Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Výpočtová zložitosť hry Net

(diplomová práca)

Marek Zeman

Vedúci: RNDr. Michal Forišek PhD.

Bratislava, 2010

Čestne prehlasujem, že som túto prácu vypracoval samostatne s použitím citovaných zdrojov.

. . . . . .

Obsah

1 Úvod **1**

2 Trieda NP a prehľad problematiky **3**
2.1 Plumber 6
2.2 Perly 8

3 Hra Net a jej variácie **11**
3.1 Hra Net 11
3.2 Existencia riešenia hry Net 14
3.3 Variácie hry 15

4 Výpočtová zložitosť hry Net a jej variácií **17**
4.1 Príslušnosť do NP 17
4.2 Pomalé metódy riešenia 19
4.3 Analýza problému KITJP-Net 28
4.4 Zložitosť hry Net pre niektoré množiny potrubí 37

Kapitola 1

Úvod

Táto práca je venovaná hre, ktorú možno nájsť na internete pod menom Net. Je to pútavá a zaujímavá logická hra, ktorá má množstvo variácií. Predpokladáme, že koncept hry je známy už dlho, avšak jej označenie Net, ktoré používame v tejto práci, preberáme od autora jednej z jej implementácií[1]. Keďže toto pomenovanie pravdepodobne nebude mnohým čitateľom známe, odporúčame sa s hrou zoznámiť v príslušnej kapitole tejto práce.

Hra sa dá jednoducho matematicky definovať, rovnako ako aj niekoľko problémov, ktoré s ňou súvisia. Základným problémom prirodzene vyvstala otázka, či má daný počiatočný stav riešenie, prípadne nejaké nájsť. Cieľom práce je klasifikovať tento problém z hľadiska výpočtovej zložitosti. Rovnako sa venujeme aj modifikovaným problémom pre rôzne variácie hry, pomocou ktorých sa pokúšame ozrejmiť, ktoré miesta sú najťažšie a kde musíme problém modifikovať, aby sme si ho zľahčili.

Od začiatku panovalo podozrenie, že pre niektoré verzie hry je otázka o existencii riešenia NP-úplná. Tento záver sa bohužiaľ nepodarilo dokázať, avšak ani vyvrátiť. Existuje veľa podobných hier, ktorých riešenie je NP-úplné a stále očakávame, že sa medzi ne zaradí aj hra Net. Problém $P = NP$ je v posledných desaťročiach snáď najvýznamnejším otvoreným problémom v teoretickej informatike. Clay Mathematics Institute ho zaradil na prvé miesto medzi Millenium Prize Problems a tomu, kto ho vyrieši, sľúbil milión dolárov[2]. Pri prevládajúcom pesimistickom pohľade na rovnosť uvedených tried by bol prínosom uvedného dôkazu fakt, že široká hráčska verejnosť prestane hľadať polynomiálne algoritmy na riešenie hry Net. V práci uvádzame aj niekoľko algoritmov, ktoré riešia zjednodušené verzie tejto hry.

Kapitola 2

Trieda NP a prehľad problematiky

V tejto kapitole si zhrnieme poznatky, ktoré budú užitočné pri našich úvahách o hre Net. Pripomenieme si triedu NP a niektoré problémy, ktoré do nej patria. Okrem toho si spomenieme niektoré známe hry a výsledky klasifikácie problémov s nimi súvisiacich. Fakt, že tieto problémy sú NP-úplné, viedol od začiatku k podozreniu, že by mohol platiť podobný výsledok aj pri hre Net.

V ďalšom budeme používať štandardnú notáciu z teórie formálnych jazykov, ktorú si môže čitateľ pripomenúť napríklad v [3]. Na úvod stručne zopakujme niekoľko definícii.

Definícia 1. *$TIME(f(n))$ je množina jazykov, ktoré sa dajú rozpoznávať deterministickým Turingovym strojom v čase $f(n)$ bez obmedzenia pamäťovej zložitosti.*

Definícia 2. *$NTIME(f(n))$ je množina jazykov, ktoré sa dajú rozpoznávať nedeterministickým Turingovym strojom v čase $f(n)$ bez obmedzenia pamäťovej zložitosti.*

Akceptáciu v časovom obmedzení chápeme ako obmedzenie na dĺžku výpočtu.

Definícia 3. $P = \cup_{k\in\mathbb{N}} TIME(n^k)$.

Definícia 4. $NP = \cup_{k\in\mathbb{N}} NTIME(n^k)$.

Definícia 5. *Nech L_1 a L_2 sú jazyky. Nech A je deterministický Turingov stroj, ktorý prekladá jazyk L_1 na jazyk L_2. Ak pre každé slovo x platí, že $x \in L_1 \iff A(x) \in L_2$, potom A je many-to-one redukciou jazyka L_1 na jazyk L_2. Ak A pracuje v polynomiálnom čase, hovoríme, že L_1 je polynomiálne many-to-one redukovateľný na L_2.*

Definícia 6. *Jazyk L nazývame NP-úplný, ak platí:*

1. *$L \in NP$*
2. *Pre každý jazyk L_1 z NP platí, že L_1 je polynomiálne many-to-one redukovateľný na L.*

Na NP-úplné problémy sa teda dá pozerať ako na najťažšiu množinou problémov v rámci triedy NP – riešenie ľubovoľného z nich vieme využiť na riešenie každého problému v NP, samozrejme s potrebou polynomiálneho času na transformáciu. Špeciálne, ľubovoľné riešenie NP-úplného problému v deterministickom polynomiálnom čase by znamenalo, že na každý problém v triede NP existuje deterministický polynomiálny algoritmus, čím by prišlo k stotožneniu tried P a NP. Takýto algoritmus sa ale doteraz nenašiel a na jeho existenciu panuje prevažne pesimistký názor.

Poznamenávame, že vo zvyšku práce sa nám budú pojmy jazyk a problém prekrývať. Pre každý jazyk totiž možeme formulovať problém zisťovania príslušnosti do tohto jazyka. Voľnejšie povedané, ak sú slová z jazyka objekty s nejakou vlastnosťou, potom problém jazyka spočíva v zisťovaní, či má daný objekt požadovanú vlastnosť a teda či patrí do daného jazyka.

Zaujímavým problémom je hľadanie okružnej jazdy mestom. Máme daný neorientovaný graf mesta, pozostávajúci z križovatiek a ulíc medzi nimi, a stojíme na jednej z križovatiek. Chceli by sme spraviť okružnú prechádzku mestom, pri ktorej by sme prešli všetkými križovatkami, žiadnou dva alebo viac krát a vrátili sa na miesto štartu.

Definícia 7. *Jazyk* HAM *je jazyk tých grafov, ktoré obsahujú cyklus, ktorý prechádza každým vrcholom grafu práve raz.*

Cyklus, ktorý prechádza každým vrcholom práve raz a vráti sa do počiatočného, sa nazýva hamiltonovský okruh. Grafom, ktoré obsahujú hamiltonovský okruh, sa niekedy hovorí skrátene hamiltonovské.

Veta 2.0.1. *Jazyk* HAM *je NP-uplný.*

Otázka zistenia existencie Hamiltonovského okruhu zostáva obtiažna aj pre niektoré podtriedy grafov. Pripomeňme si, že grafu, ktorého každý vrchol má stupeň tri hovoríme kubický a grafu, ku ktorému existuje nejaké jeho nakreslenie do roviny hovoríme planárny. Ak sa vrcholy grafu dajú rozdeliť na dve časti A, B tak, aby všetky hrany spájajú vrchol z A a vrchol z B, potom je tento graf bipartitný. Časti A a B nazývame partície.

Definícia 8. *Jazyk* HC3P *je jazyk kubických planárnych grafov, ktoré majú hamiltonovský okruh.*

Veta 2.0.2. HC3P *je NP-úplný jazyk.*

Dôkaz tejto vety vyplýva z dôkazu nasledujúcej vety a z faktu, že HC3P patrí do NP. Uveďme si ešte jedno obmedzenie navyše – požadujme, aby graf bol aj bipartitný.

Definícia 9. *Jazyk* HCB3P *je jazyk bipartitných planárnych kubických grafov.*

Veta 2.0.3. *Jazyk* HCB3P *je NP-úplný.*

Dôkaz tejto vety môže čitateľ nájsť v [10],[11]. Aj veľa ďalších modifikácii jazyka HAM si zachováva NP-úplnosť. Spomeňme ešte iné NP-úplné jazyky. Jedným z najznámejších a najstarších problémov je problém splniteľnosti booleovskej formuly. Formula je splniteľná, ak existuje taká ohodnotenie vstupných premenných, pri ktorom je formula pravdivá.

Vo zvyšku tejto kapitoly budeme uvažovať len formuly zapísané pomocou operácii NOT, AND a OR. Toto obmedzenie nie je príliš významné, pretože každá formula sa dá napísať v tvare, v ktorom sa používajú len uvedené operácie. Navyše budeme predpokladať, že formuly su v konjunktívnom normálnom tvare. Pripomeňme si, že premennú alebo jej negáciu nazývame literál a disjunkciu literálov nazývame klauzula. Formula v konjuktívnom normálnom tvare je konjunkcia klauzúl.

Definícia 10. *Jazyk* SAT *je jazyk splniteľných booleovských formúl.*

Tvrdenie o NP-úplnosti jazyka SAT je predmetom Cook-Levinovej vety. Čitateľ môže dôkaz nájsť napríklad v [4]. Aj pri mnohých modifikáciach a obmedzeniach na uvažované formuly zostáva obtiažnosť problému rovnaká. Spomenieme jednu, ktorú budeme potrebovať.

Definícia 11. *Nech ϕ je nejaká booleovská formula. Uvažujme nasledovný graf: pre každú premennú, ktorý sa vo ϕ vyskytuje (či už v negatívnom alebo pozitívnom kontexte) vytvoríme v grafe vrchol. Pre každú klauzulu vo ϕ vytvoríme vrchol. Hranou spojíme vrchol klauzuly a premennej vtedy, ak sa príslušná premenná vyskytuje v príslušnej klauzule. Okrem toho pridáme ešte cyklus, prechádzajúci cez všetky vrcholy prislúchajúce premenným. V prípade, že je tento graf planárny, hovoríme, že ϕ je planárna formula.*

Definícia 12. *Jazyk* 3SAT *je jazyk splniteľných formúl v konjuktívnom normálnom tvare, pre ktoré platí, že každá klauzula obsahuje práve tri literály. Jazyk* (1,3)SAT *je jazyk formúl v konjuktívnom normálnom tvare, v ktorom každá klauzula obsahuje práve tri literály a ku ktorému existuje také ohodnotenie vstupných premenných, že v každej klauzule je práve jeden literál pravdivý.* Planar (1,3)-SAT *je podmnožina jazyka* (1,3)SAT*, ktorá obsahuje len formuly, ktoré sú planárne.*

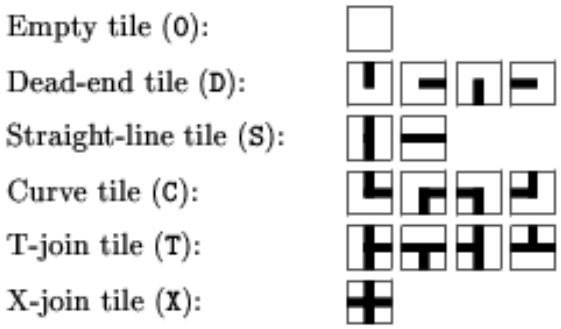
Veta 2.0.4. Planar (1,3)-SAT *je NP-úplný jazyk.*

Pre dôkaz tejto vety odkazujeme čitateľa na [5],[6],[7].

