INFORMATIKA

Nakreslení grafu minimálním počtem tahů

(Úlohy z MO – kategorie P, 14. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

V našem volném seriálu článků se zajímavými úlohami z Matematické olympiády kategorie P se po delší době vracíme opět k problematice grafových úloh. Různé grafové algoritmy se v úlohách MO kategorie P využívají poměrně často. Nejběžnější z nich (jako určení komponent souvislosti, nalezení nejkratší cesty v grafu apod.) jsou všeobecně známé a uvádí je každá učebnice programování. Budeme se proto raději věnovat úlohám, v jejichž řešení je třeba kromě standardních algoritmů použít ještě nějakou zajímavou myšlenku navíc. Jedna taková se objevila v praktické části celostátního kola 51. ročníku MO – kategorie P (školní rok 2001/2002). Původní zadání i vzorové řešení připravili organizátoři olympiády z Fakulty matematiky, fyziky a informatiky Univerzity Komenského v Bratislavě. Pro potřeby tohoto článku si zadání trochu zkrátíme a zjednodušíme, aniž bychom tím ovšem změnili smysl úlohy. Rovněž popis řešení vychází z původního slovenského originálu, je však zčásti přepracován a doplněn.

⋆⋆⋆

Kanalizační síť je tvořena N uzly očíslovanými od 1 do N, mezi kterými vede M trubek. Každá trubka spojuje dvojici uzlů a nemá žádné odbočky, ani se nikde nevětví. Mezi každou dvojicí uzlů vede nejvýše jedna trubka. Tuto kanalizační síť je třeba pročistit pomocí speciálních čisticích robotů.

Postup při čištění trubek je jednoduchý: Čisticí robot musí projít každou trubkou právě jednou; kdyby robot prošel již vyčištěnou trubkou, hrozilo by nebezpečí, že tuto trubku poškodí. Robot může do trubky vstoupit z jejího libovolného konce; pokud však již do trubky vstoupí, musí jí celou projít. Po skončení práce se může robot nacházet v jiném uzlu, než v jakém s čištěním začínal.

Za těchto podmínek se může stát, že jeden robot na vyčištění celé kanalizační sítě nebude stačit. Potřebujeme tedy do kanalizace poslat více robotů najednou a naprogramovat je tak, aby dohromady pročistili celou kanalizační síť. Čisticí robot ale není zrovna levný, a proto je potřeba navrhnout takový postup, aby celá kanalizace byla vyčištěna pomocí co nejmenšího počtu robotů.

Soutěžní úloha:

Vytvořte program, který pomůže naplánovat trasy pro čisticí roboty. Program načte popis kanalizační sítě, určí minimální počet robotů postačující pro vyčištění celé sítě a navrhne jejich trasy tak, aby každou trubkou prošel právě jeden z nich právě jednou. Jestliže existuje více řešení, váš program má za úkol nalézt a vypsat jedno libovolné z nich.

Formát vstupu:

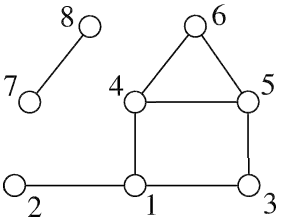
První řádek vstupního souboru ROURY.IN obsahuje dvě kladná celá čísla: počet uzlů N ($1 \leq N \leq 500$) a počet trubek M, oddělená mezerou. Každý z následujících M řádků popisuje jednu trubku; obsahuje vždy dvě čísla oddělená mezerou, což jsou čísla koncových uzlů trubky.

Formát výstupu:

Na prvním řádku výstupního souboru ROURY.OUT bude zapsáno jediné celé číslo R – nejmenší počet robotů, který postačuje k vyčištění celé kanalizační sítě. Následuje R řádků, z nichž i-tý popisuje trasu i-tého robota. Pokud i-tý robot začíná svou práci v uzlu a_1, odtud pokračuje trubkou do uzlu a_2, pak do a_3, atd., až skončí v uzlu ak, potom příslušný řádek výstupního souboru obsahuje k čísel a_1, a_2, ..., a_k (v tomto pořadí) oddělená od sebe vždy jednou mezerou.

