**
AD	CD	70 + 200 = **270**
AD	CE	90 + 300 = **390**
AD	AB	85 + 200 = **285**
DE	CE	105 + 200 = **305**
DE	CD	95 + 100 = **195**

B) Iterace 4

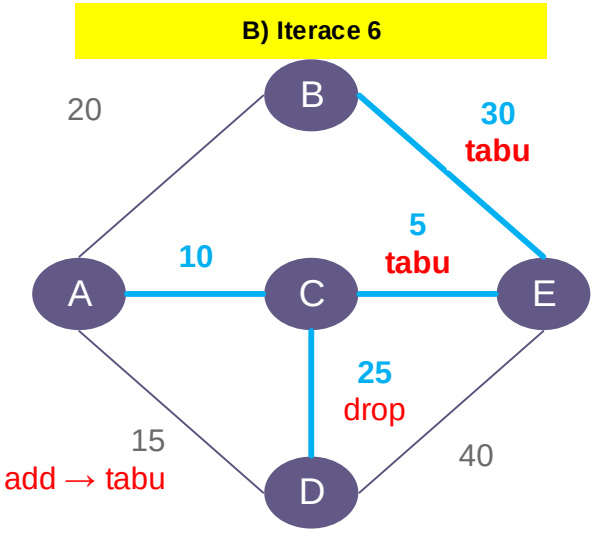


+	-	f
AD	CD	75 + 200 = **275**
AD	AC	90 + 200 = **290**
CE	BE	60 + 100 = **160**
CE	AC	80 + 100 = **180**
CE	AB	70 + 0 = **70**
DE	BE	95 + 200 = **295**
DE	CD	100 + 100 = **200**
DE	AB	105 + 100 = **205**

B) Iterace 5



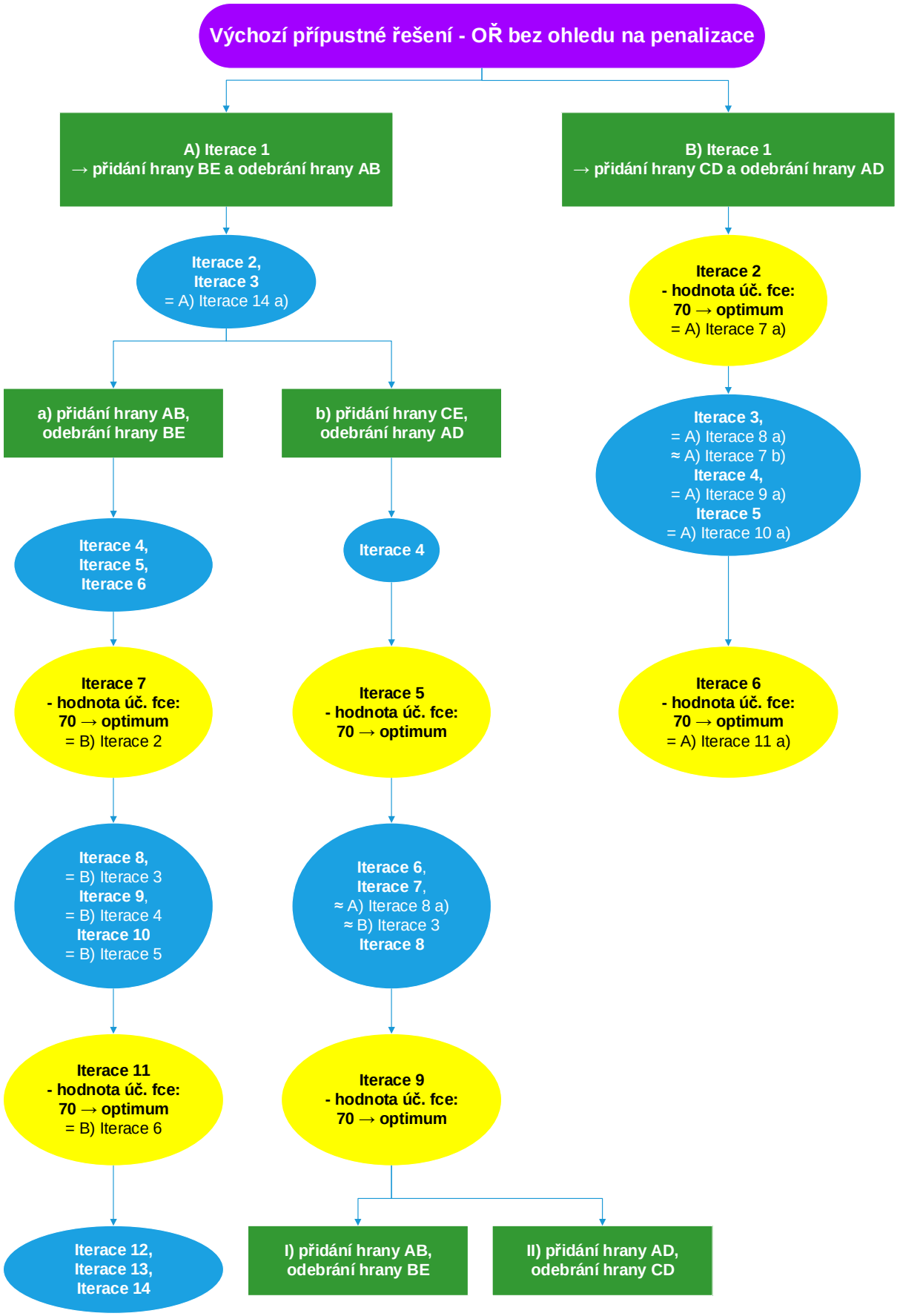
+	-	f
AD	AC	65 + 200 = **265**
AD	CD	50 + 200 = **250**
BE	AB	70 + 0 = **70**
BE	CE	85 + 100 = **185**
BE	AC	80 + 100 = **180**
DE	CE	95 + 200 = **295**
DE	CD	75 + 100 = **175**



+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CD	85 + 100 = **185**
DE	CE	105 + 100 = **205**

Pro přehlednost zobrazuje Obrázek 6-1 rozhodovací strom řešení výše podrobně rozebrané úlohy minimální kostry s omezeními řešené metaheuristikou Tabu Search. Jak je z daného obrázku patrné, k optimálnímu řešení nejrychleji konverguje větev B), kde hned ve druhé iteraci je dosaženo nejlepšího možného řešení vzhledem k zadaným omezením. V šesté iteraci větve B) je dosaženo stejného řešení jako ve druhé iteraci větve B), pouze se liší seznam zakázaných hran. Stejně tomu tak je i u dalších dosažených optimálních řešení až na řešení v iteraci 7 větve A) a podvětve a), kde se shoduje i seznam zakázaných hran. Dalším identickým optimálním řešením včetně seznamu zakázaných hran je iterace 11 větve A) a podvětve a) s iterací 6 větve B). Po větvi B) nejrychleji konverguje k optimálnímu řešení podvětev b) větve A, kdy je dosaženo optimálního řešení v páté iteraci. Nejpomaleji je tedy optimální řešení získáno postupem znázorněným v podvětvi a) větve A, kdy je optimálního řešení dosaženo poprvé v sedmé iteraci a poté v jedenácté iteraci.

Po optimálním řešení v iteraci 11 podvětve a) a větvě A) následují iterace shodné s iteracemi na počátku větve A) včetně seznamu zakázaných hran. Je tedy zřejmé, že dalšího optimálního řešení, natož ještě lepšího řešení, nemůže být dosaženo. Po optimálním řešení dosaženého v iteraci 5 podvětve b) větve A) následuje počínaje sedmou iterací identické iterace, kterých bylo dosaženo v podvětvi a) větve A) počínaje iterací 8. Obdobně je tomu v případě větve B), kde po dosažení optimálního řešení v iteraci 6 následují identické iterace včetně seznamu zakázaných hran jako ve větvi A počínaje iterací 1.



Obrázek 6-1: rozhodovací strom metaheuristiky Tabu Search s krátkodobou pamětí

6.2 Použití středně-dobé paměti

V předcházejícím případě byla každá nově zařazená hrana do kostry grafu zahrnuta do tabu listu po dobu následujících dvou iterací. Jak zobrazuje Obrázek 6-1, různé větve řešení konvergovaly k nejlepšímu nalezenému řešení zcela odlišně. Nyní bude prozkoumána rychlost konvergence k nejlepšímu nalezenému řešení, za předpokladu, že každá nově přidaná hrana bude zařazena do tabu listu po dobu následujících tří iterací. Ukončovací pravidlo je nastaveno na ukončení výpočtu po 10. iteraci ve všech nalezených větvích.

Výchozí řešení kostry grafu pro použití metaheuristiky Tabu Search je shodné jako v předchozím případě. Znovu dojde k rozdělení řešení na dvě větve, A a B jako v předchozí verzi. Nicméně větev A se již dále nedělí na podvětev a) a b). Další postup je znázorněn opět na obrázkovém souboru iterací.

Iterace 1: Mezi současnými možnostmi přidání nebo odebrání hran na kostře grafu výchozího přípustného řešení pro vytvoření nové kostry grafu existují opět dvě nevýhodnější změny se stejnou hodnotou celkových nákladů. První variantou (A) je přidat hranu BE a odstranit hranu AB jako v předchozím případě. Druhou variantou (B) je přidat hranu CD a odstranit hranu AD jako v minulém případě.