2.1 Plumber

V tejto časti si priblížime známu hru pre jedného hráča pod menom Plumber. Hra je štandardnou súčasťou niektorých linuxových distribúcii. V hre Plumber dostane používateľ mriežku veľkosti $N \times M$. V každom políčku mriežky sa nachádza jedno zo šiestich typov potrubí (viď obrázok 2.1). Ťahy hráča spočívajú v otáčaní potrubií o 90 stupňov. Na začiatku sú políčka otočené náhodne.

Pod susednými políčkami budeme rozumieť políčka, ktoré zdieľajú hrany. Každé políčko má teda najviac štyroch susedov. Ak susedia oproti sebe pri natočení nasmerujú potrubia, môže sa medzi nimi šíriť voda. V prípade, že nejaké políčko by nasmerovalo vodu k susedovi, kde by nebolo potrubie, ktoré by ju prijalo, potom by voda by mohla vytiecť von, čo by bolo nepríjemné. Riešenie danej inštancie hry Plumber je teda natočenie, v ktorom sú všetky dvojice susedných políčok orientované tak, že alebo majú oproti sebe nasmerované potrubia, alebo nemajú. Cieľ hráča je natáčaním políčok nájsť riešenie. Prirodzeným problémom je pre danú mriežku zistiť, či nejaké riešenie existuje.



Obr. 2.1: Typy potrubí v hre Plumber.

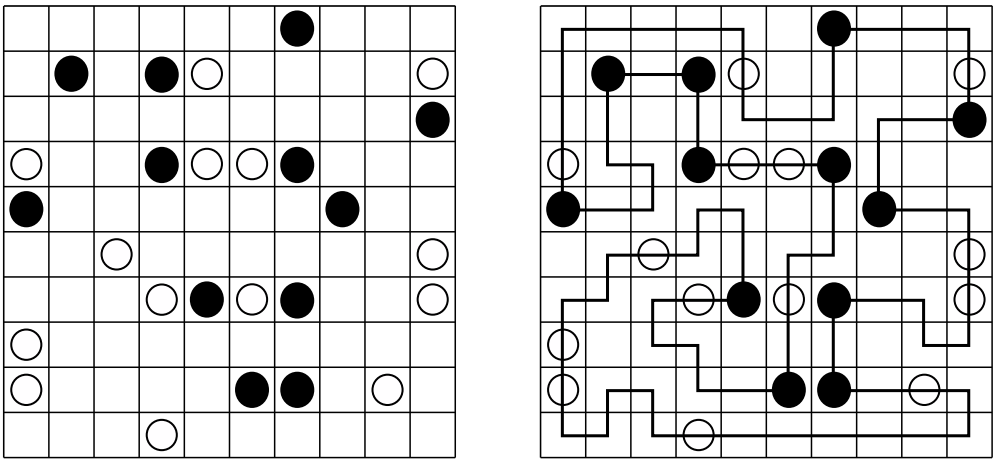
Keď uvažujeme rôzne množiny potrubí, ktoré sa môžu v mriežke vyskytnúť (spomedzi tých na obrázku 2.1), potom problém zisťovania existencie riešenia nadobúda rôzne obtiažnosti. Každému z typov potrubí sa dá na označenie priradiť písmeno podľa obrázka 2.1. Nech $S \subseteq \{O, D, S, C, T, X\}$. Potom S-Plumber označuje jazyk mriežok, ku ktorým existuje riešenie a v ktorých sa vyskytujú iba potrubia typov z S.

Autori v [9] analyzujú problém zisťovania existencie riešenia pre rôzne množiny S. Výsledky sú zhrnuté v tabuľke 2.1. Ľahko by sa ukázalo, že pre každé S je jazyk S-Plumber v NP.

V tabuľke 2.1 označuje $\surd$ prítomnosť v mriežke, $\times$ označuje neprítomnosť a $\star$, že to nie je dôležité. Zrejme najzaujímavejsie výsledky sú pre množiny potrubí, ktoré sú NP-úplné. Stačí, aby boli v mriežke zákruty, rovné potrubia a slepé konce a problém sa stáva veľmi obtiažnym. Obdobne stačí zobrať mriežku, v ktorej sa nachádzajú rovné potrubia, križovatky typu T

Tabuľka 2.1: Tabuľka obtiažností hry Plumber

O	C	D	S	T	X	Zložitosť
$\star$	$\star$	$\star$	$\times$	$\star$	$\star$	Polynomiálna
$\star$	$\star$	$\times$	$\surd$	$\times$	$\star$	Polynomiálna
$\star$	$\surd$	$\surd$	$\surd$	$\star$	$\star$	NP-úplné
$\star$	$\surd$	$\star$	$\surd$	$\surd$	$\star$	NP-úplné
$\star$	$\times$	$\times$	$\surd$	$\star$	$\star$	Polynomiálna
$\star$	$\times$	$\surd$	$\surd$	$\star$	$\star$	Neznáma



Obr. 2.2: Príklad hry Perly s riešením.

a zákruty. Pri dôkazoch NP-úplnosti sa využila redukcia problému Planar (1-3)-SAT na problém OSCDTX-Plumber. Rovnako zaujímavé sú aj niektoré výsledky o polynomiálnej časovej zložitosti - stačí odobrať rovné potrubie a problém zisťovania existencie riešenia sa stáva redukovateľným na problém hľadania perfektného párenia v bipartitnom grafe, ktorého veľkosť je rádovo veľkosť mriežky. Dôkazy všetkých tvrdení sa nachádzajú v [9].

Hra Plumber je, ako sa neskôr dozvieme, dosť podobná hre Net. V prípade hry Net ale riešime ťažší problém, v ktorom je riešenie definované striktnejšími podmienkami. Preto, bohužiaľ, nebudeme môcť túto zaujímavú redukciu priamo použiť pri riešení hry Net.

2.2 Perly

Ďalšou zaujímavou hrou, ktorú si spomenieme, je hra Perly. Anglický názov tejto hry je Pearls. V tejto pôvodom japonskej hre dostane hráč mriežku, na ktorej sú niektoré políčka prázdne a na niektorých sú perly, ktoré môžu byť biele alebo čierne. Úlohou hráča je nakresliť cez políčka uzavretú čiaru. Pre ňu musí platiť:

1. Neprekrižuje sa

2. Prechádza všetkými perlami

3. Zahýba v každej čiernej perle, avšak nie bezprostredne pred ňou ani za ňou

4. Nezahýba v žiadnej bielej perle, avšak zahýba bezprostredne pred ňou alebo za ňou

Podobne ako pri ostatných hrách, aj v tejto sa ponúka otázka zisťovania existencie riešenia. Podarilo sa ukázať, že z hľadiska časovej zložitosti je tento problém NP-úplný[8]. Na demonštráciu tohto faktu autor zredukoval problém existencie hamiltonovského okruhu v kubických planárnych grafoch na problém existencie riešenia. V konštrukcii sa graf najprv nakreslí do roviny tak, aby jeho hrany pozostávali len z vertikálnych a horizontálnych úsečiek. Potom sa k tomuto nakresleniu skonštruuje inštancia hry Perly tak, že hrany a obdĺžnikové oblasti sa nahrádzajú zhlukmi bielych perál. Čierne perly sa v konštrukcii nikde nevyužívajú.

Kapitola 3

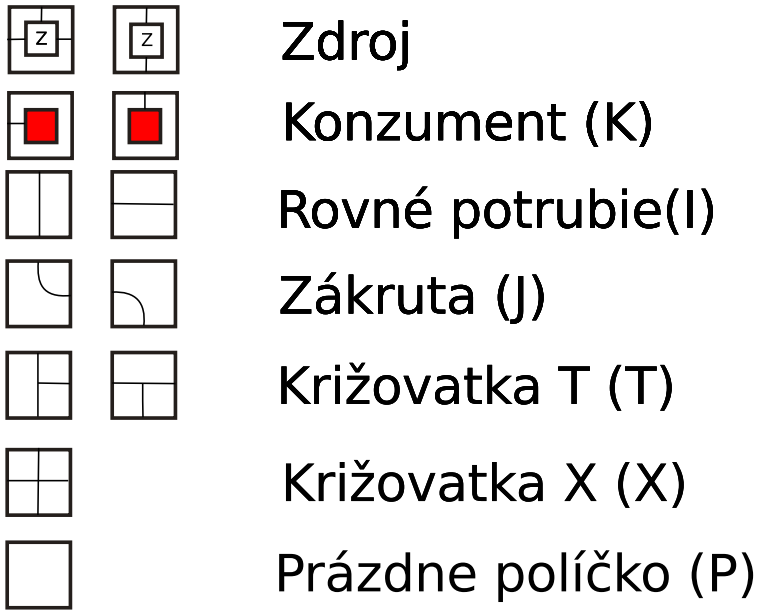
Hra Net a jej variácie

V tejto kapitole si popíšeme, ako hra Net vyzerá a aký je cieľ hráča. Skúsime spomenúť niektoré z veľkého množstva variácií, ktoré pre túto hru existujú. Potom si formulujeme otázky, ktoré s hrou súvisia a ktorých výpočtovú zložitosť budeme skúmať.

3.1 Hra Net

V hre Net sa vyskytuje mriežka obdĺžnikovej veľkosti. Na každom zo štvorcov mriežky sa nachádza potrubie, ktoré môže byť niekoľkých typov (pozri obrázok 3.1). V tejto práci uvažujeme, že políčko typu zdroj sa v mriežke nachádza vždy práve raz. Čitateľ sa neskôr môže sám presvedčiť, že tento predpoklad nie je vôbec obmedzujúci a všetky tvrdenia a postupy by sa dali ekvivalentne upraviť pre viacero zdrojov. Zdroj produkuje vodu, ktorá vyteká z jeho otvorených potrubí. Ich počet môže byť medzi 1 až 4 vrátane a môžu byť nasmerované na ľubovoľné strany. Konzument má vždy jedno otvorené potrubie, ktorým vodu prijíma a spotrebováva. Susedstvo budeme chápať tak, aby medzi susedmi mohla prúdiť voda. Preto za susedov budeme považovať políčka, ktoré s daným políčkom zdieľajú hranu. Potrubie vnútri mriežky má teda štyroch susedov, políčka na kraji mriežky majú troch susedov a štyri políčka v rohoch majú dvoch susedov.

Hráč na začiatku teda dostane mriežku s potrubiami. Jeho ťahy spočívajú v otáčaní potrubí na jednotlivých políčkach o 90 stupňov. Takto môže docieliť, že voda sa bude pri vhodnom natočení potrubí šíriť políčkami. Cieľom hry je dosiahnuť, aby všetky políčka boli zásobované vodou. Na obrázku 3.2



Obr. 3.1: Typy potrubí v hre Net

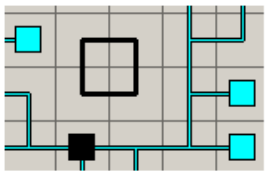
je príklad nesprávneho natočenia. Teraz si presnejšie zadefinujeme pojmy, s ktorými budeme pracovať.

