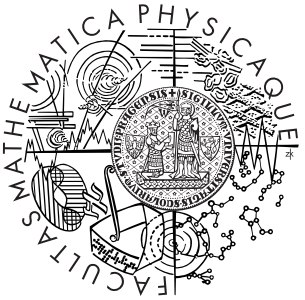
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Tibor Vansa

Implementace algoritmu pro transformaci acyklického orientovaného grafu na esenciální graf

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: Mgr. Petr Šimeček, MSc.

Studijní program: Obecná matematika

2006

Děkuji vedoucímu bakalářské práce za vedení a pomoc během přípravy bakalářské práce.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne Tibor Vansa

Obsah

Název práce: Implementace algoritmu pro transformaci acyklického orientovaného grafu na esenciální graf
Autor: Tibor Vansa
Katedra: Katedra pravděpodobnosti a matematické statistiky
Vedoucí bakalářské práce: Mgr. Petr Šimeček, MSc.
e-mail vedoucího: simecek@atrey.karlin.mff.cuni.cz

Abstrakt: Cílem předložené práce je seznámit čtenáře se základní tématikou Bayesovských sítí a jejich použitím. Ty poskytují přirozený nástroj pro práci s informacemi zatíženými neurčitostí a hrají důležitou roli v oblasti návrhu a analýzy samoučících se algoritmů. Dále představuje program implementující algoritmus pro transformaci acyklického orientovaného grafu na esenciální graf. Jeho jádro tvoří algoritmus RNDr. Milana Studeného, DrSc. Podstatou zmíněného algoritmu je metoda korektního slučování komponent grafu.

Klíčová slova: Bayesovské sítě, esenciální graf, korektní slučování komponent řetězcového grafu

Title: Implementation of Algorithm for Transformation of Acyclic Directed Graph to Essential Graph
Author: Tibor Vansa
Department: Department of Probability and Mathematical Statistics
Supervisor: Mgr. Petr Šimeček, MSc.
Supervisor's e-mail address: simecek@atrey.karlin.mff.cuni.cz

Abstract: The aim of the work is to introduce the reader to the theory of Bayesian Networks and their applications. They provide a natural tool for dealing with the information interfered by uncertainty and play important role in the design and analysis of machine learning algorithms. Further, the thesis presents the program implementing the algorithm for transformation of an acyclic directed graph to an essential graph. The main idea behind the algorithm is taken from the work of RNDr. Milan Studený, DrSc. The algorithm is based on the method of legal merging of components.

Keywords: Bayesian Networks, essential graph, legal merging of chain graph components

Kapitola 1

Základní vlastnosti Bayesovských sítí

1.1 Úvod

Bayesovské sítě (dále BN - Bayesian networks) jsou třídou pravděpodobnostních modelů, kde vrcholy grafu reprezentují náhodné veličiny a hrany závislosti mezi nimi. Umožňují nám výpočty podmíněných pravděpodobností i v případech, kdy jednoduché použití Bayesova vzorce pro výpočet podmíněné pravděpodobnosti selhává (viz např. [1]). Grafické uspořádání nám umožní lepší intuitivní náhled na daný problém. BN se využívají hlavně v umělé inteligenci, medicíně a jiných problémech, kde vztah příčina-efekt hraje důležitou roli a můžeme ho explicitně vyjádřit pomocí orientované hrany.

BN můžeme sestavovat dvěma různými způsoby, které lze kombinovat. Buďto využijeme znalosti lidských expertů o jednotlivých závislostech a jejich pravděpodobnostech a sestrojíme BN přímo nebo můžeme mít k dispozici velké množství pokusů (dat) a na jejich základě parametry BN určíme. Stejně variabilní jsou i možnosti vyvozování závěrů. Z informace o libovolné náhodné veličině můžeme usuzovat o změně pravděpodobnostních rozdělení všech ostatních náhodných veličin.

1.2 Konstrukce BN

Za BN považujeme dvojici (G,P). $G = (V, E)$ je acyklický orientovaný graf, kde V je konečná množina vrcholů a E množina hran mezi nimi. Vrchol v BN představuje náhodná veličina $X_i \in V$. Hrany z množiny E představují „závislostní vztahy“ mezi nimi. P je sdružená hustota náhod-

ných veličin $X_i \in V$. Navíc předpokládáme že (G, P) splňuje tzv. Markovskou vlastnost, viz. 2.kapitola. Ke každému vrcholu náleží jeho marginální rozdělení. V případě diskrétního rozdělení je často zobrazováno jako tabulka. Ta udává pravděpodobnost nabývání jednotlivých hodnot pro každou kombinaci hodnot rodičovských vrcholů.