A) Iterace 1: Došlo k přidání hrany BE a odstranění hrany AB. Tento redukuje výši penalizace ze 200 na 100, zatímco zvyšuje zbývající složku nákladů z 50 na 60. Zmíněný pohyb získává tabu status, čímž je propůjčen tabu stav na pohyby, které odstraní tuto hranu. Jednotlivé zakázané hrany jsou uváděny v tom pořadí, v jakém vstoupily do seznamu zakázaných hran, jelikož zachování správného pořadí je pro řešení zadané úlohy velmi důležité. U každé hrany je tedy poznámka v závorce, kolikátou iteraci je již daná hrana uvedena v tabu listu.

A) Iterace 2: Došlo k přidání hrany DE a odebrání hrany AC. Jelikož tímto pohybem nejsou porušeny zadané podmínky, žádné penalizace se neprojeví do celkové hodnoty nákladů dané kostry grafu. Celkové ohodnocení kostry grafu je 90 jednotek. Zatím jsou pouze dvě hrany v seznamu zakázaných hran, a to hrana BE a DE.

A) Iterace 3: Po odstranění hrany CE a přidání hrany AC vznikla kostra grafu s celkovým ohodnocením 95. Opět celkové ohodnocení vybrané kostry grafu není zatíženo penalizací. Nyní jsou v tabu listu uvedeny již tři hrany, a to hrana AC, BE a DE. Následující vývoj se tedy bude lišit od předchozího případu s krátkodobou pamětí.

A) Iterace 4: Přestože by bylo momentálně nejvýhodnější vybrat jako další pohyb přidání hrany CE a odebrání hrany AC, nedochází k tomuto kroku ani s pomocí aspiračního kritéria. Důvodem je skutečnost, že takto získaná kostra grafu by nabyla celkového ohodnocení 90 jednotek, jenže této hodnoty již bylo v současné větvi A již dosaženo. Aspirační kritérium tedy do dalšího vývoje výpočtu zatím nezasahuje. K odebrání je vybrána hrana AD a do nové

kostry grafu je přidána hrana CE. Celkové ohodnocení zvolené kostry grafu dosahuje 185 jednotek.

A) Iterace 5: Nyní se opět nabízí zvolit momentálně nejvýhodnější řešení, a to přidáním hrany CD a odebráním hrany DE. Hrana DE je aktuálně zakázanou hranou po dobu tří iterací. V této situaci se uplatní aspirační kritérium, protože celkového ohodnocení kostry grafu nedosáhlo prozatím v této větvi hodnotu 70. Zmíněný pohyb je tedy uskutečněn a je odstraněna hrana z kostry grafu, která byla uvedena v aktuálním tabu listu. Je získáno optimální řešení v této větvi.

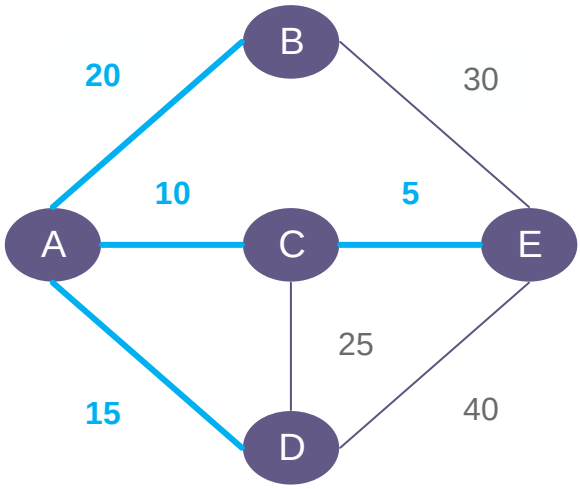
A) Iterace 6 až *A) Iterace 10:* Obdbonným způsobem je pokračováno ve výpočtu dále až do ukončení výpočtu ukončovacím pravidlem. Aspirační kritérium již nebylo aplikováno, a to z toho důvodu, že již bylo dosaženo optima s celkovým ohodnocením nákladů ve výši 90 jednotek.

B) Iterace 1: Z výchozího řešení se dále postupuje přidáním hrany CD a odebráním hrany AD. Zvolená kostra grafu dosahuje 160 jednotek nákladů, a to z důvodu porušení omezení hranou AB a CD.

B) Iterace 2: Po přidání hrany BE a odebrání hrany AB je dosaženo optimálního řešení již ve druhé iteraci. Toto řešení je identické s optimálním řešením dosažením v iteraci 7 a) větve A včetně pořadí hran uvedených v tabu listu.

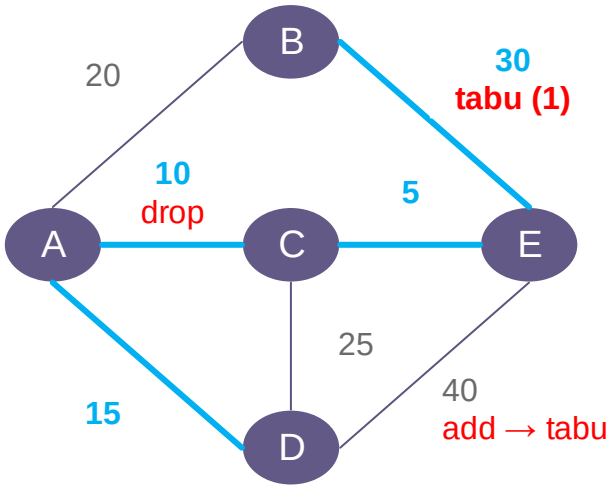
B) Iterace 3 až B) Iterace 6: Tyto iterace jsou naprosto shodné s již popsanými iterace ve větvi A, a to s iterací 8 a) až 11 a). V iteraci 6 větve B a v iteraci 11 a) větve A je opět dosaženo optima. Dále není větev B rozvíjena, protože zde byl odhalen cyklus řešení.

Výchozí řešení



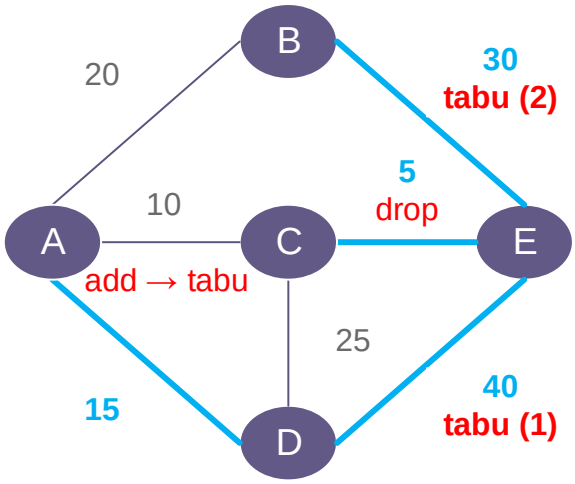
+	-	f
BE	CE	75 + 200 = **275**
BE	AC	70 + 200 = **270**
BE	AB	60 + 100 = **160**
CD	AD	60 + 100 = **160**
CD	AC	65 + 300 = **365**
DE	CE	85 + 100 = **185**
DE	AC	80 + 100 = **180**
DE	AD	75 + 100 = **175**

A) Iterace 1



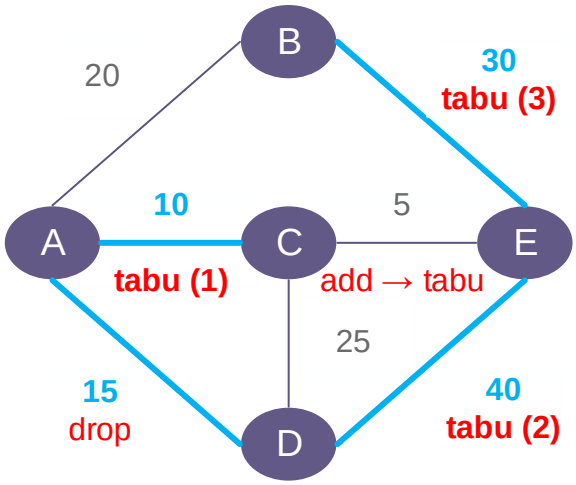
+	-	f
AB	BE	50 + 200 = **250**
AB	AC	70 + 200 = **270**
AB	CE	75 + 200 = **275**
CD	AD	70 + 100 = **170**
CD	AC	75 + 200 = **275**
DE	CE	95 + 0 = **95**
DE	AD	85 + 100 = **185**
DE	AC	90 + 0 = **90**

A) Iterace 2



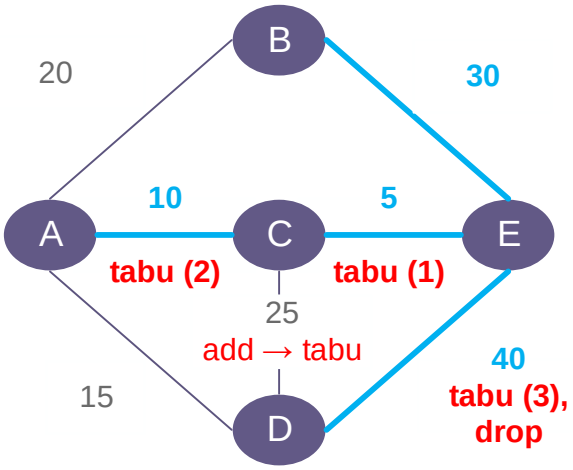
+	-	f
AB	BE	80 + 100 = **180**
AB	DE	70 + 200 = **270**
AB	AD	95 + 100 = **195**
CD	CE	110 + 100 = **210**
CD	DE	75 + 200 = **275**
AC	AD	85 + 100 = **185**
AC	CE	95 + 0 = **95**
AC	DE	60 + 100 = **160**

