

# Hľadanie chybného uzla distribuovaných systémov pomocou mobilných agentov

(Ročníkový projekt)

Peter Glaus

# Obsah

<b>1 Úvod</b>	<b>3</b>
1.1 Hľadanie čiernej diery . . . . .	3
1.2 Členenie práce . . . . .	4
<b>2 Definícia problému</b>	<b>5</b>
2.1 Základné pojmy . . . . .	5
2.1.1 Distribuovaný systém . . . . .	5
2.1.2 Skupina agentov . . . . .	5
2.1.3 Čierna diera . . . . .	6
2.1.4 Protivník . . . . .	6
2.1.5 Hodnotenie riešenia . . . . .	7
2.2 Modifikácie zadania . . . . .	7
2.2.1 Anonymita . . . . .	7
2.2.2 Druh výpočtu . . . . .	7
2.2.3 Komunikácia agentov . . . . .	8
2.2.4 Počiatočné znalosti . . . . .	9
2.2.5 Topológia siete . . . . .	9
2.3 Po lopate . . . . .	10
<b>3 Hľadanie v asynchrónnom modeli</b>	<b>11</b>
3.1 Opatrné kráčanie . . . . .	11
3.2 Obmedzenia . . . . .	12
3.3 Prehľadávanie náhodného grafu . . . . .	13
3.3.1 Bez znalosti topológie . . . . .	13
3.3.2 So zmyslom orientácie . . . . .	14
3.3.3 Úplná znalosť topológie . . . . .	15
3.4 Prehľadávanie dobre spojitých grafov . . . . .	16
3.5 Prehľadávanie s použitím žetónov . . . . .	16

<i>OBSAH</i>	2
<b>4 Hľadanie v synchrónnom modeli</b>	<b>18</b>
4.1 BHS ako optimalizačný problém . . . . .	18
4.1.1 Základná myšlienka . . . . .	18
4.1.2 Optimalizačný problém . . . . .	18
4.2 NP-Ťažkosť problému . . . . .	19
4.2.1 Modifikovaný problém hamiltonovskej kružnice . . . . .	19
4.3 Aproximácia problému . . . . .	19
<b>Záver</b>	<b>20</b>
<b>Literatúra</b>	<b>21</b>

# Kapitola 1

## Úvod

### 1.1 Hľadanie čiernej diery

V mojej diplomovej práci sa plánujem venovať oblasti s názvom distribuované algoritmy. Predbežný názov práce je: "Mobilné entity v grafoch s chybnými komponentmi". Tento názov je naozaj iba predbežný, pretože zatiaľ nemám vymedzené, na čo presne sa chcem zamerať. Cieľ je skúmať algoritmy na distribuovaných systémoch s použitím jednoduchých agentov. Tieto algoritmy budú riešiť jeden problém a to identifikáciu chybného uzla ďalej len BHS<sup>1</sup>.

Chybný uzol v našom ponímaní je akási čierna diera grafu. Je to uzol, ktorý sa nedá zvonka rozlíšiť od ostatných uzlov. Ale keď do neho príde agent, je okamžite zničený bez toho, aby vedel poslať nejakú správu. Z týchto vlastností hneď vieme dve základné pravidlá. Čiernu dieru môžu tušiť agenti tak, že jeden do nej vojde a už nevyjde. Druhé pravidlo je, že minimálne jeden agent musí byť obetovaný, aby sme určili kde je čierna diera. Väčšinou sa skúmajú grafy s maximálne jednou čiernou dierou, ale sú riešenia aj pre grafy s mnohými čiernymi dierami.

Táto úloha sa dá jednoducho zdefinovať, avšak v skutočnosti existuje veľmi veľa modifikácií BHS. Tieto modifikácie sa zakladajú na zmene parametrov distribuovaného systému teda grafu a na zmene parametrov agentov, ktorí daný systém prehľadávajú. Efektívnosť riešenia problému BHS sa môže líšiť pre rôzne typy grafov. Ďalej záleží na tom čo všetko vedia jednotliví agenti o grafe. Najťažšie je riešiť BHS keď agenti vôbec nepoznajú topológiu a najjednoduchšie je keď majú na začiatku úplnú mapu grafu. V neposlednom rade záleží na tom akú výpočtovú silu majú agenti a akým spôsobom môžu interagovať a spolupracovať.

Z tohto dôvodu bude príprava na diplomovú prácu v tomto projekt spo-

---

<sup>1</sup>Black Hole Search

čítať v prehľade už nájdených riešení pre jednotlivé modifikácie. Budem sa venovať hlavne hotovým riešeniam a ich vlastnostiam.

Pri čítaní publikácií o BHS som sa zatiaľ stretol s dvoma základnými smermi. Jeden smer sa venuje BHS s použitím asynchrónnych agentov[6, 5, 3, 8, 4]. Tu sa skúmajú riešenia bez znalosti topológie grafu aj riešenia so znalosťou topológie. Asynchrónnosť nám prináša prvok neurčitosti keďže nikdy neviem s určitosťou povedať či sa niekde nachádza čierna diera alebo sú agenti, ktorí smerovali k danému uzlu len veľmi spomalení.

Druhý smer sa na problém pozerá z hľadiska optimálneho traverzovania[11, 2, 10, 9]. Agenti pracujú synchronne a na začiatku väčšinou poznajú topológiu grafu. Ako si neskôr ukážeme synchronnosť nám prináša mnohé výhody. Úlohou je nájsť čo najlepšie traverzovanie grafu tak, aby bola odhalená resp. boli odhalené všetky čierne diery.

## 1.2 Členenie práce

V prvej časti je priblížená problematika distribuovaných systémov. Okrem toho presnejšie zadefinujem problém hľadania chybného uzla BHS. No a nakoniec opíšem modifikácie zadania, ktoré je možné skúmať.