Definícia 13. *Inštancia hry pre dané rozmery je matica znakov, ktoré popisujú, aký typ potrubia sa na jednotlivých políčkach nachádza.*

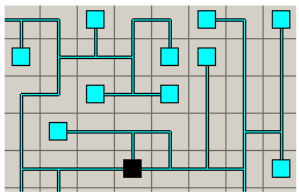
Definícia 14. *Stavom hry budeme nazývať informáciu, ktorá hovorí, ako je ktoré políčko otočené.*

Stav hry budeme niekedy v rovnakom význame označovať aj slovom natočenie. Stav si môžeme reprezentovať napríklad ako maticu veľkosti mriežky. Keďže pre typ potrubia sú najviac štyri rôzne otočenia, každý prvok tejto matice môže nadobúdať hodnoty 0 až 3. Počet všetkých stavov pre danú inštanciu veľkosti $N \times M$ je presne 4^{NM}, avšak niektoré sa môžu zhodovať, pretože napríklad rovné potrubie má v skutočnosti len dve rôzne možnosti orientácie. Pre formuláciu ďalšej požiadavky si definujeme nasledujúci intuitívny pojem.

Definícia 15. *Natočenie dvoch susedných políčok je konzistentné, ak platí, že k ich spoločnej hrane obidve alebo majú nasmerované potrubie, alebo obidve*



Obr. 3.2: Príklad časti neprípustného stavu (sieť nie je súvislá)



Obr. 3.3: Príklad časti neprípustného stavu (v sieti sa vyskytuje cyklus)

nemajú. Pre daný stav nazveme jedno políčko konzistentné, ak pre všetkých jeho susedov platí, že sú s ním konzistentné. Konzistentný stav je taký, v ktorom sú všetky políčka konzistentné.

V konzistentnom natočení teda všetka voda v sieti preteká potrubiami a žiadna sa z nich nevylieva – nenastáva situácia, že by nejaké políčko nasmerovalo potrubie k susedovi a ten nenasmeroval potrubie oproti. Okrem požiadavky konzistencie je na vyriešenie tiež vyžadované, aby do každého políčka pritekala voda práve jedným smerom. Teraz uvedieme definíciu riešenia inštancie.

Definícia 16. *Koncový stav je taký stav danej inštancie, v ktorom sú potrubia natočené tak, aby voda tiekla zo zdroja do všetkých neprázdnych políčok mriežky, stav je konzistentný a do žiadneho políčka voda nepriteká z dvoch rôznych strán.*

V prípade, že nájdeme koncový stav, inštanciu sme úspešne vyriešili. Stav, ktorého časť je na obrázku 3.3, nie je koncový. Všimnime si, že požiadavka na zásobovanie každého políčka práve z jednej strany sa dá ekvivalentne formulovať ako požiadavka, aby šírenie vody v sieti malo stromovú štruktúru. V pomyselnom strome by bol koreň zdroj, a pre každé políčko by platilo, že

prijme vodu potrubím smerujúcim k zdroju a prípadnými zvyšnými potrubiami ju rozposiela ďalej. Listami tohto stromu by boli konzumenti, pretože prijatú vodu ďalej nerozposielajú.

3.2 Existencia riešenia hry Net

Keď už máme zadefinovanú hru a jej riešenie, zamyslime sa nad otázkami, ktoré by nás mohli zaujímať pre dané inštancie.

Prirodzenou otázkou je, či má daná inštancia riešenie – či existuje koncové natočenie. Podobne by sme sa mohli pýtať trochu ťažšiu otázku – koľko rôznych koncových natočení existuje. Mohli by sme klásť obmedzujúce podmienky tým, že uvedieme, ako majú byť otočené niektoré políčka a pýtať sa, či existuje riešenie za takých podmienok. Ak by sme chceli minimalizovať počet otočení od počiatočného (ktoré je náhodné), mohli by sme sa pýtať, koľko najmenej potrubí musíme otočiť, aby sme dospeli ku koncovému natočeniu. Samozrejme, ku každej z optimalizačných otázok by sme dokázali formulovať aj rozhodovaciu verziu. Napríklad namiesto najmenšieho počtu políčok, ktoré treba otočiť, by sme sa mohli pýtať, či existuje koncové natočenie, ku ktorému sa dokážeme dostať na otočenie najviac K políčok.

Základným parametrom, ktorý budeme pri presnej formulácii problému uvažovať, je množina typov potrubí, ktoré sa v mriežke vyskytujú. Každému typu potrubia priradíme veľké písmeno anglickej abecedy podľa obrázka 3.1. Hlavnou otázkou, nad ktorou sa budeme zamýšľať, je existencia riešenia pre danú inštanciu, v ktorej sa vyskytuje nejaká pevne zvolená množina typov potrubí.

Definícia 17. *Nech $S \subseteq \{K, I, T, J, X, P\}$.* S-Net *je označenie jazyka tých inštancií hry Net, ku ktorým existujú koncové natočenia a ktoré pozostávajú z potrubí, ktorých písmenkové označenia sú z množiny S.*

Napríklad KITJP-Net bude označovať všetky riešiteľné inštancie, v ktorých sa vyskytuje konzument, rovné potrubie, križovatka v tvare T a prázdne políčko. V množine prípustných typov potrubí chýba zdroj. Vždy budeme uvažovať, že ten je v mriežke prítomný práve raz a nebudeme ho zapisovať medzi množinu potrubí. Zamýšľanie sa nad otázkami okolo inštancií hry Net, v ktorých chýba zdroj, je totiž triviálne.

3.3 Variácie hry

Okrem manipulácie s množinou potrubí si spomeňme ešte niekoľko možností, ako upraviť hru. Niektoré možnosti nastavení nám poskytuje implementácia [1]. V tejto implementácií hry je zdroj umiestnený v strede hracej plochy, avšak my sa tohto obmedzenia v tejto práci nebudeme pridržiavať. Jedným z možných nastavení je aj prepojenie krajov mriežky – v takom nastavení majú všetky políčka 4 susedov a políčko vo vrchnom riadku môže vodu nasmerovať nahor, kde ju prijme zodpovedajúce políčko v dolnom riadku (obdobne aj pre políčka v pravom alebo ľavom stĺpci). Toto nastavenie zodpovedá nakresleniu mriežky nie do roviny, ale na torus. My budeme v našej práci uvažovať len rovinné nakreslenie, v ktorom nie sú krajné políčka prepojené.

Jednou z ďalších možností je variant hry, pri ktorom umiestnime medzi niektoré políčka steny. Medzi týmito dvoma políčkami nemôže pretekať voda. V tomto prípade sa nám zodpovedajúco upravuje definícia konzistentnosti dvoch susedných políčok. Na steny sa môžeme dívať ako na obmedzujúce podmienky – z množiny všetkých možných otočení potrubí nám každá stena niektoré vylúči. Z tohto pohľadu nám stena slúži ako akási pomôcka – vieme, že istým spôsobom potrubia vedľa nej natočené nebudú. Všimnite si, že v prípade križovatky tvaru T alebo rovného potrubia stačí na informáciu o natočení tohto potrubia v každom prípadnom koncovom stave jedna jediná stena v susedstve políčka.

Kapitola 4

Výpočtová zložitosť hry Net a jej variácií

V tejto kapitole sa budeme venovať riešeniu problému S-Net pre rôzne S. Podobne, ako v predchádzajúcej kapitole, budeme pre rozmery mriežky používať označenie N a M. Riadky očíslujeme smerom od vrchného k spodnému od 1 do N. Obdobne si označíme aj stĺpce, ľavý bude mať číslo 1, pravý M.

Pri návrhu riešení v tejto kapitole sa trochu odkloníme od formálneho modelu Turingových strojov a naše postupy budeme popisovať myšlienkovo. Naše algoritmy budu prezentované tak, aby pre čitateľa nebolo obtiažne domyslieť technické detaily implementácie na štandardnom počítači napríklad v jazyku C alebo Pascal, ale v prípade záujmu aj vo forme Turingového stroja, ktorý sme spomínali v predchádzajúcej kapitole.

4.1 Príslušnosť do NP

Venujme sa najprv jednoduchšiemu problému, ktorým je pre danú inštanciu a daný stav zistiť, či je stav koncový. Treba overiť, či je každé políčko zásobované vodou, či je každé políčko konzistentné a či žiadne neprijíma vodu z dvoch rôznych strán. O jednoduchosti tohto problému hovorí nasledujúca veta.

Veta 4.1.1. *Nech $S \subseteq \{K, I, T, J, X, P\}$ je ľubovoľná množina potrubí. Existuje polynomiálny algoritmus vzhľadom na rozmery mriežky, ktorý pre danú inštanciu, v ktorej sa vyskytujú potrubia z S a daný stav zistí, či je stav*

koncový. Navyše existuje jeho implementácia v štandardnom programovacom jazyku, ktorej časové nároky sú lineárne od veľkosti mriežky – $O(MN)$.

Dôkaz. Uvažujme, ako by vyzeral program v bežnom programovacom jazyku. Bez újmy na všeobecnosti, nech dostane inštanciu hry ako maticu s hodnotami jednotlivých políčok a stav hry ako maticu čísel popisujúcich jednotlivé natočenia.

Najprv sa zamyslime nad podmienkou, či sú všetky políčka konzistentné. Toto sa pre zadanú maticu a stav hry overí veľmi jednoducho. Pre každé políčko a každú stranu sa podľa vstupných matíc zistí, či tam smeruje potrubie alebo nie. V prípade, že sused na tej strane nie je, potrubie tam smerovať nesmie, v druhom prípade musíme overiť, či na druhej strane nenastáva nekonzistencia. V prípade, že niekde nekonzistencia nastane, môžeme prehlásiť, že stav nie je koncový.

Na overenie zvyšných dvoch podmienok si formulujme problém v reči teórie grafov. Vrcholy budú políčka mriežky. Pre daný stav skonštruujeme hrany – tie budú medzi susednými políčkami, ak majú oproti sebe nasmerované potrubia. Našou úlohou je zistiť, či je graf stromom, teda súvislým acyklickým grafom. V prípade, že je graf súvislý, potom sa voda musí dostať ku každému políčku v mriežke. Ak je graf strom, potom platí, že každé políčko je zásobované z najviac jednej strany.

Koreň stromu bude zdroj – vieme, že ten je práve jeden. Jedným prehľadaním grafu (napríklad do hĺbky) zo zdroja dokážeme overiť obidve podmienky. Ak je graf strom, potom platí, že voda do každého vrchola okrem zdroja priteká hranou smerujúcou ku koreňu a odteká všetkými ostatnými. Listy v našom strome tvoria vrcholy, ktorých políčka sú konzumenti.

Prvá fáza algoritmu potrebuje počet operácií úmerný počtu políčok v mriežke, pretože konzistenciu každého políčka mriežky overuje v konštantnom čase, keďže každé má najviac štyroch susedov. Druhá fáza spočíva v skonštruovaní a prehľadaní grafu s $O(NM)$ vrcholmi a $O(NM)$ hranami. Na túto fázu nám postačuje čas $O(NM)$, takže výsledné nároky algoritmu na počet operácií sú $O(NM)$. □

Všimnime si, že pri dôkaze sme nemuseli predpokladať nič o množine potrubí, a preto tento algoritmus funguje pre každé prípustné S zo znenia vety. Okrem toho treba poznamenať, že mu nevadia ani prípadné iné modifikácie – steny medzi políčkami, otvorené kraje mriežky a ľahko sa dá modifikovať aj pre viac zdrojov.