A) Iterace 3



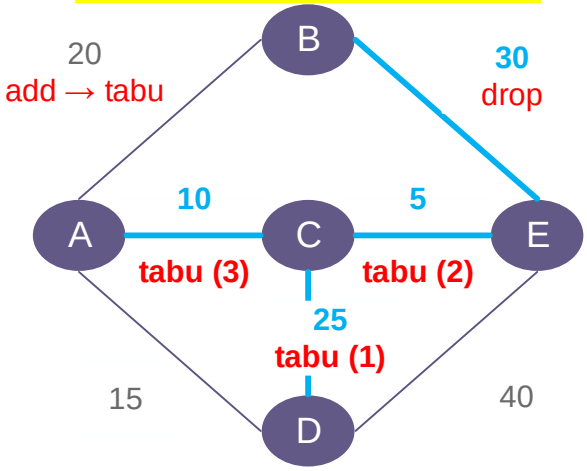
+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

A) Iterace 4



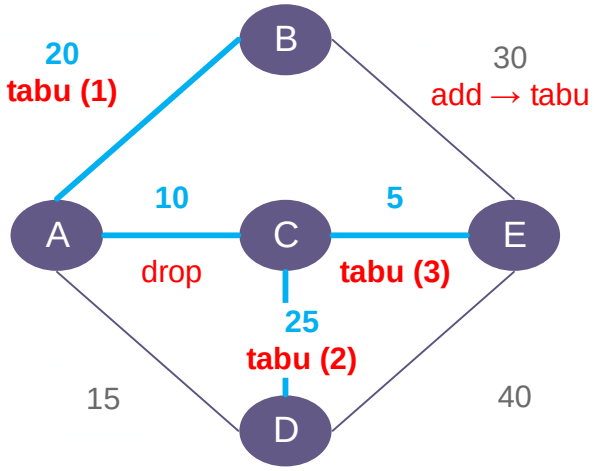
+	-	f
AB	AC	95 + 100 = **195**
AB	BE	75 + 100 = **175**
AB	CE	100 + 100 = **200**
CD	DE	70 + 0 = **70**
CD	CE	105 + 100 = **205**
AD	AC	90 + 0 = **90**
AD	DE	60 + 100 = **160**
AD	CE	95 + 0 = **95**

A) Iterace 5



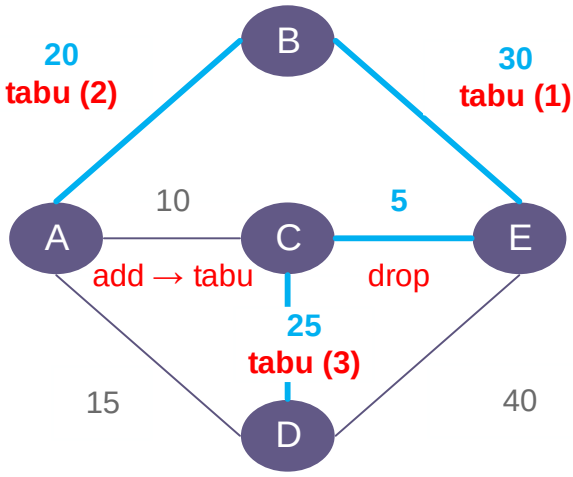
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 2000 = **275**
AD	CD	60 + 100 = **160**
DE	CD	85 + 100 = **185**
DE	CE	105 + 100 = **205**

A) Iterace 6



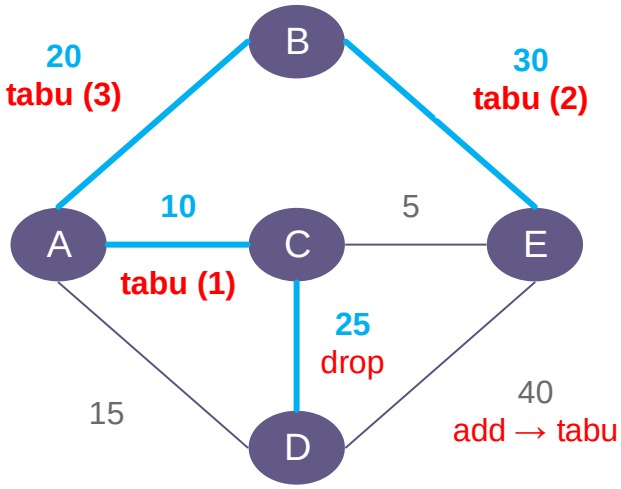
+	-	f
AD	CD	50 + 200 = **250**
AD	AC	65 + 300 = **365**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CD	95 + 100 = **195**
DE	CE	95 + 200 = **295**

A) Iterace 7



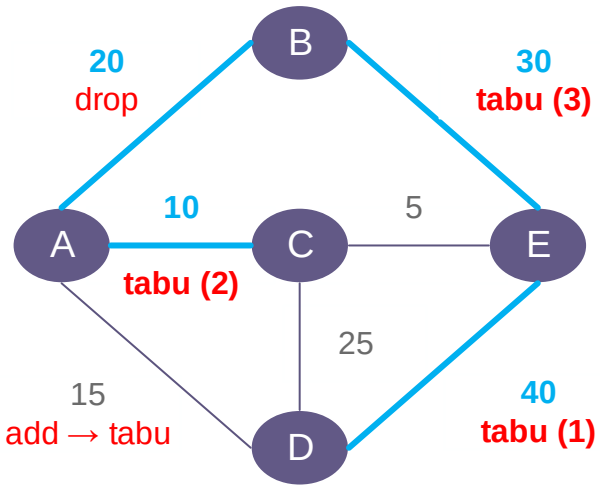
+	-	f
AC	AB	70 + 0 = **70**
AC	CE	85 + 100 = **185**
AD	CD	70 + 200 = **270**
AD	AB	75 + 200 = **275**
AD	CE	90 + 300 = **390**
AD	BE	65+ 300 = **365**
DE	CD	95 + 100 = **195**
DE	CE	105 + 200 = **305**

A) Iterace 8



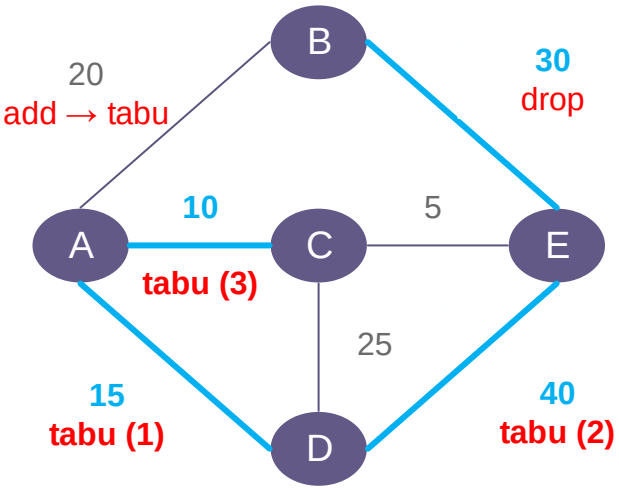
+	-	f
AD	CD	75 + 200 = **275**
AD	AC	90 + 200 = **290**
CE	BE	60 + 100 = **160**
CE	AC	80 + 100 = **180**
CE	AB	70 + 0 = **70**
DE	BE	95 + 200 = **295**
DE	CD	100 + 100 = **200**
DE	AB	105 + 100 = **205**

A) Iterace 9



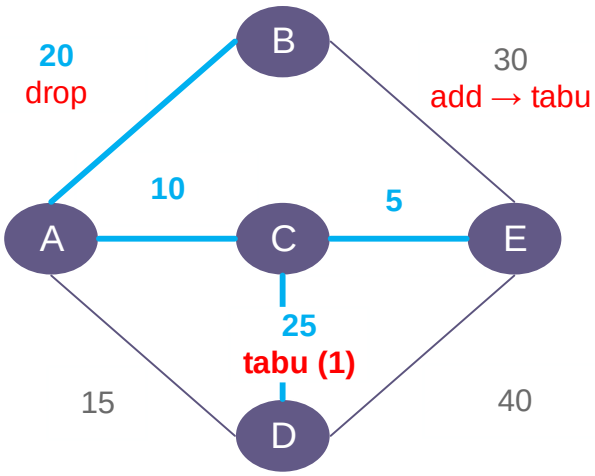
+	-	f
AD	AB	95 + 0 = **95**
AD	DE	75 + 200 = **275**
AD	BE	85 + 100 = **185**
CD	AC	115 + 200 = **315**
CD	DE	85 + 100 = **185**
CE	AC	95 + 100 = **195**
CE	BE	75 + 100 = **175**
CE	AB	85 + 100 = **185**