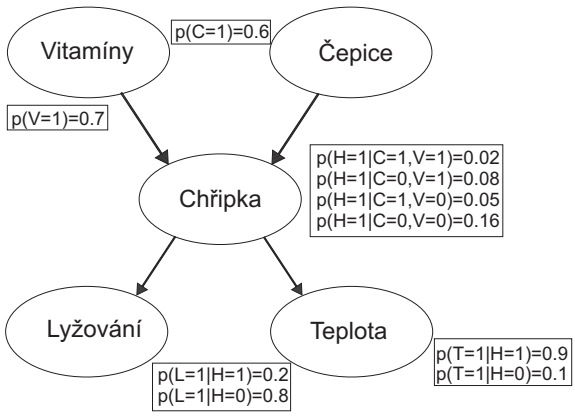
V praxi většinou známe podmíněná rozdělení jednotlivých vrcholů a zajímá nás, jak vypočítat sdružené pravděpodobnostní rozdělení. To lze v obecném případě vyjádřit pomocí řetízkového pravidla jako

$$P(x_1, x_2, \ldots x_n) = P(x_1) \times P(x_2|x_1) \times \ldots \times P(x_n|x_1, \ldots, x_{n-1}) =$$

$$= \prod_i P(x_i|x_1, \ldots, x_{i-1})$$

Jak později uvidíme, v BN podmíněná pravděpodobnost vrcholu závisí pouze na vrcholech, z kterých do něj vede hrana, a předchozí vzorec se proto zjednoduší na

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|Rodiče(x_i))$$



Obrázek 1.1: Příklad Bayesovské sítě

Příklad 1. V BN na obrázku 1.1 událost „dostanu Chřipku“ může být ovlivněna dvěma příčinami, zda jsem „bral Vitamíny“ a zda jsem „nosil Čepici“. Chřipka může mít dva následky, jestli „dostanu Teplotu“ a jestli „pojedu za dva týdny Lyžovat“. Výpočet sdruženého rozdělení se zde zredukuje na

$$P(v, c, h, l, t) = P(v) \times P(c|v) \times P(h|c, v) \times P(l|c, v, h) \times P(t|c, v, h, l) =$$

$$= P(v) \times P(c) \times P(h|c,v) \times P(l|h) \times P(t|h)$$

Budeme-li chtít např. vypočíst, s jakou pravděpodobností jsem pojedu Lyžovat, jestliže jsem měl Teplotu. Použijeme k tomu Bayesův vzorec:

$$P(L=1|T=0) = P(L=1,T=0,H=1) + P(L=1,T=0,H=1) =$$

$$= P(L=1|H=0)\cdot P(H=0|T=0) + P(L=1|H=1)\cdot P(H=1|T=0) =$$

$$= P(L=1|H=0) \times \frac{P(T=0|H=0)P(H=0)}{P(T=1)} +$$

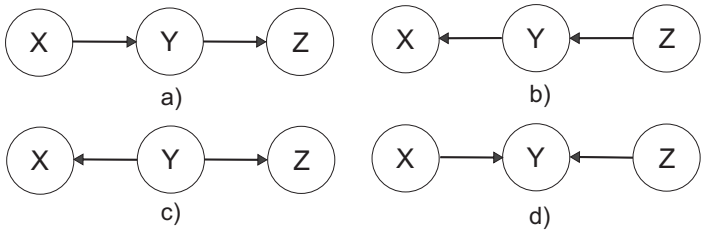
$$+ P(L=1|H=1) \times \frac{P(T=0|H=1)P(H=1)}{P(T=1)}$$

Za $P(H=1)$ se dále dosadí

$$\sum_{c,v} P(H=1|C=c,V=v)$$

1.3 Problém kauzality

V této oblasti nepanuje v komunitě okolo BN consensus. Otázka zní, zda můžeme odlišit pouhou pravděpodobností interpretaci od příčinné interpretace. Často je totiž možné za užití Bayesova pravidla změnit orientaci hrany v grafu, aniž by se nám změnilo výsledné sdružené rozdělení P. Je však zřejmé, že oba grafy nemohou zároveň správně prezentovat příčinu a důsledek (př. viz. obrázek 1.2)



Obrázek 1.2: BN s grafy na obrázcích a), b) a c) mohou vždy reprezentovat stejné sdružené pravděpodobnostní rozdělení. V obecném případě však toto rozdělení nemůže být reprezentováno BN s grafem na obrázku d). Jedno pravděpodobnostní rozdělení tedy může mít různé příčinné interpretace.

Výhoda příčinné interpretace Bayesovských sítí spočívá v možnosti předvídání důsledků aktivních zásahů, zatímco nekauzální sítě pouze pasivně zaznamenávají dané děje. Tato vlastnost je důležitá například v

oblasti kontroly procesů, výrobních postupů a rozhodovací podpory v medicíně.

1.4 Odhadování parametrů BN

V praxi se často můžeme setkat se situací, kdy nemáme k dispozici experta, který by znal jednotlivé příčinné vztahy mezi vrcholy a pomohl nám sestrojit potřebnou BN. Nebo je čas těchto expertů příliš drahý, zato máme k dispozici rozsáhlé soubory dat. Naštěstí můžeme využít schopnosti BN automaticky se naučit pravděpodobnosti a strukturu ze statistických dat. Současné soubory jsou někdy až tak rozsáhlé, že vyhledávat v nich strukturu bez automatizované pomoci je nemožné. Výpočetní náročnost závisí na tom, jak úplná data máme k dispozici a jak rozsáhlé znalosti jsme získali od expertů.