Příklad:

Soubor ROURY.IN
8 8
1 2
1 3
1 4
3 5
4 5
4 6
5 6
7 8
Soubor ROURY.OUT
3
2 1 3 5 4 1
4 6 5
8 7



Obr. 1

⋆⋆⋆

Jistě jste se již někdy setkali s úlohami, kterým se říká jednotažky. Úkolem v nich je nakreslit zadaný čárový obrazec jedním tahem, aniž byste některou jeho čáru kreslili vícekrát. Pokud se na obrázek podíváme jako na nakreslení grafu, jedná se vlastně o grafovou úlohu nalézt v zadaném grafu tah obsahující všechny hrany grafu. Tahem rozumíme posloupnost vrcholů a hran v_0, e_1, v_1, e_2, …, v_{k-1}, e_k, v_k takovou, že hrana e_i spojuje vrcholy v_{i-1} a v_i a přitom všechny hrany obsažené v tahu jsou navzájem různé.

Naše úloha o kanalizační síti problém jednotažek trochu zobecňuje. Budeme-li chápat kanalizační síť jako graf, pak naším úkolem je nakreslit

tento graf co nejmenším počtem tahů. Přitom každá hrana grafu musí být obsažena pouze v jednom tahu, tzn. tahy jsou hranově disjunktní. Řečeno jinými slovy a ještě trochu přesněji, v zadaném grafu máme nalézt nejmenší množinu tahů takovou, že každá hrana grafu je obsažena v právě jednom z nich. Potřebný minimální počet tahů bude určovat nezbytný počet čisticích robotů a řešení problému zahájíme několika základními pozorováními. Pokud je graf nesouvislý, jistě můžeme úlohu řešit pro každou jeho komponentu souvislosti zvlášť. Nadále tedy stačí řešit úlohu jen pro souvislé grafy. Ve výsledném algoritmu nesmíme ovšem zapomenout nejprve určit komponenty souvislosti zadaného grafu (některým ze standardních grafových algoritmů) a určení minimálního počtu tahů poté provést zvlášť v každé komponentě.

Stupněm vrcholu grafu rozumíme počet hran, které z tohoto vrcholu vycházejí. Součet stupňů všech vrcholů grafu je sudý, neboť každá hrana tento součet zvýší o 2 (u každého ze svých dvou konců o 1). Každý graf proto musí mít sudý počet vrcholů lichého stupně. Předpokládejme tedy, že náš graf má $2k$ vrcholů lichého stupně pro nějaké vhodné nezáporné celé číslo k.

Máte-li zkušenosti s řešením jednotažek, jistě si teď uvědomíte, jak souvisejí stupně vrcholů grafu s možností nakreslit tento graf jedním tahem. Podmínkou je, aby buď graf měl všechny vrcholy sudého stupně (pak existuje dokonce tah, který začíná i končí na stejném místě), nebo aby měl přesně dva vrcholy lichého stupně (potom hledaný tah v jednom vrcholu lichého stupně začíná a ve druhém končí). Mnozí tato pravidla intuitivně znáte a používáte, my si ovšem nyní jejich platnost exaktně odvodíme a dokážeme a poté je využijeme i pro řešení naší původní úlohy.

Podíváme se nejprve na grafy, ve kterých mají všechny vrcholy sudý stupeň (tzn. $k = 0$). Dokážeme, že v nich vždy existuje uzavřený tah obsahující všechny hrany. Připomeňme si, že tah nazýváme uzavřený, pokud začíná a končí ve stejném vrcholu grafu. Uvažujme nejdelší tah v našem grafu, tj. ten, který obsahuje co nejvíce hran. Ukážeme nejprve, že tento tah musí být uzavřený. Potom dokážeme, že obsahuje všechny hrany grafu. Kdyby tah nebyl uzavřený, pak by z jeho posledního vrcholu vycházel lichý počet hran obsažených v našem tahu, ale protože stupeň každého vrcholu je sudý, vycházela by z něj i nějaká nepoužitá hrana, o kterou by bylo možné náš tah prodloužit. To je spor se skutečností, že jsme začínali s nejdelším tahem v grafu. Proto nejdelší tah je nutně uzavřený. Pokud tento tah neobsahuje všechny hrany, pak existuje vrchol w, ze kterého vychází jak

hrana obsažená v našem tahu, tak i hrana, která v našem tahu není (víme totiž, že graf je souvislý). Protože náš tah je uzavřený, můžeme předpokládat, že w je jeho první a zároveň poslední vrchol a tento tah lze potom prodloužit přidáním nepoužité hrany, která vede z vrcholu w. To je opět spor s předpokladem, že je náš tah nejdelší možný. Tím jsme dokázali, že nejdelší tah obsahuje všechny hrany grafu.