A) Iterace 10



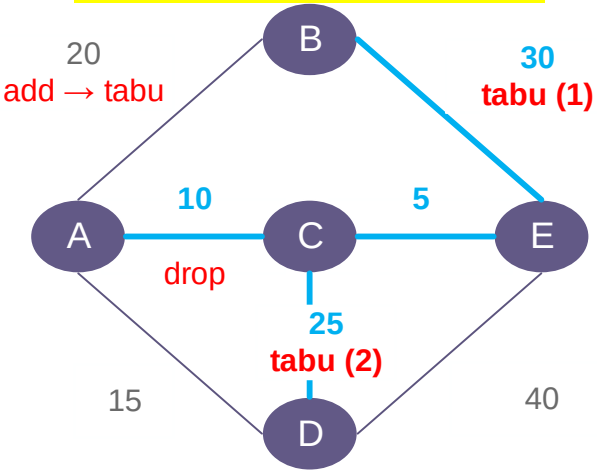
+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

B) Iterace 1



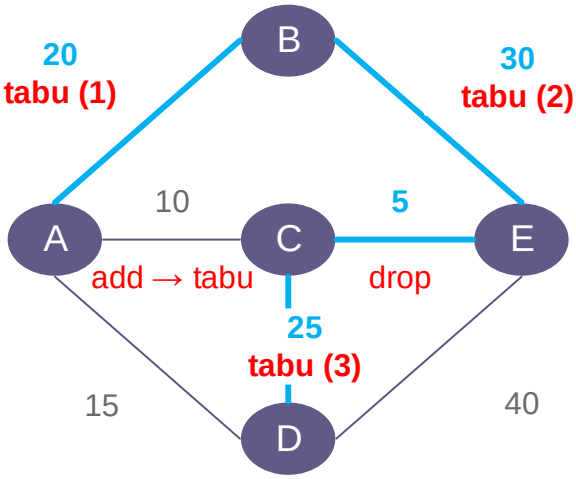
+	-	f
AD	AC	65 + 300 = **365**
AD	CD	50 + 200 = **250**
BE	AB	70 + 0 = **70**
BE	AC	80 + 100 = **180**
BE	CE	85 + 100 = **185**
DE	CE	95 + 200 = **295**
DE	CD	95 + 100 = **195**

B) Iterace 2



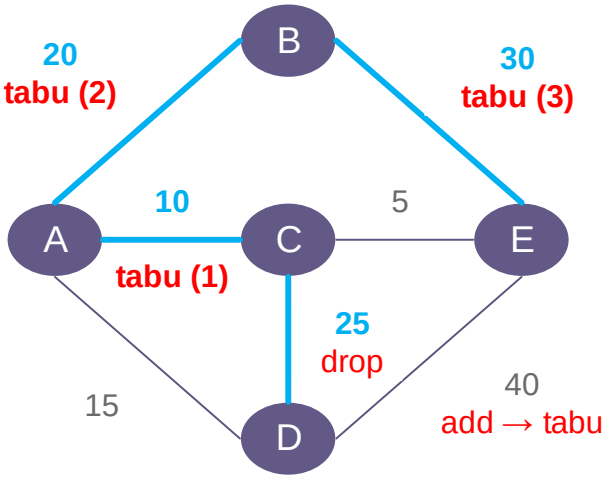
+	-	f
AB	BE	60 + 100 = **160**
AB	AC	80 + 100 = **180**
AB	CE	85 + 100 = **185**
AD	AC	75 + 200 = **275**
AD	CD	60 + 100 = **160**
DE	CE	105 + 100 = **205**
DE	CD	85 + 100 = **185**

B) Iterace 3



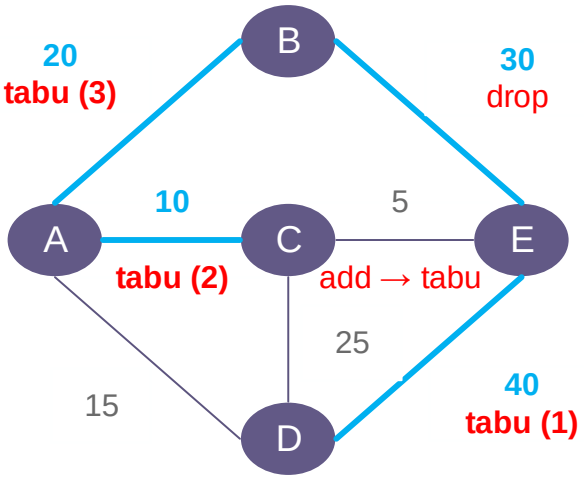
+	-	f
AC	AB	70 + 0 = **70**
AC	CE	85 + 100 = **185**
AC	BE	60 + 100 = **160**
AD	CD	70 + 200 = **270**
AD	CE	90 + 300 = **390**
AD	AB	85 + 200 = **285**
DE	CE	105 + 200 = **305**
DE	CD	95 + 100 = **195**

B) Iterace 4



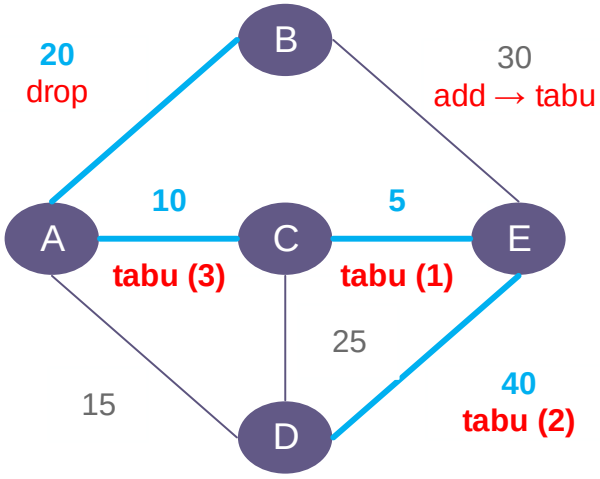
+	-	f
AD	CD	75 + 200 = **275**
AD	AC	90 + 200 = **290**
CE	BE	60 + 100 = **160**
CE	AC	80 + 100 = **180**
CE	AB	70 + 0 = **70**
DE	BE	95 + 200 = **295**
DE	CD	100 + 100 = **200**
DE	AB	105 + 100 = **205**

B) Iterace 5



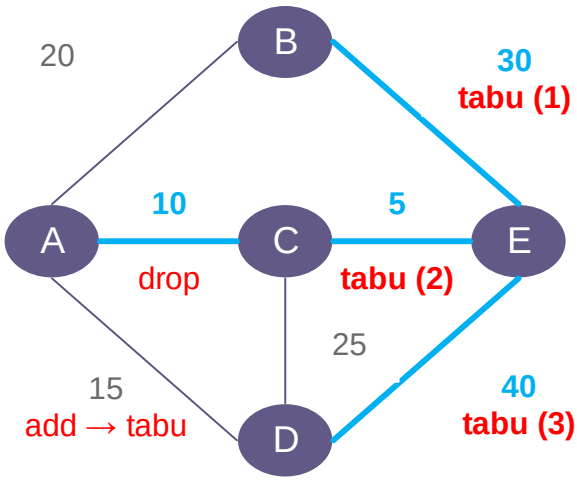
+	-	f
AD	AB	95 + 0 = **95**
AD	DE	75 + 200 = **275**
AD	BE	85 + 100 = **185**
CD	AC	115 + 200 = **315**
CD	DE	85 + 100 = **185**
CE	AC	95 + 100 = **195**
CE	BE	75 + 100 = **175**
CE	AB	85 + 100 = **185**

B) Iterace 6



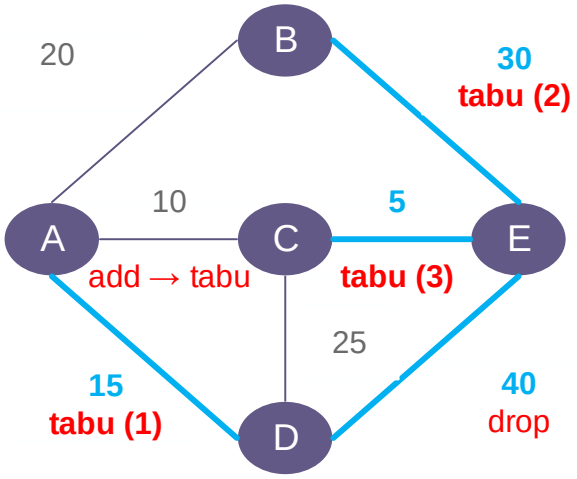
+	-	f
AD	AC	80 + 100 = **180**
AD	CE	85 + 100 = **185**
AD	DE	50 + 200 = **250**
CD	CE	95 + 200 = **295**
CD	DE	60 + 100 = **160**
BE	AB	85 + 100 = **185**
BE	AC	95 + 100 = **195**
BE	CE	100 + 100 = **200**