Pro hledání vhodné BN potřebujeme dva základní prvky: Ohodnocovací funkci a algoritmus pro procházení množinou všech možných struktur (neboli množiny hran). Ohodnocovací funkce nám udává nakolik vhodnou BN jsme našli. Většinou se skládá ze tří samostatných funkcí, které reflektují hlavní požadavky na vhodnou BN. Prvním z nich je, aby se nalezená BN co nejvíce podobala původní BN sestavené experty. Za druhé by měla BN co nejvíce odpovídat datům. Třetím požadavkem je co nejjednodušší struktura grafu.

Algoritmus pro procházení všech možných struktur grafu má za úkol najít struktury s nejlepšími ohodnoceními pomocí ohodnocovací funkce (s co nejvyšším skóre), neboť vzhledem k rozsáhlosti prohledávaného prostoru možných struktur je výpočetně nemožné ohodnotit všechny možnosti. Navíc, jak již víme, různé struktury mohou reprezentovat stejné pravděpodobnostní rozdělení, a proto se ohodnocují pouze zástupci tříd ekvivalentních struktur.

1.5 Aplikace

BN pomáhají mnoha různým vědním disciplínám se vyrovnat s problémem neurčitosti v reálném světě. Oblastí, kde se BN hojně využívají, je medicína. Ta poskytuje dostatek vědomostí aplikovatelných v BN a na druhou stranu grafická podoba BN umožňuje lékařům snadné pochopení dosažených výsledků. Jeden z hlavních problémů medicíny, hledání příčiny nemoci a volba vhodných způsobů léčení, je navíc velmi vhodný pro BN.

Jako příklad můžeme uvést PATHFINDER, diagnostický systém pro nemoci lymfatických uzlin. MUNIN je BN s okolo 1000 vrcholy pro diagnostisiku neuromoskulárních poruch. Z dalších oblastí je to např. HAILFINDER, systém předpověď krupobití v severovýchodním Coloradu. GEMS je expertní monitorovací systém, kterému se podařilo uspět tam, kde pravidlové systémy selhaly. Vista je systém vyvinutý NASA pro řízení telemetrie pohonů vesmírných raketoplánů. Dalšími použití jsou v nápovědách softwarových aplikacích, při hraní her, inteligentních učících programech pro studenty, předpovídání pohybu cen ropy, rozpoznávání řeči, monitorování pohybu robotů nebo dopravních zácpy. (Seznam příkladů byl převzat z [2] str.120)

1.6 Software

Pro práci s BN můžeme využít několika programů, freewarových i komerčních. Zde je seznam nejznámějších:

- Analytica (http://www.lumina.com)
- BayesiaLab (http://www.bayesia.com)
- BayesNetToolbox (http://www.ai.mit.edu./∼murphyk/software/BNT/bnt.html)
- GeNIe (http://www.sis.pitt.edu/∼genie/)
- gR (http://www.r-project.org)
- Hugin (http://www.hugin.com)
- JavaBayes (http://www.pmr.poli.usp.br/ltd/software/javabayes/)
- Netica (http://www.norsys.com)
- WinBugs (http://www.mrc-bsu.cam.ac.uk/bugs/)

Kapitola 2

Markovská vlastnost

2.1 Zavedení značení

Nejprve zaveďme několik pojmů z teorie grafů. Orientovaným grafem rozumíme dvojici $G = (V, E)$, kde V je konečná neprázdná množina vrcholů a E je množina orientovaných hran. Orientovanou hranu nad množinou V definujeme jako uspořádanou dvojici $(a, b) \in E$, $a, b \in V$, $a \neq b$, $(b, a) \notin E$ (značení $a \rightarrow b$). Cyklus v grafu je posloupnost vrcholů $a_1, a_2, \ldots, a_n$, $n \geq 3$, taková, že platí $a_1 = a_n$ a $(a_i, a_{i+1}) \in E$ $\forall i = 1, \ldots, n-1$. Řetězcem v grafu je posloupnost vrcholů $a_1, a_2, \ldots, a_n$, $n > 1$, taková, že $(a_i, a_{i+1}) \in E$ nebo $(a_{i+1}, a_i) \in E$ $\forall i = 1, \ldots, n-1$. Orientovaným acyklickým grafem budeme rozumět orientovaný graf bez cyklů a budeme pro něj používat zkratku DAG (directed acyclic graph). Jestliže z vrcholu a vede orientovaná hrana do vrcholu b, a nazýváme rodičem b. Jestliže existuje posloupnost vrcholů $a = c_1, c_2, \ldots, c_{n-1}, c_n = b$, $n > 1$ takových, že pro $i = 1, \ldots, n-1$ je c_i rodičem c_{i+1}, nazýváme b potomkem a. Vrcholy, které nejsou potomky vrcholu a, budeme nazývat nepotomky (kromě samotného vrcholu a). Množinu všech rodičů vrcholu a budeme značit PA_a (jako parents) a množinu všech nepotomků ND_a (nondescendents).

2.2 Markovská vlastnost

Kdybychom konstruovali BN jakou úplné grafy, velikost tabulek by exponenciálně závisela na počtu náhodných veličin. Proto je důležité, abychom do grafu dávali hrany pouze v tom případě, kdy je mezi příslušnými náhodnými veličinami přímá závislost.