Vraťme se nyní ke grafům, které obsahují $2k$ vrcholů lichého stupně pro nějaké $k > 0$. V každém z těchto $2k$ vrcholů aspoň jeden z hledaných tahů začíná nebo končí. To opět snadno dokážeme sporem. Nechť w je vrchol lichého stupně, kde žádný tah nezačíná ani nekončí. Každý tah musí obsahovat sudý (třeba i nulový) počet hran vycházejících z vrcholu w a protože všechny tahy dohromady obsahují každou hranu grafu právě jednou, musel by být stupeň vrcholu w sudý, což je spor. Z toho plyne, že každé řešení musí obsahovat alespoň k tahů.

Ukážeme nyní naopak, že k tahů pro nakreslení grafu s $2k$ vrcholy lichého stupně postačuje. Do grafu přidáme k hran, které budou spojovat dvojice vrcholů lichého stupně. Takto vytvořený graf zůstane souvislý (když byl souvislý i původní graf) a všechny jeho vrcholy mají sudé stupně. Jak jsme dokázali dříve, existuje v něm tedy uzavřený tah, který obsahuje každou hranu právě jednou. Pokud z tohoto tahu vynecháme k přidaných hran, pak se tento tah rozpadne na k hranově disjunktních tahů, které dohromady obsahují všechny hrany původního grafu. Těchto k tahů je řešením úlohy.

Zbývá vyřešit otázku, jak nalézt v grafu, jehož všechny stupně vrcholů jsou sudé, uzavřený tah obsahující všechny hrany. Algoritmus spočívá v postupném přidávání hran do vytvářeného tahu. Vyjdeme z libovolného výchozího vrcholu a po dosud nepoužitých hranách budeme postupovat tak dlouho, dokud to bude možné, tzn. dokud nepřijdeme do vrcholu, z něhož už nevede žádná nepoužitá hrana. Protože stupně všech vrcholů jsou sudé, musí být tímto vrcholem opět původně zvolený výchozí vrchol. Tím jsme dostali nějaký uzavřený tah. Pokud na tomto tahu existuje vrchol, jehož všechny hrany jsme ještě nepoužili, zvolíme ho za nový výchozí vrchol a postupujeme z něj stejným způsobem. Dostaneme tak další uzavřený tah, který snadno spojíme s původním tahem do jednoho uzavřeného tahu. Stejným způsobem pokračujeme, dokud existuje nějaký vrchol s nepoužitou hranou. Jakmile takový vrchol neexistuje, máme nalezen požadovaný uzavřený tah obsahující všechny hrany.

Zamyslíme se ještě, jak popsaný algoritmus efektivně implementovat. Strukturu grafu si zaznamenáme pomocí seznamu hran, který si můžeme uložit v poli. Pro každou hranu si budeme evidovat nejen to, mezi kterými dvěma vrcholy vede, ale také zda už byla tato hrana použita ve vytvářeném tahu. Zvlášť si označíme ty hrany, které jsme do grafu přidali dodatečně. Pro každý vrchol předem sestrojíme seznam všech hran, které z něj vedou. Tyto seznamy mohou být různě dlouhé, takže bude vhodné reprezentovat je dynamicky formou lineárního spojového seznamu. Seznam hran vycházejících z vrcholu budeme postupně procházet a hrany budeme používat v tom pořadí, jak jsou uloženy v seznamu. Potřebujeme si tedy pro každý vrchol také pamatovat, kam až jsme v jeho seznamu hran došli (tzn. které jeho hrany jsou již jistě použité). Vytvářené tahy budeme reprezentovat jako seznamy hran, což nám umožní navazující tahy snadno spojovat dohromady.