B) Iterace 7



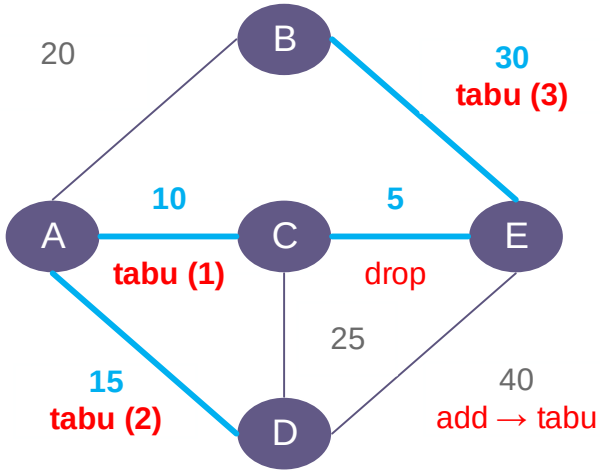
+	-	f
AB	AC	95 + 100 = **195**
AB	BE	75 + 100 = **175**
AB	CE	100 + 100 = **200**
CD	DE	70 + 0 = **70**
CD	CE	105 + 100 = **205**
AD	AC	90 + 0 = **90**
AD	DE	60 + 100 = **160**
AD	CE	95 + 0 = **95**

B) Iterace 8



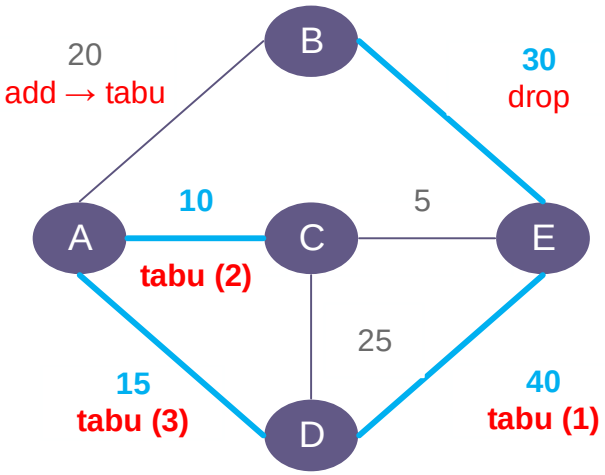
+	-	f
AB	BE	80 + 100 = **180**
AB	DE	70 + 200 = **270**
AB	AD	95 + 100 = **195**
CD	CE	110 + 100 = **210**
CD	DE	75 + 200 = **275**
AC	AD	85 + 100 = **185**
AC	CE	95 + 0 = **95**
AC	DE	60 + 100 = **160**

B) Iterace 9



+	-	f
AB	BE	50 + 200 = **250**
AB	AC	70 + 200 = **270**
AB	CE	75 + 200 = **275**
CD	AD	70 + 100 = **170**
CD	AC	75 + 200 = **275**
DE	CE	95 + 0 = **95**
DE	AD	85 + 100 = **185**
DE	AC	90 + 0 = **90**

B) Iterace 10

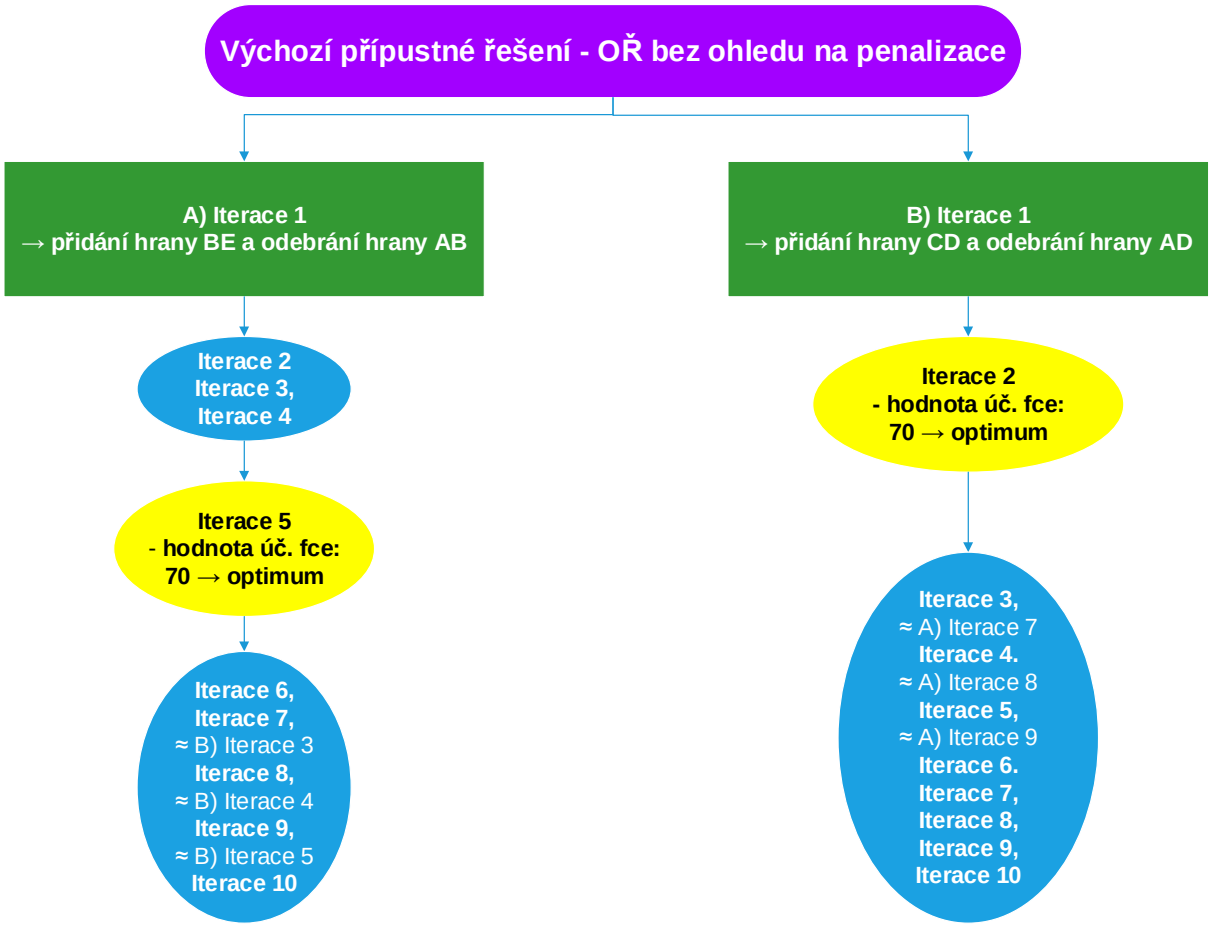


+	-	f
AB	BE	85 + 100 = **185**
AB	AC	105 + 100 = **205**
AB	AD	100 + 100 = **200**
CD	AC	110 + 100 = **210**
CD	AD	105 + 100 = **205**
CE	AC	90 + 0 = **90**
CE	AD	85 + 100 = **185**
CE	DE	60 + 100 = **160**

Jak je patrné ze série iterací, kromě iterace 4 ve větvi A nebylo využito aspiračního kritéria a vzhledem k seznamu aktuálních zakázaných hran pro jednotlivé iterace se vždy nabízela jediná hrana pro odstranění ze zvolené kostry grafu. Jak zobrazuje Obrázek 6-2, optimálního řešení vzhledem k zadaným omezením a penalizacím bylo ve větvi A dosaženo již v páté iteraci a ve větvi B opět ve druhé iteraci jako v případě řešení metaheuristikou Tabu Search s krátkodobou pamětí. Za povšimnutí stojí skutečnost, že konvergence k optimu se oproti předchozímu případu s krátkodobou pamětí zrychlila. Po dosažení optima však nebylo po dobu pěti (ve větvi A) až osmi iterací (ve větvi B) dosaženo opět optima. V předcházejícím případě se výpočet navracel do optima rychleji, jednotlivé výsledky se lišily seznamem aktuálních zakázaných hran.

Během výpočtu se střednědobou pamětí nebyl odhalen žádný cyklus, dokud neskončil výpočet kvůli ukončovacímu pravidlu. Jednotlivé větve řešení se již dále nevětvily, jak tomu bylo v případě s krátkodobou pamětí. Z výše uvedeného vyplývá, že pro zajištění rychlejší

konvergence výpočtu k optimálnímu řešení, je výhodnější zvýšit velikost tabu listu ze dvou na tři hrany. Problematikou optimání velikosti tabu listu zjištěnou na základě experimentů se zabývá [29].



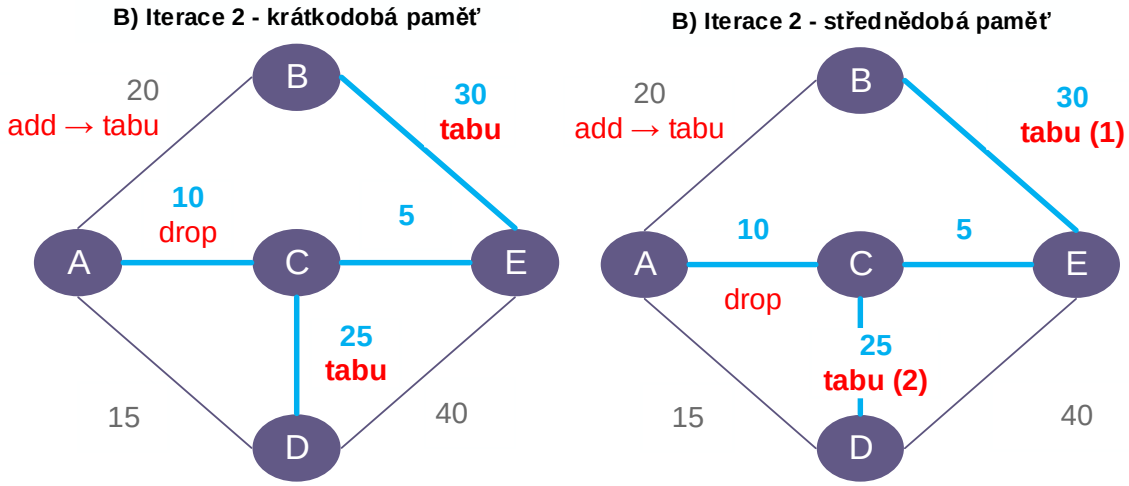
Obrázek 6-2: rozhodovací strom metaheuristiky Tabu Search se střednědobou pamětí

6.3 Srovnání optimálního řešení s řešením pomocí metaheuristiky

Řešení pomocí exaktní metody zaručuje u méně obtížných úloh dosažení optimálního řešení neboli globálního extrému. Metaheuristiky sice nezaručují nalezení optima, nicméně jsou schopné získat řešení, které je optimu velmi blízké, v reálném čase.