Teraz si ukážeme, že problém zistenia existencie koncového stavu pre danú inštanciu hry je v triede NP, ktorú spomíname podrobne v predchádzajúcej kapitole. Nedeterministický polynomiálny algoritmus budeme popisovať v neformálnej rovine, pretože nepoužívame žiadny model, v ktorom by sme mali definovaný nedeterministický výpočet.

Veta 4.1.2. *Nech $S \subseteq \{K, I, T, J, X, P\}$ je ľubovoľná množina potrubí. Jazyk* S-Net *patrí do triedy NP.*

Dôkaz. Popíšeme algoritmus, ktorý spočíva v myšlienke natipovania si stavu a jeho overenia. Bez ujmy na všeobecnosti začneme v ľavom hornom rohu. Nedeterministicky si tipneme, ako bude toto políčko natočené a prejdeme na vedľajšie políčko. Takto skúsime natočiť všetky políčka. Potom len overíme algoritmom z vety 4.1.1, či sme tipovali dobre. V prípade, že existuje koncový stav, existuje aj zodpovedajúca postupnosť odhadov na otočenie políčok a tento stav nájdeme. V opačnom prípade sa nám to určite nepodarí.

Tento algoritmus vyžaduje polynomiálny nedeterministický čas. Tipnutie otočenia jedného políčka nás stojí konštantný čas. Všetky políčka nás teda stoja $O(NM)$ a v polynomiálnom čase podľa vety 4.1.1 stíhame overenie. □

Opäť si môžeme všimnúť, že nikde nepredpokladáme nič o množine potrubí S a preto uvedený postup funguje pre každú prípustnú množinu.

4.2 Pomalé metódy riešenia

V tejto časti si uvedieme niektoré naše algoritmy na riešenie problému existencie koncového stavu. Hoci majú exponenciálnu časovú zložitosť v závislosti od parametra N alebo M, posledný z nich je najlepší známy postup. Všetky algoritmy v tejto sekcii riešia problém S-Net pre každé S.

Prístup skúšania a overovania

Úplne najjednoduchšie riešenie by spočívalo vo vyskúšaní a overení všetkých možných stavov. Ako sme spomenuli pri definícii hry, stav je možné reprezentovať prirodzeným číslom v rozsahu do 4^{NM}. Dekódovanie čísla stavu do matice dokážeme v čase $O(NM)$, preto bude obtiažnosť celého postupu s využitím vety 4.1.1 $O(4^{NM}NM)$. Takto už pre malé N a M potrebujeme neúnosné množstvo operácií.

Dynamické budovanie riešenia

Možnosťou, ako zredukovať nároky, je skúsiť uplatniť princíp dynamického programovania. Najprv demonštrujeme algoritmus, ktorý je jednoduchší, ale neoveruje všetky podmienky, a potom ho upravíme, aby sme dostali správny. Využijeme pri tom, že pri budovaní riešenia niekde v mriežke nie sme až tak veľmi závislí od toho, ako konkrétne máme otočené potrubie niekde inde v mriežke ďaleko od tohto miesta. Pre políčko na i-tom riadku a j-tom stĺpci budeme používať označenie $[i, j]$.

Základom algoritmu je postupovanie po riadkoch. Po každom riadku si budeme udržovať zoznam možností, ako mohol vyzerať predchádzajúci riadok z hľadiska pokračovania potrubí smerom k aktuálnemu riadku. Následne vyskúšame všetky možnosti otočenia spracovávaného riadku a skúsime ich napojiť na všetky predchádzajúce. Ak sa nedostaneme do sporu s konzistenciou, môžeme pridať toto otočenie do zoznamu pre ďalší riadok. Keď sa nám podarí napojiť posledný riadok, algoritmus úspešne skončíme.

Označme množinu rôznych otočení písmenom V. Každé otočenie sa dá reprezentovať prirodzeným číslom do 2^M, keďže v tomto postupe nás zaujíma len to, či z nejakého políčka smeruje k ďalšiemu riadku potrubie. Postup algoritmu je nasledovný:

1. Vytvoríme prázdny zoznam V_1. Vyskúšame všetky otočenia prvého riadku. O každom overíme, či nenastáva nekonzistencia medzi políčkami $[1, j]$ a $[1, j + 1]$ pre všetky $j, 1 \leq j < M$, pri danom otočení. Podobne treba overiť, že nesmeruje potrubie smerom do steny. V prípade, že overenie dopadlo úspešne, zakódujeme otočenie do čísla v tak, že v jeho binárnej reprezentácii nastavíme j-ty bit na 1 vtedy, ak z políčka na $[1, j+1]$ smeruje smerom k políčku $[2, j+1]$ potrubie. Zoznam V_1 si stačí udržiavať ako množinu – my síce môžeme dostať rôznym natočením políčok prvého riadku rovnaké číslo, ale z hľadiska ďalšieho postupu to pre nás budú ekvivalentné situácie.

2. Budeme postupne spracovávať i-ty riadok, pre $i = 2, \ldots, N-1$. Najprv vytvoríme prázdny zoznam V_i. Budeme postupovať skúšaním všetkých 4^M otočení. Najprv overíme konzistentnosť otočenia v rámci riadku i: $\forall j, 1 \leq j < M$ overíme, či je dvojica políčok na $[i, j]$ a $[i, j + 1]$ konzistentná. Napokon treba overiť, či krajné políčka $[i, 0]$ a $[i, M]$ nenasmerovali potrubia smerom k stene. V prípade, že tento test dopadne úspešne, budeme pokračovať s daným natočením riadka i a overovať

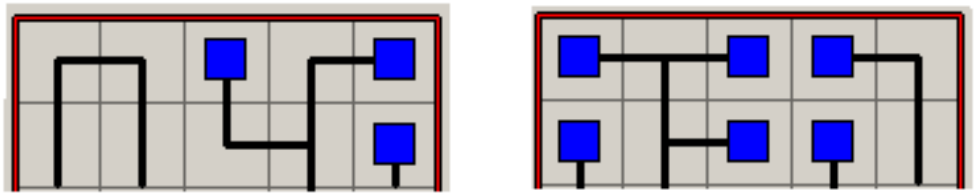
pre všetky prvky zoznamu V_{i-1}, či sa na neho dá aktuálne natočenie riadka i napojiť. Treba overiť konzistenciu medzi políčkami $[i-1][j]$ a $[i][j]$ pre $j = 1, \ldots, M$. V prípade, že toto overenie dopadne dobre, môžeme daný riadok na predchádzajúci napojiť a preto do zoznamu V_i pridáme číslo v, ktoré sme zakódovali obdobne ako v predchádzajúcom bode podľa skúmaného otočenia.

3. Obdobne ako predchádzajúci odsek, skúsime všetky natočenia riadku N. Najprv overíme konzistentnosť dvojíc $[N, j]$ a $[N, j+1]$ pre $j = 1, \ldots, M-1$. Obdobne pre všetky políčka overíme, či z nich nie sú nasmerované potrubia smerom k stene. V prípade, že je otočenie tohto riadku konzistentné, skúsime ho napojiť na každé z otočení predchádzajúceho riadku v zozname V_{N-1}. V prípade, že sa nám to podarí, môžeme skončiť úspešne.

Je ľahké vidieť, že tento algoritmus skončí úspešne práve vtedy, keď existuje konzistentný stav. Avšak ak skončí úspešne, nemusí existovať stav, ktorý je koncový, pretože nekontroluje zvyšné dve podmienky – či je každé políčko zásobené a či do žiadneho nepriteká voda z dvoch strán.

Ukážeme teraz, že malou úpravou môžeme získať algoritmus, ktorý bude korektne zisťovať existenciu koncového stavu. Zjavne nestačí, keď si v zoznamoch V pamätáme len fakt, či je smerom k políčku na ďalšom riadku nasmerované potrubie alebo nie. My totiž musíme vedieť, či v prípade, že tam potrubie je, treba do tohto potrubia vodu priviesť, alebo naopak nám odtiaľ tečie. Nestačí ale ani rozšírenie na tri možnosti. Obrázok 4.1 ukazuje, že si potrebujeme pamätať ešte niečo - ktoré potrubia sú spolu spojené v spracovanej časti mriežky. Preto zmeníme kódovanie stavov v zoznamoch V pridaním tejto informácie. Keďže stĺpec má M políčok, potrebujeme odlíšiť potenciálne až M rôznych hodnôt pre potrubie bez vody. V prípade, že v potrubí voda je, tak analogickú informáciu nepotrebujeme – vodu totiž do potrubia neprivádzame, ale z nej čerpáme. Preto na zakódovanie jedného políčka pri otočení riadka potrebujeme $M+2$ rôznych hodnôt – M pre potrubie bez vody, jednu pre potrubie s vodou a jedno pre políčko bez potrubia. Preto majú odteraz jednotlivé natočenia smerom k ďalšiemu riadku kódy v rozsahu do $(M+2)^M$. Všimnime si, že dve otočenia, ktoré sme v predchádzajúcom riadku stotožnili, môžu odteraz byť rôzne. Pridaním ďalšej informácie sme si totiž začali všímať nové rozdiely.

Dá sa ukázať, že pre zmenené kódovania v zoznamoch V vieme aplikovať ten istý postup. Pre dané otočenie a predchádzajúci riadok totiž vieme



Obr. 4.1: Kódovanie ľavého natočenia druhého riadku bude pri upravenom algoritme 11023, kódovanie pravého 12034. V pôvodnom postupe by sme obe zakódovali 11011.

efektívne zisťovať aj číselné označenia potrubí bez vody v ďalšom riadku. Pri overovaní treba kontrolovať jednak to, či sme niekde neuzavreli oblasť mriežky bez vody, ale aj to, či sme niekde nespojili dve potrubia, ktoré obe už boli zásobované vodou. Rovnako treba dávať pozor, aby sme niekde nespojili dve potrubia s rovnakým číslom, ktoré síce ešte nemajú vodu, ale keďže ju niekedy dostať musia, potom by nastala situácia, že nejaké políčko bude zásobované z dvoch strán.

Tento algoritmus zisťuje existenciu koncových stavov. Popri overovaní konzistencie celej mriežky overuje aj zásobovanie vodou každého políčka a skutočnosť, či je každé políčko zásobované z práve jednej strany.