Vlastní hledání a spojování tahů realizujeme nejlépe rekurzivní procedurou. Algoritmus nejdříve nalezne počáteční uzavřený tah. Vyjde přitom například z prvního vrcholu a bude postupovat po jednotlivých vrcholech grafu tak, že se v každém vrcholu vydá vždy po jeho první nepoužité hraně. Potom budeme procházet vrcholy na tomto tahu, dokud nenarazíme na první vrchol s nějakou dosud nepoužitou hranou. Pro tento vrchol zavoláme tutéž rekurzivní proceduru, která stejným způsobem nalezne z něj vycházející uzavřený tah a spojí ho s původním tahem. Při vhodné implementaci této rekurzivní procedury nebude již třeba pro vrcholy na přidávaném tahu kontrolovat, zda jsou všechny jejich hrany použity – zkontroluje to sama procedura, případně se přitom podle potřeby rekurzivně znovu zavolá. Výpočet končí ve chvíli, kdy na počátečním tahu nezbývá žádný vrchol s nepoužitou hranou. V tom okamžiku totiž již v celém grafu není žádný vrchol s nepoužitou hranou a máme tedy uzavřený tah obsahující všechny hrany grafu.

Časová a paměťová složitost popsané rekurzivní procedury sloužící k nalezení tahu je $O(N+M)$, kde N je počet vrcholů a M počet hran grafu – pro každý vrchol grafu musíme jednou projít seznam hran z něho vycházejících a tyto seznamy pro všechny vrcholy dohromady mají celkovou délku $O(M)$. Při řešení celé zadané úlohy musíme uvažovat ještě určení komponent souvislosti grafu, v každé komponentě doplnění hran mezi dvojicemi vrcholů lichého stupně a na závěr vypuštění těchto přidaných hran z nalezeného tahu. Komponenty souvislosti grafu dokážeme určit standardním algoritmem prohledávání grafu do hloubky nebo do šířky v čase $O(N+M)$.

Operace s dočasně přidávanými a odebíranými hranami mají celkovou časovou složitost $O(N)$, neboť v grafu může být nejvýše N vrcholů lichého stupně. Časová i paměťová složitost celého řešení je tedy $O(N+M)$.

Pokud byste se chtěli podívat na programovou realizaci popsaného řešení, navštivte na Internetu stránky věnované kategorii P Matematické olympiády. Najdete je na adrese http://mo.mff.cuni.cz/. Vedle aktuálních informací o soutěži je na těchto webových stránkách umístěn také archív soutěžních úloh a jejich vzorových řešení za posledních třináct let.

Dokazování a objevování vět v geometrii pomocí metod počítačové algebry

PAVEL PECH

Pedagogická fakulta JU, České Budějovice

(*Dokončení*)

4. Automatické objevování vět

V této kapitole se budeme věnovat automatickému objevování vět. Automatické objevování vět spočívá v hledání dodatečných předpokladů tak, aby se tvrzení, které obecně není pravdivé, stalo pravdivým. Podstatu automatického objevování si opět nejlépe objasníme na příkladě. Dokažte následující tvrzení:

Tvrzení A: *Uvažujme trojúhelník ABC. Označme K, L, M po řadě paty kolmic spuštěných z libovolného bodu P na strany BC, AC, AB trojúhelníka ABC. Potom jsou body K, L, M kolineární.*

Abychom dokázali dané tvrzení, zvolme systém souřadnic tak, že $A = [a, 0], B = [b, c], C = [0, 0], P = [p, q], K = [m, n], L = [k, 0], M = [r, s]$. Předpoklady jsou následující:

$PL \perp AC \Leftrightarrow p - k = 0$
$K \in BC \Leftrightarrow cm - bn = 0$
$PK \perp BC \Leftrightarrow (p - m)b + (q - n)c = 0$
$M \in AB \Leftrightarrow (b - a)s - (r - a)c = 0$
$PM \perp AB \Leftrightarrow (p - r)(b - a) + (q - s)c = 0.$

Závěr tvrzení je vyjádřen rovnicí

K, L, M jsou kolineární $\Leftrightarrow (m - k)s - (r - k)n = 0.$

V CoCoA napíšeme

```
Use R::= Q[abcpqkmnrs];
I:=Ideal(p-k,cm-bn,(p-m)b+(q-n)c,(b-a)s-(r-a)c,(b-a)(p-r)
   +(q-s)c);
NF((m-k)s-(r-k)n),I);
```

Odpověď je $kn - nr - ks + ms$, tedy normální forma není rovna nule a platnost tvrzení dokázána není (tvrzení není pravdivé na první pohled, zvolíme-li např. bod P v průsečíku výšek).