Ďalšia časť ročníkového projektu sa venuje hlavne riešeniam využívajúcim asynchrónnych agentov. Sú popísané myšlienky jednotlivých riešení. Na záver je tam zhrnutie ďalších mnou nájdených výsledkov.

V tretej časti je opísaný problém z hľadiska hľadania optimálneho traverzovania pre synchronných agentov. Tu sa nachádza definícia problému a jeho základné úskalia. Rovnako je tu náčrt dôkazu, že problém je *NP-ťažký*. Okrem toho sú tu zhrnuté iba jednotlivé známe aproximácie a dolné odhady.

# Kapitola 2

## Definícia problému

### 2.1 Základné pojmy

#### 2.1.1 Distribuovaný systém

Distribuovaný systém v našom ponímaní je graf  $G = (V, E)$ .  $V$  je množina vrcholov a  $E$  je množina hrán. Hovoríme, že dva vrcholy  $u, v \in V$  sú susedné ak existuje hrana  $(u, v)$  z  $E$ . Vrcholy grafu budeme nazývať aj uzly. Hrany budú naopak linky. Napojenie hrany na uzol budeme nazývať *port*. Vo všetkých spomínaných riešeniach autori uvažujú graf jednoduchý a neorientovaný. To znamená, že hrana  $(u, v)$  je tá istá ako  $(v, u)$  a maximálne jedna medzi ľubovoľnými dvoma vrcholmi. Rovnako množina  $E$  neobsahuje hrany typu  $(u, u)$ . Pokiaľ nebude uvedené inak budeme uvažovať práve takýto graf.

Súčasťou grafu je označenie portov každého vrchola. Označme  $\lambda_u(u, v)$  ako pomenovanie vo vrchole  $u$  pre port hrany  $(u, v)$ . Potom pomenovania pre všetky porty vrcholu  $u$  označíme ako  $\lambda_u$ . Následne môžeme zdefinovať pomenovanie resp. označenie grafu  $G$  ako  $\lambda = \{\lambda_x \mid x \in V\}$ . Pomenovanie portov nejakého uzla je pre agenta nachádzajúceho sa v tom uzle známe. Vďaka nemu vedia agenti rozlíšiť hrany v jednotlivých uzloch. Rovnako sa môže zdefinovať pomenovanie vrcholov, my ho však nebudeme používať.

Dvojica  $(G, \lambda)$  tvorí distribuovaný systém.

#### 2.1.2 Skupina agentov

Hovoríme, že po distribuovanom systéme sa pohybuje skupina agentov. Agent sa občas označuje aj ako proces. Agent sa nachádza v nejakom uzle. Keď sa nachádza v nejakom uzle  $u$ , vidí linky, ktoré z neho vedú a rovnako vidí ich pomenovania  $\lambda_u$ . V uzle môže vykonať nejakú činnosť následne môže čakať na nejakú udalosť alebo prejsť nejakou hranou. Čakanie na udalosť budeme

nazývať aj zaspaním. Akonáhle sa agent rozhodne ísť nejakou hranou, nemôže si to už rozmyslieť. Vo všeobecnosti sa po nejakom konečnom čase ocitne v uzle na opačnej strane hrany.

Agenti sa na začiatku nachádzajú v nejakom uzle  $h$ <sup>1</sup>, základni. Všetci agenti vykonávajú rovnaký program, teda sa správajú úplne rovnako. Jediný rozdiel je buď keď sú asynchrónni alebo keď nie sú anonymní.

Ich úlohou je aby mal každý agent na konci vytvorenú lokálnu mapu grafu, ktorá zodpovedá skutočnosti a mal na nej vyznačené všetky hrany vedúce do čiernych dier. To znamená, že na konci musí prežiť aspoň jeden agent resp. nemôžu všetci popadať do čiernych dier.

Agenti majú možnosť komunikácie. Je triviálne zrejmé, že keby nemohli komunikovať, tak by buď nemohli získať žiadne nové informácie o grafe alebo by mohli všetci hneď uhynúť v čiernej diere. Navzájom sa môžu, ale nemusia vidieť. Treba si uvedomiť, že už to že sa vidia je istý druh komunikácie.

Akcie agentov sa pokladajú za atomické. Teda keď jeden začne vykonávať akciu, môžeme rátať s tým, že ju dokončí skôr ako niečo stihne vykonať iný agent. Podstatná vec je aj *FIFO* vlastnosť liniek. Táto vlastnosť znamená, že keď dvaja agenti chcú prejsť z vrchola  $u$  po hrane  $(u, v)$ , tak ten ktorý odíde z vrchola  $u$  ako prvý, rovnako ako prvý príde do vrchola  $v$ .

### 2.1.3 Čierna diera

Predpokladá sa, že jeden alebo viac uzlov grafu je chybných. Chybný je takým spôsobom, že každý agent, ktorý sa rozhodne ísť hranou vedúcou do tohto uzla, ukončí svoju činnosť, zomrie. Takýto uzol budeme volať čierna diera. Agent, ktorý ide do čiernej diery, sa o tom nijak dopredu nevie dozvedieť a nemôže o tom ani poslať správu ostatným agentom. Zjavné je že jediný spôsob ako sa agenti môžu dozvedieť o čiernej diere je ak vedia, že iný agent prešiel po nejakej hrane, o ktorej nevedia kam vedie, a daný agent sa nevrátil. Problém je ten, že pri asynchrónnom modeli agent môže byť len veľmi spomalený.

### 2.1.4 Protivník

Pri riešení BHS sa nechceme venovať priemerným prípadom. Chceme, aby nami zvolený postup fungoval aj pre najhorší možný prípad. Z tohto dôvodu si zavedieme pojem protivníka alebo protihráča. Počas toho ako agenti prehľadávajú sieť, protivník sa snaží nám to znemožniť. Presnejšie sa snaží, aby agenti úlohu nesplnili alebo aby im to trvalo čo najdlhšie.