Pri analýze časovej zložitosti tohto postupu treba pripomenúť, že jednotlivé zoznamy V_i môžu mať až $(M+2)^M$ prvkov. Prvý riadok nás stojí menej času ako ostatné, lebo skúmame len $O(4^M)$ otočení a každé overujeme v čase $O(M)$. Prvá fáza teda trvá $O(M4^M)$. Každý ďalší riadok ponúka ďalších $O(4^M)$ otočení, ktoré môžu chcieť byť skombinované so všetkými $O((M+2)^M)$ prvkami zo zoznamu V. Overenie jednej náväznosti nás stojí čas $O(M)$. Zoznam budeme implementovať ako pole, kde bude pre každé možné otočenie informácia, či sa v zozname nachádza alebo nie. Takýto zoznam musíme inicializovať, preto nás jeho vytvorenie stojí čas $O((M+2)^M)$. Vloženie a zisťovanie prítomnosti prvku vo zozname sme schopní robiť v čase $O(1)$. Preto nás každý riadok stojí čas $O((M+2)^M+(M+2)^M4^MM) = O((M+2)^M4^MM)$. Týchto riadkov je $N-1$, takže výsledný čas bude $O(M4^M+(N-1)(M+2)^M4^MM) = O(N(M+2)^M4^MM)$. Napriek tomu, že tento čas vyzerá hrozivo, je to omnoho pomalšie rastúca funkcia pri algoritme skúšania všetkých stavov. Dôležité je spomenúť, že tento postup

môžeme realizovať aj po stĺpcoch a preto si môžeme vybrať to menšie z čísel N, M a to mať v časovej zložitosti v exponente. Vo všeobecnosti si tak ale nemusíme pomôcť, napríklad v prípade štvorcovej mriežky.

Vylepšené dynamické budovanie riešenia

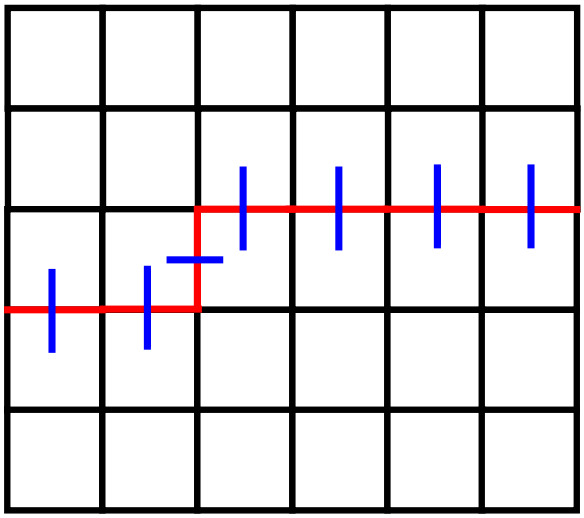
Uvedené riešenie sa dá zoptimalizovať použitím ďalších myšlienok. Prvým jednoduchým zlepšením je pozorovanie, že nepotrebujeme $M + 2$ hodnôt pri kódovaní stavov. Uvedomme si, že potrubia, ktorými už zo spracovanej oblasti vyteká voda, sú jeden komponent. A pri konštrukcii riešenia nám až tak nezáleží, či voda z komponentu vyteká, alebo do neho musí byť privedená - v oboch prípadoch musíme dávať pozor rovnako a nespájať spolu potrubia rovnakých komponentov. Ak má byť riešenie korektné, potom do komponentu aj tak vodu napokon privedieme a potom by sme spôsobili, že sa v nejakom políčku stretne z dvoch smerov.

Dôležitým vylepšením je nepripájať ďalšie riadky ako celok, ale postupne pridávať jednotlivé políčka nového riadku. Toto budeme robiť zľava doprava (viď obrázok 4.2). Hranica už vybudovanej oblasti bude teda pozostávať z niekoľko, aj nula, políčok, ktoré sú v aktuálne budovanom riadku a zvyšku políčok z predchádzajúceho riadku. Hranica teda pozostáva z nanajvýš $M+1$ hrán medzi políčkami. Keď pripájame nové políčko, uvažujeme všetky možnosti, ako mohla vyzerať hranica doteraz, a po pridaní políčka dostaneme nové hranice, ktorého budú od predchádzajúcich lokálne zmenené v niekoľkých hranách.

Poslednou optimalizáciou je skúsiť presnejšie odhadnúť počet takýchto hraníc, pretože hodnota $(M+1)^{M+1}$ je veľmi hrubá (niektoré možnosti sa nám nevyskytnú). Keďže parametrom počtu rôznych hraníc je M, označme tento počet zatiaľ $P(M)$. Neskôr tento počet skúsime presnejšie charakterizovať. Celý algoritmus bude teda vyzerať nasledovne:

1. Inicializujeme prázdnu hranicu, zodpovedajúcu vrchnému riadku. Táto hranica bude mať samé nuly.

2. Postupne pridávame políčka po riadkoch a v rámci riadkov zľava doprava. Všetky aktuálne prípustné hranice skúšame kombinovať so všetkými štyrmi natočeniami nového políčka. Podobne ako v predchádzajúcom algoritme si musíme dať pozor na konzistenciu nového políčka s



Obr. 4.2: Budovanie riešenia pridávaním políčok s označenou hranicou spracovanej časti mriežky.

okolitými dvomi v pôvodnej hranici. Rovnako musíme prečíslovať komponenty v prípade, že prišlo k ich spojeniu, a to nielen v mieste pridania políčka, ale v celej hranici.

3. Posledný riadok postupujeme obdobne s rozdielom, že smerom do steny mriežky samozrejme nesmú políčka smerovať potrubia. Rovnako musíme dávať pozor, aby sme na konci po pridaní posledného políčka v pravom dolnom rohu nenechali viac ako jeden komponent. V opačnom prípade by sme totiž nesplnili podmienku, že všetky políčka budú zásobované vodou, pretože zdrojové políčko by patrilo len do jedného z komponentov. Ak nám ostane po pridaní posledného políčka nejaká korektná hranica, potom riešenie existuje.

Poďme teraz skúsiť nájsť čo najpresnejší odhad počtu $P(M)$. Táto hodnota je dôležitá pre časovú zložitosť celého algoritmu, keďže počet rôznych hraníc je najvýraznejším prispievateľom do počtu potrebných krokov tohto algoritmu. Rátame teda postupnosti dĺžky M, ktoré zodpovedajú tomu, ktoré hrany idú a ktoré nejdú cez hranicu a ktoré sú ako prepojené. Napríklad postupnosť (1, 0, 1, 2, 0, 1) znamená, že cez hranicu ide 1., 3., 4. a 6. hrana a hrany 1, 3 a 6 sú v už spracovanej časti mriežky prepojené.

Všimnime si, že ak je počet stĺpcov mriežky M, potom môže mať hranica až $M+1$ prvkov. Avšak v prípade, že políčko, ktoré bolo pridané naposledy, má nasmerované smerom k hranici obidve hrany, potom určite patria rovnakému komponentu a v postupnosti by sa vyskytli dve rovnaké čísla po sebe. Okrem tohto prípadu môžu nastať tri situácie - potrubie má smerom k hranici toto políčko nasmerované len po prvej možnej hrane, len po druhej možnej hrane, alebo ani po jednej. Preto budeme rátať pre jednoduchosť počet postupností pre dĺžku M a ich počet na konci vynásobíme štyrmi. Keďže nám ale pôjde o asypmtotický odhad rastu, potom to pre nás aj tak napokon nebude podstatné.

Pre odhad rastu uvedených postupností si najprv uveďme nasledovnú lemu:

Lema 4.2.1. *Pre $n \geq 3$ definujme $s(n,i) = (i-2)^{-3/2}(n+1-i)^{-3/2}$ a následne $W(n) = \sum_{i=3}^{n} s(n,i)$. Potom platí: $W(n) < 6.1n^{-3/2}$.*

Dôkaz. Pre $n \leq 50$ overíme ručne.

Všimnime si, že skúmaná suma je symetrická: $s(n,3) = s(n,n)$, $s(n,4) = s(n,n-1)$, atď. Navyše ľahko nahliadneme, že od začiatku po polovicu sumy (teda od $i = 3$ po $i = \lfloor (n+3)/2 \rfloor$) veľkosť sčítancov klesá. S využitím týchto dvoch pozorovaní môžeme hodnotu $W(n)$ odhadnúť tak, že všetky členy sumy okrem prvých a posledných dvoch odhadneme zhora integrálom.

Dostávame:

$$\begin{aligned} W(n) &\leq 2\left(s(n,3) + s(n,4) + \int_4^{(n+3)/2} s(n,x)dx\right) \\ &= 2(n-2)^{-3/2} + 2^{-1/2}(n-3)^{-3/2} + \int_4^{(n+3)/2} 2s(n,x)dx \end{aligned}$$

Pre $n \geq 50$ platí $(n-3)^{-3/2} < 1.1n^{-3/2}$, preto

$$W(n) < 2.2n^{-3/2} + 0.78n^{-3/2} + I(n)$$

Výpočtom integrálu dostávame:

$$I(n) = \int_4^{(n+3)/2} 2s(n,x)dx = -\frac{4(n-2x+3)}{(n-1)^2\sqrt{x-2}\sqrt{n+1-x}} + C$$

A teda platí:

$$I(n) = \int_4^{(n+3)/2} = \frac{4(n-5)}{(n-1)^2\sqrt{2}\sqrt{n-3}}$$

Zjavne teraz

$$I(n) < \frac{4(n-1)}{(n-1)^2\sqrt{2}\sqrt{n-3}} = \frac{2\sqrt{2}}{(n-1)\sqrt{n-3}} < 2\sqrt{2}(n-3)^{-3/2}$$

A teda pre $n \geq 50$ je

$$I(n) < 3.12n^{-3/2}$$

Sčítaním odhadov dostávame

$$W(n) < 6.1n^{-3/2}$$

□

Okrem toho využijeme ešte jedno pomocné tvrdenie:

Lema 4.2.2. *Nech* $n \geq 20$. *Potom* $(n-1)^{-3/2} \leq 1.1 \cdot n^{-3/2}$.

Dôkaz. Zjavne platí:

$$\lim_{n\to\infty} \left(\frac{n}{n-1}\right)^{3/2} = 1$$

Dôkaz dokončíme vyhodnotením

$$\left(\frac{20}{19}\right)^{3/2} = 1.079977213$$

□

Náš odhad budeme robiť pomocou nasledovných hodnôt:
$P(M)$ - počet všetkých postupností.
$Q(M)$ - počet tých z nich, ktoré majú na prvom mieste 1.
Platia nasledovné rekurzívne vzťahy:
$P(M) = Q(M) + P(M-1)$, lebo na prvom mieste je 1 alebo 0.
$Q(M) = P(M-1) + \sum_{i=2}^{M} P(i-2)Q(M-i+1)$. O platnosti tejto rovnosť sa presvedčíme rozborom prípadov. Zamyslime sa, kde môže byť ďalšia jednotka. Ak sa už v postupnosti žiadna nenachádza, potom sme sa zbavili jedného čísla a pre zvyšok postupnosti máme $P(M-1)$ možností. V opačnom prípade zložíme zvyšok postupnosti z dvoch podpostupností. Nájdeme najbližšiu jednotku a na pozíciach po ňu spravíme ľubovoľnú postupnosť. Na zvyšných spravíme postupnosť začínajúcu 1.