Nyní budeme hledat dodatečné podmínky, které je nutné přidat k předpokladům tak, aby se tvrzení stalo pravdivým. Přidáme tedy polynom závěru $(m - k)s - (r - k)n$ k polynomům ideálu I a budeme eliminovat závislé proměnné k, m, n, r, s, tj. eliminační ideál bude obsahovat pouze polynomy s proměnnými a, b, c, p, q. Napíšeme

```
Use R::= Q[abcpqkmnrs];
I:=Ideal(p-k,cm-bn,(p-m)b+(q-n)c,(b-a)s-(r-a)c,(b-a)(p-r)
   +(q-s)c,
(m-k)s-(r-k)n);
Elim(k..s,I);
```

Odpovědí je jediný polynom ve tvaru $ac^2(-acp+cp^2+abq-b^2q-c^2q+cq^2)$. Rovnice $a = 0$ nebo $c = 0$ znamenají, že vrcholy trojúhelníka leží na jedné přímce, což můžeme vyloučit. Zbývající vztah

$$-acp + cp^2 + abq - b^2q - c^2q + cq^2 = 0 \tag{16}$$

je podmínka pro to, aby bod P ležel na kružnici opsané trojůhelníku ABC. To můžeme nahlédnout z toho, že budeme-li uvažovat body A, B, C jako pevné a bod P jako proměnný, potom je rovnice (16) rovnicí kružnice opsané trojúhelníku ABC, jak snadno zjistíme dosazením.

Rovnice (16) by tedy měla být ona dodatečná podmínka, kterou je nutné přidat k předpokladům Tvrzení A, aby se toto tvrzení stalo pravdivým.

Dostáváme následující nové tvrzení:

Tvrzení B: *Uvažujme trojúhelník ABC. Označme K, L, M po řadě paty kolmic spuštěných z bodu P, který leží na kružnici opsané trojúhelníku ABC, na strany BC, AC, AB trojúhelníka ABC. Potom jsou body K, L, M kolineární.*

Provedeme důkaz Tvrzení B. Napíšeme

```
Use R::= Q[abcpqkmnrs];
I:=Ideal(p-k,cm-bn,(p-m)b+(q-n)c,(b-a)s-(r-a)c,(b-a)(p-r)
+(q-s)c, ac^2(-acp+ cp^2 + abq - b^2q - c^2q + cq^2));
NF((m-k)s-(r-k)n,I);
```

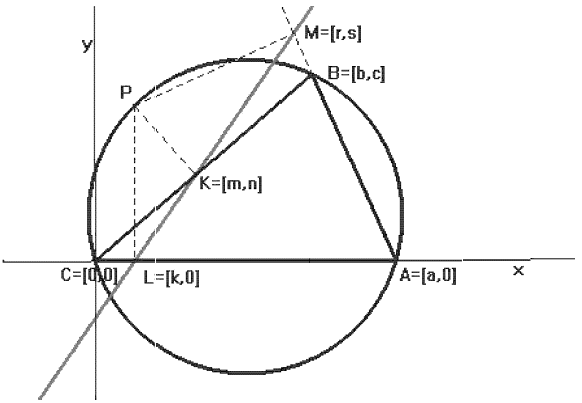
Odpověď je $kn - nr - ks + ms$, tedy Tvrzení B jsme nedokázali. Pokusme se zjistit příčinu. Předpokládejme, že platí dané předpoklady a zároveň neplatí závěr, podobně jako při důkazu sporem. Abychom vyjádřili negaci rovnice $(m-k)s-(r-k)n = 0$, zavedeme pomocnou proměnnou t a napíšeme $((m-k)s-(r-k)n)t-1 = 0$. Pokud by totiž platila původní rovnost $(m-k)s-(r-k)n = 0$, dostaneme $1 = 0$. V ideálu J, který obsahuje polynomy předpokladů a negaci závěru budeme eliminovat všechny závislé proměnné, tj. všechny proměnné kromě těch, pomocí kterých jsme zadali úlohu. Napíšeme

```
Use R::= Q[abcpqkmnrst];
J:=Ideal(p-k,cm-bn,(p-m)b+(q-n)c,(b-a)s-(r-a)c,(b-a)(p-r)
+(q-s)c, -acp + cp^2 + abq - b^2q - c^2q + cq^2,
((m-k)s-(r-k)n)t-1);
Elim(p..t,J);
```