---

<sup>1</sup>Homebase

Protivník môže meniť topológiu grafu. To znamená, že ak nie je daný počet vrcholov môže nejaké uberať resp. pridávať. Hrany, po ktorých sme ešte nešli môže presúvať. Rovnako môže presúvať konce hrán, o ktorých ešte nemáme zaručené kam idú. Takýmto spôsobom sa buď snaží podsúvať agentom čiernu diery, aby do nej spadli pri nesprávnom kroku alebo vytvára neznáme časti grafu tak, aby výpočet trval čo najdlhšie. Okrem toho má možnosť spomaliť agentov idúcich po nejakej hrane. Nemôžeme sa preto spoliehať na to, že niečo trvá príliš dlho.

### 2.1.5 Hodnotenie riešenia

Pri hodnotení algoritmu sa pozeráme na dva základné parametre. Prvý parameter je počet použitých agentov. V nejakých podmienkach stačia dvaja agenti. Pre niektoré modely je však potrebné viac agentov, väčšinou tento počet závisí od maximálneho stupňa čiernej diery alebo od počtu čiernych dier.

Druhé kritérium je čas potrebný na výpočet. Čas výpočtu sa môže rátať v počte krokov urobených agentom. Jeden krok v tomto prípade znamená preskúmanie situácie v uzle, vykonanie nejakej jednoduchšej činnosti v uzle<sup>2</sup> a následnom prechode nejakou hranou alebo čakaním na udalosť. Pri synchronnom modeli nás môže zaujímať celkový počet synchronných krokov. V každom takomto kroku každý agent urobí nejakú akciu.

## 2.2 Modifikácie zadania

### 2.2.1 Anonymita

Jedna z prvých vlastností, ktorú môžu mať agenti je anonymita. Agenti môžu, ale nemusia mať identifikátor. Týmto identifikátorom sa odlišujú navzájom od seba a rovnako od neho môže záležať aj výpočet, ktorý vykonávajú.

Anonymita sa môže týkať aj uzlov. Väčšinou sa uvažuje, že uzle nemajú žiadne označenie, jediný rozdiel, ktorý možno poznať je počet hrán.

### 2.2.2 Druh výpočtu

Agenti používajú dva druhy výpočtu, synchronný a asynchronný. Keď pracujú agenti asynchronne, algoritmus musí byť o niečo robustnejší resp. agenti musia mať nejakú výhodu. Hlavný problém je v tom, že keď agent prejde po

---

<sup>2</sup>napr. odovzdanie správy

nejakej linke, je nemožné zistiť či prešiel do čiernej diery alebo nie. Asynchrónny výpočet znamená, že prechod linkou môže trvať ľubovoľne konečne dlho. To znamená, že keď sa agent dlho nevracia<sup>3</sup> môže byť kvôli tomu, že agent spadol do čiernej diery a aj kvôli tomu, že mu prechod trvá dlho. Z tohto dôvodu sa väčšinou vyžaduje, aby agenti na začiatku vedeli počet vrcholov siete a čierna diera je práve jedna. Bližšie sa algoritmom s asynchrónnymi agentmi venujem v kapitole 3.

Agenti, ktorí pracujú synchronne majú veľkú výhodu. Ak sa nejaký z nich vydá po nejakej linke ostatní môžu počkať či sa vráti. Ak sa vráti do nejakého času tak vedia, že na druhej strane nie je čierna diera. Keď sa však do nejakého času nevráti a bol naprogramovaný tak, aby sa vrátil, vedia, že na druhej strane je čierna diera. Synchronnosť znamená, že agenti vykonávajú všetky svoje akcie v synchronných krokoch. Nám v podstate stačí, že existuje horné ohraničenie koľko agentovi trvá prechod nejakou linkou a to, že ostatní toto ohraničenie poznajú.

Synchronní agenti vedia odhaliť ľubovoľný počet čiernych dier v grafe aj bez znalosti ich počtu alebo počtu vrcholov. Kapitola 4 je práve o algoritmoch so synchronnými agentmi.

### 2.2.3 Komunikácia agentov

Ako bolo už spomenuté v úvode, bez toho aby agenti spolu komunikovali nie je možné vyriešiť problém BHS. Buď by sa agenti nemohli ani pohnúť alebo by protivník zariadil to, aby každý agent išiel rovnakou cestou rovno do čiernej diery. Podobne bolo spomenuté, že to že sa agenti navzájom nejak vidia je tiež spôsob komunikácie.

Prvý spôsob komunikácie je pomocou správ. Agenti si medzi sebou priamo posielajú správy. Tento spôsob sa využíva v synchronnom modeli a agenti sú schopní si posielat správy len keď sú v tom istom uzle. To znamená, že agenti o sebe vedia a vidia sa keď sú viacerí v jednom uzle. Na linkách sa agenti nemôžu dobehnúť a nevidia sa ani keď idú oproti sebe.

Druhý spôsob je komunikácia pomocou odkazov. V každom uzle sa nachádza tabuľa<sup>4</sup>. Na túto tabuľu môžu agenti písať a môžu z nej čítať. V jednom momente má k tabuli prístup len jeden agent. Na týchto tabuliach sa vykonáva všetka komunikácia. Agenti o sebe vôbec nevidia ani keď sú v tom istom uzle. Pri použití tabule sa zvykne uviesť aj veľkosť tabule v bitoch.