Definice 2.0. *Nechť máme tři náhodné veličiny X,Y,Z a jejich sdružené*

rozdělení P. Potom X je podmíněně nezávislá na Y za předpokladu Z, jestliže platí

$$P(X = x, Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z)$$

pro všechna x,y,z, $P(Z = z) > 0$. Pokud je množina náhodných veličin $\{X_\alpha \in A\}$ podmíněně nezávislá na $\{Y_\beta \in B\}$ za předpokladu $\{X_\gamma \in C\}$, kde $A, B, C \subseteq V$, píšeme zkráceně $I_P(A, B|C)$.

Definice 2.1. *Nechť máme sdružené rozdělení P náhodných veličin z množiny V a DAG G=(V,E). Potom (G,P) splňuje Markovskou vlastnost, pokud každá náhodná veličina $X_i \in V$ je nezávislá na množině nepotomků za předpokladu množiny rodičů. Neboli*

$$I_P(X_i, ND_{X_i}|PA_{X_i})$$

Věta 2.2. *Jestliže (G,P) splňuje Markovskou vlastnost, pak je sdružené rozdělení P jednoznačně určené pouze podmíněnými pravděpodobnostmi jednotlivých vrcholů za předpokladu jejich rodičů, kdykoliv tato podmíněná rozdělení existují. Z nich lze P určit pomocí řetízkového pravidla. Důkaz viz.[4] str. 39*

Zde spočívá síla BN, neboť v takovýchto případech můžeme sdružené rozdělení popsat pouze s pomocí tabulek závislých jen na rodičích příslušného vrcholu. Z Markovské vlastnosti vyplývají i jiné nezávislosti než ty, založené na rodičích vrcholu. Zavedení pojmu d-separace nám je umožní v grafu najít.

Definice 2.3. *Nechť X,Y,Z jsou vrcholy v G. Pokud orientované hrany v řetězci XZY mají tvar $X \rightarrow Z \rightarrow Y$, nazýváme tento řetězec lineární. Jestliže mají tvar $X \leftarrow Z \rightarrow Y$, nazýváme ho větvení. Jestliže mají tvar $X \rightarrow Z \leftarrow Y$, jde o obrácené větvení.*

Definice 2.4. *Nechť G=(V,E) je DAG, $A \subseteq V$, S a T jsou různé vrcholy náležící $V \setminus A$ a ρ je řetězec mezi S a T. Potom ρ je blokován A pokud nastane alespoň jedna z následujících možností:*

1. *Existuje takový vrchol $Z \in A$ na ρ, že v řetězci ρ tři za sebou jdoucí vrcholy X,Z,Y tvoří lineární řetězec nebo větvení.*

2. *Existuje takový vrchol Z na ρ, že Z a všichni jeho potomci nenáleží do A. Zároveň X,Z,Y jsou tři za sebou jdoucí vrcholy v ρ tvořící obrácené větvení.*

Definice 2.5. *Nechť G=(V,E) je DAG, $A \subset V$, X a Y jsou různé vrcholy z $V \setminus A$. Pak X a Y jsou d-oddělitelné množinou A v G pokud každý řetězec mezi X a Y je blokován A.*

Následující definice a věta nám zaručí, že všechny nezávislosti vynucené Markovskou vlastností můžeme identifikovat pomocí d-oddělitelnosti.

Definice 2.6. *Nechť G=(V,E) je DAG, A,B,C jsou navzájem disjunktní podmnožiny V. Pak A a B jsou d-oddělitelné množinou C, jestliže pro každé X∈A, Y∈B, X a Y jsou d-oddělitelné C. Píšeme*

$$I_G(A, B|C).$$

Jestliže C je prázdná množina, píšeme pouze

$$I_G(A, B).$$

Věta 2.7. *Nechť P je sdružené pravděpodobnostní rozdělení veličin ve V a G=(V,E) je DAG. Potom (G,P) splňuje Markovskou vlastnost právě když pro každé tři množiny $A, B, C \subset V$, které jsou navzájem disjunktní, platí následující implikace: Kdykoliv A a B jsou d-oddělitelné pomocí C, potom A a B jsou podmíněně nezávislé na C v P. Tedy, (G,P) splňuje Markovskou vlastnost právě když*

$$I_G(A, B|C) \Rightarrow I_P(A, B|C).$$

Důkaz viz.[4] str.77

Definice 2.8. *Nechť $G_1 = (V, E_1), G_2 = (V, E_2)$ jsou dva DAGy na stejné množině náhodných veličin V. Potom G_1 a G_2 jsou Markovsky ekvivalentní, pokud pro každé tři navzájem disjunktní množiny A,B a C platí následující ekvivalence: A a B jsou d-oddělitelné za podmínky C v G_1 právě když jsou d-oddělitelné v G_2. Neboli*

$$I_{G_1}(A, B|C) \Leftrightarrow I_{G_2}(A, B|C).$$

Uvedeme si ještě jednu možnou charakterizaci Markovovsky ekvivalentních grafů, která se nám bude hodit v následující kapitole. Za imoralitu budeme považovat obrácené větvení $X \rightarrow Y \leftarrow Z$, kde mezi X a Z nevede hrana.