Odpovědí je jediný polynom, který má po rozkladu na činitele tvar $(b^2+c^2)(a^2-2ab+b^2+c^2)$. Předpokládejme, že je tento polynom různý od nuly a přidejme tuto podmínku $(b^2+c^2)((a-b)^2+c^2)v-1 = 0$, kde v je další pomocná proměnná, do ideálu J. Nově vzniklý ideál označme K. Geometricky to znamená, že pro vrcholy trojúhelníka platí $B \neq C$ a $B \neq A$. Opět ověříme pravdivost tvrzení.

```
Use R::= Q[abcpqkmnrsvt];
K:=Ideal(p-k,cm-bn,(p-m)b+(q-n)c,(b-a)s-(r-a)c,(b-a)(p-r)
+(q-s)c,-acp+cp^2+abq-b^2q-c^2q+cq^2,
(b^2+c^2)((a-b)^2+c^2)v-1);
NF((m-k)s-(r-k)n,K);
```

Výsledek 0 říká, že Tvrzení B je pravdivé, za předpokladu, že $B \neq C$ a $B \neq A$. Těmto podmínkám říkáme podmínky nedegenerace. Obvykle se tyto podmínky do zadání úloh elementární geometrie nedávají. Přirozeně se předpokládá, že trojúhelník je skutečný trojúhelník a ne úsečka či trojice bodů v přímce, tři splývající body apod. Můžeme tedy říci, že Tvrzení B je pravdivé. Mnozí čtenáři jistě poznali v Tvrzení B známou Wallace-Simsonovu větu [5], kterou jsme takto „objevili“, obr. 8.



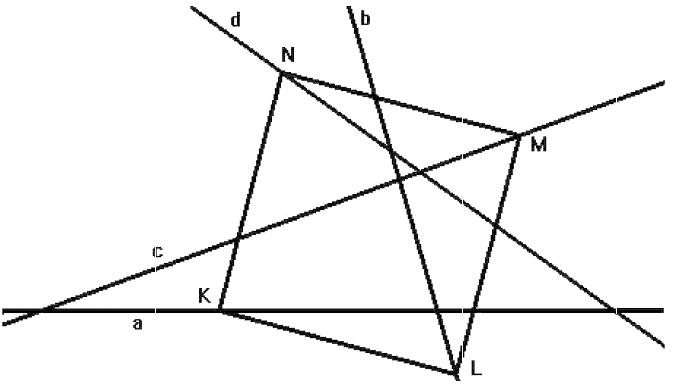
Obr. 8 Paty kolmic K, L, M spuštěných z bodu P kružnice trojúhelníku opsané na strany trojúhelníka ABC leží na přímce

5. Řešení neelementárních konstrukčních úloh

V této části podáme příklad neelementární konstrukce, kterou není snadné sestrojit pomocí kružítka a pravítka. Jedná se o následující úlohu, viz [8].

V rovině jsou dány přímky a, b, c, d. Sestrojte čtverec $KLMN$, jehož vrcholy leží po řadě na daných přímkách a, b, c, d.

Zvolme systém souřadnic tak, že $K = [k_1, k_2]$, $L = [l_1, l_2]$, $M = [m_1, m_2]$, $N = [n_1, n_2]$ a nechť přímky a, b, c, d mají rovnice $a : a_1x + a_2y + a_3 = 0$, $b : b_1x + b_2y + b_3 = 0$, $c : c_1x + c_2y + c_3 = 0$, $d : d_1x + d_2y + d_3 = 0$. Předpokládejme že $K \in a$, $L \in b$, $M \in c$, a $N \in d$. Otočíme vektor $L - K$ o 90^0 v kladném smyslu okolo K a dostaneme vektor $N - K$ (značíme $rot(L - K) = N - K$). Potom otočíme vektor $K - N$ o 90^0 v témž smyslu okolo N a dostaneme vektor $M - N$ a tak dále, viz obr. 9.