Modifikácia tabuľovej komunikácie je použitie žetónov<sup>5</sup>. Žetóny fungujú veľmi podobne ako tabule s konečným počtom bitov. Každý agent má určitý

---

<sup>3</sup>a bol naprogramovaný tak, aby sa vrátil

<sup>4</sup>whiteboard

<sup>5</sup>token

počet žetónov. Agent môže v nejakom uzle položiť žetón, ktorý budú ostatní vidieť. Žetóny sa ukladajú buď do stredu miestnosti alebo k nejakej hrane. Silu žetónov ovplyvňuje aj to či sú všetky žetóny rovnaké alebo si ich agenti rozoznajú.

### 2.2.4 Počiatočné znalosti

Vlastností distribuovaného systému, ktoré agenti na začiatku môžu a nemusia vedieť, je viacero. V prvom rade agenti môžu vedieť počet vrcholov grafu. Pri asynchrónnom modeli je veľmi vhodné vedieť veľkosť grafu. Ak agenti nevedia počet vrcholov, tak si nikdy nemôžu byť istí či už majú skončiť prehľadávanie grafu alebo nie<sup>6</sup>. Okrem toho sa občas predpokladá aj znalosť počtu čiernych dier resp. maximálny stupeň čiernej diery. Od tohto parametra sa často odvíja aj počet aktívnych agentov.

Niektoré modely predpokladajú, že agenti poznajú na začiatku topológiu celého grafu aj s pozíciou základne. V tomto prípade agenti musia len do tejto mapy označiť vrcholy, ktoré sú čiernymi dierami. Táto vlastnosť je veľmi silná a umožňuje prehľadávať sieť len s  $|B + 1|$  agentmi, kde  $|B|$  je počet čiernych dier. Okrem toho často zrýchli algoritmus.

Existujú modely so slabšou znalosťou ako je kompletná topológia. Agenti v tomto prípade majú *zmysel orientácie*. Tento zmysel orientácie je reprezentovaný pomocou pomenovania hrán vo vrchoch. Hovoríme, že agenti majú zmysel orientácie ak z dvoch postupností označení hrán, začínajúcich v tom istom vrchole, vedia jednoznačne povedať či aj končia v tom istom vrchole. Dá sa ukázať, že pre ľubovoľný graf existuje označenie portov také, že graf má zmysel orientácie[6].

### 2.2.5 Topológia siete

Väčšinou sa pri probléme BHS uvažuje s náhodným grafom. Avšak existujú aj špeciálne riešenia pre niektoré topológie[3]. Pri týchto riešeniach sa ukázalo, že pre dané topológie vieme urobiť rýchlejší algoritmus ako pre náhodný graf.

Špeciálne postavenie majú algoritmy na kruhoch. Kruh je veľmi zjednodušený graf preto aj algoritmy pracujúce len na kruhoch sú oveľa jednoduchšie. Napriek tomu sa práve vďaka kruhom dá jednoducho ukázať dolný odhad na počet krokov pre náhodné grafy.

---

<sup>6</sup>bližšie sa tomuto problému venujeme v kapitole 3

## 2.3 Po lopate

Ako hovorí názov tejto časti, skúsime "po lopate" vysvetliť problém BHS. Určite existuje mnoho prirovnaní pre distribuovaný systém, ale mne najbližší a najvhodnejší pre BHS je bludisko.

Distribuovaný systém je ako bludisko. Skladá sa z miestností čoby uzlov. Tieto miestnosti sú pospájané veľmi úzkymi tunelmi, ktorými sa vieme plaziť len tým smerom, ktorým sme do tunela vošli. Ale tunel je dosť široký natoľko, aby sme sa vedeli obísť s protiídúcou osobou bez toho, aby sme o nej vedeli. Vchody do tunelov sú porty a sú nejak označené.

Niektorá z miestností nemá podlahu a je to bezodné prepadlisko. Zvuk sa šíri tunelmi len minimálne.

Na začiatku sa v bludisku nachádza skupina odvážlivcov, ktorí majú za úlohu čo najrýchlejšie označiť všetky porty, ktorých hrany vedú do čiernej diery. Samozrejme, že prvoradé je, aby mali čo najmenšie straty.

Napríklad ak majú na začiatku mapu bludiska, tak môžeme hovoriť, že poznajú topológiu siete. Ak mapu nemajú, ale platí tam metrika a hrany sú priamočiare, tak majú v podstate istý zmysel pre orientáciu. No a keď im zaviažeme uši a ústa a dáme do každej miestnosti tabulu a jednu kriedu, tak môžu komunikovať pomocou odkazov.

# Kapitola 3

## Hľadanie v asynchrónnom modeli

### 3.1 Opatrné kráčanie

Opatrné kráčanie je metóda, ktorá sa používa pri ľubovoľnej modifikácii BHS. Najviac ho však vyzdvihujú práve pri práci s asynchrónnymi agentmi. Je to spôsob prechodu grafu počas hľadania, ktorým sa snažíme predísť stratám na agentoch.

Hlavná myšlienka je rozdeliť si porty do troch kategórií: *neznáme*, *nebezpečné*, *bezpečné*. Neznáme sú tie, o ktorých nevieme kam vedú a žiadny agent po nich nešiel. Nebezpečné sú také, do ktorých vošiel agent, ale ešte nevieme či žije. Bezpečné sú tie do, do ktorých niekto vošiel a vrátil sa. Ďalej platia tri základné pravidlá:

1. Žiadny agent nevojde do portu, ktorý je označený ako nebezpečný.
2. Ak idem cez neznámy port, tak ho pred tým označím ako nebezpečný.
3. Keď som prešiel neznámym portom, tak pred tým ako urobím hoci akú inú činnosť sa vrátim naspäť do pôvodného vrchola a označím port ako bezpečný. Ak agent nemá takúto možnosť<sup>1</sup>, tak aspoň odstráni značenie o nebezpečnom porte.

Značenie nebezpečných portov je veľmi dôležité. Zaručuje nám to, že do čiernej diery vojde maximálne toľko agentov, koľko do nej vedie hrán.