Věta 2.9. *Dva grafy jsou Markovsky ekvivalentní právě když mají stejné hrany (bez ohledu na orientaci) a pokud mají stejnou množinu imoralit. Důkaz viz.[4] str.90*

Definice 2.10. *Nechť G=(V,E) je DAG. Potom podmíněná nezávislost $I_P(A, B|C)$ je vynucená Markovskou vlastností grafu G, jestliže platí*

$$I_P(A, B|C), \forall P \in \mathbf{P},$$

kde $\mathbf{P}$ *je množina všech sdružených rozdělení P, pro které (G,P) splňuje Markovskou vlastnost.*

Lze odvodit, že dva DAGy jsou Markovsky ekvivalentní právě tehdy, když si vynucují stejné nezávislosti, neboli pokud pro každé sdružené rozdělení P platí: P splňuje Markovskou vlastnost vůči $G_1 = (V, E_1)$ právě tehdy pokud splňuje Markovskou vlastnost vůči $G_2 = (V, E_2)$.

Další poznatky o Markovské vlastnosti a ekvivalenci grafů lze nalézt nalézt v [4] či [3].

Kapitola 3

Studeného algoritmus

3.1 Úvod

Jak již víme z 1.kapitoly, některé přístupy k učení v BN jsou založeny na maximalizaci skóre grafu. Ty využívají metody cestování prostorem tříd Markovsky ekvivalentních grafů nad množinou náhodných veličin. Proto potřebujeme vhodnou metodu pro reprezentaci takových grafů. Jednou z populárních reprezentací je idea esenciálního grafu. Objevují se i jiné termíny pro tento typ: úplný vzor, maximální orientovaný graf pro vzor nebo úplný pdag. Cílem Studeného algoritmu je vylepšit efektivnost při hledání esenciálního grafu během učení BN.

3.2 Zavedení značení

V této kapitole budeme pracovat s hybridními grafy, tj. grafy $H = (V, L, E)$, kde V je množina vrcholů, L množina neorientovaných hran (značení $a - b$) a E je množina orientovaných hran (značení $a \to b$). Neorientovanou hranu nad množinou V definujeme jako dvojici (a, b), $a, b \in V, a \neq b$. Množina vrcholů C je spojitá, jestliže pro každé dva vrcholy $a, b \in C$ existuje neorientovaná cesta je spojující, neboli posloupnost vrcholů $a = a_1, a_2, \ldots, a_n = b$ kde $a_i - a_{i+1} \in L$ pro $i = 1, \ldots, n-1$. Komponenta spojitosti grafu H je maximální spojitá množina vrcholů v H (maximální ve smyslu inkluze). Řetězcovým grafem rozumíme hybridní graf ve kterém existuje uspořádané rozdělení vrcholů grafu na neprázdné podmnožiny $B_1, B_2, \ldots, B_n$, $n \geq 1$ takové že

- jestliže $a - b \in L$, pak a, b náleží B_i pro nějaké i
- jestliže $a \to b$, pak $a \in B_i$, $b \in B_j$ a $i < j$

Množinou rodičů podmnožiny $C \subset V$ bude

$$pa_H(C) = \{a \in V;\, \exists b \in C,\, a \to b \,\text{náleží}\, E\}$$

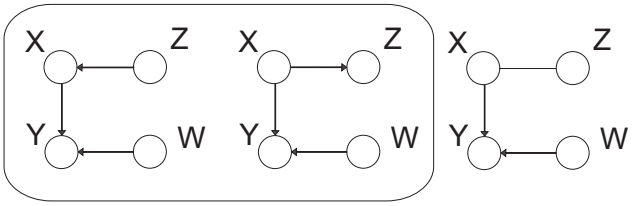
Vlajkou budeme rozumět podgraf tvaru $a \to b - c$, kde mezi a a c nevede hrana. Množina $K \subseteq V$ je úplná v hybridním grafu H, pokud $\forall a, b \in K, a \neq b$, neorientovaná hrana $a - b$ náleží do L.

3.3 Ekvivalentí třídy

Z předchozí kapitoly víme, že dva grafy BN jsou Markovovsky ekvivalentní (dále již jen ekvivalentní), jestliže mají stejné hrany (bez ohledu na orientaci) a stejnou množinu imoralit. Ke každé třídě ekvivalentních grafů sestrojíme reprezentanta.