Obr. 9 Čtverec $KLMN$, jehož vrcholy leží na daných přímkách a, b, c, d

Dostaneme následující relace:

$K \in a \Leftrightarrow a_1k_1 + a_2k_2 + a_3 = 0$, $L \in b \Leftrightarrow b_1l_1 + b_2l_2 + b_3 = 0$, $M \in c \Leftrightarrow c_1m_1 + c_2m_2 + c_3 = 0$, $N \in d \Leftrightarrow d_1n_1 + d_2n_2 + d_3 = 0$, $\mathrm{rot}(L - K) = N - K \Leftrightarrow -(l_2 - k_2) = n_1 - k_1$, $l_1 - k_1 = n_2 - k_2$, $\mathrm{rot}(K - N) = M - N \Leftrightarrow -(k_2 - n_2) = m_1 - n_1$, $k_1 - n_1 = m_2 - n_2$, $\mathrm{rot}(N - M) = L - M \Leftrightarrow -(n_2 - m_2) = l_1 - m_1$, $n_1 - m_1 = l_2 - m_2$, $\mathrm{rot}(M - L) = K - L \Leftrightarrow -(m_2 - l_2) = k_1 - l_1$, $m_1 - l_1 = k_2 - l_2$.

Máme systém dvanácti rovnic, který budeme řešit vzhledem k osmi neznámým k_1, k_2, l_1, l_2, m_1, m_2, n_1, n_2. Např. pro k_1 dostaneme

$$k_1 = \frac{b_2c_1 + b_2c_2 + c_1d_1 + c_2d_1 - b_2d_2 - c_1d_2 + c_2d_2 + c_1 - c_2 - d_1}{b_2c_1d_1 + b_2c_2d_1 - b_2c_1d_2 + c_1d_1 - c_1d_2 + c_2d_2}.$$

Podobně najdeme ostatní neznámé. Podívejme se na jedno z řešení $KLMN$ na obrázku, který byl sestrojený s pomocí interaktivního software Cabri II. Z konstrukce vidíme, že existují nejvýše čtyři čtverce daných vlastností.

6. Závěr

V tomto článku jsme se seznámili se základy teorie automatického dokazování, automatického odvozování a automatického objevování vět elementární (i neelementární) geometrie. Na příkladech jsme poznali podstatu této teorie, která zvláště v posledních letech nabývá na významu. Zvyšující se výkonnost počítačů, jakož i objevy nových algoritmů umožňují tento nebývalý pokrok.

V případě, že klasická – syntetická metoda selhává, můžeme použít k řešení problému některou z naznačených počítačových metod. Zvláště v kombinaci s dynamickým software (např. Cabri II nebo Cabri II plus) můžeme dostat velmi zajímavé a působivé výsledky.

Autor článku děkuje dr. Antonínu Vrbovi za řadu podnětných připomínek a návrhů.

Literatura

[1] *Bartsch, H. J.:* Matematické vzorce. SNTL, Praha 1983.
[2] *Berger, M.:* Geometry I. Springer-Verlag, Berlin – Heidelberg 1987.
[3] *Bottema, O.:* Hoofstukken uit de elementaire meetkunde. Servire, Den Haag, 1944, Epsilon, Utrecht 1987.
[4] *Cox, D. – Little, J. – O'Shea, D.:* Ideals, Varieties, and Algorithms. Second Edition, Springer 1997.
[5] *Coxeter, H. S. M. – Greitzer, S. L.:* Geometry revisited. Toronto – New York 1967.
[6] *Dörrie, B. H.:* Triumph der Mathematik. Breslau 1933.
[7] *Hora, J. – Pech, P.:* Using computer to discover some theorems in geometry. Acta Acad. Paed. Agriensis 29 (2002), 67-75.
[8] *Karger, A.:* Classical Geometry and Computers. Journal for Geometry and Graphics **2** (1998), 7-15.
[9] *Staudt, Ch. R.:* Über die Inhalte der Polygone und Polyeder. Journal für die reine und angewandte Mathematik **24** (1842), 252-256.
[10] *Wang, D.:* Gröbner Bases Applied to Geometric Theorem Proving and Discovering. In: Gröbner bases and applications. B. Buchberger, F. Winkler (eds), Lecture Notes of Computer Algebra, Cambridge Univ. Press, Cambridge, 1998, 281-301.
[11] *Pech, P:* Klasické vs. počítačové metody při řešení úloh v geometrii. Jihočeská univerzita, Č. Budějovice 2005.