---

<sup>1</sup>napr. keď sa používajú jednoduché tokeny

## 3.2 Obmedzenia

Asynchrónny model má niektoré už spomenuté obmedzenia.

**Lema 1** *Nie je možné vyriešiť BHS pokiaľ sú v grafe dva alebo viac nepreskúmaných vrcholov.*

*Dôkaz:* Predstavme si dva nepreskúmané vrcholy  $u$  a  $v$ . Vďaka nášmu protivníkovi sa nedá v konečnom výpočte rozlíšiť či  $u$  je čierna diera a cesta do  $v$  trvá veľmi dlho alebo naopak. Preto nevieme kde je čierna diera.  $\square$

**Dôsledok 1** *Pokiaľ má graf viac ako 2 vrcholy, sú potrební aspoň dvaja agenti.*

*Dôkaz:* Ak sa v sieti nachádza viac ako 2 uzly, tak na začiatku sú všetky uzly okrem základne nepreskúmané. Z predchádzajúcej lemy vyplýva, že úloha nie je vyriešená. Preto aspoň jeden agent sa musí pohnúť zo základne. Náš protivník vie následne umiestniť čiernu diery tak, aby tento agent padol do nej. Preto musí byť minimálne ešte jeden agent, ktorý ostane nažive.  $\square$

**Lema 2** *Nie je možné overiť či v grafe naozaj čierna diera je.*

*Dôkaz:* Je celkom zrejmé, že pokiaľ nevieme či je v sieti čierna diera, náš protivník by mohol spomaliť všetkých agentov idúcich do vrchola  $u$ . Algoritmus, ktorý má skončiť v konečnom čase by nemohol s určitosťou povedať či  $u$  je alebo nie je čierna diera.  $\square$

Budeme pracovať len s modelmi s práve jednou čiernou diery. Okrem toho budeme aj pri neznalosti topológie vyžadovať, aby agenti vedeli počet vrcholov  $n$  a maximálny stupeň čiernej diery  $\Delta$ .

**Lema 3** *Nie je možné odhaliť čiernu diery pokiaľ nepoznáme veľkosť grafu.*

*Dôkaz:* Predstavme si sieť a v nej vrchol  $v$ . Agenti, ktorí idú do  $v$  sú spomalení natolko, že ostatní čakajú len na nich. Keďže nevedia aký je počet vrcholov, tak sa nemôžu rozhodnúť či je  $v$  čierna diera alebo či sa nachádza nejaký vrchol za ním.  $\square$

Dôležitá požiadavka je aj nutnosť dvojspojitého grafu.

**Lema 4** *Pokiaľ by graf nebol dvojspojité, tak nie je možné s určitosťou skončiť.*

*Dôkaz:* Predstavme si artikuláciu  $v$ , kde na jednej strane je základňa  $h$  a čierna diera je buď  $v$  alebo je na opačnej strane vrchola  $v$ . Agenti ktorí vojdú do portov smerujúcich do  $v$  sa nevracajú. To teraz môže znamenať, že  $v$  je čierna diera alebo sú spomalení našim protivníkom. Zvyšní agenti si však musia vybrať jednu z možností. Buď  $v$  prehlásia za čiernu dieru alebo budú čakať kým sa niektorý agent vráti. Ani jedna možnosť nemusí byť správna.  $\square$

### 3.3 Prehľadávanie náhodného grafu

V práci [6] sa podarilo dokázať niekoľko dolných aj horných ohraničení pre počet krokov algoritmov pracujúcich na náhodných grafoch, kde agenti komunikujú pomocou tabúľ. Horné ohraničenia samozrejme dokázali konštrukčne. Pozrieme sa na ne postupne podľa znalosti grafu.

#### 3.3.1 Bez znalosti topológie

##### Dolný odhad

V prvom rade autori dokázali výsledok, že na vyriešenie BHS na neznámom náhodnom grafe treba aspoň  $\Delta + 1$  agentov s tým, že ak  $n - 4 < \Delta < n$  tak ich možno stačí  $\Delta$ . To, že sa dá problém vždy vyriešiť s  $\Delta + 1$  agentmi je celkom zrejmé.

Dolný odhad časovej zložitosti je ukázaný pre ľubovoľný graf s maximálnym stupňom aspoň  $\Delta \geq 3$ . Myšlienka je vytvoriť prstencovitý graf, ktorý agenti z jedného miesta postupne do kruhu odkrývajú. Graf pozostáva z buniek tvorených  $2 \times \Delta + 2$  vrcholmi, ktoré sú usporiadané tak, aby tam bolo naraz  $\Delta$  neznámych portov. Bunky sú medzi sebou spojené vždy iba jednou hranou a tvoria kruh.

Na začiatku môžu ísť agenti doprava alebo doľava. Protivník sa správa tak, že na jednej strane nechá zablokovaný<sup>2</sup> jeden port (napr. naľavo) a na druhej strane dovolí agentom nájsť  $\Delta$  neznámych. Aby algoritmus fungoval korektne, musia zvyšní agenti ísť prehľadávať tieto porty, ktoré sú napravo. Následne protivník odblokuje port na ľavej strane, teda otvorí prechod do novej bunky. Napravo zablokuje práve jednu hranu a to prechod do novej bunky. Pre agentov to znamená nútený prechod naľavo.

Takto protivník stále strieda zablokované porty naľavo a napravo. Ak by sa agenti, ktorí nie sú na žiadnej linke rozhodli zaspáť a čakať, tak protivník

<sup>2</sup>spomalí agenta, ktorý sa ním vybral

vie nasmerovať všetky zablokované porty do čiernej diery. Potom by sa agenti, ktorí zaspali už nezobudili.

Preto musia agenti chodiť stále z jednej strany na druhú prechádzať za každým dlhé stredné územie. Po konkrétnych výpočtoch dostávame spodnú hranicu  $\Omega(n^2)$  krokov.