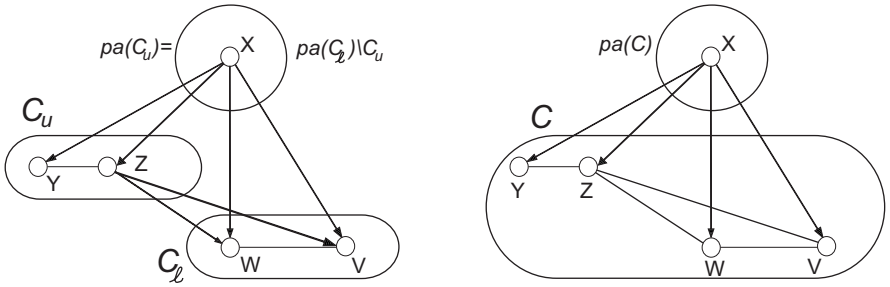
Definice 3.1. *Ekvivalentní třídu $\mathcal{G}$ v BN reprezentujeme jako esenciální graf G^*, který je hybridní a*

- *$a \to b$ náleží G^* právě když $a \to b$ náleží G pro všechna $G \in \mathcal{G}$*
- *$a - b$ náleží G^* právě když existují $G_1, G_2 \in \mathcal{G}$ takové že $a \to b \in G_1$ a zároveň $a \to b \in G_2$*



Obrázek 3.1: Příklad dvou různých ekvivalentních grafů a jejich esenciálního grafu

Pro algoritmus se nám bude hodit charakterizovat esenciální graf jako člena určité třídy řetězcových grafů, speciálně řetězcových grafů bez vlajek. To nám dá nové nástroje pro testování, zda je nalezený graf esenciální. Za zmínku stojí fakt, že esenciální graf není jediný způsob jak charakterizovat ekvivalentní třídy BN. Dalším je např. koncept „ Největšího řetězcového grafu “.



Obrázek 3.2: Příklad korektního sloučení

3.4 Vlastní algoritmus

Studeného algoritmus používá metodu korektního sloučení komponent, kterou si nyní popíšeme. Je složen ze sledu elementárních operací aplikovaných na graf s výsledkem ekvivalentního řetězcového grafu bez vlajek. Každý krok se skládá ze sloučení dvou komponent, první budeme říkat horní a značit C_u a druhou dolní C_l.

Definice 3.2. *Nechť H je řetězcový graf bez vlajek a C_u, C_l jsou dvě jeho komponenty pro které platí*

- $K \equiv \{pa_H(C_l) \cap C_u\}$ *je neprázdná úplná množina v H*
- $pa_H(C_u) = pa_H(C_l) \setminus C_u$

Jestliže všechny orientované hrany vedoucí z vrcholů z C_u do C_l jsou nahrazeny neorientovanými potom říkáme, že výsledný hybridní graf H' vzešel z H operací korektního sloučení z horní komponenty C_u a dolní komponenty C_l.

Věta 3.3. *Za předpokladu předchozí definice, sestrojený graf H' je řetězcový graf bez vlajek, který je ekvivalentní původnímu grafu H a má $C \equiv \{C_u \cup C_l\}$ jako komponentu. Důkaz viz.[5] str.8*

Graf, ke kterému budeme chtít nalézt esenciální graf, je DAG. Ten je acyklický a orientovaný, tudíž řetězcový, a neobsahuje žádné orientované hrany, proto je i bez vlajek. Předchozí věta nám tedy zaručuje v každém kroku algoritmu na vstupu řetězcový graf bez vlajek.

Věta 3.4. *Nechť H je řetězcový graf bez vlajek ekvivalentní BN. Potom H je esenciální graf právě když neexistují dvě jeho komponenty, které by mohly být korektně sloučeny. Důkaz viz.[5] str.11*

Definice 3.5. *Nechť H je hybridní graf nad V a $a \in V$. Rodičovskou komponentou pro $a \in V$ bude spojitá komponenta P z H taková, že*

- $pa_H(a) \cap P$ *je neprázdná úplná množina v* H
- $pa_H(P) = pa_H(a) \setminus P$

Dá se dokázat, že jestliže komponenta P existuje, je určena jednoznačně. Předchozí věty nám nyní dávají teoretický základ pro sestrojení vlastního algoritmu.

Algoritmus

Nechť G je BN nad V a $a_1, a_2, \ldots, a_n$ $n \geq 1$, je pořadí vrcholů, které je konzistentní se směrem orientovaných hran v grafu G. Sestrojme sekvenci hybridních grafů $G^1, G^2, \ldots, G^n$ následujícím způsobem:

- Položme $G^1 = G$
- pro $k = 2, \ldots, n$ otestujme zda existuje rodičovská komponenta pro a_k v G^{k-1}. Jestliže existuje, definujme G^k jako graf obdržený z G^{k-1} nahrazením všech orientovaných hran z $pa_{G^{k-1}}(a_k) \cap P$ do a_k neorientovanými, jinak položme $G^k = G^{k-1}$.

Algoritmus se zastaví na grafu G^n a ten je esenciální.

DAG jako vstupní graf nám zajistí že takovéto očíslování můžeme nalézt. Zároveň nám zajistí, že C_l v operaci slučování bude vždy jen jeden vrchol, neboť nahrazujeme hrany příslušející pouze níže očíslovaným vrcholům. Tudíž zbytek grafu zůstává orientovaný a má tedy jednoprvkové komponenty. Podrobný důkaz správnosti algoritmu naleznete v [5] str.14.