Zvolený příklad, na kterém byly demonstrovány schopnosti a především postup metaheuristiky Tabu Search, nebyl složitý. Příklad menšího rozměru byl vybrán za účelem možného získání exaktního řešení za účelem porovnání s výsledky dosažené zvolenou metaheuristikou. Optimální řešení získané exaktní metodou pro případ minimální kostry s penalizacemi znázorňuje Obrázek 4-1 (vpravo). V daném příkladě bylo získáno optimální řešení i metaheuristikou Tabu Search, nejrychleji právě vždy ve větvi B hned ve druhé iteraci, a to v případě s krátkodobou i střednědobou pamětí, viz Obrázek 6-3. Jednotlivá

nalezená optima jsou shodná zvolenými hranami v kostře grafu, nicméně jednotlivá řešení se navzájem liší hranami a jejich pořadím v aktuálním tabu listu.



Obrázek 6-3: OŘ ve větvi B s krátkodobou pamětí (vlevo), OŘ ve větvi B s dlouhodobou pamětí (vpravo)

7 Úvaha o vlivu a rozměru aspiračního kritéria

Aspirační kritérium umožňuje za splnění určitých podmínek nerespektovat zákazy z tabu seznamu při výběru přechodu z aktuálního řešení. Nejběžnější a nejjednodušší aspirační kritérium spočívá v povolení určitého pohybu, přestože je uveden v tabu listu, a to jen v případě, jestliže daný pohyb zajistí lepší hodnotu účelové funkce, než byla dosud získána. Jsou používána jiná a daleko komplikovanější aspirační kritéria, ale pouze zřídka, jak je například praktikováno v [32], [33]. Aspirace tahu vede buď k dosud nejlepšímu řešení nebo vede k odlišnému řešení, než kam dosud řešení směrovalo. Klíčovým aspektem je následující podmínka - jestliže se zacyklení nemůže objevit, nepřihlíží se k tabu listu. Aspirační kritérium umožňuje přijmutí i horšího řešení, aby bylo možné přejít přes lokální optima a nezabřednout v nich při hledání co nejlepšího řešení. Důležitos aspiračního kritéreia spočívá v zajištění určité flexibility metaheuristiky Tabu Search směřováním k atraktivním pohybům vzhledem k hodnotě účelové funkce.

Hlavním úkolem tabu listu je zabránění zacyklení a zpětnému chodu, tedy zkoumání potenciálního řešení, jestliže již bylo v nedávné době zkoumáno. V případě, že jsou všechny přechody z aktuálního řešení zakázané, používá se tzv. výchozí aspirační kritérium, které zaručuje přechod k řešení s nejlepší hodnotou účelové funkce.

V řešeném příkladě se ukázala volba délky paměti jako klíčová. V případě krátkodobé paměti, kdy v tabu seznamu byly uvedeny pouze dvě zakázané hrany, resp. všechny pohyby s těmito hranami, se strom řešení, viz Obrázek 6-1 rozrostl do dvou hlavních větví A a B a poté ještě do dvou podvětví a) a b) ve větvi A. Zatímco v případě střednědobé paměti, kdy v tabu seznamu byly uváděny tři zakázané hrany, byl strom řešení jednodušší a obsahoval pouze dvě větve A a B, které se dále nedělily, viz Obrázek 6-2. Konvergence k optimálnímu řešení se zrychlila ve větvi A, ale ve větvi B zůstala stejná – optima bylo dosaženo opět hned ve druhé iteraci. V mnoha případech se ale volí použití krátkodobé paměti vzhledem k náročnosti udržení údajů v paměti. Je-li třeba zintenzivnit procec prohledávání množiny přípustných řešení, volí se střednědobá či dlouhodobá paměť.

Hlavní problém metaheuristiky Tabu Search ovšem spočívá v udržování seznamu zakázaných přechodů, seznam musí obsahovat počáteční řešení, přechod a odpovídající řešení. Obdobné údaje musí být zahrnuty pro další zakázané přechody. Jedná se tedy o velký objem dat, při řešení rozsáhlých úloh se tedy primárně řeší problematika velikosti tabu listu, jež je většinou stanovena experimentálně před samotným řešením rozsáhlého problému s podmínkou zachování efektivnosti algoritmu. Do tabu seznamu se mohou dále ukládat jen ty nejdůležitější atributy o daném přechodu, a to např. předchůdce a následovník invertovaného řetězce. Důsledkem je poté zákaz všech přechodů s touto charakteristikou.

8 Závěr

Tato práce měla za cíl představení metaheuristiky Tabu Search na jednoduchém příkladě minimální kostry s omezeními. Nejdříve došlo v úvodní části k základnímu představení úlohy kostry grafu a jejích modifikací, a to minimální kostry grafu a minimální kostry grafu s omezeními. Dále byly v jednoduchosti popsány tři základní algoritmy pro hledání minimální kostry grafu včetně jejich složitosti z důvodu řešené úlohy minimální kostry grafu s omezeními v praktické části práce. V poslední teoretické části byl dán prostor pro nastínění problému řešitelnosti úloh v kategorii NP-H a NP-C, dále pro prezentaci a vzájemné porovnání možných metod řešení optimalizačních problémů, kterými jsou exaktní metody, heuristické a metaheuristické přístupy.

V praktické části práce bylo nejdříve získáno optimální řešení úlohy minimální kostry s omezeními pomocí exaktní metody, která byla naprogramována v jazyce JAVA 1.7. Poté se hledalo co nejlepší řešení úlohy pomocí metaheuristiky Tabu Search. Dané výsledky byly následně porovnány na základě těchto dvou odlišných přístupů. Došlo se k závěru, že nejlepší řešení získané metaheuristikou Tabu Search odpovídá optimálnímu řešení získanému exaktním přítupem. Na základě tohoto zjištění bylo možné prohlásit nejlepší dosažené řešení metaheuristikou Tabu Search za optimální řešení.

Jednotlivé kroky řešení pomocí metaheuristiky Tabu Search byly prováděny velmi názorně s obrázkovou podporou zobrazující detailně každý krok v dané iteraci a také s podrobným popisem událostí v každé iteraci. Celkový přehled nad jednotlivými iteracemi podávají odpovídající stromy řešení, jeden pro případ s krátkodobou pamětí a druhý pro případ se střednědobou pamětí. Volba délky paměti se na daném příkladě ukázala jako klíčová.

Během řešení dané minimální kostry grafu s omezeními bylo zapotřebí si vést seznam všech možných hran, které mohou být v následujícím kroku do kostry grafu zařazeny a které se naopak v dalším kroku mohou vypustit. Daný seznam se musel po každé iteraci přepočítávat a aktualizovat dle momentálního seznamu zakázaných hran. V případě řešení se střednědobou pamětí, kdy bylo nastaveno expirační pravidlo na hodnotu tří iterací, se osvědčilo si zaznamenávat u každé zakázané hrany počet iterací, ve kterých již daná hrana setrvává v tabu seznamu. Záměna pořadí by totiž vedla k odlišnému vývoji hledání optimálního řešení. Některá nalezená optimální řešení se navzájem lišila zakázanými hranami či jen jejich pořadím v tabu listu. Pouze v jednom případě bylo aplikováno aspirační kritérium, a to až v řešení se střednědobou pamětí. Důvodem bylo, že jen v tomto případě bylo zapotřebí nebrat ohled na seznam zakázaných hran pro získání lepšího než dosud nalezeného řešení. V ostatních případech již nemělo smysl po získání optima aplikovat v dalším postupu aspirační

kritérium, jelikož žádná hodnota celkových nákladů nově získané minimální kostry grafu s omezeními nemohla být nižší než hodnota optima.