### Horný odhad

Efektívny algoritmus na prehľadávanie neznámeho náhodného systému, ktorý je popísaný v [6] využíva to, že graf je jednoduchý. Zo žiadneho vrchola nevedú dve linky do čiernej diery. To znamená, že stačia dvaja agenti na prehľadanie všetkých portov nejakého vrchola a minimálne jeden z nich nespadne do čiernej diery.

Keď objaví agent nový uzol, tak poznačí v základni, že treba dvoch agentov na prehľadanie toho uzla. Keď nejaký agent skončil prehľadávanie nejakého uzla, tak ide do základne, vyberie si jeden z neprehľadaných uzlov, poznačí na tabulu, že ho ide on prehľadávať a ide prehľadávať ten uzol.

To že vždy žije aspoň jeden agent zaručuje bezpečné kráčanie a počet agentov  $\Delta + 1$ . To že každý nájdený uzol je prehľadaný, je zaručené tým, že po jeho objavení sa to poznačí na tabulu v základni a voľný agent ide vždy do základne. No a je potom zrejmé, že agenti časom nájdu každý uzol.

Agenti sa po odkrytej časti grafu pohybujú po kostre. To znamená že prejsť do nejakého známeho uzla trvá maximálne  $n$  krokov. No a zakaždým keď agent nájde nový vrchol ide do základne. Nových vrcholov je maximálne  $n - 2$ , pretože nerátame základňu a čiernu diery. Je zrejmé, že agent urobí maximálne  $n^2$  krokov.

Takže horný odhad na riešenie BHS za daných podmienok na počet agentov je  $\Delta + 1$  a odhad časovej zložitosti je  $\Theta(n^2)$ .

### 3.3.2 So zmyslom orientácie

#### Horný odhad

Zmysel pre orientáciu nám zaručí jednu vec. Na korektné riešenie problému nám stačia dvaja agenti. Z dôsledku 1 vieme, že treba aspoň dvoch. Ak jeden padne do čiernej diery, ten druhý vie vďaka označeniam portov a zmyslu orientácie nespahnúť do tej istej čiernej diery. No a keďže čierna diera je iba jedna, tak tento druhý agent prežije.

Algoritmus nebude tak jednoduchý ako sme to pred chvíľou načrtli. Problém je v tom, že nevieme kedy už je agent v čiernej diere. Z toho dôvodu zavedieme pojem nebezpečného vrcholu namiesto nebezpečnej hrany. Nebez-

pečný vrchol je taký, do ktorého sa vybral jeden z agentov po nejakej hrane. Druhý agent vie jasne identifikovať takýto vrchol.

Agenti budú postupne objavovať vrcholy. Objaviť alebo preskúmať vrchol znamená vyskúšať prejsť všetkými jeho portami<sup>3</sup>. Ak jeden z agentov skončí prehľadávanie vrcholu  $u$ , stáva sa akýmsi vodcom a ide hľadať iný nepreskúmaný vrchol  $v$  a preskúmava ten. Keď druhý agent zistí, že zvyšok vrcholu  $u$  už je preskúmaný tak hľadá vodcu, aby mu pomohol preskúmať vrchol  $v$ , ktorý vodca momentálne prehľadáva.

Bezpečnosť aspoň jedného agenta je opäť zaručená opatrným kráčaním, ktoré je trochu upravené. Agenti musia prejsť všetkými hranami aspoň raz, samozrejme okrem hrán vedúcich do čiernej diery. Nájdenie nového nepreskúmaného vrchola trvá prehľadanie už objavenej kostry  $O(n)$ . Dobehtie vodcu trvá rovnako. No a maximálne  $n - 2$  krát sa presúvajú k novému vrcholu. Časový odhad  $O(n^2)$  krokov.

### Dolný odhad

Dolný odhad sa dá ukázať podobne ako v predchádzajúcej časti pomocou prstencové ho grafu. Budeme predpokladať, že graf má jednoznačne pomenované aj vrcholy<sup>4</sup>. Pomenovania portov budú podľa vrchola kam vedú. Toto pomenovanie spĺňa podmienky z definície. Akurát pomenovanie na porte hovorí iba o nasledujúcom vrchole a nehovorí nič o vrchole za ním. Vďaka tomuto vie protivník blokovaním liniek donútiť agentov chodiť stále z prava do ľava a naspäť.

Zakaždým keď sa agentovi odkryje cesta musí si skontrolovať či sa druhému agentovi neodkryla cesta do toho istého vrchola. Ak si to neskontrolujú, tak ich protivník nasmeruje oboch do čiernej diery. Aby si to skontroloval musí prejsť celou už preskúmanou časťou. Preto urobia aspoň  $\Omega(n^2)$  krokov<sup>5</sup>.

### 3.3.3 Úplná znalosť topológie

#### Horný a dolný odhad

Je zrejmé, že úplná znalosť topológie je vylepšený zmysel pre orientáciu, takže na prehľadanie grafu nám stačia dvaja agenti. Títo agenti však môžu pracovať rýchlejšie.

Kompletný algoritmus a dôkaz z [6] neuvediem avšak načrtnem aspoň hlavnú myšlienku. Tým, že agenti majú na začiatku mapu siete, vedľa si ju

<sup>3</sup>okrem toho, ktorý vedie do nebezpečného vrcholu

<sup>4</sup>to dáva výhodu ľubovoľnému algoritmu

<sup>5</sup>Presnejší popis tohto dôkazu sa dá nájsť v [6]

rozdeliť na dva nezávislé súvislé komponenty. Každý z nich prehľadá ten svoj.