Teraz vyslovíme hlavné tvrdenie:

Veta 4.2.3. *Pre postupnosti P a Q platia nasledovné ohraničenia:*

$$P(M) \leq 0.5 \cdot 6^M \cdot M^{-3/2}$$

$$Q(M) \leq 0.4 \cdot 6^M \cdot M^{-3/2}$$

Dôkaz. Tieto ohraničenia budeme dokazovať indukciou. Zjavne $P(1) = 2$, $Q(1) = 1$ a $P(0) = 0$. Pre M menšie ako 20 sme urobili dôkaz numerickým vyhodnotením. Nech $M \geq 20$ a nech odhady platia pre obe postupnosti pre všetky $m < M$. Uvažujme najprv postupnosť Q. Vychádzame z rekurzívnej definície

$$Q(M) = P(M-1) + \sum_{i=2}^{M} P(i-2)Q(M-i+1)$$

Využitím indukčných predpokladov dostávame

$$Q(M) \leq 0.5{\cdot}6^{M-1}{\cdot}(M{-}1)^{-3/2}{+}\sum_{i=2}^{M} 0.5{\cdot}6^{i-2}{\cdot}(i{-}2)^{-3/2}{\cdot}0.4{\cdot}6^{M-i+1}{\cdot}(M{-}i{+}1)^{-3/2}$$

Využitím lemy 4.2.2. dostávame

$$Q(M) \leq 0.092 \cdot 6^M \cdot M^{-3/2} + 6^M \cdot 0.034 \cdot \sum_{i=3}^{M} (i-2)^{-3/2} \cdot (M-i+1)^{-3/2}$$

Pomocou odhadu z lemy 4.2.1 dostávame

$$Q(M) \leq 0.092 \cdot 6^M \cdot M^{-3/2} + 6^M \cdot 0.2074 \cdot M^{-3/2}$$

$$Q(M) \leq 0.2994 \cdot 6^M \cdot M^{-3/2}$$

Pre dôkaz postupnosti P si znovu rozpíšme rekurzívnu definíciu

$$P(M) = Q(M) + P(M-1)$$

Využitím práve dokázaného odhadu pre Q a indukčného predpokladu dostávame

$$P(M) \leq 0.4 \cdot 6^M \cdot M^{-3/2} + 0.5 \cdot 6^{M-1} \cdot (M-1)^{-3/2}$$

Využitím lemy 4.2.2. dostávame

$$P(M) \leq 6^M \cdot M^{-3/2} \left(0.4 + \frac{0.5 \cdot 1.1}{6}\right)$$

$$P(M) \leq 0.492 \cdot 6^M \cdot M^{-3/2}$$

□

Hodnotu $P(M)$ teda dokážeme ohraničiť výrazom $O(6^M M^{-3/2})$, ktorý v ďalše analýze zjednodušíme na $O(6^M)$.

Pridanie každého políčka nás stojí prejdenie celého predchádzajúceho zoznamu hraníc a skombinovanie so štyrmi natočeniami nového políčka. Vytvorenie novej hranice trvá čas $O(M)$, pretože môžeme potrebovať prečíslovať komponenty. Celkový čas pridania políčka je teda $O(6^M M)$. Políčok je $N \times M$ a čas potrebný na inicializáciu a ukončenie algoritmu je konštantný, preto je celkový čas behu $O(NM^2 6^M))$. Tento čas je omnoho lepší ako v prípade predchádzajúceho neoptimalizovaného algoritmu. Výraz 6^M pôsobí možno optimisticky a mohlo by sa zdať, že ak by sme posnažili viac, tak by sme sa mohli premennej v exponente zbaviť úplne a dostať celé riešenie do polynomiálneho času. Ukazuje sa ale, že takáto výrazná optimalizácia už nie je možná.

4.3 Analýza problému KITJP-Net

V tejto sekcii sa budeme venovať problému KITJP-Net a jeho obtiažnosti. Vďaka vete 4.1.2. vieme, že KITJP-Net patrí do NP. Ak by sme chceli ukázať NP-úplnosť, potrebujeme dokázať, že je aspoň tak ťažký ako niektorý iný NP-úplný problém.

V ďalšom ukážeme spôsob, ako by mohol vyzerať dôkaz NP-úplnosti jazyka KITJP-Net. Uvedená konštrukcia nie je úplne správna, avšak možno je dobrým základom nejakej inej správnej konštrukcie. Skúsime použiť otázku o existencii hamiltonovského okruhu v planárnom bipartitnom kubickom grafe. Jazyk prislúchajúci tomuto problému sme označovali HCB3P a podľa vety 2.0.3. je NP-úplný.

Súčiastka S

Na úvod si ukážeme konštrukciu z potrubí, ktorá bude užitočná v ďalšej časti. Táto konštrukcia nám umožní preniesť obtiažnosť rozhodnutí pri ťažkých problémoch do rozhodnutí, ako natočiť potrubie a usmerniť v sieti tok vody.

Našou úlohou bude skonštruovať zhluk políčok, do ktorého ústia tri vonkajšie potrubia. Jedno dopredu určené je takzvané vstupné a priteká ním voda. Chceme, aby jedným zo zvyšných dvoch táto voda vytiekla. Posledným potrubím pritečie druhá voda, ktorá má byť korektne pohltená. Samozrejme, chceme vedieť natočiť zhluk tak, aby boli všetky políčka napojené vodou a aby bolo natočenie konzistentné. Avšak nevieme, ktorým z dvoch potrubí voda pritečie a ktorým odtečie. Chceme byť preto pripravení na obe možnosti a vedieť konzistentne natočiť potrubia v oboch prípadoch. Budeme sa tak môcť prispôsobiť natočeniu mimo zhluku, ktoré rozhodne, ktorá z dvoch situácii nastane. Schéma požadovanej konštrukcie je na obrázku 4.3.

Uvedenú vlastnosť má napríklad konštrukcia na obrázku 4.4. Túto konštrukciu označíme ako súčiastka S. Slovo súčiastka je názorná, pretože miesto potrubí si môžeme predstaviť aj elektrické vodiče a miesto vody elektrický prúd. Súčiastku S neskôr využijeme pri konštrukcií zložitejších sietí. Celá súčiastka je obkolesená prázdnymi políčkami, s výnimkou potrubí A,B,C.

Veta 4.3.1. *Nech priteká voda zo smeru A. Potom pre zhluk potrubí v súčiastke S platí:*

1. *Ak druhá voda priteká z C, existuje práve jedno konzistentné natočenie, pri ktorom sa dostane voda do všetkých potrubí z práve jednej strany. Navyše, v tomto natočení vyteká voda z B.*

2. *Ak druhá voda priteká z B, existuje práve jedno konzistentné natočenie, pri ktorom sa dostane voda do všetkých potrubí z práve jednej strany. Navyše, v tomto natočení vyteká voda z C.*

Dôkaz. Pri dôkaze sa budeme odvolávať na obrázok 4.4 a odkazovať sa na políčka s potrubiami podľa súradnicového označenia - riadky budeme číslovať zvrchu nadol od 1 do 6, pričom najvrchnejší riadok s prázdnymi políčkami nebudeme brať do úvahy. Stĺpce budeme označovať zľava doprava písmenami od A po G. Sú len dve možnosti, ako otočiť zákrutu na políčku 1D, pretože v ostatných prípadoch by nebol konzistentný prívod A. Tieto dve možnosti povedú k dvom bodom v znení vety.

Obr. 4.3: Schematicky znázornená požadovaná konštrukcia. Musíme byť pripravení konzistentne vyriešiť obe situácie: alebo voda pritečie potrubím B a potrubím C odtečie voda pritekajúca cez A, alebo si potrubia B a C svoje úlohy vymenia.

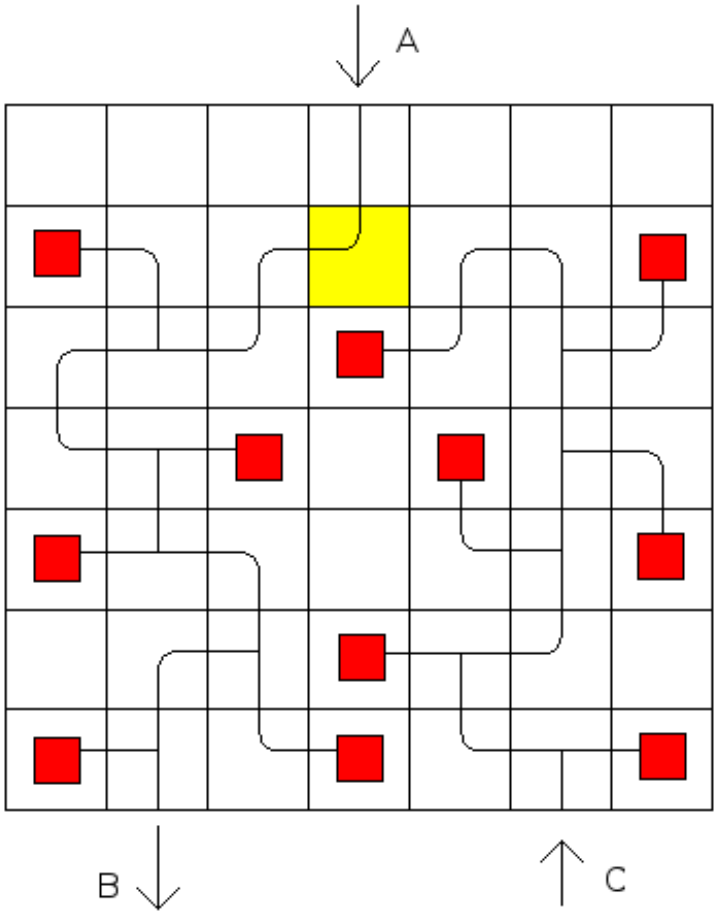
Dokážme najprv (1). Natočenie na obrázku 4.4 je konzistentné. Za predpokladu, že z C priteká voda, potom pri tomto natočení príde voda do každého políčka práve z jednej strany. Potrebujeme ešte ukázať, že toto natočenie je jediné, ktoré spĺňa uvedené podmienky. Čitateľ ľahko vidí, že ak potrubie na 1D natočíme inak ako na obrázku, do ľavej časti sa nemá šancu dostať voda.

Všimnime si potrubie na 1C. Jediný spôsob, ako ho natočiť, je na obrázku 4.4. Nad týmto políčkom je totiž prázdne políčko. Potom je ale jednoznačne určené natočenie políčka na 1B, a vďaka tomu aj natočenie 1A. Podobnými úvahami sa bude niesť celý dôkaz. O niektorých políčkach môžeme s istotou povedať, ako budú otočené, bez ohľadu otočenia ich okolia. Sú to políčka 6A a 6G. Uvažujme nasledovnú postupnosť niektorých zvyšných neprázdnych políčok:

$$1D, 1C, 1B, 1A, 1E, 1F, 1G, 2A, 2G, 3A, 3G, 4A,$$

$$4G, 4F, 5F, 6F, 6E, 6D, 6C, 6B, 5B, 4B$$

Každé z políčok má, ak chceme dodržať konzistenciu, jednoznačne ur-



Obr. 4.4: Možná realizácia súčiastky S.

čené natočenie. Toto tvrdenie by sa dalo dokázať indukciou na poradie v postupnosti. Otočenie potrubia na 1D je predpoklad.