V průběhu řešení zvoleného problému metaheuristikou Tabu Search se naskýtala otázka ohledně velikosti tabu listu, délky expirační doby a rozsahu aspiračního kritéria. Na základě uvedeného řešení se osvědčilo použití tabu seznamu zahrnující tři momentální zakázané hrany v každé iteraci, jelikož řešení rychleji konvergovalo k optimu než v případě krátkodobé paměti. Možnost vypuštění pouze jedné hrany ze současné kostry grafu je sice trochu omezující, na druhou stranu je velmi úspěšně zabráněno vzniku cyklů či zpětnému pohybu v řešení. Na rozsáhlejších úlohách by bylo dobré provést experiment, nicméně na základě zkušeností z tohoto příkladu se dá doporučit zařadit spíše více možných hran do aktuálního tabu seznamu, aby nebylo možné se brzy vrátit zpět do již zkoumaného řešení. Jelikož byl daný problém málo rozsáhlý, nebyly údaje zahrnované v tabu seznamu nijak limitovány. Bylo možné uvádět atributy přechodů včetně aktuálního ohodnocení potencionálních koster grafu. V rozsáhlejším případě by bylo zapotřebí tyto údaje eliminovat na nejnižší možnou míru a uvést např. jen předchůdce a následovníky invertovaného řetězce.

Čtenář mohl následovat jednotlivé kroky a osvojit si tak postup řešení úlohy minimální kostry s omezeními pomocí metaheuristiky Tabu Search a poté být schopen jej aplikovat na složitější úlohy. Rešeršní část poskytuje přehled problémů, které se pomocí dané metaheuristiky řešily a také které se v současné době začínají nově řešit. Tato proměna poskytuje nový náhled na zmíněnou metaheuristiku a nabízí se zavedení nových principů a nástrojů k efektivnímu řešení nově zvolených úloh.

Je třeba zdůraznit, že různé, i metaheuristické, přístupy se mohou navzájem kombinovat. Metaheuristika Tabu Search je často doprovázena přístupy genetických algoritmů, Lagrangeových relaxací, programování s omezeními (Constraint Programming) a celočíselného programování (Integer Programming). Metaheuristika Tabu Search je výkonným nástrojem, který byl s úspěchem aplikován na řadu těžkých reálných kombinatorických úloh. Tento přístup si dokáže také poradit s řadou reálných a velmi komplikovaných omezení.

Co se týká nových trendů ve využívání metaheuristiky Tabu Search, již se opouští od klasických aplikací (problémy z teorie grafů, rozvrhovací problémy, rozvozní problémy) a přechází se k novým, a to ke kontinuální optimalizaci (Continuous Optimization), vícekriteriální optimalizaci (Multi-criteria Optimization), stochastickému programování (Stochastic Programming), smíšenému celočíselnému programování (Mixed Integer Programming), problémům rozhodování se v reálném čase (Real-time Decision Problems), atd. Tyto nové aplikace mohou vést k rozšíření technik zahrnutých v aparátu metaheuristiky Tabu Search.

Literatura

[1] FIALA, Petr. *Řízení projektů*. 2. přeprac. vyd. Praha: Oeconomica, 2008. 186 s. ISBN 978-80-245-1413-0.

[2] VAN DEN BOOMEN, Janneke. *Non-isomorphic spanning trees of graphs*. Nijmegen, Netherlands, 2009. [cit. 2014-05-12]. Dostupné online: <http://www.math.ru.nl/~bosma/Students/JannekevandenBoomen/JannekevdBoomen MScthesis.pdf>. Master Thesis.

[3] MILKOVÁ, Eva. *Teorie grafů a grafové algoritmy*. Vyd. 1. Hradec Králové: Gaudeamus, 2013, 123 s. ISBN 978-80-7435-267-6.

[4] WU, Bang Ye a Kun-Mao CHAO. *Spanning trees and optimization problems*. Boca Raton, FL: Chapman & Hall/CRC, c2004, xv, 184 p. ISBN 1584884363.

[5] KOLÁŘ, Josef. *Teoretická informatika*. Vyd. 1. V Praze: České vysoké učení technické, 2009, 206 s. ISBN 978-80-01-04331-8.

[6] PETTIE, Seth. a RAMACHANDRAN, Vijaya. *An optimal minimum spanning tree algorithm*. Journal of the ACM. 2002, 49(1): 16-34. DOI: 10.7717/peerj.739/fig-6.

[7] VOLEK, Josef a Bohdan LINDA. *Teorie grafů - aplikace v dopravě a veřejné správě*. Vyd. 1. Pardubice: Univerzita Pardubice, 2012, 190 s. ISBN 978-80-7395-225-9.

[8] MAREŠ, Martin. *Graph Algorithms (The Saga of Minimum Spanning Trees)*. Praha, 2008 [cit. 2014-12-20]. Dostupné také z: <http://mj.ucw.cz/papers/saga/saga.pdf>. Disertační práce. Karlova Univerzita. Vedoucí práce Prof. RNDr. Jaroslav Nešetřil, DrSc.

[9] UNČOVSKÝ, Ladislav. *Modely sieťovej analýzy*. 1. vyd. Bratislava: Alfa, 1991, 236 s. Edícia ekonomickej literatúry. ISBN 80-05-00812-0.

[10] HUMMER, Waldemar. *Constrained Minimum Spanning Tree Algorithms*. TU Wien [online]. Vienna, Austria, 2008 [cit. 2015-01-10]. Dostupné online: <https://www.ads.tuwien.ac.at/teaching/ws08/Seminar/MST_hummer.pdf>

[11] NADIRADZE, Giorgi. *Bounded Diameter Minimum Spanning Tree*. Budapešť, Maďarsko, 2013 [cit. 2015-01-10]. Dostupné také z: <http://www.etd.ceu.hu/2013/nadiradze_giorgi.pdf>. Disertační práce. Central European University. Vedoucí práce Profesor Ervin Györi.

[12] BUI, T. N. a ZRNCIC, C. M. 2006. *An ant-based algorithm for finding degree-constrained minimum spanning tree*. In GECCO ’06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pages 11–18, New York, NY, USA. ACM

[13] MLADENOVIC, Nenad, BRIMBERG, Jack, HANSEN, Pierre, a MORENO-PÉREZ, José A. (2007). *The p-median problem: A survey of metaheuristic approaches*. European Journal of

Operational Research, 179(3), 927-939. Dostupné online z: <http://jamoreno.webs.ull.es/www/papers/EJOR07.pdf>.

[14] DREYER, D.R. and OVERTON, M.L. Two Heuristics for the Euclidean Steiner Tree Problem. *Journal of Global Optimization*, 1998, vol. 13, no. 1. pp. 95-106 ProQuest Central. ISSN 09255001. DOI Dostupné online z: <http://dx.doi.org/10.1023/A:1008285504599>.

[15] PELIKÁN, Jan. *Diskrétní modely*. Praha: Vysoká škola ekonomická, 1999. 163 s. ISBN 80-7079-179-9.

[16] DAEGENE, S. "*The P versus NP Problem in Quantum Physics*." Neuroquantology 12, no. 4 (December 2014): 350-354. Academic Search Complete, EBSCOhost

[17] ŠANDERA, Čeněk. *Hybridní model metaheuristických algoritmů*. Brno, 2013 [cit. 2015-01-16]. Dostupné z: <http://hdl.handle.net/11012/35853>. Disertační práce. Vysoké učení technické v Brně. Fakulta strojního inženýrství. Vedoucí práce Prof. RNDr. Ing. Miloš Šeda, Ph.D.

[18] Pearl, Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. New York, Addison-Wesley, 1984.

[19] TUZAR, A. a PASTOR, O. *Teorie dopravních systémů*. vyd. 1. Praha: ASPI a.s., 2007. ISBN 978-80-7357-285-3.

[20] GLOVER F.. *Future paths for integer programming and links to artificial intelligence*. Computers and Operational Research, 13(5), 1986.

[21] REEVES C.. *Modern heuristic techniques for combinatorial problems*., 2005. Oxford: Scientific Publications.

[22] DORIGO, Marco a Thomas STÜTZLE. *Ant colony optimization*. Cambridge, Mass.: MIT Press, c2004, xi, 305 s. ISBN 0-262-04219-3.

[23] OSMAN Ibrahim H. and KELLY James P.. *Meta-heuristics: theory & applications*. Springer, Kluwer Academic Publisher, USA. 1996.

[24] VOSS Stefan, OSMAN Ibrahim, a ROUCAIROL Catherine. *Metaheuristics: Advances and trends in local search paradigms for optimization*. Kluwer Academic Publishers, 1999.

[25] WEBER Ben. *Distributed Hybrid Metaheuristics for Optimization*. California Polytechnic State University. 2005.

[26] TALBI El-Ghazali. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons Publishing, 2009.

[27] YANG X.-S.. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[28] SÖRENSEN, Kenneth, GLOVER, Fred W., *Encyclopedia of Operations Research and Management Science, Metaheuristics*. 960-970 pp. Springer US, 2013.

[29] GLOVER, Fred. *Tabu Search: A Tutorial.* Interfaces. 20, 4, 74-94, July 1990. ISSN: 00922102.

[30] GLOVER, Fred. *Tabu search: Part-I*. ORSA Journal on computing 1, no. 3, 1989. 190-206 pp.

[31] GLOVER, Fred. *Tabu search: Part-II*. ORSA Journal on computing 2, no. 1, 1990. 4-32 pp.

[32] HERTZ, Alain., de WERRA, Dominique: *The tabu search metaheuristic: how we used it*. Annals of Mathematics and Artificial Intelligence 1, no. 1, 1990. 111–121 pp.

[33] de WERRA, Dominique, HERTZ, Alain: *Tabu search techniques: a tutorial and an application to neural networks*. Operations Research Spektrum 11, no. 3, 1989. 131-141 pp.