Minimálne jeden z nich skončí a to ten, ktorý vo svojom komponente nemá čiernu dieru. Tento agent následne nájde nebezpečný port, po ktorom sa vybral druhý agent. Pri hľadaní používa iba bezpečné hrany, ktoré overil jeden z agentov. Keď nájde nebezpečný port, tak opäť rozdelí nepreskúmanú časť na dve a nechá o tom správu druhému agentovi.

Ak si agenti takto delia mapu vždy na dve podobne veľké polovičky, tak ju musia deliť  $O(\log n)$  krát. Po každom delení musia v najhoršom prípade prejsť  $O(n)$  hranami k druhému agentovi. Z tohto nám vychádza časová zložitosť  $O(n \log n)$  krokov.

Dolný odhad časovej zložitosti pre hľadanie čiernej diery na kruhu je  $\Omega(n \log n)$ , bližšie o tom pojednáva [5]. Preto tento odhad platí aj pre náhodné grafy.

### 3.4 Prehľadávanie dobre spojených grafov

V článku [3] sa venovali autori niektorým špeciálnym triedam grafov. Riešili na nich problém BHS so znalosťou topológie. Boli to nasledujúce triedy grafov: hyperkocky, hviezdicové grafy, cube-connected-cycles, wrapped butterflies, chordal rings.

Na týchto grafoch autori ukázali, že BHS ide riešiť pomocou dvoch agentov s časovou zložitosťou  $O(n)$  krokov. Tento výsledok dosiahli vďaka štruktúre grafov, ktorá umožňuje agentom rýchlo sa premiestňovať po grafe. Agenti sa presúvali na základe efektívneho smerovacieho algoritmu a tým sa podarilo dosiahnuť takúto dobrú časovú zložitosť.

### 3.5 Prehľadávanie s použitím žetónov

Na záver tejto kapitoly uvediem konkrétny výsledok uvedený v [4]. Autori sa tu zamerali na prehľadávanie neznámych náhodných grafov pomocou žetónov. Agenti mali jedinou možnosť komunikácie a to pomocou žetónov. Navyše, každý agent môže niesť iba jeden žetón. Žetóny sú nerozlíšiteľné. Agent môže žetón položiť buď do stredu miestnosti alebo k ľubovoľnému portu v miestnosti.

Treba si uvedomiť ako málo informácie vie agent žetónom odovzdať alebo uložiť. V prvom rade museli využiť žetóny na označenie nebezpečných portov. Tým pádom na ostatné signály im zostala iba možnosť uložiť žetón do stredu nejakej miestnosti.

Prvý spôsob akým si pomohli je ten, že ako špeciálny signál používajú

žetón v základni. Druhý spôsob je žetón v pomocnej miestnosti. Pomocnú miestnosť si zadefinovali ako miestnosť susediacu so základňou, ktorá je bezpečná a má v nejakom usporiadaní najnižšie číslo portu zo základne.

Následne potrebujú agenti prekonať dva problémy. Systém ako sa budú uspávať a budiť, keď nemajú aktuálne čo robiť. Tento problém je riešený položením signálneho žetónu do základne a následným zaspaním. Budiaci agent zdvihne žetón a presunie ho do pomocnej miestnosti. Tento *protokol* je však zložitejší, aby sa predišlo zablokovaniam.

Druhý veľký problém je s objavovaním. Keď agent príde do neznámeho vrchola, vrchol môže, ale nemusí byť novým vrcholom. Agent musí overiť či tento vrchol už náhodou nemá zaznačený v mape. S použitím nerozlišiteľných žetónov to nie je jednoduché. Preto si agent zvolí cyklus cez nový vrchol, bezpečné hrany smerom do pomocnej miestnosti a späť. Keď urobí  $n^2$  krokov po tomto cykle a všetky označenia portov sedeli s jeho vytvorenou mapu, tak je veľká šanca, že nájdený vrchol sa zhoduje s tým v jeho mape, ale ešte stále to tak byť nemusí. Keby bol ním prejdený cyklus násobkom cyklu, ktorý si na začiatku zvolil, tak sa mohol oklamať. Preto ešte pomocou žetónu v pomocnej izbe musí zistiť či to tak je. Tu nastáva problém s ostatnými agentmi, ktorý je opäť vyriešený niekoľko-násobným opakovaním špeciálneho *protokolu*.

Takéto *porovnanie* musí absolvovať pri objavení každého nového vrchola, s každým vrcholom mapy, ktorý má rovnaké názvy portov. Ak sa ani s jedným nezhodne, tak nájdený vrchol je skutočne novo objavný.

Autori pri tomto algoritme použili  $\Delta + 1$  agentov, ktorí urobia  $O(\Delta^2 m^2 n^7)$  krokov.

# Kapitola 4

## Hľadanie v synchrónnom modeli

### 4.1 BHS ako optimalizačný problém

#### 4.1.1 Základná myšlienka

Pri synchrónnom modeli sa na problém BHS pozerá trochu inak. Vďaka tomu, že agent môže s istotou prehlásiť vrchol za čiernu diery, je výsledný algoritmus viac deterministický. Protivníkovi sme zobrali jednu veľmi dôležitú schopnosť spomaľovať agentov a tým nás zneisťovať. Navyše majú agenti na začiatku znalosť topológie prehľadávanej siete.

Z tohto dôvodu sa už na problém prestávame pozeráť z hľadiska či nájdeme čiernu diery a ktorá z tých hrán kde zmizol agent naozaj vedie do čiernej diery. Nemusíme ani vylučovať rôzne možnosti, ak prideme k novému vrcholu, tak vieme jednoducho a rýchlo zistiť či je čiernou diery. Vieme akoby celý algoritmus pohybu všetkých agentov navrhnuť pred tým ako sa začnú hýbať.

BHS na synchrónnej sieti s vedomosťou topológie sa stáva problémom nájsť optimálny traverzovací algoritmus, ktorý nájde čiernu diery. Z tohto dôvodu sa BHS stáva optimalizačným problémom.