Zamyslime sa teraz nad otočením políčka 2C. Máme dve možnosti: toto políčko prevedie vodu z 1C na 2B alebo na 2D. V prípade, že by sme vodu nasmerovali na 2D, potom by celá ľavá časť siete zjavne nemohla byť napojená, čím by sme nesplnili jednu z podmienok. Doplňme teda 2C a pokračujme ďalej:

$$2C, 2B, 3B, 3C, 4C, 5C, 5D, 5E, 4E, 3E, 3F, 2F, 2E, 2D$$

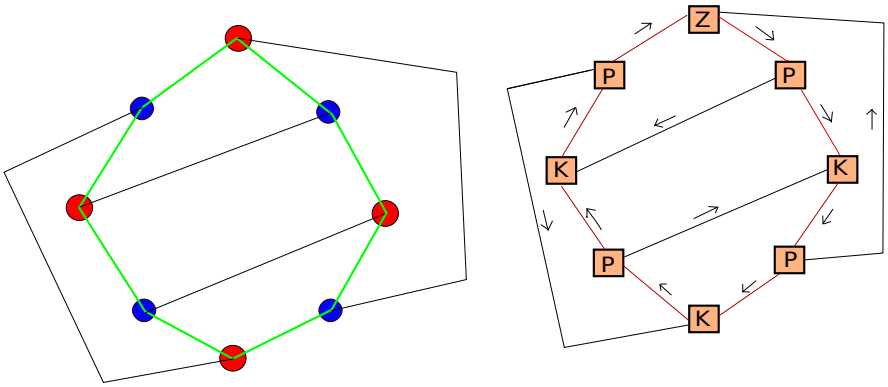
Obdobne ako v prvej časti postupnosti by sa jednoznačnosť natočenia políčok dala dokázať indukciou podľa poradia v postupnosti. Okrem spomínaného políčka 2C nám pri všetkých políčkach na určenie natočenia stačila jedna z požadovaných podmienok - konzistentnosť. V postupnosti sa vyskytlo 36 políčok súčiastky S. Ak pripočítame 4 prázdne políčka a 2, ktoré majú jednoznačné natočenie bez ohľadu na natočenie okolia, potom dostávame 42, čo sú všetky potrubia súčiastky S. Iniciatívny čitateľ môže jednoznačnosť natočenia preveriť.

Na dôkaz (2) si všimnime si, že typy potrubí sú symetrické podľa stredného stĺpca D. To znamená, že v prípade, že sa rozhodneme natočiť žltú zákrutu na políčku 1D opačne, môžeme pootáčať potrubia na oboch častiach tak, aby plnili opačnú funkciu. Jednoznačnosť sa dá dokázať analogicky ako v bode jedna. □

Konštrukcia inštancie KITJP-Net z grafu

V tejto časti skúsime nájsť redukciu HCB3P na KITJP-Net. Potrebujeme teda nájsť takú transformáciu, ktorá z grafu, ktorý je planárny, bipartitný a kubický, vyrobí sieť potrubí. Pre túto sieť potrubí musí platiť, že existuje koncové natočenie práve vtedy, keď mal pôvodný graf hamiltonovský okruh. Naša konštrukcia tieto vlastnosti mať nebude. Je však zaujímavým základom pre ďalšie bádanie.

Požadujeme, aby sieť potrubí bola z grafu skonštruovateľná v polynomiálnom čase. Okrem toho je potrebné, aby sa sieť potrubí skladala z prázdnych políčok, rovných potrubí, zákrut, križovatiek T a konzumentov. Majme teda kubický, planárny a bipartitný graf G. Predpokladajme, že G je súvislý, pretože ak by nebol, môžeme rovno prehlásiť, že nie je hamiltonovský.



Obr. 4.5: Naľavo je príklad planárneho kubického bipartitného hamiltonovského grafu. Napravo je sieť potrubí s koncovým otočením. V potrubiach medzi zhlukmi je naznačený smer toku vody. Zhluky potrubí sú označené ako zdroj, konzument a producent.

Vychádzajme z nejakého zobrazenia G do roviny. Z tohto zobrazenia dostaneme sieť potrubí tak, že hrany nahradíme dlhými úsekmi potrubí vedúcimi vodu a vrcholy nahradíme zhlukmi potrubí. Vrcholy budú rozdelené na takzvaných konzumentov a producentov podľa partície, do ktorej patria. Všetci konzumenti budú realizovaní rovnakou sieťou potrubí a tiež všetci producenti budú skonštruovaní rovnako. Jeden z konzumentov prehlásime za zdroj a ten bude realizovaný špeciálnym zhlukom potrubí. V tomto zdrojovom zhluku potrubí sa bude nachádzať aj políčko zdrojového potrubia. Ako sa presvedčíme neskôr, na výbere zdroja spomedzi konzumentských vrcholov nezáleží.

Myšlienka je nasledovná: zdrojový vrchol bude produkovať takzvanú okruhovú vodu. Otáčaním potrubí budeme šíriť túto okruhovú vodu v sieti medzi zhlukmi, ktoré reprezentujú vrcholy grafu. Ak nájdeme takú postupnosť navštívenia týchto zhlukov, v ktorej voda obíde všetky a vráti sa ku zdroju, potom zodpovedajúci okruh vrcholov v grafe bude hamiltonovský. Keďže sme vychádzali zo zobrazenia grafu do roviny, v sieti sa potrubia spájajúce zhluky zodpovedajúce vrcholom nikdy nekrižujú.

Zhluky prislúchajúce vrcholom musia byť skonštruované tak, aby vedeli jednou hranou vodu prijať a druhou odovzdať – podľa toho, ako je nasmerovaný a orientovaný tok vody podľa hamiltonovského okruhu. Okrem toho majú ale vrcholy ešte tretiu hranu. Tu príde na pomoc bipartitnosť – pro-

ducenti do potrubí reprezentujúcich túto hranu vodu pošlú a konzumenti ju príjmu. Túto vodu budeme volať odpadová.

Pre každý zhluk potrubí realizujúci nejaký vrchol musíme byť schopní natočiť potrubia v zhluku tak, aby boli konzistentné a aby boli všetky políčka v zhluku uspokojené vodou z práve jedného smeru. Samozrejme, každý zhluk musíme byť schopní natočiť tak, aby dokázal usmerniť okruhovú vodu, keďže v čase konštrukcie siete nevieme, kadiaľ si budeme želať tok vody.

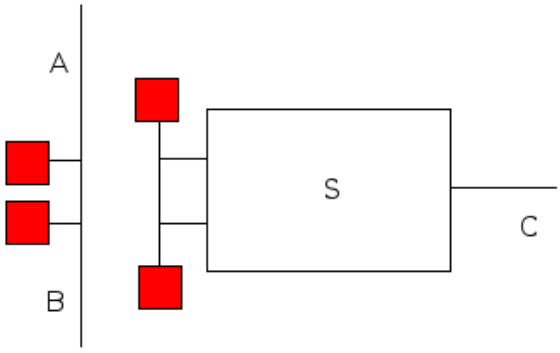
Jednoduchou časťou našej konštrukcie bude realizácia producentov. Tento zhluk jednou hranou vodu prijíma a dvoma ju vysiela von – raz ako odpadovú a raz ako okruhovú. Je potrebné, aby bol schopný prijať vodu z každého z troch potrubií. Na jeho realizáciu nám stačí jedno políčko s T-križovatkou – voda sa nám rozdistribuuje správne bez ohľadu na to, odkiaľ priteká.

Zložitejšou časťou je skonštruovanie konzumenta. Tento zhluk má dvoma hranami vodu prijať, pričom jedna je okruhová a druhá je odpadová. Okruhovú vodu má treťou hranou odviesť ďalej. V skutočnosti nie je potrebné rozlišovať vstupné vody na okruhovú a odpadovú. Od tohto faktu ale pri našej konštrukcii odhliadneme.

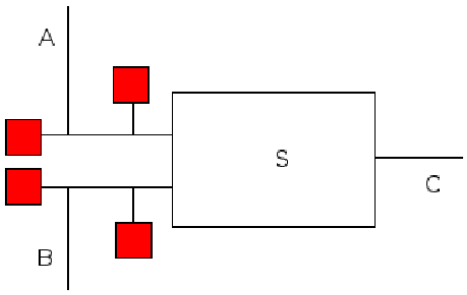
Teoreticky existuje 6 možností, ako môžeme rozdeliť funkcie medzi hrany. Tieto možnosti sú ale z hľadiska vnútornej konštrukcie len tri – stačí nám vedieť natočiť potrubia tak, aby sme pohltili odpadovú vodu a spojiť zvyšné dve hrany. Keď tieto zvyšné dve hrany spojíme, potom už nie je z hľadiska daného konzumenta dôležité, ktorým smerom voda tečie. Potrebovali by sme teda zhluk potrubí, ku ktorému existujú aspoň tri konzistentné natočenia, ktoré privedú vodu ku každému políčku v zhluku práve z jednej strany. Ku každej z troch možností požiadaviek na dvojicu prepojených vonkajších hrán musí existovať natočenie.

Na obrázku 4.6 je schéma konštrukcie konzumenta a jeho natočenie v prípade, že chceme skonzumovať odpadovú vodu z C. Odpadová voda síce vyjde zo súčiastky S, ale jej druhým výstupom sa do nej hneď vráti a privedie vodu aj k potrubiam v jej druhej časti. V ostatných prípadoch natočíme potrubia ako na obrázku 4.7. V takom prípade natočenie potrubia v použitej súčiastke S určí, či sa skonzumuje odpadová voda z A alebo B. Zvyšné dve potrubia sa spoja, aby v nich mohla prúdiť okruhová voda.

Zostalo nám ukázať, ako skonštruovať zdroj. Súčasťou zdroja je aj políčko so zdrojovým potrubím. Tento zhluk je napojený na tri potrubia, takže by sa zdalo, že musíme byť pripravení na 6 rôznych alternatív, aké funkcie budú jednotlivé potrubia plniť. Uvedomme si ale, že nám nezáleží na tom, či do potrubia príde okruhová alebo odpadová voda, pretože v oboch prípadoch



Obr. 4.6: Konštrukcia konzumenta v natočení, v ktorom konzumujeme vodu z C.



Obr. 4.7: Konštrukcia konzumenta v natočení, v ktorom konzumujeme vodu z A alebo B podľa rozhodnutia v S.

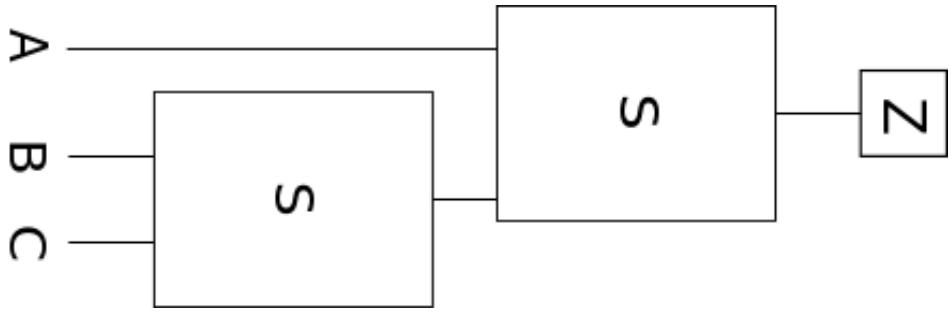
ju musíme konzistentne skonzumovať. Stačí teda uvažovať len tri možnosti – potrebujeme vedieť každým z troch potrubí vodu zo zdroja vyviesť von a zvyšnými dvoma prijať.

Na obrázku 4.8 je možná realizácia zdrojového vrcholu. V súčiastkach S robíme dve otáčacie rozhodnutia. Pri každej kombinácii rozhodnutí voda bude vytekať práve jedným z potrubí A,B,C a pre každé z potrubí existuje dvojica rozhodnutí, že voda bude vytekať práve týmto potrubím.