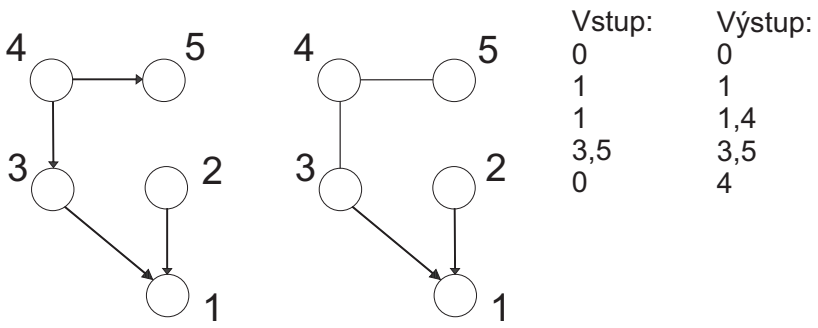
Kapitola 4

Dokumentace programu

Hlavním cílem této práce je implementovat Studeného algoritmus pro transformaci acyklického orientovaného grafu na esenciální. Program je přiložen ve dvou verzích. Zdrojový kód první z nich je napsán pouze za pomocí jazyka C a program je proto zkompilovatelý pro všechny operační systémy. Nemá ale uživatelské rozhraní. Druhá verze využívá komponent WinApi a je obohacena o uživatelské rozhraní pro operační systém Windows. Na CD jsou rovněž v adresáři bakalarka přiloženy elektronické verze této bakalářské práce ve formátech ps, pdf a tex. V adresáři literatura je ve formátu ps přiložena práce RNDr. Milana Studeného, DrSc., popisující implementovaný algoritmus podrobněji.

4.1 První verze

Název programu této verze je Egraf1.exe (zkompilováno pro Windows XP), zdrojového kódu Egraf1.cpp a obojí naleznete v adresáři program1. Zdrojový kód vlastního algoritmu (tedy ne celého programu) je v Příloze A. Vstup programu je z textového souboru, který má následující formát: Vrcholy DAGu si libovolně očíslujeme, a na i-tý řádek vypíšeme čísla vrcholů, do kterých vede z i-tého vrcholu orientovaná hrana. Čísla jednotlivých vrcholů oddělujeme čárkou. V případě že z vrcholu nevede žádná hrana, necháme řádek prázdný nebo do něj napíšeme 0. Výstup je ve stejném formátu, neorientované hrany jsou vyjádřeny jako dvě opačně orientované hrany. V případě že souboru zadáme cyklický graf, tuto chybu oznámí na výstupu. Programu můžeme jako parametry zadat jména vstupního a výstupního souboru. V případě že tak neučiníme, jako přednastavené hodnoty jsou určeny vstup.txt a vystup.txt. Maximální velikost vstupního DAGu je přednastavena na 100 vrcholů,



Obrázek 4.1: Vstupní DAG a jeho esenciální graf, jejich zápis ve formátu pro vstup a výstup z programu

ale lze změnit v závislosti na výpočetním výkonu počítače. To lze provést změnou hodnoty v makru ♯define VELIKOST na začátku zdrojového kódu. Algoritmus má kvadratickou časovou složitost.

4.2 Druhá verze

Druhou verzi naleznete na CD v adresáři program2 a jmenuje se Egraf.exe. Její zdrojový kód je rozdělen na dvě částí, main.c (obsahující hlavně uživatelské rozhraní) a algoritmus.h (obsahující vlastní algoritmus). Jsou spojeny do projektu Egraf.dev. Zdrojový kód algoritmu je téměř identický se zdrojovým kódem první verze, jsou zde pouze drobné rozdíly umožňující spouštění po jednotlivých krocích. Vstupní a výstupní soubor je odlišný od první verze, neboť musí uchovávat navíc informaci o poloze jednotlivých vrcholů. Maximální velikost vstupního grafu je 100 vrcholů nebo hran. Opět lze zvětšit.

Kliknutím na plochu levým tlačítkem myši získáte nový vrchol, nebo označíte již existující vrchol. Kliknutím pravým tlačítkem vedete hranu z aktivního vrcholu (označeného zeleným kolečkem) do vybraného vrcholu. Podržením levého tlačítka myši můžete přesouvat jednotlivé vrcholy. Tlačítko „Smazat bod“ smaže aktivní vrchol a všechny hrany z něj nebo do něj vedoucí. Tlačítko „Smazat hrany“ smaže všechny hrany vycházející z aktivního vrcholu. Tlačítkem „Transfor.“ program nalezne esenciální graf. Tlačítkem „Další krok“ tuto operaci povedete po jednotlivých krocích. Modré kolečko označuje vrchol, pro který bude operace korektního sloučení provedena v následujícím kroku. Během transformace po krocích již nemůžete graf upravovat, neboť v tomto případě již nelze jednoznačně určit jak pozměnit očíslování grafu.

Literatura

[1] Castilo, Hali, S.A., Gutierez, J.M.: *Expert Systems and probabilistic network models*, Springer 1996

[2] Korb, K.B., Nicholson, A. E.: *Bayesian Artificial Intelligence*, Chapman&Hall/CRC,New York, 2004.

[3] Lauritzen, S.L.: *Graphical Models*, Clarendon Press, Oxford 1996.

[4] Neapolitan, R. E.: *Learning Bayesian Networks*, Prentice Hall, New York, 2003.