#### 4.1.2 Optimalizačný problém

Našou úlohou je pre daný graf a polohu základne, povedať počet agentov, ktorí ho budú prehľadávať a najrýchlejší traverzovací algoritmus, ktorý budú dodržiavať. Úlohou agentov je nájsť všetky čierne diery. Niekedy je povedané, že je maximálne jedna, inokedy je daný ich počet  $|B|$ .

## 4.2 NP-Ťažkosť problému

Optimalizačný problém BHS je NP-Ťažký [10]. Dôkaz sa robí redukciami na modifikovaný problém hamiltonovskej kružnice.

### 4.2.1 Modifikovaný problém hamiltonovskej kružnice

**Zadanie:** Kubický planárny graf  $G = (V, E)$  a jeho hrana  $(x, y) \in E$ .

**Riešenie:** Odpoveď na otázku, či v grafe existuje hamiltonovská kružnica prechádzajúca hranou  $(x, y)$ .

Náš problém:

**Zadanie:** Graf  $G' = (V', E')$ , základňa  $h$  a celé číslo  $X$ .

**Riešenie:** Odpoveď na otázku, či existuje traverzujúca schéma začínajúca v  $h$  taká, že BHS algoritmus na nej založený stojí najviac  $X$ .

Tu nebudem uvádzať dôkaz. Pre môj ročníkový projekt je dôležitý výsledok a jeho dôsledky.

## 4.3 Aproximácia problému

V tom istom článku [10] autori následne ukazujú, že problém nejde aproximovať s lepším pomerom ako  $3/2$ . Okrem toho ponúkajú algoritmus aproximujúci daný problém s pomerom  $7/2$ .

Ďalšie výsledky dosiahnuté tou istou skupinou autorov je aproximácia problému s pomerom  $7/2 - 1/8$  [11].

Okrem toho sa pán Klasing a Radzik venovali aj modifikácii tohto problému pre viac čiernych dier [2]. V takomto prípade navrhli algoritmus so zložitou  $O(\frac{n}{k} \times \frac{\log n}{\log \log n} + b \times D_b)$ , kde  $k$  je počet agentov,  $b$  je počet čiernych dier a  $D_b$  je najväčší priemer grafu po odstránení najviac  $b$  čiernych dier. Okrem toho majú predpoklad  $b \leq \frac{k}{2}$ . Dolná hranica časovej zložitosti takéhoto algoritmu je pritom  $\Omega(\frac{n}{k} + D_b)$ .

# Záver

V tomto projekte som sa snažil v prvom rade zdefinovať problém hľadania čiernej diery. Mojm cieľom bolo popísať zadaný problém a jeho modifikácie. Zjavne, problém má mnoho rôznych variánt a pri niektorých sa k riešeniu dá pristupovať z úplne iného smeru. Je ťažké sa venovať všetkým modifikáciám podrobne a ani to nebolo náplňou tohto projektu.

Okrem toho som v skratke zhrnul najzaujímavejšie výsledky dosiahnuté pre BHS. Popísal som aj základnú myšlienku niektorých riešení a dôkazov. Tento popis by mal slúžiť ako zdroj nápadov pre niekoho, kto sa chce problému bližšie venovať. Mnohé myšlienky môžu byť veľmi inšpiratívne pri riešení ďalších modifikácií.

Možností ďalšieho výskumu v tejto oblasti je viacero ako vidieť z mnohých modifikácií zadania problému. Napríklad, žiadne z riešení, ktoré som doteraz stretol, sa nevenovali orientovaným grafom. Pri orientovaných grafoch nám prestávajú fungovať základné metódy, ktoré boli použité pri tu menovaných riešeniach. Okrem toho sú zaujímavé aj kombinácie BHS s inými známymi problémami v oblasti distribuovaných systémov.

# Literatúra

- [1] Hagit Attiya, Marc Snir, and Manfred K. Warmuth. Computing on an anonymous ring. *J. ACM*, 35(4):845–875, 1988.
- [2] Colin Cooper, Ralf Klasing, and Tomasz Radzik. Searching for black-hole faults in a network using multiple agents. In *Proceedings of the 10th International Conference on Principles of Distributed Systems (OPODIS 2006)*, volume 4305 of *Lecture Notes in Computer Science*, pages 320–332. Springer Verlag, December 2006.
- [3] S. Dobrev, P. Flocchini, R. Kráľovič, P. Ružička, G. Prencipe, and N. Santoro. Black hole search in common interconnection networks. *Netw.*, 47(2):61–71, 2006.
- [4] Stefan Dobrev, Paola Flocchini, Rastislav Kralovic, and Nicola Santoro. Exploring an unknown graph to locate a black hole using tokens. In *TCS*, 2006.
- [5] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Mobile search for a black hole in an anonymous ring. *Lecture Notes in Computer Science*, 2180:166–179, 2001.
- [6] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Searching for a black hole in arbitrary networks: optimal mobile agent protocols. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 153–162, New York, NY, USA, 2002. ACM.
- [7] Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Multiple agents rendezvous in a ring in spite of a black hole. In *OPODIS*, volume 3144 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2003.

- [8] Stefan Dobrev, Rastislav Kralovic, Nicola Santoro, and Wei Shi. Black hole search in asynchronous rings using tokens. In *CIAC*, pages 139–150, 2006.
- [9] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Approximation bounds for black hole search problems. *Networks*, 2007. to appear.
- [10] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Hardness and approximation results for black hole search in arbitrary graphs. *Theoretical Computer Science*, 384(2–3):201–221, October 2007. to appear.
- [11] Ralf Klasing, Euripides Markou, Tomasz Radzik, and Fabiano Sarracco. Hardness and approximation results for black hole search in arbitrary networks. *Theor. Comput. Sci.*, 384(2–3):201–221, 2007.