[34] CRAINIC, Teodor Gabriel, GENDREAU, Michel: *Towards an evolutionary method-Cooperative multi-thread parallel tabu search heuristic hybrid*. Meta-Heuristics, Springer US, 1999. 331-344 pp.

[35] FLEURENT, Charles., FERLAND, Jacques A.: *Genetic and hybrid algorithms for graph colouring*. Annals of Operations Research 63, no. 3, 1996. 437-461.

[36] GRÜNERT, Tore: *Lagrangean tabu search*. Essays and Surveys in Metaheuristics, Springer, US, Kluwer, Boston, 2002. 379–397 pp.

[37] de BACKER, Bruno, FURNON, Vincent, SHAW, Paul, KILBY Philip, Prosser Patrick: *Solving vehicle routing problems using constraint programming and metaheuristics*. Journal of Heuristics 6, no. 4, 2000. 501–523 pp.

[38] CASEAU Yves, LABURTHE, Francois, LE PAPE, Claude, ROTTEMBOURG, Benoit: *Combining local and global search in a constraint programming environment*. The Knowledge Engineering Review 16, no. 1, 2001. 41–68 pp.

[39] PESANT, Gilles, GENDREAU, Michel: *A constraint programming framework for local search methods*. Journal of Heuristics 5, no. 3, 1999. 255–280 pp.

[40] GLOVER, Fred, LAGUNA, Manuel: *Tabu Search*. Springer, US, 1999.

[41] BATTITI, Roberto, TECCHIOLLI, Giampietro: *The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization*. Annals of Operations Research 63, no. 2, 1996. 151–188 pp.

[42] CHELOUAH, Rachid, SIARRY, Patrick: *Tabu Search applied to global optimization*. Eur. J. Oper. Res. 123, no. 2, 2000. 256–270 pp.

[43] CHELOUAH, Rachid, SIARRY, Patrick *A hybrid method combining continuous tabu search andNelder-Mead simplex algorithms for the global optimization of multiminima functions*. European Journal of Operations Research 161, no. 3, 2005. 636–654 pp.

[44] JAEGGI, D.M., PARKS, Geoffrey T., KIPOUROS, Timoleon, CLARKSON, P. John: *The development of a multiobjective tabu search algorithm for continuous optimisation problems*. European Journal of Operations Research. 185, no. 3, 2008. 1192–1212 pp.

[45] ROLLAND, E.: *A tabu search method for constrained real-number search: Applications to portfolio selection*. Technical Report, Department of Accounting and Management Information Systems, Ohio State University, Columbus, 1997.

[46] GLOVER, Fred: *Ejection chains, reference structures and alternating path methods for traveling salesman problems*. Discrete Applied Mathematics 65, no. 1, 1996. 223-253.

[47] ARINGHIERI, Roberto: *Solving chance-constrained programs combining tabu search and simulation.* Experimental and Efficient Algorithms. Springer Berlin Heidelberg, 2004. 30-41 pp.

[48] LøKKETANGEN, Arne, WOODRUFF, David L.: *Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming*. Journal of Heuristics 2, no. 2, 1996. 111–128 pp.

[49] CRAINIC, Teodor Gabriel, GENDREAU, Michel, FARVOLDEN, Judith M.: *Simplex-based tabu search for the multicommodity capacitated fixed charge network design problem*. INFORMS J. Comput. 12, 200. 223–236 pp.

[50] OSMAN, Ibrahim H.: *Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem*. Annals of Operations Research 41, no.4, 1993. 421–451 pp.

[51] GENDREAU, Michel, GUERTIN, Francois, POTVIN, Jean-Yves, SÉGUIN, René: *Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries*. Transportation Research Part C: Emerging Technologies 14, no. 3, 2006. 157–174 pp.

[52] CHU, Andrej. *Metaheuristická metóda mravčej kolónie pri riešení kombinatorických optimalizačných úloh*. Praha, 2009. [cit. 2015-02-15] Dostupné z: <https://isis.vse.cz/auth/lide/clovek.pl?zalozka=7;id=49592;studium=85439;zp=21305;download_prace=1>. Disertační práce. Vysoká škola ekonomická. Vedoucí práce Prof. Ing. Josef Jablonský, CSc.

[53] SINGH, Alok., a BAGHEL, Anurag S. *New metaheuristic approaches for the leaf-constrained minimum spanning tree problem*. Asia-Pacific Journal of Operational Research 25, no. 4, 2008. 575-589 pp.

[54] KASPERSKI, Adam, MAKUCHOWSKI, Mariusz a ZIELISKI, Pawel. *A Tabu Search Algorithm for the Minmax Regret Minimum Spanning Tree Problem with Interval Data.* Journal of Heuristics 18, no. 4, 2012. 593-625 pp. ProQuest Central. ISSN 13811231.

[55] BREEDAM, Alex V. *Comparing descent heuristics and metaheuristics for the vehicle routing problem*. Computers & Operations Research 28, no. 4, 2001. 289–315 pp.

[56] CHIANG, W.C., RUSSELL, R.A.: *A reactive tabu search metaheuristic for the vehicle routing problem with time windows*. INFORMS Journal of Computing 9, no. 4, 1997. 417-430 pp.

[57] CRAINIC, Teodor Gabriel, GENDREAU, Michel, SORIANO, Patrick, TOULOUSE, Michel: *A tabu search procedure for multicommodity location/allocation with balancing requirements*. Annals of Operations Research 41, no. 4, 1993. 359–383 pp.

[58] GENDREAU,M., HERTZ, A., LAPORTE, G.: *A tabu search heuristic for the vehicle routing problem*. Management Science 40, no. 10, 1994. 1276–1290 pp.

[59] APPLEGATE, David A., BIXBY, Robert E., CHVATAL, Vaclav, COOK, William J.: *The Traveling Salesman Problem: A Computational Study*. Princeton University Press. 2011.

[60] GUTIN, Gregory, PUNNEN, Abraham P.: *The Treveling Salesman Problem and its Variations*. Kluwer, Boston. Springer Sci ence & Business Media, 2002.

[61] JOHNSON, David S., GUTIN, Gregory, McGEOCH, Lyle A., YEO, Anders, ZHANG, Weixiong, ZVEROVICH, Alexei: *The traveling salesman problem and its variations*. *Experimental Analysis of Heuristics for the ATSP*. Kluwer, Boston. Springer US, 2002.

[62] BASU, Sumanta, GHOSH, Diptesh: *A review of the tabu search literature on traveling salesman problems*. IIM Ahmedabad Working Paper Series, 2008.

[63] BASU, Sumanta. *Neighborhood reduction strategy for tabu search implementation in asymmetric traveling salesman problem*. *O*psearch 49, no. 4, 2012. 400-412 pp.

[64] BASU, Sumanta, GAJULAPALLI, RavindraS. and GHOSH, Diptesh. *A Fast Tabu Search Implementation for Large Asymmetric Traveling Salesman Problems Defined on Sparse Graphs.* Opsearch 50, no. 1, 2013. 75-88 pp. ISSN 00303887.

[65] FEILLET, Dominique, DEJAX, Pierre and GENDRAU, Michel. *Traveling Salesman Problems with Profits.* Transportation Science 39, no. 2, 2005. 188-205 pp. ProQuest Central. ISSN 00411655.

[66] GUPTA, Dharmendra Kumar. *Tabu Search for Vehicle Routing Problems (VRPs)*. International Journal of Computer Mathematics 79, no. 6, 2002. 693-701pp. ISSN: 00207160.

[67] CESCHIA, Sara, DI GASPERO, Luca a SCHAERF, Andrea. *Tabu Search Techniques for the Heterogeneous Vehicle Routing Problem with Time Windows and Carrier-Dependent Costs.* Journal of Scheduling 14, no. 6, 2011. 601-615 pp. ProQuest Central. ISSN 10946136.

Seznam obrázků

Obrázek 2-1: počet koster pro úplný graf 4
Obrázek 3-1: Grafická reprezentace problému NP 11
Obrázek 3-2: Výpočet NP a P problémů 12
Obrázek 4-1: Minimální kostra grafu bez ohledu na zadané penalizace (vlevo), minimální kostra grafu s ohledem na zadané penalizace (vpravo) 22
Obrázek 5-1: Rozhodovací strom přípustnosti dané změny 27
Obrázek 5-2: Rozhodovací strom metaheuristiky Tabu Search s krátkodobou pamětí 28
Obrázek 5-3: Rozhodovací strom výběru nejlepšího přípustného řešení 30
Obrázek 6-1: rozhodovací strom metaheuristiky Tabu Search s krátkodobou pamětí 51
Obrázek 6-2: rozhodovací strom metaheuristiky Tabu Search se střednědobou pamětí 61
Obrázek 6-3: OŘ ve větvi B s krátkodobou pamětí (vlevo), OŘ ve větvi B s dlouhodobou pamětí (vpravo) 62

Seznam tabulek

Tabulka 2-1: Přehled algoritmů 7
Tabulka 3-1: Přehled výpočetního času algoritmů 12
Tabulka 4-1: Výčet neizomorfních koster grafu bez penalizací 23
Tabulka 4-2: Výčet neizomorfních koster grafu s penalizacemi 23
Tabulka 5-1: Ilustrativní přehled aplikací metaheuristiky Tabu Search 31