[5] Studený, M.: *Characterization of essential graphs by means of the operation of legal merging of components*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems vol.**12**, str. 43–62, 2004.

Příloha A

Zdrojový kód

```
  int okolni(int vrchol,int k,int paa[VELIKOST+1][2])
{
if (paa[vrchol][1]!=0) return 0;
int i,j;
for (i=1;i<pocetvrcholu+1;i++)
     {
     paa[vrchol][1]=k;
     if (predchudci[paa[vrchol][0]][i]==0) break;
     if ((predchudci[paa[vrchol][0]][i]<0) &&
      (nalezi(abs(predchudci[paa[vrchol][0]][i]),paa)))
         {
         j=0;do j++; while (paa[j][0]!=(abs(predchudci[paa[vrchol][0]][i])));
         okolni(j,k,paa);
         }
     }
return 0;
}

  int main(int argc, char *argv[])
{
  int a,paa[VELIKOST+1][2],panP[VELIKOST+1][2],P[VELIKOST+1];
    int pocetkomponent,komponentaP,neplati;
    for (a=1;a<pocetvrcholu+1;a++)
        {
         for (i=0;i<VELIKOST+1;i++)
             {
             paa[i][0]=0;paa[i][1]=0;panP[i][0]=0;panP[i][1]=0;P[i]=0;
             }
```

```
 for (i=1;i<pocetvrcholu+1;i++)
     {
     if (predchudci[a][i]==0) break;
     paa[i][0]=predchudci[a][i];
     }
 /*timhle rozdelime paa na komponenty souvislosti*/
 pocetkomponent=0;
 for (i=1;i<pocetvrcholu+1;i++)
     {
     if  (paa[i][0]==0) break;
     if  ((paa[i][0]!=0) && (paa[i][1]==0))
         {
         pocetkomponent++;
         okolni(i,pocetkomponent,paa);
         }
     }
 /*nyni otestuju zda nektera komponenta paa splnuje podminku*/
 /*paP+P=paa*/
 /*dle proposition 2 je komponenta dana jedenoznacne*/
 for (i=1; i<pocetkomponent+1; i++)
      {
      k=0;m=0;
      for (j=1; j<VELIKOST+1;j++) P[j]=0;
      for (j=1; j<pocetvrcholu+1;j++)
          {
          if (paa[j][0]==0) break;
          if (paa[j][1]!=i)
             {
             m++;
             panP[m][0]=paa[j][0];
             }
          if (paa[j][1]==i)
             {
             k++;
             P[k]=paa[j][0];
             }
          }
/*uplnost*/
neplati=0;
for (j=1;j<pocetvrcholu+1;j++)
```

```
for (l=1;l<pocetvrcholu+1;l++)
    {
    if (P[j]==0) break;
    if (P[l]==0) break;
    if (P[j]==P[l]) continue;
    m=0;
    do
    {
        m++;
        if (predchudci[P[j]][m]==-P[l])
        {
            m=0;
            break;
        }
    }
    while (predchudci[P[j]][m]!=0);
    if (m!=0) neplati++;
    }
/*inkluze jednim smerem*/
      for (j=1; j<k+1;j++)
      for (l=1; l<pocetvrcholu+1;l++)
          {
          if (predchudci[P[j]][l]==0) break;
          if ((predchudci[P[j]][l])>0)
          if (!(nalezi(abs(predchudci[P[j]][l]),panP))) neplati++;
          for (m=1; m<pocetvrcholu+1; m++)
             {
             if (panP[m][0]==(abs(predchudci[P[j]][l]))) panP[m][1]=1;
             if (panP[m][0]==0) break;
             }
          }
/*inkluze druhym smerem*/
      for (j=1; j<m+1;j++)
      if ((panP[j][0]!=0) &&  (panP[j][1]==0))neplati++;
      if (neplati) komponentaP=0;
      if (!neplati) komponentaP=i;
      }
/*nyni premenim sipky na hrany*/
if (!neplati)
   {
```

```
k=0;m=0;
for (j=1; j<VELIKOST+1;j++) P[j]=0;
for (j=1; j<pocetvrcholu+1;j++)
   {
   if (paa[j][0]==0) break;
   if (paa[j][1]==komponentaP)
      {
      k++;
      P[k]=paa[j][0];
      }
   }
for (j=1;j<pocetvrcholu+1; j++)
    {
    if (P[j]==0) break;
    l=0;
    do l++; while (naslednici[P[j]][l]!=a);
    naslednici[P[j]][l]=-a;
    }
for (j=1; j<pocetvrcholu+1; j++)
    {
    l=0;
    if (P[j]==0) break;
    do l++; while (naslednici[a][l]!=0);
    naslednici[a][l]=-P[j];
    }
for (j=1; j<pocetvrcholu+1; j++)
    {
    l=0;
    if (P[j]==0) break;
    do l++; while (predchudci[P[j]][l]!=0);
    predchudci[P[j]][l]=-a;
    }
for (j=1;j<pocetvrcholu+1; j++)
    {
    if (P[j]==0) break;
    l=0;
    do l++; while (predchudci[a][l]!=P[j]);
    predchudci[a][l]=-P[j];
    }}}
```