Dokončili sme jednotlivé diely konštrukcie. Pre daný graf G by sme teda najprv našli jeho nakreslenie do roviny a potom z tohto nakreslenia získali inštanciu KITJP-Net tak, že hrany nahradíme potrubiami a jednotlivé vrcholy príslušnými zhlukmi potrubí. Konštrukcia je uskutočniteľná v deterministickom polynomiálnom čase a má polynomiálnu veľkosť.

Táto konštrukcia bohužiaľ nemá potrebnú vlastnosť, že riešenie inštancie hry Net existuje práve vtedy, ak je graf Hamiltonovský. Môže sa totiž stať, že ak graf nie je hamiltonovský, potom pre sieť existuje koncové natočenie. Preto konštrukcia nemá požadované vlastnosti many-to-one redukcie.

Pre ďalšie skúmanie a hľadanie dôkazu NP-úplnosti jazyka sa nám na-



Obr. 4.8: Konštrukcia zdrojového vrcholu.

priek našim negatívnym skúsenostiam aj naďalej javia variácie jazyka SAT, prípadne HAM. Tieto ťažké problémy sú často využívané pri dôkazoch NP-úplnosti hier pre jedného hráča.

4.4 Zložitosť hry Net pre niektoré množiny potrubí

Pre niektoré obtiažne varianty hry Net je hypotéza, že zisťovanie existencie riešenia je NP-úplný problém a za predpokladu $P \neq NP$ prakticky neexistuje lepší algoritmus ako exponenciálny, ktorý prezentujeme v tejto práci. V tejto časti si spomenieme niekoľko množín potrubí, pre ktoré existuje jednoduchý polynomiálny algoritmus.

Ak poznáme pre nejakú množinu typov potrubií S algoritmus na riešenie S-Net, potom pre ľubovoľnú podmnožinu S môžeme použiť ten istý algoritmus. Žiadnemu algoritmu totiž nemôže vadiť, ak sa niektoré z políčok, ktoré pripúšťa, v mriežke nevyskytne, pretože by nebol korektný. Preto bude usporiadanie obtiažností podľa jednotlivých množín typov potrubí zodpovedať usporiadaniu týchto množín podľa inklúzie.

Prvým zjavným faktom je, že ak má byť otázka o existencii riešenia netriviálna, potom sa musí v množine potrubí nachádzať konzument. Všimnime si, že pre každé z ostatných piatich typov potrubí voda v koncovom natočení priteká z práve jedného smeru a vyteká aspoň jedným smerom. Konzument je jediný typ políčka, z ktorého voda nevyteká. Ak má teda existovať riešenie, musí byť v mriežke prítomný konzument, pretože predpokladáme, že zo zdroja vyteká voda aspoň jedným smerom. Ďalším pozorovaním je nízka

obtiažnosť prázdneho políčka, pri ktorom sa nerobia žiadne rozhodnutia a navyše slúži ako rada, ako majú byť natočené okolité políčka. Je preto očakávateľné, že ak má jazyk S-Net pre $S, P \in S$ svoju obtiažnosť, tak keď položíme $S_2 = S \setminus \{P\}$, tak jazyk S_2-Net nebude jednoduchší.

Veta 4.4.1. *Nech* $S \subseteq \{K, I, J, P\}$. *Potom jazyk* S-Net *patrí do* P.

Dôkaz. Najprv si uvedomme, že v tejto sieti sa nám voda nikde nevetví. Preto poznáme nutnú podmienku pre existenciu koncového natočenia – počet konzumentov musí byť rovnaký ako počet výstupov zo zdroja. Keďže vieme, že je to najviac 4, potom je počet možností natočenia konzumentov a zdroja konštantný a nízky. Nemáme preto problém všetky možnosti vyskúšať. Presnejšie, pre štyroch konzumentov je to 256, pretože v tomto prípade má zdroj len jedno rôzne natočenie.

Predpokladajme teda, že máme nejaké pevné natočenie konzumentov a zdroja. Budeme postupne natáčať aj ostatné políčka, aby sme dostali jediný možný konzistentný stav. Všimnime si, že pre všetky políčka z množiny S okrem konzumenta platí, že netreba veľa informácie o natočení susedov, aby sme dostali jednoznačné konzistentné natočenie alebo prehlásili, že také neexistuje. Stačí, aby sme vedeli o dvoch susedoch, ktorí nie sú oproti, či z nich prichádza alebo neprichádza potrubie.

Budeme postupovať po riadkoch a v rámci riadkov zľava doprava. Ak je políčko v ľavom hornom rohu konzument alebo zdroj, je jeho natočenie už pevné. V opačnom prípade využijeme informáciu, že ho z dvoch strán obklopujú steny a natočíme ho. Následne ale vieme o políčku napravo od neho dosť informácii na jeho natočenie. Takto vieme postupne ponatáčať políčka v hornom riadku a potom následne aj v nasledujúcich riadkoch. Môže nastať aj situácia, v ktorej natočenie nie je možné – napríklad ak bude políčko zdroj, avšak jeho natočenie nebude sedieť s natočením vrchného a pravého suseda. Takto pre dané natočenie zdroja a konzumentov dostaneme najviac jeden prípustný konzistentný stav. Podľa výsledku z vety 4.1.1. môžeme overiť, či je koncový. Takto pre dané rozhodnutie o natočení zdroja a konzumentov dokážeme v lineárnom čase od veľkosti mriežky zistiť existenciu koncového stavu. □

Bohužiaľ, uvedený dôkaz nie je možné použiť, ak do množiny potrubí pridáme križovatku typu T alebo križovatku typu X. V ďalšom sa budeme teda snažiť pridať niektoré z vetviacich potrubí. Aby sme vykompenzovali nárast zložitosti, budeme musieť niektoré iné typy políčok odobrať. Mohlo

by sa zdať, že pridanie križovatky X nám žiadnu obtiažnosť nepridá, pretože v nej nerobíme žiadne otáčacie rozhodnutie. Argument v predchádzajúcom dôkaze ale padá, pretože sa nám v mriežke môže vyskytnúť veľa konzumentov.

Skúsme sa teda zamyslieť nad množinami potrubí, do ktorých sme pridali križovatku typu X. Jednoduchý výsledok predstavuje nasledujúca veta. Tá hovorí, že stačí odobrať zákrutu a hľadanie riešenia je opäť jednoduché.

Veta 4.4.2. *Nech* $S \subseteq \{K, I, X, P\}$. *Potom jazyk* S-Net *patrí do P.*

Dôkaz. Uvažujme jazyk KIXP-Net. Zisťovanie existencie koncového natočenia v danej inštancii budeme robiť jednoduchým natáčaním potrubí v mriežke postupujúc do zdrojového vrcholu. Vieme, že zdroj má najviac štyri rôzne natočenia, preto nie je problém ich všetky vyskúšať. Pri križovatke typu X nerobíme žiadne rozhodnutie a v prípade rovného potrubia nám stačí, ak poznáme s určitosťou natočenie jedného suseda.

Majme teda nejaké pevné natočenie zdroja. Budeme sa postupne rozširovať do mriežky a natáčat políčka s potrubiami s myšlienkou, že dané natočenie je nutné pre konštrukciu koncového stavu. V terminológii algoritmov z oblasti teórie grafov budeme robiť prehľadávanie do šírky. Ak sme natočili nejaké políčko a vedľa sa nachádza rovné potrubie, prázdne políčko alebo križovatka X, potom vieme, ako toto vedľajšie políčko natočiť. V prípade konzumenta to ale nie je také priamočiare. Zamyslime sa nad stavom, že sme postupné natočenie dokončili pre všetky políčka ostatných typov a už nedokážeme rozhodnúť o žiadnom ďalšom. V takom prípade je natočená časť mriežky súvislá. Ukážeme, že vieme rozhodnúť, ako majú byť natočené všetky políčka a či existuje koncové natočenie. Uvažujme teda ľubovoľné políčko mriežky okrem zdroja:

- Políčko je typu P. O jeho natočení je triviálne rozhodnuté a navyše nie je potrebné do neho priviesť vodu.
- Políčko je typu I alebo X. Ak patrí do oblasti, ktorá bola prejdená a ponatáčaná, potom je o jeho natočení už rozhodnuté. V prípade, že do tejto oblasti nepatrí, koncové natočenie určite neexistuje, pretože sa do tohto políčka nemôže dostať voda. Toto políčko totiž nesusedí so žiadnym iným políčkom s rovným potrubím alebo križovatkou X, do ktorého dokáže pritiecť voda zo zdroja.
- Políčko je konzument. Podľa predpokladu z predchádzajúceho bodu môže susediť len s už natočenými políčkami iných typov alebo inými

konzumentmi. V takom prípade ale vieme spočítať počet susedov, ktorí v prípade, že konzistentné natočenie existuje, nasmerujú k uvažovanému konzumentovi potrubie. Ak je tento počet rôzny od 1, koncové natočenie neexistuje. V opačnom prípade vieme výsledné natočenie tohto políčka. Využili sme pozorovanie, že konzument nemôže nasmerovať svoj vstup k susednému konzumentovi, pretože tak určite nedosiahneme koncové natočenie.

Pre pevné natočenie zdroja takto dokážeme skonštruovať jediné prípustné konzistentné natočenie mriežky. Táto fáza trvá lineárny čas od veľkosti inštancie hry, pretože pre každé políčko sme spravili len konštantný počet operácii. Na overenie, či je nájdené konzistentné natočenie aj koncové, môžeme použiť výsledok z vety 4.1.1. □

Podobne ako v prípade predchádzajúceho dôkazu, ani tento algoritmus nemôžeme rozšíriť o ďalší typ potrubia v množine S. Zákruta aj križovatka T majú totiž nepríjemnú vlastnosť, že ak chceme zistiť ich natočenie podľa susedov, tak musíme poznať vo všeobecnosti až natočenie troch z nich.

Literatúra

[1] Simon Tatham *http://www.chiark.greenend.org.uk/ sgtatham/puzzles/*

[2] Clay Mathematics Institute *http://www.claymath.org/millennium/*

[3] Michal Forišek *Skriptá z teórie formálnych jazykov a automatov*

[4] *http://en.wikipedia.org/wiki/Cook-Levin_theorem*

[5] J. Kratochvíl *A special planar satisfability problem and some consequences of its NP-completness* Discrete Appl. Math. 52 (1994), 233-252

[6] D. Lichtenstein *Planar formulae and thier uses* SIAM J. Computing 11 (1982), 329-343

[7] A. Mansfield *Determining the thickness of graphs is NP-hard* Math. Proc. Comb. Phil. Soc. 93 (1983), 9-23

[8] Erich Friedman *Pearl Puzzles are NP-complete*, Technical report, Stetson University, DeLand, FL 32723 (2002)

[9] D. Král, V. Majerech, J. Sgall, T. Tichý, G. Woeginger, *It is tough to be a plumber*, Theor. Comput. Sci., 313, 3 (2004), 473–484

[10] J. Plesnik *The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two*, Information Processing Letters, 8(4), 1979, 199-201

[11] A. Itai and C.H. Papadimitriou and J.L. Szwarc *Hamilton paths in grid graphs* SIAM J. Comput., 11(4), 1982, 676-686